Université catholique de Louvain
Faculté des Sciences Appliqués
Département d'Ingénierie Informatique

Constructivist Learning: An Operational Approach for Designing Adaptive Learning Environments Supporting Cognitive Flexibility

Vu Minh Chieu

Thèse présentée en vue de l'obtention
du grade de Docteur en Sciences Appliquées

Membres du jury :

Prof. Y. Deville
Université catholique de Louvain, Département d'Ingénierie Informatique
Prof. M. Frenay (co-promotrice)
Université catholique de Louvain, Département de Psychologie de l'Education et du Développement
Prof. J. D. Legat (président)
Université catholique de Louvain, Département d'Electricité
Prof. E. Milgrom (co-promoteur)
Université catholique de Louvain, Département d'Ingénierie Informatique
Prof. C. Vander Borght
Université catholique de Louvain, Département de Biologie
Prof. W. van Joolingen
Universiteit Twente, Faculté des Sciences de Comportement

Septembre, 2005

Abstract

Constructivism is a learning theory that states that people learn by actively constructing their own knowledge, based on prior knowledge. Many different perspectives exist on constructivist pedagogical principles and on how to apply them to instructional design. It is thus not only difficult to evaluate the conformity of existing learning systems with constructivist principles, it is also quite hard to ensure that a new learning system being designed will ultimately facilitate and stimulate constructivist learning.

A critical characteristic often mentioned in learning systems is adaptability. That is, the ability to provide a learning experience that is continuously tailored to the needs of the individual learner.

The present research aims to help designing truly constructivist and adaptive learning systems. For that purpose, it is necessary to clarify what constructivism entails in an *operational* manner: I propose a set of criteria for certain aspects of constructivism and use it both as guidelines for designing learning systems and for evaluating the conformity of learning systems with these constructivist principles.

One facet often mentioned as being strongly relevant to constructivism is cognitive flexibility, meaning the ability to spontaneously restructure one's knowledge, in many ways, in adaptive response to radically changing situational demands.

The claim I make in the present thesis is that *the operational approach I proposed makes the design and use of adaptive learning environments supporting cognitive flexibility straightforward and effective*. More specifically, the dissertation makes four main contributions to the interdisciplinary field of learning and e-Learning technology.

Firstly, the thesis proposes operational criteria for cognitive flexibility and presents both justifications and examples of their use. The set of criteria may be used in different instructional situations for designing and evaluating conditions of learning.

Secondly, on the basis of the criteria for cognitive flexibility, the thesis proposes an operational instructional design process and shows an example of its use. The process may also be applied in a variety of instructional situations for the design and use of learning systems fostering cognitive flexibility.

Thirdly, the thesis introduces a new, open-source, domain-independent, Web-based adaptive e-Learning platform, named COFALE, and illustrates an example of its use. The platform may be used for designing adaptive learning systems supporting cognitive flexibility in various domains.

And fourthly, the thesis reports on a preliminary evaluation of the example handled by COFALE with actual learners. The study provides a certain number of encouraging results for fostering cognitive flexibility by means of ICT-based learning conditions.

Keywords

Constructivism

Cognitive flexibility

Instructional design

Operational criteria

Operational approach

Teaching recursion

E-Learning

Adaptive learning systems

Computer-based instruction

Distance learning

Learning objects

Open-source learning platform

Acknowledgements

I would like to thank "Commissariat Général aux Relations Internationales de la Communauté Wallonie-Bruxelles" (CGRI) for partly supporting the current research project.

I also would like to thank many people for their support, encouragement, and guidance during my years as a PhD student here at the Department of Computing Science and Engineering, *Université catholique de Louvain* (INGI).

First and foremost, this dissertation represents a great deal of time and effort not only of my part, but on the part of my advisors, Elie Milgrom and Mariane Frenay. Without their invaluable assistance, this domain-interdisciplinary dissertation would not have finished.

On the one hand, as a specialist in the domains of computing science and e-Learning technology, Elie Milgrom has helped me frame my research from day one, pushed me to get through the inevitable research setbacks, and encouraged me to achieve to the best of my ability. He always tells me: "I am here to help you develop the best of your intellectual ability. It is you, not me, who are responsible for your research project: You should always think about how to bring your research objectives about and how to convince people that you have attained your research objectives." I am really interested in this constructivist manner he has exploited to help me in my research.

On the other hand, as a specialist in the domains of learning psychology and pedagogy, Mariane Frenay has provided me with a lot of invaluable discussions so that I have been able to finish my thesis successfully in an interdisciplinary domain. I really appreciate her because of many detailed comments she has given to me during my PhD research.

I also thank my other three committee members, Yves Deville, Cécile Vander Borght, and Wouter van Joolingen, for valuable discussions and comments regarding my research, and my committee president, Jean-Didier Legat, for his organization of my PhD defenses.

I am indebted to many other people at INGI who have helped me finishing the present dissertation. Marc Lobelle, the former president of INGI, has provided me with much help in my everyday life and in my participation in international scientific conferences. Yves Deville, the president of INGI, also has encouraged and helped me to participate in international scientific conferences. Christine Jacqmot has encouraged me, especially in the early states of my research. Claude Daubies, Francis Degey, Viviane Dehut, Fabienne Delbrouck, Chantal Poncin, Pierre Reinbold, Stéphanie Remacle-Landrain, Phan Manh Tien, Marie-France Declerfayt, Freddy Gridelet, and Michèle Piquart have provided me with much support for administrative and technique issues regarding not only my research but also my everyday life.

On a more personal note, I have been lucky to have many close friends who have helped to make my time as a PhD student enjoyable. Because of limited space, I could not list here the name of those friends.

Finally, I thank my family for their continual support: My parents, Vu Tien Nam and Vu Thi Cham, my brothers Quang and Vinh, and my sister Anh have always been, and continue to be, there for me at all times. And most of all, I wish to express thanks forever to my wife Lien and my son Duc for enduring so many lonely evenings and weekends while I was pre-occupied with my research. It is my privilege to dedicate this dissertation to my wife and my son, with love.

Contents

List of Figures

List of Tables

Introduction

I had lived in the countryside until my adolescence. The first time I came to Hanoi, the capital of Vietnam, I wanted to buy a new book in which I was very interested. I asked my uncle to accompany me to a bookstore because I knew nothing about the very complex transportation system of Hanoi. Instead of satisfying my desire, he gave me a street map of Hanoi and several bookshop addresses, and told me to find out my way around by myself. It was very hard for me to get to the right bookshop where I bought the book. The result, however, was great: I learned not only how to get to several bookstores but also how to use a map, how to ask people in the street about transportation, and so on; I was able to get to any place in Hanoi with the techniques I learned; in other words, *I knew what it is to know my way around*. My experience is a kind of *active learning* (Perkins, 1996).

I concern myself with learning. I must therefore first know what learning is. To know what learning is, I must rely on a learning theory. There are, however, many theories of learning (Kearsley, 2003). I see that various forms of constructivism have emerged during the past fifteen years (Driscoll, 2000). Constructivism, as defined by Santrock (2001), is a learning theory that “emphasizes that individuals learn best when they *actively* construct their knowledge and understanding” (p. 318). Bourgeois and Nizet (1999) stated that constructivist learning is a process of *active* construction and transformation of knowledge. Many researchers in science education, educational psychology, and instructional technology accept constructivism (Driscoll, 2000; Santrock, 2001). Personally, I also like the model provided by constructivism describing how people learn, for example my learning experience presented in the previous paragraph. Therefore, I decided to do research on constructivism, and particularly, I want to know how to design *constructivist learning environments*.

Constructivist researchers (e.g., Driscoll, 2000; Santrock, 2001; Wilson, 1996) have claimed that information and communication technology (ICT) could provide significant help in implementing constructivist learning conditions. This claim has been also evidenced by the appearance of a significant number of ICT-based "constructivist" learning systems (Kinshuk et al., 2004). I wanted to know how ICT could facilitate and stimulate constructivist learning. So, I decided to investigate on how to design *ICT-based constructivist learning environments*.

I also concern myself with adaptive learning systems. Adaptation is a technique of providing a particular student with the most appropriate learning conditions such as learning contents and activities to facilitate his or her process of knowledge construction and transformation (Bourgeois & Nizet, 1999; Santrock, 2001). Adaptation support is useful because most learners within a learning environment have different personal characteristics such as prior knowledge, learning preferences, and learning progress (Brusilovsky, 1999; Milgrom et al., 1997; Stoyanov & Kirschner, 2004). The main goal of my research is thus to help designing *ICT-based constructivist and adaptive learning environments*.

The question addressed by my thesis:

How to exploit ICT effectively to design constructivist and adaptive learning environments?

More specifically:

- How to exploit ICT to provide the individual learner with appropriate learning conditions that truly facilitate and stimulate constructivist learning?
- How to help the teacher design ICT-based adaptive learning environments from a constructivist point of view?

A major problem to answer the previous thesis question has been that, while many descriptions and pedagogical principles for constructivism exist, there is little *practical* advice on how to design constructivist learning environments and on how to evaluate the conformity of learning environments with constructivist principles (Driscoll, 2000; Jonassen & Rohrer-Murphy, 1999). Indeed, educational theorists tend to accept the central assumption of constructivism presented previously; they derive, however, many different pedagogical implications from the same basic principles. Driscoll (2000), for instance, examined multiple perspectives on constructivism and identified at least five major facets of constructivism related to instructional design:

1. *Reasoning, critical thinking, and problem solving* (Cognition and Technology Group at Vanderbilt, 1991a; Perkins, 1991a).
2. *Retention, understanding, and use* (Edelson et al., 1996).
3. *Cognitive flexibility* (Feltovich et al., 1996; Spiro et al., 1991).
4. *Self-regulation* (Duffy & Cunningham, 1996).
5. *Mindful reflection and epistemic flexibility* (Language Development and Hypermedia Group, 1992).

To help educators design and evaluate constructivist learning conditions, educational theorists have suggested various guidelines and criteria. For example, both the Cognition and Technology Group at Vanderbilt (1991a) and Jonassen (1999) argued that learners must cope with complex situations for problem-solving skills to be maximally facilitated. Regarding cognitive flexibility, Spiro and colleagues (1991) advocated: “Revisiting the same material, at different times, in rearranged contexts, for different purposes, and from different conceptual perspectives is essential for attaining the goals of advanced knowledge acquisition” (p. 28). Bourgeois and Nizet (1999) stressed that social negotiation is required for students to come to understand another’s point of view. Reeves and Okey (1996) and Shepard (1991) argued for methods of assessment such as interviewing, observations, and holistic task performance (e.g., to ask students to write an essay, conduct an experiment, or carry out a project).

From these indications, I deduce that course designers should examine constructivist learning conditions in four key components of learning systems:

1. *Learning contents* (e.g., concept introductions, examples, exercises, and case studies).

2. *Pedagogical devices* (e.g., methods and tools provided for learners for exploring learning contents).
3. *Human interactions* (e.g., means and techniques for engaging tutors and learners in exchanges).
4. *Assessment* (e.g., problems and tools for determining whether learners have achieved the objectives of the instruction).

I believe, however, that the previous indications are still too general for educators to be able to imagine concrete steps when they want to design or evaluate constructivist learning environments in specific situations. This is why I propose *operational* criteria (stressing the qualifier "operational"). And I decided to choose *cognitive flexibility* among many facets of constructivism previously presented for proposing criteria.

According to Spiro and Jehng (1990), cognitive flexibility is "the ability to spontaneously restructure one's knowledge, in many ways, in adaptive response to radically changing situational demands" (p. 165). Driscoll (2000) examined the assumptions proposed by Spiro and Jehng and identified two principal learning conditions fostering cognitive flexibility:

1. *Multiple modes of learning* (i.e., multiple representations of contents, multiple ways and methods for exploring contents).
2. *Multiple perspectives on learning* (i.e., expression, confrontation, and treatment of multiple points of view).

I chose cognitive flexibility because of three main reasons. Firstly, I see that the pedagogical principles underlying cognitive flexibility reflect the basic characteristics of constructivism (Spiro et al., 1988, 1990, 1991). Secondly, I see that cognitive flexibility is a major common point among many constructivist researchers (e.g., Bourgeois & Nizet, 1999; Driscoll, 2000; Spiro & Jehng, 1990). And thirdly, I believe that ICT may facilitate the implementation of learning situations supporting cognitive flexibility, as Driscoll (2000) and Wilson (1996) showed with several hypermedia examples.

My operational approach:

To facilitate instructional design, I transform the pedagogical principles underlying cognitive flexibility into operational criteria. An operational criterion for cognitive flexibility is a test that allows a straightforward decision about whether or not a learning situation reflects the pedagogical principles underlying cognitive flexibility. The way I propose criteria for cognitive flexibility is to examine each of the two learning conditions fostering cognitive flexibility in each of the four components of learning environments identified earlier. For example, regarding multiple modes and learning contents, I propose a criterion to determine whether a learning content is represented in different forms such as text, images, and simulations.

I argue in this thesis that such criteria provide a useful framework both for designing and for evaluating learning environments supporting cognitive flexibility. My thesis makes the following four main contributions to the interdisciplinary field of learning and e-Learning technology.

Thesis contributions:

- *A set of operational criteria*. The set of criteria for cognitive flexibility may be used to devise and evaluate learning conditions *easily* in different instructional situations such as traditional instruction, computer-based instruction, and distance education. The way I have proposed criteria for cognitive flexibility may be *reused* to propose criteria for other facets of constructivism.
- *An operational instructional design process*. The process consists of a number of instructional design activities. The process ensures the design of a course satisfies all the criteria for cognitive flexibility.
- *A domain-independent e-Learning platform*. COFALE is an open-source, Web-based adaptive learning environment supporting cognitive flexibility. One can use many instructor tools and guidelines provided by COFALE to design online courses with support both for cognitive flexibility and for adaptability. One may also modify COFALE's source code to exhibit other pedagogical principles than cognitive flexibility.
- *A preliminary evaluation of COFALE*. A short-term study was performed with a small number of first-year engineering students in FSA/UCL to formatively evaluate the COFALE learning environment. Several encouraging results were reported. For example, students were satisfied with and interested in learning with the assistance of COFALE.

In what follows, I shortly describe the content and objectives of each chapter of the thesis. I have organized the thesis into four parts. The first part, regrouping the first three chapters, explains the pedagogical framework I propose for designing and evaluating learning systems. The second part, regrouping chapters 4 and 5, presents some background and related work. The third part, regrouping the next three chapters, describes the COFALE learning environment. And the fourth part presents the last chapter concerning a preliminary evaluation of my approach.

Chapter 1 describes the assumptions of constructivism and multiple facets of constructivism related to instructional design. This chapter does not contribute anything new; it simply explains the educational paradigm I follow in this thesis. Reading this chapter is important for understanding the main contributions of the thesis presented in the next chapters.

Chapter 2 proposes criteria for cognitive flexibility and presents both justifications and examples of their use. After examining this chapter, one can use the criteria to design or evaluate one's own instructional situations. One can also propose one's own criteria for any pedagogical principle in the same way I have proposed such criteria for cognitive flexibility.

Chapter 3 presents an operational instructional design process taking into account all the criteria presented in chapter 2, together with an example about teaching recursion in computing science. The design process helps clarifying how I have devised learning conditions to satisfy all the criteria for cognitive flexibility. It is neither final, neither normative, nor prescriptive. After reading this chapter, the practitioner should be able to find out his or her own way to exhibit the desired characteristics of cognitive flexibility.

Chapter 4 explains several key concepts related to ICT-based learning systems: learning content management systems, learning objects, and adaptive learning systems.

Chapter 5 provides an analysis of work related to ICT-based constructivist and adaptive learning systems. Firstly, I use the set of criteria for cognitive flexibility introduced in chapter 2 to analyze the conformity of several "constructivist" learning systems with cognitive flexibility. Secondly, I analyze adaptation support of several adaptive learning systems. The purpose of those analyses is to show the new features that COFALE (presented in the next chapters) adds on to the state of the art.

Chapter 6 illustrates the learning conditions presented in chapter 3 in a new adaptive e-Learning platform (COFALE). COFALE is based on ATutor, an open-source, Web-based learning content management system provided by the Adaptive Technology Resource Center (2004). The demonstration aims to show that it is possible to create ICT-based adaptive learning conditions satisfying all the criteria for cognitive flexibility identified in chapter 2.

Chapter 7 presents COFALE's authoring tools and guidelines allowing the course designer to create learning environments such as the one presented in chapter 6. Chapters 6 and 7 argue that the operational approach of the thesis is useful for exploiting ICT and the pedagogical principles underlying cognitive flexibility.

Chapter 8 gives an overview of the implementation of the COFALE system.

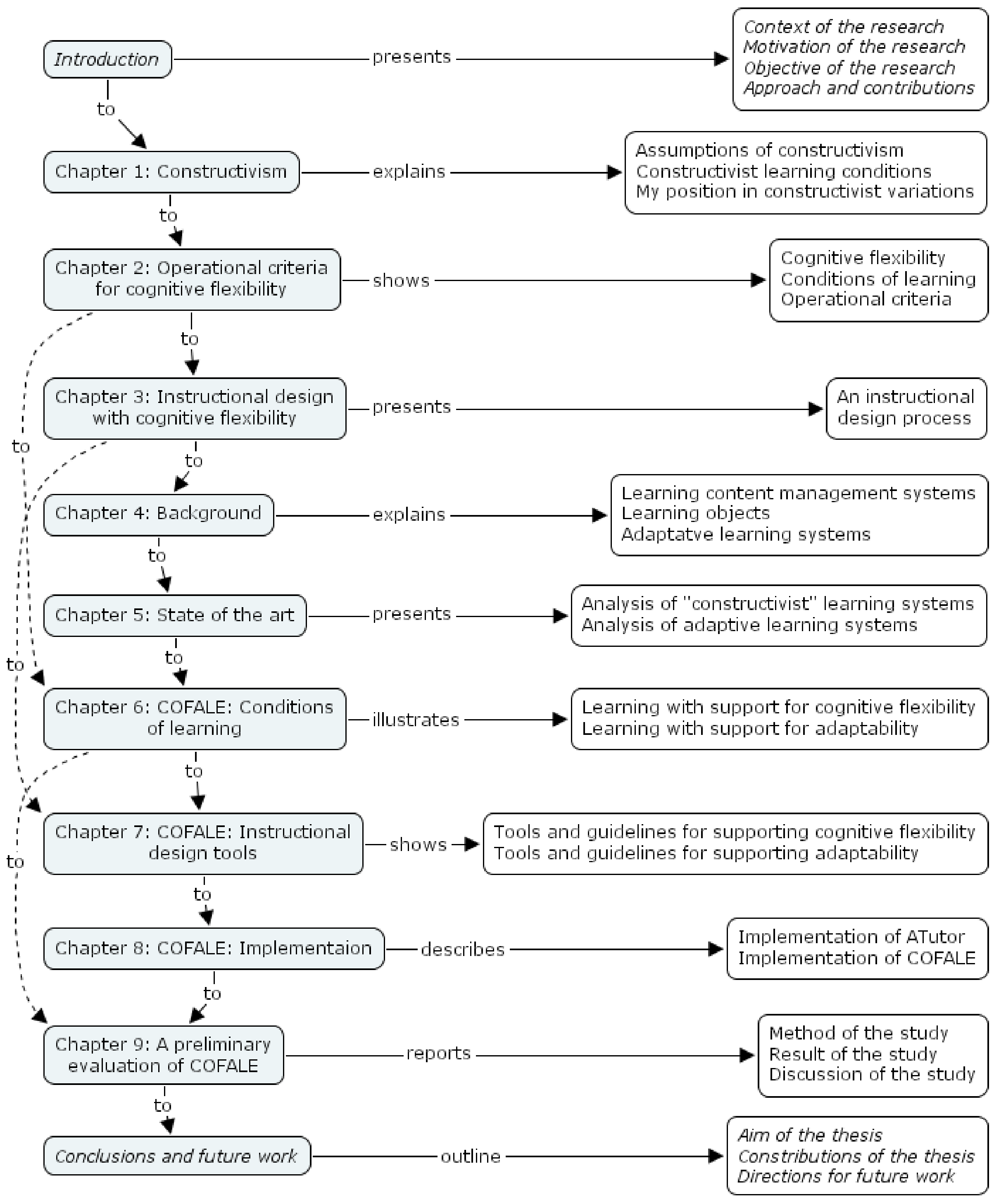
Chapter 9 reports on a preliminary study carried out to formatively evaluate COFALE. The 2-week-long experiment was performed with nine first-year engineering students in FSA/UCL. Several encouraging results were reported for learning with the help of COFALE. Feedback was also analyzed to ameliorate the design and use of the COFALE system.

To conclude, I look again at the thesis question: "*How to exploit ICT effectively to design constructivist and adaptive learning environments?*" I give my following answer with the arguments presented in the previous chapters. I also outline several promising directions for future work.

My affirmation in this thesis:

The operational approach used in this thesis makes the design and use of adaptive learning environments supporting cognitive flexibility straightforward and effective.

A schematic diagram of the structure of the thesis


Introduction
presents
Context of the research
Motivation of the research
Objective of the research
Approach and contributions
to
Chapter 1: Constructivism
explains
Assumptions of constructivism
Constructivist learning conditions
My position in constructivist variations
to
Chapter 2: Operational criteria for cognitive flexibility
shows
Cognitive flexibility
Conditions of learning
Operational criteria
to
Chapter 3: Instructional design with cognitive flexibility
presents
An instructional design process
to
Chapter 4: Background
explains
Learning content management systems
Learning objects
Adaptatve learning systems
to
Chapter 5: State of the art
presents
Analysis of "constructivist" learning systems
Analysis of adaptive learning systems
to
Chapter 6: COFALE: Conditions of learning
illustrates
Learning with support for cognitive flexibility
Learning with support for adaptability
to
Chapter 7: COFALE: Instructional design tools
shows
Tools and guidelines for supporting cognitive flexibility
Tools and guidelines for supporting adaptability
to
Chapter 8: COFALE: Implementaion
describes
Implementation of ATutor
Implementation of COFALE
to
Chapter 9: A preliminary evaluation of COFALE
reports
Method of the study
Result of the study
Discussion of the study
to
Conclusions and future work
outline
Aim of the thesis
Constributions of the thesis
Directions for future work

⟶ : Logical links across the structure

- - -▶ : Possible shortcuts across the structure

PART ONE: CONSTRUCTIVISM, COGNITIVE FLEXIBILITY, AND INSTRUCTIONAL DESIGN

This part presents the extended work of the following paper:

Chieu, V.M., Milgrom, E., & Frenay, M. (2004). Constructivist learning: Operational criteria for cognitive flexibility. *The Fourth IEEE International Conference on Advanced Learning Technology*, Joensuu, Finland, 221–225 (full paper).

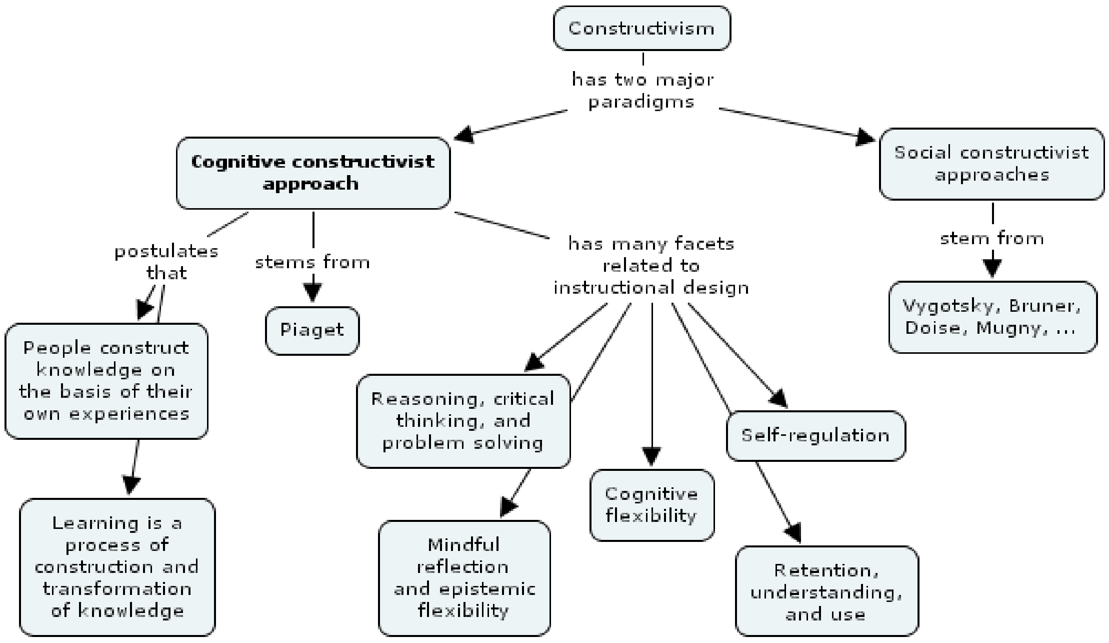
CHAPTER 1

Constructivism

"*The world, as we perceive it, is our own invention.*"

Heinz von Foerster (1988, p. 45–46), Austrian Constructivist, 1911 – 2002

In this chapter, I first give an overview of a variety of perspectives on constructivism: a cognitive constructivist approach stemming from the views of Piagetian theorists and social constructivist approaches stemming from the views of Vygotsky, Bruner, Doise, Mugny, … Then, following the cognitive constructivist approach, I examine three main issues of any theory of learning and instruction: (a) what knowledge is, (b) what learning is, and (c) what conditions that facilitate and stimulate learning are. The main objective of the chapter is to clarify my position in those constructivist variations. Reading this chapter is important for understanding my contributions presented in the next chapters of the thesis. After reading this chapter, one *should* construct one's own perspective on constructivism.



Constructivism
has two major paradigms
Cognitive constructivist approach
Social constructivist approaches
postulates that
stems from
has many facets related to instructional design
stem from
People construct knowledge on the basis of their own experiences
Piaget
Vygotsky, Bruner, Doise, Mugny, ...
Reasoning, critical thinking, and problem solving
Self-regulation
Cognitive flexibility
Learning is a process of construction and transformation of knowledge
Mindful reflection and epistemic flexibility
Retention, understanding, and use

Summary

1.1 Constructivism: Multiple paradigms

Many educational theorists tend to accept that, from a constructivist point of view, people learn best when they actively construct their own knowledge and understanding. There is, however, no single constructivist theory of learning as well as of instruction (Driscoll, 2000). Rather, researchers in fields from science education to educational psychology and instructional technology are articulating various aspects of constructivism. These constructivist variations include generative learning (Cognition and Technology Group at Vanderbilt, 1991a, 1991b; Wittrock, 1985a, 1985b), discovery learning (Bruner, 1986), embodied cognition (Johnson, 1987; Lakoff, 1987), and cognitive flexibility theory (Spiro et al., 1991). Constructivism is only one of the labels used to describe those constructivist variations.

Key concept: *Constructivism* is a learning theory that emphasizes that individuals learn best when they actively construct their own knowledge and understanding.

Constructivist researchers in the field of learning psychology (e.g., Bourgeois and Frenay, 2002; Santrock, 2001) have classified different constructivist approaches into two major paradigms: (a) a cognitive constructivist approach supported by the view of Piaget (1975) and (b) a number of social constructivist approaches supported by the views of Bruner (1996), Doise and Mugny (1997), Vygotsky (1962), and so on. Researchers who follow the first paradigm believe that learners construct knowledge by transforming, organizing, and reorganizing previous knowledge and information. And researchers who follow the second one believe that learners construct knowledge through social interactions with others. Moving from the first paradigm to the second one, the conceptual shift is from the individual development to collaboration and social interaction (Rogoff, 1998). This statement does not mean that the first paradigm neglects social interaction or the second one ignores individual development. Both of them take into account both aspects but with different emphases (Bourgeois & Nizet, 1999). According to Bourgeois and Nizet, socio constructivism mainly tackles the following question to which cognitive constructivism does not attach importance: In which conditions and according to which methods social interactions foster learning?

Key concept: Cognitive constructivism underlines individual development whereas social constructivism emphasizes collaboration and social interaction.

Sometimes the distinctions among constructivist approaches are not clear-cut and questionable (Marshall, 1996). It seems to me that certain constructivist researchers, for instance, Jonnaert and Vander Borght (2003) with the social constructivist and interactive approach, emphasize both individual development and social negotiation. So, when the word "socio" is present in an educational approach, it does not necessarily mean that the approach is social constructivist. A number of resources (e.g., Bourgeois & Nizet, 1999, chapters 3 & 7; Driscoll, 2000, chapter 11; Santrock, 2001, chapter 9) provide further discussions on the debate of constructivist variations.

Although the previous distinction of the two paradigms is debatable, I follow it because it provides at least a mean to clarify my position in constructivist variations. In this thesis I decided to choose the first paradigm: The cognitive constructivist approach stemming from the views of Piagetian theorists (e.g., Bourgeois & Nizet, 1999; Driscoll, 2000). I chose the first paradigm because several constructivist researchers (e.g., Kinshuk et al., 2004; Wilson, 1996) have claimed that ICT is a very promising means for creating learning conditions exhibiting the pedagogical principles underlying this paradigm. Hereafter when I use the term “constructivism”, I mean this cognitive constructivism.

Key concept: The paradigm the present thesis follows is cognitive constructivism.

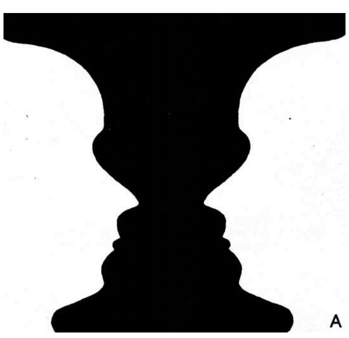
To present constructivism, as any theory of learning and instruction, I first examine three main issues: (a) what knowledge is, (b) what learning is, and (c) what constructivist learning conditions are. Then, I clarify the position I follow in this thesis among a variety of perspectives on constructivist learning conditions. Understanding this position is essential for understanding the thesis.

1.2 Constructivism and knowledge

What is knowledge in a constructivist point of view? I begin with examining several examples. Then, I give my own definition of knowledge on the basis of the views of Piagetian theorists.

Looking at Figure 1.1, what can we see? Certain people could see a vase whereas other people could see two faces. This example shows that when confronting the same information, people construct different knowledge (or representations).

Figure 1.1. The vase of Rubin (1915)



Now let me examine another example:

> One day, King Tang Tai Zong (China, 626–649 AD) asks his high-ranking mandarin Xu Jing Sun:
> – I can see that you are not a bad person. So why are there slanders about you and many people dislike you?
> Xu Jing Sun answers:
> – Your Majesty, during spring time it rains very frequently, the farmers are so glad that their fields are watered but the pedestrians are not happy because the rain makes the road so slippery. When the moonlight is brightest in autumn like a mirror on the sky, poets are happy to see such beautiful sight but burglars are afraid of its brightness. God is fair but people can blame him even when it is sunny or rainy. I am not perfect so I cannot avoid being the subject of slanders. So towards these slanders, I think you should take time to consider them and should not rush into any conclusion. If a King believes in those slanders people said, the mandarins will become victims. If parents believe in slanders regarding their children, their children will suffer. If the husband and wife believe slanders about each other, their relationship will be damaged. Slanders are even more poisonous than the venom of snakes, sharper than knives, and kill without leaving a blood stain (translated from Vietshare.com, 2004).

This classic reference provides direct evidence that people, through their own experiences, construct their own understanding about the environment surrounding them. De facto, individuals create different knowledge about the same natural phenomena. Even the same individual, at different times, constructs different knowledge of the same information. For example, moving from a farmer to a pedestrian or vice versa, he or she makes different sense of the rain.

The situations such as those in the previous classic reference are also frequent in our today life. Here is an example proposed by Kuhn (1983):

> To know what scientists' representation about the atomic theory is, a North-American researcher questions two specialists recognized by the international scientific community in their respective field: chemistry and physics. The researcher asks them whether or not a helium atom is a molecule. The answer of the chemist and the answer of the physicist do not agree to each other. For the chemist, the helium atom is a molecule. He argues for his answer by referring to the kinetic theory of gases. For the physicist, the helium atom is not a molecule. He argues for his answer by affirming that he cannot see the molecular spectrum of the helium atom. (cited in Jonnaert and Vander Borght, 2003, p. 23)

The two answers, of the chemist and the physicist, are not contradictory at all. Each one constructs his own definition of the helium atom according to his reference field. If a scientist is both a chemist and a physicist, he or she may have both definitions of the helium atom at the same time.

What could we deduce about the concept of knowledge from the previous examples? Knowledge is not out there, external to the individual and waiting to be acquired. It is neither wholly preformed within the individual and ready to emerge as the individual develops. Instead, knowledge is invented and reinvented as the individual develops and interacts with the environment surrounding him or her. Those assumptions about knowledge are consistent with Piaget's views (Driscoll, 2000). In addition, Piaget believed that the individual organizes knowledge as *cognitive structures* or schemata and that, when confronting new information, the individual could use his or her prior cognitive structures and his or her cognitive ability to yield a new set of cognitive structures or new knowledge. That is cognitive or intelligence development.

According to Bourgeois and Nizet (1999) and Santrock (2001), there are two main types of knowledge: *declarative knowledge* and *procedural knowledge*. The first one is "the conscious recollection of information, such as specific facts or events that can be verbally communicated" (Santrock, 2001, p. 282), for example the assumptions about constructivism. The second one is "[cognitive structures] in the form of skills and cognitive operations about how to do something" (Santrock, 2001, p. 282), for instance, the way to teach students in a manner consistent with constructivist assumptions.

Key concepts:

Cognitive structure is a concept or framework that exists in an individual's mind to organize and interpret information (Santrock, 2001, p. 49).

Knowledge is cognitive structures an individual constructs about the new information on the basis of his or her own experiences and the interaction with the environment surrounding him or her.

1.3 Constructivism and learning

If people invent and reinvent knowledge through their own experiences and interactions with the environment, what happens exactly "in the mind" of an individual when he or she invents new knowledge or learns something? I begin with examining an example. Then, on the basis of the views of Piagetian theorists, I give my own definition of learning.

Here is an example of constructivist learning proposed by Bruner (1973) in his approach about discovery learning:

> The concept of prime numbers appears to be more readily grasped when the child, through construction, discovers that certain handfuls of beans cannot be laid out in completed rows and columns. Such quantities have either to be laid out in a single file or in an incomplete row-column design in which there is always one extra or one too few to fill the pattern. These patterns, the child learns, happen to be called prime. It is easy for the child to go from this step to the recognition that a multiple table, so called, is a record sheet of quantities in completed multiple rows and columns. Here is factoring, multiplication and primes in a construction that can be visualized (cited in Kearsley, 2003).

To give more explanation for the previous example, I show here two figures. Figure 1.2 shows that it is easy for the child to lay out 12 beans in completed rows and columns whereas it is impossible for the child to lay out 11 beans in such row-column designs except for a single file (Figure 1.3). Notice that the teacher gives the child a small number of beans so that it can try every possibility of laying out the beans in row-column designs.

Now let me look further into the learning process in the previous example from a Piagetian-constructivist point of view. Suppose that the child possesses a set of cognitive structures (prior knowledge) in its memory, for instance, factoring and multiplication. When the child confronts the previous situation, its learning process may be articulated around the following two inseparable mechanisms:

1. *Assimilation*. The child activates a certain number of cognitive structures to fit the new information it confronts in the given situation into its existing knowledge. For example, when we give 12 beans to the child, it will apply its existing knowledge of multiplication and factoring to lay out the beans in completed rows and columns (Figure 1.2).
2. *Accommodation*. Sometimes, however, while treating new information, the child could find that its existing knowledge is inadequate to treat the information. In this case we say that a cognitive conflict has occurred. For instance, given 11 beans, the child could encounter an anomalous experience because it cannot lay out the beans in multiple tables whatever the number of rows or columns that is greater than one (Figure 1.3). This cognitive conflict motivates the child to find the way to overcome it. A possible way is that the child transforms its existing cognitive structures into new ones to be able to account for the incompatible information. For example, the child recognizes the difference between two groups of numbers: the number 12 belongs to the first one and the number 11 to the second one. The second group of numbers, the child learns, happens to be called prime.

Figure 1.2. Laying out 12 beans in completed rows and columns

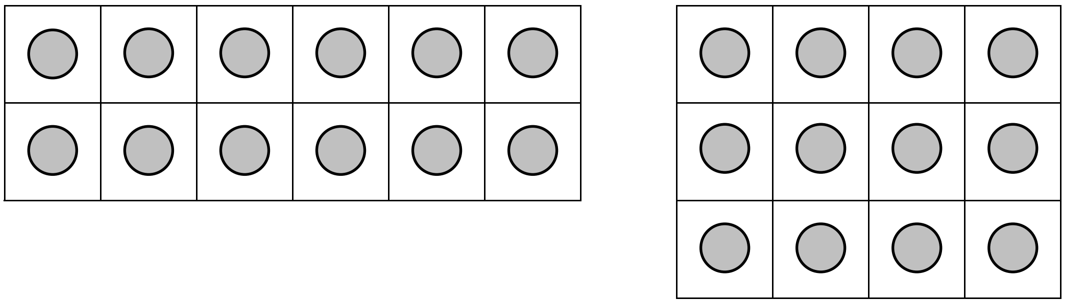
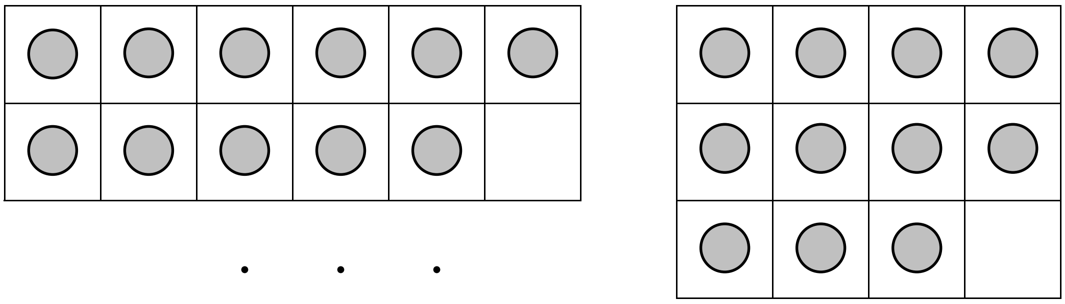


Figure 1.3. 11 beans cannot be laid out in a “multiple” table



Here is another example of assimilation and accommodation that I observed: My infant knows how to grab its favorite little rattle and thrust it into its mouth. When it comes across a new object, for instance its mother's expensive watch, it easily learns to transfer its "grab and thrust" cognitive structure to the new object. That is *assimilation*. When my infant comes across another object again, for example a beach ball or a big rattle, it will try its old cognitive structure of grab and thrust. This of course works poorly with the new object. Therefore, my infant will adjust its cognitive structure to adapt to the new object: In the example of the beach ball, "squeeze and drool" would be an appropriate title for the new cognitive structure; and in the example of the big rattle, my infant turns the rattle to find an appropriate point that it can thrust into its mouth. That is *accommodation*.

According to Piaget (1975), assimilation and accommodation exist in an inseparable relationship. An inadequate attempt to assimilate new information into existing cognitive structures may result in some adjustment of those cognitive structures (thus, accommodating the information). And vice versa, such accommodation affects subsequent assimilation, which will now proceed in accord with the new structure.

Moreover, assimilation and accommodation together constitute the master development process: Equilibration. Equilibration particularly characterizes the individual's transition from one stage of development to the next, for example, from a child who could solve concrete problems in a logical fashion to an adult who could solve abstract problems in a systematic and logical fashion. Through the trajectory of life, an individual unceasingly encounters anomalous experiences that create states of disequilibrium. The individual can only resolve a state of disequilibrium when he or she adopts a more adaptive, more sophisticated mode of thought. In this case, the individual attains a new equilibrium. That is *learning*.

Note that Piaget stated that cognitive changes occur from immediately after birth (Driscoll, 2000). According to Piaget, newborns come into the world with a number of innate reflexes, claimed to be primitive cognitive structures, for example, sucking, reacting to noises. Within a short time, they begin to modify these reflexes to make them more adaptive, for instance sucking a finger becomes a different action from sucking a nipple. More information about Piaget's theory of cognitive and knowledge development is presented in Driscoll's book (2000, chapter 6).

Key concepts:

Assimilation is the process in which individuals incorporate new knowledge into existing cognitive structures.

Accommodation is the process in which individuals adjust existing cognitive structures to account for new information.

Learning is the process in which individuals construct and transform cognitive structures.

1.4 Constructivism and conditions of learning

If learning is a process of construction and transformation of knowledge, what are conditions that facilitate and stimulate (or disadvantage) learning? Let me give a simple example: In the previous example proposed by Bruner, the conditions of learning proposed for the child stimulate constructivist learning, because they evoke a cognitive conflict in its mind. If the child, however, is given a textual definition to learn the concept of prime numbers, this condition of learning could lead to "rote" or passive learning rather than constructivist learning.

Although there is no set of teaching practices that constitutes a Piagetian approach to instruction, many educational theorists have suggested broad constructivist pedagogical principles consistent with Piaget's development theory. Driscoll (2000) examined a variety of perspectives on constructivism and identified five important facets of constructivism related to instructional design:

1. *Reasoning, critical thinking, and problem solving*. Regarding this facet, The Cognition and Technology Group at Vanderbilt (1991a) named “the ability [of the learner] to write persuasive essays, engage in informal reasoning, explain how data relate to theory in scientific investigations, and formulate and solve moderately complex problems that require mathematical reasoning” (p. 34).
2. *Retention, understanding, and use*. Perkins (1991a) stated: “The basic goals of education are deceptively simple. To mention three, education strives for the retention, understanding, and active use of knowledge and skills” (p.18). Regarding this facet, the author means the ability of the student to actively apply the new knowledge in various situations, particularly in interactions with other people, in order to reinforce his or her retention and understanding of the new knowledge.
3. *Cognitive flexibility*. Spiro and associates (1991) declared the need for learners to actively use cognitive flexibility, meaning the ability to modify one’s cognitive structures, in many ways, to be able to adapt to a variety of new situations.
4. *Self-regulation*. Self-regulation is the ability of learners to identify and pursue their own learning goals (Driscoll, 2000).
5. *Mindful reflection and epistemic flexibility*. Culler (1990) spoke of the need to foster post-structuralist thinking, a kind of reflective criticism or mindful reflection. Cunningham (1987, 1992) defined reflexivity as “the ability of students to be aware of their own role in the knowledge construction process”. Morrison and Collins (1996) spoke of epistemic fluency meaning the ability to identify and use different ways of knowing.

Driscoll also identified five main constructivist learning conditions corresponding to the previous five facets, respectively:

1. Provide learners with complex and relevant learning environments.
2. Engage learners in social negotiation.
3. Provide learners with multiple modes of learning and multiple perspectives on learning.
4. Encourage the ownership of learners in learning.
5. Make learners be aware of their own role in the knowledge construction process.

I follow the distinction of the five facets proposed by Driscoll because it appears to embody different points of view proposed by constructivist researchers. In the following paragraphs, I give an overview of constructivist learning conditions, according to Driscoll's five facets. The difference between Driscoll’s synthesis and mine is that Driscoll presents both constructivist and social constructivist pedagogical principles whereas I outline only the first ones.

It is worth noting that the distinctions of Driscoll's five facets of constructivism are obviously not clear-cut. For example, we can find several principles about multiple perspectives or social interactions both in the facet about retention, understanding, and use and in the facet about cognitive flexibility.

1.4.1 Complex, realistic, and relevant learning environments

The main learning conditions fostering reasoning, critical thinking, and problem solving are complex, realistic, and relevant learning environments. The Cognition and Technology Group at Vanderbilt (1991a) stated that students cannot be expected to learn to think critically and solve the complex problems they will face in real life unless they have the opportunity to do so. Both Jonassen (1999) and Spiro and his colleagues (1991) argued that, for problem-solving skills to be maximally facilitated, learners must cope with very complex situations.

Regarding this facet, Bourgeois and Nizet (1999), Frenay and Bédard (2004), and Frenay (1994) also argued for relevant learning environments. They advocated that learners should examine appropriate learning situations to be able to identify the essential characteristics of the new knowledge (they named this the comprehension process). On the other hand, they added, learners should also use and apply the new knowledge in a variety of concrete situations (e.g., solving problems) that are different from the ones in the comprehension process (they named this the exploration process). In addition, according to Bourgeois and Nizet (1999), it is by tests, errors, and reasoning in the exploration process that learners gradually forward to the acquisition of the new knowledge. Thus, we must also provide learners with appropriate assessment devices.

Complex learning situations include the use of "construction kits" such as Legos or software such as *Geometric Supposer* (Wilson, 1996). Construction kits allow learners to assemble "not just things, such as TinkerToys, but more abstract entities, such as commands in a program language, creatures in a simulated ecology, or equations in an environment supporting mathematical manipulations" (Perkins, 1991a, p. 19). Another kind of complex learning environments is computer-based micro-worlds that emphasize the instructional nature of simulations (Wilson, 1996). *SimCity* (2004), for example, is a simulation of real-world cities that allows learners to explore what it means to build and manage a variety of aspects of city life.

1.4.2 Social interactions in groups

It is important to note that although I am saying "social negotiation", it does not mean that what I say belongs to social constructivist approaches. In section 1.1, I explained that cognitive constructivism takes into account social negotiation, but with less emphasis than cognitive development.

A number of constructivist authors have argued for social negotiation as the critical condition of learning to stimulate the retention, understanding, and active use of knowledge by students. For instance, Cunningham and associates (1993), Knuth and Cunningham (1993) declared that intellectual development is significantly influenced through social interactions;

therefore, learning should reflect, more or less, collaboration between both tutors and learners, and learners and learners.

Bourgeois and Nizet (1999) also advocated the importance of groups in education. Firstly, learning in groups encourages learners to actively express their personal points of view without being afraid of errors. Secondly, learning in groups multiplies feedback on the work and understanding of each member. And finally, each individual could learn so much from observing the work and errors of peers.

Recently, a new genre of research and application has emerged as computer-supported collaborative learning (Koschmann, 1996). For example, the Collaborative Visualization project (Pea, 1993) has been designed to connect learners across classrooms and outside of classrooms.

1.4.3 Multiple modes of learning and multiple perspectives

In chapter 2, I will look further into the concept of cognitive flexibility as well as into learning conditions fostering cognitive flexibility. In this sub-section, I outline only a certain number of general principles underlying cognitive flexibility.

Researchers who adhere to cognitive flexibility have concentrated on the use of multiple modes of learning and on the confrontation of multiple perspectives (Driscoll, 2000). For instance, using multiple modes of representation can serve as a means of juxtapositions, that is, viewing the same content through different modes such as visual and auditory. Juxtaposition thus enables different aspects of the content to be seen. On the other hand, there are typically multiple ways to think about and solve problems. So, learners must engage in activities that allow them to evaluate alternative solutions to problems as a means of testing and enriching their own understanding (Cunningham et al., 1993; Knuth & Cunningham, 1993).

Regarding cognitive flexibility, Bourgeois and Nizet (1999) described the need of the "reversibility of the thought" for learners. According to this approach, the teacher should be responsible for the following three activities: (a) engage learners in expressing their personal points of view, (b) organize the confrontation of learners' points of view, and (c) provide methodological tools allowing learners to treat these different points of view.

Several constructivists (e.g., Cunningham, 1992) accept that hypermedia can be effectively used to stimulate learners to think about ideas, theories, literary work, and so forth, from a variety of points of view. For instance, in the Lab Design Project (Honebein et al., 1992, 1993), graduate students could investigate the sociology of a building by using a rich hypermedia database to explore different aspects of the building.

1.4.4 Ownership in learning

The principal condition of learning to bring self-regulation by students about is ownership in learning. Perkins (1991b) advocated: "Students are not likely to become autonomous thinkers and learners if they lack an opportunity to manage their own learning" (p. 20). Learners are not passive recipients of instruction that the teacher has designed for them. Instead, they play an active role in identifying what their own learning needs are and how those needs can best be satisfied (Cunningham et al., 1993; Hannafin, 1992). The teacher, however, should not leave learners alone in their own management of learning tasks. The teacher acts as a facilitator who helps learners frame their learning objectives in meaningful contexts (Cunningham et al., 1993; Knuth and Cunningham, 1993).

To give an example, here is a report of a project with elementary school students:

> In Harel and Papert's [1992] work, elementary school students who displayed a great dislike for fractions tackled the task of learning about fractions with great enthusiasm when their role was changed from students to software designers. They were asked to design a computer program in LOGO (software they were already familiar with) that would teach the basics of fractions to children one year younger than themselves. In order to do this, they first had to teach themselves what was important to know about fractions. When the project was complete, the students had learned not only about fractions, but also about software design and instructional design. (Honebein et al., 1993, p. 9)

1.4.5 Self-awareness of knowledge construction

According to Driscoll (2000), the important condition of learning to bring mindful reflection and epistemic flexibility about is to engage learners in being aware of the knowledge construction process (she named this condition "self-awareness of knowledge construction"). Examples include providing multiple modes of learning, confronting multiple perspectives, and encouraging ownership in learning (Driscoll, 2000).

Bourgeois and Nizet (1999) also argued for the confrontation of a variety of concurrent points of view. But they strongly stressed the need of "personal thought" that helps learners investigate on and give their own opinion to those different points of view, like a social actor. The teacher, therefore, should encourage learners' personal thought by presenting, as objectively as possible, a diversity of points of view, not only his or her own ones but also those of other people.

Here is an example presented in a book about learning psychology and instruction:

> Consider, for example, the different views of learning that are presented in this book. What do they each imply about your own learning of their assumptions and knowledge? […] from a constructivist point of view, you might be expected to recognize that all these theories are constructed to make sense of the phenomenon of learning. Their different assumptions lead to different pictures of learning, and consequently, of instruction. From discussion with your classmates and others, you might develop a personal view as to what theory (or theories) is the most right or useful. Or you may reject the assumptions upon which all these theories have been built in order to pose a new set of assumptions and explore a potentially new theory of learning. (Driscoll, 2000, p. 390)

Key concept: Constructivism has many facets related to instructional design.

1.5 Discussion

What can be concluded from various points of view on constructivist learning conditions presented in section 1.4, what is the problem of constructivism related to instructional design, and what is my position in those variations of perspectives? The following paragraphs should answer these questions.

1.5.1 Definition of a constructivist learning environment

From the five main conditions of learning presented in section 1.4, I conclude that the course designer should examine constructivist learning conditions in four key components of learning environments (hereafter I use the term *learning materials* to denote these four components together):

1. *Learning contents*. These are any source of information provided for learners for exploring their learning objectives, for example, concept introductions, examples, exercises, and case studies. Other names for this component include content objects, information banks, and so on.
2. *Pedagogical devices*. Pedagogical devices include methods and tools provided for learners for exploring learning contents, for instance, a reference book, a Web platform in which we can deliver various kinds of information such as text, images. Cognitive tools and construction kits are also different types of pedagogical devices.
3. *Human interactions*. These include means and techniques for engaging the tutor and learners in exchanges, for example, meeting rooms, mailing lists, forums, chat rooms.
4. *Assessment*. Assessment or assessment devices are problems, methods, and tools for determining whether learners have achieved the learning objectives, for instance posttests.

In the following textbox, I give my definition of constructivist learning environments. Of course, this is a general definition and it may have no universal acceptance among constructivist authors (see other definitions in Wilson, 1996). I look further into this concept in the next chapters.

> **Key concept:** A *constructivist learning environment* is a place where learners may use a variety of information resources, pedagogical and assessment devices, and interact with the tutor and peers through communication means in their guided pursuit of learning objectives, according to constructivist principles.

1.5.2 Need of an operational approach

I believe that the indications suggested by constructivist theorists in section 1.4 are too general for teachers to be able to imagine concrete steps when they want to design or evaluate constructivist learning environments in their own instructional situations. For example, given an ICT-based learning environment, it is difficult for a teacher to determine whether the learning contents delivered in this environment actually reflect constructivist principles. Or when designing a new learning system, the course designer does not know whether the system finally facilitates and stimulates constructivist learning. Both Driscoll (2000) and Jonassen and Rohrer-Murphy (1999) also claimed this problem of constructivism related to instructional design. Therefore, I argue for the need of proposing *operational* criteria for constructivism (stressing on the qualifier "operational"). I believe that operational criteria could provide a useful framework both for designing and evaluating constructivist learning systems.

Among the five facets of constructivism identified in section 1.4, cognitive flexibility is a pedagogical principle that is often mentioned by constructivist authors (e.g., Bourgeois & Nizet, 1999; Driscoll, 2000; Spiro & Jehng, 1990). The pedagogical principles underlying cognitive flexibility reflect the basic characteristics of constructivism (Spiro et al., 1988, 1990, 1991). Moreover, a significant number of examples have showed that ICT may facilitate the implementation of learning conditions fostering cognitive flexibility (Driscoll, 2000; Spiro & Jehng, 1990; Wilson, 1996). Therefore, the main point of the thesis is to exploit the constructivist facet about cognitive flexibility. In chapter 2, I propose a set of operational criteria for cognitive flexibility. And in the following chapters, I use this set of criteria as a framework for validating various issues such as instructional design, evaluation of learning situations and systems.

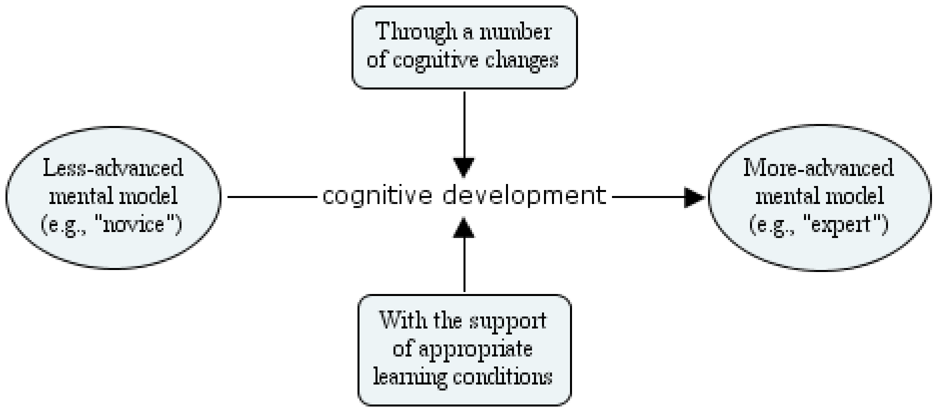
Key concept: An *operational criterion* for cognitive flexibility is a test that allows a straightforward decision about whether or not a learning situation reflects the pedagogical principles that are underlying cognitive flexibility.

1.5.3 Characteristics of the learner

In the Piagetian-constructivist point of view presented earlier, each learner possesses a *mental model* (i.e., a mental representation or knowledge structure) about a concept or a situation at any point in time. The purpose of learning is to have the mental model get closer and closer to that subsumed by the learning objectives. Through personal experience, the learner may undergo a certain number of cognitive changes and then possess a different mental model (Sasse, 1991). Students' mental models are one of essential characteristics that affect the learning of students (Sasse, 1991).

To illustrate an example of mental models and cognitive development, I construct Figure 1.4. In this example, a beginner could start with a "novice" model on a given subject, for instance a child with no knowledge of the concept of prime numbers. Through a number of interactions with the appropriate conditions of learning provided by the teacher, for instance laying out beans in row-column design (see also section 1.3), the learner is expected to undergo several cognitive changes and gradually evolve an "expert", for instance after constructing tables of beans, the child could discover that certain handfuls of beans cannot be laid out in a multiple table, and therefore grasp the concept of prime numbers (see also section 1.3).

Figure 1.4. An example of mental models and cognitive development


Through a number of cognitive changes
Less-advanced mental model (e.g., "novice")
cognitive development
More-advanced mental model (e.g., "expert")
With the support of appropriate learning conditions

There are other characteristics of the learner than mental models that the course designer should also take into account for designing constructivist learning environments, for example, the motivation for learning, the social attitude and behavior when working in groups (Bourgeois & Nizet, 1999; Driscoll, 2000; Santrock, 2001). In this thesis, however, I concentrate on studying the role of mental models in instructional design, that is the way to devise learning situations taking into account learners' mental models. In chapters 6 and 7, I discuss this issue further.

Key concept: A *mental model* is a conceptual structure of declarative knowledge or procedural knowledge or both of them a person holds of a concept or a device or a system.

1.5.4 Adaptation support

If we assume the learner's cognitive development presented previously, so one of major roles of the designer of a "course" is to provide the learner with appropriate learning conditions (Figure 1.4) so that the learner's process of knowledge construction and transformation is facilitated (Bourgeois & Nizet, 1999).

A particular type of adaptation support that is often mentioned by constructivist theorists is *scaffolding*, meaning that the teacher should change the level of support over the course of a learning session of a particular learner (Santrock, 2001). In other words, a more skilled person (teacher or more-advanced peer) adjusts the amount of guidance to adapt the learner's current performance level. For instance, when the learner confronts a new situation, the tutor might guide and encourage the learner in the learning process; as the learner's competence increases, the support is adjusted or removed (Soderman, Gregory, & O'Neill, 1999). The present thesis takes into account this particular kind of adaptation.

What conditions of learning could be adaptive to the needs of the individual learner and how to perform adaptation support? According to Brusilovsky (1999), Murray (1999), and Stoyanov and Kirschner (2004), there are four main adaptation techniques concerning the four learning components identified in section 1.5.1, respectively, and one technique concerning problem-solving support, as follows:

1. *Adaptive presentation of learning contents*. The course designer should define which learning contents are appropriate to a specific learner at any given time, for example simpler situations and examples for a "novice" learner than for an "expert" one.
2. *Adaptive use of pedagogical devices*. The course designer should define which learning activities are appropriate to a specific learner, for instance simpler tasks to a "novice" learner than to an "expert" one.
3. *Adaptive communication support*. The course designer should identify which peers are appropriate to help a specific learner, for example learners with more-advanced mental models help learners with less-advanced ones.
4. *Adaptive assessment*. The course designer should identify which assessment problems and methods are appropriate to determine the actual performance of a specific learner, for instance simpler tests for a "novice" learner than for an "expert" one.
5. *Adaptive problem-solving support*. The tutor should give appropriate feedback during the problem-solving process of a specific learner, for example to show the learner his or her own difficulties and provide him or her with the way to overcome those difficulties.

I look further into the issue of adaptation support in chapters 4, 5, 6, and 7.

Key concept: *Scaffolding* is a technique of changing the level of support over the course of a learning session of a particular student.

1.6 Conclusion

In this chapter, I present some background on constructivism. This chapter does not contribute anything new. Rather, it clarifies a certain number of important educational definitions to which I adhere. These definitions provide the reader with the key for understanding my contributions presented in the next chapters.

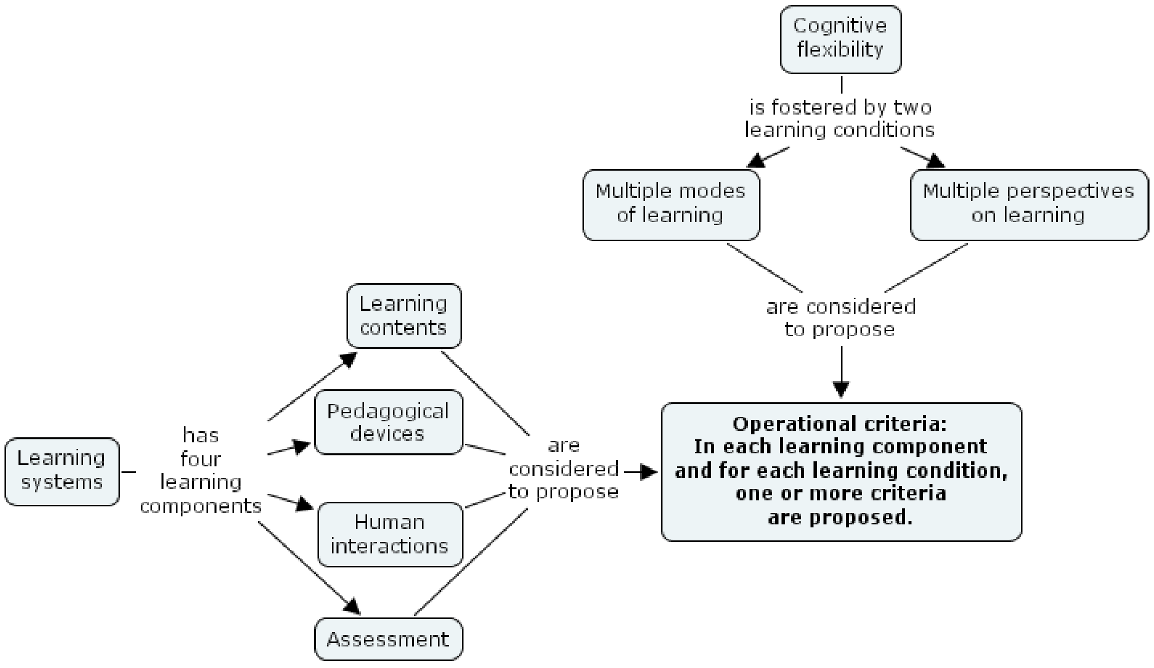
CHAPTER 2

Operational criteria for cognitive flexibility

"*Everything should be made as simple as possible, but not simpler.*"

Albert Einstein, German Scientist, 1879 – 1955 (cited in Suomela, 2005)

Cognitive flexibility is a pedagogical principle that is often mentioned among the basic characteristics of constructivism. This chapter proposes operational criteria for cognitive flexibility and presents both justifications and examples of their use. In this chapter, I argue that the set of operational criteria I have proposed make the process of instructional design more straightforward than do indications suggested by educational theorists. After reading this chapter, one *should* be able to propose and use one's own operational criteria for *any* pedagogical principle.


Cognitive flexibility
is fostered by two learning conditions
Multiple modes of learning
Multiple perspectives on learning
are considered to propose
Learning systems
has four learning components
Learning contents
Pedagogical devices
Human interactions
Assessment
are considered to propose
Operational criteria: In each learning component and for each learning condition, one or more criteria are proposed.

Summary

2.1 Introduction

Although many descriptions and pedagogical implications for constructivism exist, there have been few *operational* interpretations of what constructivist learning is and of what constructivist pedagogical principles entail (Driscoll, 2000; Jonassen & Rohrer-Murphy, 1999). To facilitate the process of instructional design exhibiting the pedagogical principles underlying cognitive flexibility, an important facet of constructivism, in this chapter I propose a set of *operational criteria*. I first clarify the concept of cognitive flexibility and the conditions of learning that foster cognitive flexibility. Then I propose operational criteria for cognitive flexibility by examining each of those learning conditions in each of the four components of constructivist learning environments identified in section 1.5.1. I support my claim that the set of criteria makes instructional design straightforward by showing how to apply it to the design of learning situations in the problem area presented next.

Context for the examples: Java variable situation

In an introductory course on object-oriented programming and Java at the *Université catholique de Louvain* (UCL), I observed a particular kind of frequent misunderstanding. The students were exposed to the principles underlying variables and value assignment; they got to read several examples. Then they took a test in which they faced the following code segment:

```
int a, b;  /* 1 */
a = 3;     /* 2 */
b = a;     /* 3 */
a = 5;     /* 4 */
```

About 20 percent of students seemed to believe that the value of variable `b` was "automatically" changed to `5` after the value of variable `a` had been changed in the fourth instruction. Several researchers in computing science (e.g., Bayman & Mayer, 1983; Du Boulay, 1986) also drew similar conclusions. I shall show in this chapter how one could devise new learning situations in a manner consistent with cognitive flexibility so that students will overcome these misconceptions.

I also summarize a part of Jonnaert and Vander Borght's work (2003) that clarified what the concept of constructivist learning in the school context entails in a set of criteria. I claim that these criteria for the concept of learning and my criteria for conditions of learning are *complementary*.

2.2 Cognitive flexibility

According to Spiro and Jehng (Spiro & Jehng, 1990), cognitive flexibility is "the ability to spontaneously restructure one's knowledge, in many ways, in adaptive response to radically changing situational demands" (p. 165).

To clarify this concept, I give a simple example (Figure 2.1) in which I answer the following three questions: (a) what the child's prior knowledge is, (b) what new situations are, and (c) how the child restructures its knowledge to respond to the new situations adaptively.

Figure 2.1. Deriving the meanings of the same word in different contexts

How does a child develop the ability to derive the meanings of the same word in different situations?

Sentence 1:

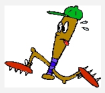
<u>Bats</u> fly at night and feed on fruit and insects.

Sentence 2:

I watched <u>the bat</u> flitting through the trees.

Sentence 3:

He gripped <u>the bat</u> tightly as he waited for the pitch.



Sentence 4:

I hope I can <u>bat</u> a home run.

The child, through personal experience and interactions with peers, parents, and teachers, acquired *prior knowledge* of the structure of simple sentences such as the role of nouns and verbs and a rich vocabulary including the meaning of the word "bat", as an animal (Figure 2.1: sentence 1). Now, given a certain number of *new situations* (Figure 2.1: sentences 2, 3, 4), the child is expected to structure or restructure its prior knowledge, as follows:

- In sentence 2, as in sentence 1, it considers the word "bat" as a noun (on the basis of the article "the"), then as an animal (on the basis of the action "flitting through the trees"), then as the actual meaning of this word (on the basis of its prior knowledge of the meaning of this word). If this situation occurs, the teacher would say that the child learns nothing because no cognitive structure is modified.
- In sentence 3, the child tries to apply the same process as in sentence 2 to derive the meaning of the word "bat" but it cannot arrive at the actual meaning of this word (we would not grip an animal tightly while waiting for the pitch). This cognitive conflict

makes the child *restructure* its own knowledge to *adapt* to this situation: It considers the word "bat" as a noun (as in sentence 2), then as a tool (on the basis of the action "gripped … tightly" that it learned in other sentences in the pass), then as the actual meaning of this noun (e.g., on the basis of the attached picture or interactions with peers). If this situation occurs, the teacher would say that the child learns a new meaning of the word "bat".

- In sentence 4, its cognitive process is similar to the one in sentence 3 but in a *new way* to *restructure* its *prior knowledge* to *adapt* to the *new situation*: It considers the word "bat" as an action verb (on the basis of the auxiliary verb "can"), then the actual meaning of this verb (on the basis of the attached picture, for instance).

In a Piagetian point of view, cognitive flexibility, as learning, is the ability that newborns already have when they come into the world. When an infant is born, it possesses a variety of innate reflexes, for instance, sucking, reacting to noises, focusing on objects within their view. Within a short time, it begins to modify these reflexes to adapt to the new environment surrounding it, for example, sucking a finger becomes a different action from sucking a nipple (see also other examples in section 1.3). As the child develops, its ability to exhibit cognitive flexibility gradually matures (Driscoll, 2000).

What are examples of students' cognitive flexibility behavior? On the basis of indications suggested by constructivist researchers (e.g., Bourgeois & Nizet, 1999; Spiro & Jehng, 1990), I have been able to deduce several instances, as follow:

- When students are faced with a new problem, they try to analyze different aspects of the problem in a systematic manner and to use different ways they have successfully used in the past to solve similar or related problems in order to find a solution, which is as complete as possible.
- When students are confronted with a new concept, they try to perform different activities in different contexts to look further into various aspects of the new concept.
- When students discuss with peers, they try to listen and ask, in a systematic manner, questions such as "Why?", "What is your source of information?" in an effort to understand other points of view.

In chapter 9, I also provide a certain number of examples I observed while working with actual students.

The individual's cognitive flexibility is there, but in instruction we need to provide *explicitly* and *systematically* learning conditions that facilitate and stimulate students' cognitive flexibility, especially in complex and ill-structured domains, that is, the domains in which cases or examples are diverse, irregular, and complex (Spiro & Jehng, 1990; Spiro et al., 1991; Feltovich et al., 1996). The next section describes those conditions of learning.

Key concept: *Cognitive flexibility* is the ability to structure or restructure one's prior knowledge, in many ways, to adapt to a diversity of new situations.

2.3 Learning conditions for cognitive flexibility

Advanced learning in ill-structured and complex domains such as biomedicine and literature gives rise to a difficult problem: What one has to do to attain a *deep understanding* of a complex concept (Spiro & Jehng, 1990). Deep understanding means that students are prepared to be ready to apply conceptual knowledge in a domain where the phenomena occur in irregular patterns, and to use knowledge in a great variety of ways that may be required in a rich domain.

Spiro and colleagues have shown in a number of studies that when students attempt to apply, to ill-structured domains, the strategies they have used effectively for understanding well-structured domains (e.g., in introductory learning), they make errors of oversimplification, overgeneralization, and "overreliance" on context-independent representations (Spiro et al., 1988). In the biomedical domain, for example, students who use only organicist metaphors or only the metaphor of the machine to help them understand how the body functions tend to analyze cases only partially. The point Spiro makes is that neither metaphor captures all aspects of body functions, although neither metaphor is wrong.

Therefore, in attempting to solve the problem of instruction in ill-structured domains, Spiro and associates have presented a new Cognitive Flexibility Theory in which they have advocated the use of multiples forms of pedagogical models, multiple metaphors and analogies, and multiple interpretations of the same information (Feltovich et al., 1996). The central metaphor of Cognitive Flexibility Theory is "learning in criss-crossed landscape": “Revisiting the same material, at different times, in rearranged contexts, for different purposes, and from different conceptual perspectives is essential for attaining the goals of advanced knowledge acquisition” (Spiro et al., 1991, p. 28). The authors have argued that by criss-crossing a conceptual landscape in many directions, knowledge that will have to be used in many ways is acquired in many ways. If taught in this manner, medical students, for instance, would be able to examine a single case from many different vantage points and see firsthand the effect of reinterpreting a particular symptom. Examining multiple cases in different contexts will help students build new cognitive structures in order to account for new cases. The example presented in section 2.2 shows that children learn the language in a very complex landscape of interrelated words: They understand a new sentence because they have examined a great variety of sentences that help them derive the meaning of the new sentence (each sentence in the past may help accounting partly for the new one). Or in chapter 1, one should examine multiple facets of constructivism to be able to fully understand, for example, the role of social negotiation in constructivist learning.

Another point of view proposed by educational theorists (e.g., Bourgeois & Nizet, 1999; Frenay & Bédard, 2004; Frenay, 1994) about cognitive flexibility in adult education stresses that teachers should encourage learners to explore new knowledge in various concrete situations, more or less different from the ones with which learners have been familiar. Those authors claim that this operation is important for knowledge transfer because it provides the chances for learning reinforcement (i.e., prior knowledge helps accounting for new knowledge). On the other hand, Bourgeois and Nizet add that, it is necessary to give means allowing learners to analyze and evaluate the new knowledge "from the outside". According to this approach, teachers are responsible for the following three activities: (a) engage learners in expressing their personal points of view, (b) organize the confrontation of learners' points of view, and (c) provide methodological tools allowing learners to treat different points of view. The point Bourgeois and Nizet make is that learners are confronted not with only one alternative point of view on a given object but with a diversity of points of view, and that learners are systematically encouraged to "come in" and "come out" different points of view with which they are confronted, and to connect those points of views one to another.

Driscoll (2000) examined the assumptions proposed by Spiro and colleagues, and identified two principal conditions of learning for cognitive flexibility: (a) *multiple modes of learning* (i.e., multiple representations of contents, multiple ways and methods for exploring contents), and (b) *multiple perspectives on learning* (i.e., expression, confrontation, and treatment of multiple points of view).

From the different points of view on cognitive flexibility presented earlier, I make two claims. Firstly, one needs to explicitly foster students' cognitive flexibility, particularly in instruction of ill-structured domains. I believe that most domains are complex and ill-structured if we anchor instruction in complex and realistic situations. For example, the application of algorithms to the problems in a word-processing program in mathematics, the learning of the concept of recursion described in chapter 3 in computing science. Secondly, I believe that the indications suggested by researchers in pedagogy are still too general for the course designer to be able to imagine concrete steps when he or she wants to design learning systems leading to cognitive flexibility. For instance, what has to be done with the learning contents, pedagogical devices, human interactions, and assessment in the Java variable problem presented earlier? That is why I propose *operational criteria* for cognitive flexibility. Here, I define an operational criterion to be a test that allows a straightforward decision about whether or not a learning situation reflects the pedagogical principles underlying cognitive flexibility.

I follow the two learning conditions for cognitive flexibility proposed by Driscoll because they appear to embody different points of view proposed by other educational theorists. In the next section, I transform the pedagogical principles underlying these learning conditions into operational criteria, so that one can design learning situations exhibiting the desired characteristics of cognitive flexibility. I present both the way I use for this transformation process and its application.

Key concept: The two main conditions of learning for cognitive flexibility are *multiple modes of learning* and *multiple perspectives on learning*.

2.4 Operational criteria for cognitive flexibility

I start by considering two learning conditions for cognitive flexibility: *multiple modes of learning* and *multiple perspectives on learning*. I also consider four main components of learning systems identified in section 1.5.1: learning contents, pedagogical devices, human interactions, and assessment. In each of the four components of learning systems and for each of the two learning conditions for cognitive flexibility, I propose criteria that can be applied for checking the presence of the learning condition in the learning component. In Table 2.1, each cell contains one criterion except for the fourth cell, which contains three criteria because of the complex pedagogical principles underlying the learning conditions in this cell (see more explanations in section 2.4.2).

Table 2.1. Operational criteria for cognitive flexibility

	Learning conditions	
Learning components	**Multiple modes of learning**	**Multiple perspectives on learning**
Learning contents	(1) MM1	(2) MP1
Pedagogical devices	(3) MM2	(4) MP2, MP3, MP4
Human interactions	(5) MM3	(6) MP5
Assessment	(7) MM4	(8) MP6

In what follows, I present the operational criteria from left to right and from top to bottom of Table 2.1; I justify each criterion and I apply each criterion for designing learning situations for the Java variable situation. I discuss the set of criteria in section 2.4.5.

2.4.1 Operational criteria for learning contents

This sub-section presents two criteria proposed for learning contents: MM1 for multiple modes and MP1 for multiple perspectives.

MM1: *The same learning content presenting concepts and their relationships is represented in different forms (e.g., text, images, audio, video, simulations)*.

I believe that a single representation of content presents only part of the characteristics of a new concept. So, the teacher should make multiple representations available to help the learner better grasp diverse aspects of the new concept. The teacher, however, should not present the learner with too many kinds of information (e.g., text and images and audio and video) at the same time, because this type of presentation (cognitive overload) may distract the learner from perceiving the new concept (Kirsh, 2000; Sweller, 2005).

In the Java variable problem, I shall use texts, images, and simulations for most presentations. Figure 2.2 shows an introduction about variables and assignment. The box metaphor used for variables may help learners master the basics of the concept of variable in Java such that a variable is used to hold a data value. In addition, a simulation showing what the computer does when we declare and initialize a variable is useful for dynamically illustrating the relationship between the concept of variable and memory locations in the computer.

MP1: *The same abstract concept is explained, used, and applied systematically with other concepts in a diversity of examples of use, exercises, and case studies in complex, realistic, and relevant situations*.

A small number of examples, exercises, and case studies cannot illustrate every different interpretation of a new concept and its relationships with other ones. Thus, the teacher should make multiple examples, exercises, and case studies available to help learners better understand and apply *multiple interpretations* of the new concept in different contexts. It should be noted here that, at the end of each chapter of many textbooks, the authors present many exercises in different contexts, but these exercises often illustrate the same interpretation of the new concept.

Figure 2.2. Introduction to variables

A *variable* is a name for a location in memory used to hold a data value …

Example 1 (text + image):		**Example 2** (text + image):		
int a;	//a = $\boxed{?}$	int a, b;	//a = $\boxed{?}$	b = $\boxed{?}$
a = 2;	//a = $\boxed{2}$	a = 3; b = a;	//a = $\boxed{3}$	b = $\boxed{3}$
		a = 5;	//a = $\boxed{5}$	b = $\boxed{3}$

Figure 2.2 shows the use of a box analogy for illustrating the variable concept in Java. Note that this analogy may make learners believe wrongly that a single variable can hold more than one value at a given time (since a physical box may contain more than one object), or that a variable is always initialized to zero (an empty box contains nothing, and zero is nothing). To alleviate these misconceptions, I prepared the librarian situation (Figure 2.3). This situation shows that a variable holds only one value at a time and that a variable has no value at the moment of declaration if we do not explicitly initialize it. Indeed, initializing the variable "`totalPrice`" to zero is necessary for the computation of the total, while the variable "`bookPrice`" is not necessarily initialized because it is only used when the librarian purchases a new book. When a new book is purchased, the variable "`bookPrice`" is used to hold only the price of the new book (but not the price of any other book), and the variable "`totalPrice`" is used to add up the total price of all purchases until now.

Figure 2.3. Librarian situation

A librarian purchases new books for her department. She doesn't stop buying until the total price of all purchases is over 1900€. Write a segment of code to perform this process.

Possible solution:

```
int totalPrice = 0;  //totalPrice = $\boxed{0}$
int bookPrice;       //bookPrice = $\boxed{?}$
while (totalPrice <= 1900) {
        System.out.println ("Enter the new book price:");
        // class Keyboard allows input operations
        bookPrice = Keyboard.readInt();    // bookPrice = $\boxed{56}$  | bookPrice = $\boxed{31}$ | …
        totalPrice += bookPrice;           // totalPrice = $\boxed{56}$ | totalPrice = $\boxed{87}$ | …
}
System.out.println("Total purchase price: " + totalPrice);
```

The librarian situation, however, does not show that assigning one variable to another means that the first variable holds the value of (but does not link to) the second one. I add a third situation (Figure 2.4), in which to approximate the income of a given year (e.g., 2004), variables "`income1`" and "`income2`" are used to hold the incomes of the previous two years (e.g., 2002 and 2003). After approximating the income in 2004, to approximate the income in 2005, the variable "`income1`" takes the value of the variable "`income2`" (i.e., the income in 2003) and the variable “`income2`” takes the value of the variable “`income`” (i.e., the income in 2004), …

Figure 2.4. IT company situation

The income of a new IT company in 2000, 2001, 2002, and 2003 is: 2, 3, 5, and 8 million euros. From 2002 onwards, the income of any given year is approximately equal to the sum of the incomes of the previous two years). Approximate the income of the company in 2010.

Possible solution:

```
int income1 = 5;   // income1 = 5
int income2 = 8;   // income2 = 8
int income = 0;    // income =  0
for (int year = 2004; year <= 2010; year++){
            income = income1 + income 2;      // income = 13   |  income = 21   |  …
            income1 = income2;                // income1 = 8   |  income1 = 13  |  …
            income2 = income;                 // income2 = 13  |  income2 = 21  |  …
}
System.out.println(“Income in 2010: ” + income);
```

These three situations also show different uses of variables and assignment (e.g., counter variables, summing with a variable), and rely on the `for` and `while` concepts. When facing the concept of variable, learners should not necessarily explore all three situations at once. They can tackle them at a different time, for example during the exploration of the `for` and `while` iterations (see also criteria MM2 and MP2 presented in the next sub-section).

Confronting the previous three situations, students are stimulated not only to read the text, but also to perform other learning activities, for instance, to test the source code, to discuss the role of each variable in the source code (see also criteria MM3 and MP5 shown in section 2.4.3).

In addition to those situations, I prepare programming exercises and the learning content for memory locations (an important concept for understanding variables). I also include different interpretations proposed by other authors in books and websites about the variable concept, for example, the reference book "Java software solutions" of Lewis and Loftus (2003), the online Java programming tutorials of Sun (2004) and OOPWeb (2004). These resources are necessary to satisfy criterion MP3 presented in the next section.

2.4.2 Operational criteria for pedagogical devices

In this sub-section I present four criteria proposed for pedagogical devices: MM2 for multiple modes and MP2, MP3, and MP4 for multiple perspectives. I separate three interdependent criteria MP2, MP3, and MP4 to facilitate their application. Indeed, the pedagogical principles underlying multiple perspectives in pedagogical devices are complex. For example, Bourgeois and Nizet (1999) and Spiro and Jehng (1990) argued for the following three suggestions (see also section 2.3): (a) learning in a criss-cross landscape, (b) learning by confronting multiple points of view, and (c) learning by producing summaries on a diversity of points of view. Therefore, if those three suggestions are merged into only one criterion, I think it may be hard for the practitioner to apply it.

MM2: *Learners are encouraged to study the same abstract concept for different purposes, at different times, by different methods including different activities (reading, exploring, discussion, knowledge reorganization, etc.).*

To satisfy criteria MM1 and MP1, the teacher should prepare the learning content taking into account multiple modes and multiple perspectives. The teacher should also provide suitable pedagogical devices allowing learners to explore this learning content in different ways and contexts. Multiple learning activities (other than mere reading) help learners better master and transfer new knowledge.

In the Java variable problem, I could deliver the learning content on a Web-based platform and engage learners in four learning activities: (a) read, (b) explore examples and cases by using hypertexts and simulations, (c) do exercises by writing and executing programs, and (d) find and discuss other interpretations about variables (i.e., in other programming languages than Java).

MP2: *When facing a new concept, learners are encouraged to explore the relationships between this concept and other ones as far as possible in complex, realistic, and relevant situations.*

Learners should learn a new concept with other ones in different meaningful contexts to understand various interpretations of the new concept and transfer new knowledge in real situations. Hence, the teacher should provide explicit tools to encourage learners to systematically explore the interrelations of concepts in relevant situations.

In the Java variable problem, when learners are led to face the variable concept, I present hypertext links in their learning environment to have them explore related concepts such as those of memory location, `while` loop, and `for` loop.

Similarly, when learners face the `while` and `for` iterations, a hypertext link to the variable concept is presented so that learners can revise the variable concept easily. The main idea here is to create a criss-cross landscape in the learners' learning hyperspace.

MP3: *When facing a new concept, learners are encouraged to explore different interpretations of this concept (by other authors and by peers), to express their personal point of view on the new concept, and to give feedback on the points of view of other people.*

In criterion MP2, the teacher encouraged learners to explore multiple interpretations proposed by the teacher about the new concept. Here, the teacher should engage them in the expression of their personal point of view and in the exploration of those of others. These learning activities may help learners understand diverse conceptions and misconceptions, and so help them overcome their own resistance to learning.

In the Java variable problem, I engage learners in three learning activities: (a) explore other interpretations of other people about the variable concept (e.g., the reference book "Java software solutions" by Lewis and Loftus 2003; the Java tutorial website of Sun, 2004; the Java tutorial website of OOPWeb, 2004); (b) find and add other examples, exercises, and case studies in their own learning hyperspace (e.g., using online search tools and the keywords “Java”, “variables”, and “assignment” to find relevant websites); and (c) explore peers’ learning hyperspaces to understand what they think and how they learn.

MP4: *When facing a new concept, learners are encouraged to examine, analyze, and synthesize a diversity of points of view on the new concept.*

In criteria MP2 and MP3, the teacher encouraged learners to explore a variety of points of view about the new concept. In this one the teacher should provide tools to stimulate them to treat these diverse points of view, so as to construct their own knowledge space about the new concept (e.g., to produce a synthesis by using text, tables, concept maps).

In the Java variable problem, I could, for instance, ask learners to produce a table stating the main points of the variable concept expressed by themselves and by peers. For each point the learner makes, the learner is asked to provide the information source used to justify the point.

2.4.3 Operational criteria for human interactions

This sub-section presents two criteria proposed for human interactions: MM3 for multiple modes and MP5 for multiple perspectives.

MM3: *The number of participants, the type of participant (learner, tutor, expert, etc.), the communication tools (e-mail, mailing lists, face to face, chat rooms, video conferencing, etc.), and the location (in the classroom, on campus, anywhere in the world, etc.) are varied.*

Multiple modes of discussion provide different learning activities, produce different learning outcomes, and help learners manage their own learning more flexibly. For example, a mailing list allows asynchronous communications, and video conferencing allows a synchronous discussion among people in different locations.

In the Java variable problem, I could organize small groups of students and provide them with meeting rooms, mailing lists, and Q&A websites, so that they can run a variety of discussions with peers and other people.

MP5: *During the discussion, learners are encouraged to diversify – as far as possible – the different points of view about the topic discussed.*

During a discussion, learners should actively express their personal points of view and stimulate those of other participants, so as to face and process effectively a variety of points of view. The teacher should therefore provide methodological tools to facilitate this process (Bourgeois & Nizet, 1999).

In the Java variable problem, I could attach an open list of general and domain-specific questions to learners’ communication tools, so that they can use them to elicit peers’ points of view. Examples of such questions are: Why do we have to initialize the variable “`totalPrice`” in the librarian situation? What happens when we change the value of a variable? Are there other explanations? Does anyone have a different opinion? What was your source of information? Why? Note that researchers in pedagogy (e.g., Wright, 1995) have suggested a variety of general questions facilitating discussions in learning (see Appendix B5).

2.4.4 Operational criteria for assessment

This sub-section shows the last two criteria proposed for assessment: MM4 for multiple modes and MP6 for multiple perspectives.

MM4: *During the learning process, learners are encouraged to use different assessment methods and tools, at different times, and in different contexts for demonstrating their ability to solve different problems*.

Multiple modes of assessment enhance multiple modes of learning and help teachers and learners understand what learners actually learn. For instance, authentic assessment help teachers and learners see how learners manage the tasks and actively use problem-solving skills in a context that approximates the real world life as closely as possible. Performance assessment enables teachers and learners to see how learners perform a task such as write an essay, conduct an experiment, carry out a project, or solve a real-world problem (Reeves & Okey, 1996).

In the Java variable situation, at different points in time, the learner could be engaged in two assessment activities. Firstly, the learner is asked to do Assessment 2.1 *individually*, for instance, after the introduction of the variable concept. The purpose of this assessment is to help learners evaluate their basic understanding of the concept of variable (e.g., a single variable cannot hold more than one value at a given time, variable assignment is totally different from mathematical equality). Secondly, the learner is asked to do a programming test (Assessment 2.2) in *small groups*, for example, after the learning of other concepts such as objects, arrays. The objective of this test is to help learners apply the concept of variable with other ones such as loops, objects, and arrays to a meaningful programming situation.

Assessment 2.1. Exchanging the value of two variables

Write a segment of code to exchange the value of two variables.

Possible solution 1:

```
int a = 3;
int b = 5;
int c = a;
a = b;
b = c;
```

Possible solution 2:

```
int a = 3;
int b = 5;
a =  a + b;
b = a - b;
a = a - b;
```

MP6: *During the problem-solving process, learners are encouraged to confront multiple ways to solve the problem and multiple possible solutions to the problem*.

I believe that when learners confront multiple solutions to a problem, they have opportunity to compare, come in and come out different solutions, so as to construct their own one. Therefore, the teacher should provide learners with problems whose nature must give rise directly to different ways to solve them and to different solutions to them.

In the Java situation, I could ask learners to confront and compare different solutions to the problem presented in each assessment. In Assessment 2.1, there are at least two possible solutions: one using an intermediate variable, and one not using an intermediate variable. In Assessment 2.2, there are also at least two possible solutions (see Appendix A): one using an instance variable "`totalCost`" to add up the total cost of all CDs in the collection until now, and one, instead of using an instance variable "`totalCost`", implementing a loop `for` in the method `toString` to sum up the total price of all CDs in the collection until now.

Assessment 2.2. Collection of compact discs

Complete a class CD, which represents a compact disc (CD) specified by its title, artist, purchase price, and number of tracks, and a class CDCollection, which represents a collection of compact discs. Write and execute a class MyTest to test the methods of the classes CDCollection and CD.

```
/* SPECIFICATION
 * Representation of a collection of compact discs.
 */
public class CDCollection {
  // Instance variables: to be completed

  // Constructor: to be completed
  public CDCollection() {}

  // Methods: to be completed
/* PRECOND
 * POSTCOND
 * Adds a CD to the collection.
 */
  public void addCD(String title, String artist, double cost, int tracks) {}

/* PRECOND
 * POSTCOND
 * Return a report describing the CD collection: the number of CDs, the total
 * cost and the average cost of all CDs.
 */
  public String toString() {}
}

/* SPECIFICATION
 * Representation of a compact disc.
 */
public class CD {
  // Instance variables
  private String title, artist;
  private double cost;
  private int tracks;

  // Constructor: to be completed
  public CD(String title, String artist, double cost, int tracks) {}

  // Methods: to be completed
  ...
/* PRECOND
 * POSTCOND
 * Return a description of this CD.
 */
  public String toString() {}
}
```

Sometimes, however, students may have difficulties to give alternative solutions to a given problem, for instance, the second solution in Assessment 2.1. In this case, I could show the solution to students and ask them to discuss, for example what the computer does when the segment of code presented in the solution is executed. The point here is that the variable concept appears to be more readily grasped when learners examine various aspects of this concept in those situations, for instance, the advantages (and disadvantages) of using an intermediate variable in Assessment 2.1, the advantages of using an instance variable in Assessment 2.2.

Main result: Ten operational criteria were proposed for cognitive flexibility. Each criterion was justified and showed with one example of application.

2.4.5 Discussion

In this sub-section, I discuss a number of points related to the pertinence and the application of the set of criteria I proposed previously.

Firstly, *the quality of criteria satisfaction is important*. The role of the set of criteria is to help the course designer frame instructional and learning situations in a manner consistent with cognitive flexibility. But the course designer's expertise in the subject of instruction is essential. For example, for criterion MM1, it is important to provide the learner with multiple representations of the learning content, but it is more important to think about which forms and which ways are appropriate to present the learner with the learning content. In the Java variable situation, the combination of text and images may be appropriate, whereas in the recursion situation (see chapter 3), the use of text, images, and simulations should be pertinent because the concept of recursion is more complex than the concept of variable. For criterion MP1, one must be an expert in programming and Java to be able to devise a diversity of meaningful instructional situations for the Java variable problem (see section 2.4.1). The point here is that preparing a *diversity* of situations (quality) that emphasize *different* aspects or interpretations of a new concept is *more important* than preparing *many* situations (quantity) that emphasize *only one or two* aspects of the new concept.

Secondly, *the quantity of criteria the course designer should satisfy may not be always critical*. Presently, there has been no evidence indicating that satisfying all of the criteria is always better than satisfying, for instance, two thirds of the set of criteria. Maybe satisfying six criteria is better than satisfying one or two criteria, but I am not sure whether satisfying 10 criteria is always better than satisfying six or seven criteria. Sometimes we may provoke cognitive overload if we stimulate the learner to perform too many cognitive activities (Kirsh, 2000; Sweller, 2005). Personally, I believe that in introductory learning, satisfying several criteria could be sufficient; in advanced learning, however, it may be necessary to satisfy most or all of the criteria. In all cases, I think the quality is more important than the quantity of criteria one satisfies. In chapter 5, I discuss this point further.

Thirdly, *a number of criteria are related or similar to each other*. Therefore, the course designer should be aware of this characteristic to be able to apply the set of criteria effectively. For instance, criteria MP3 and MP4 encourage the learner to do a variety of learning activities, so they reinforce criterion MM2. Criteria MP3 and MP5 are relatively similar. The main difference is that in criterion MP3, the learner is encouraged to use different pedagogical devices to examine different points of view of other people (perhaps without interaction with those people), whereas in criterion MP5, the learner is encouraged to actively elicit other people's points of view during discussion. Thus, it may be not necessary to always satisfy both criteria, but in some cases, satisfying criterion MP5 will reinforce the satisfaction of criterion MP3. The same conclusions could also be drawn for criteria MP3 and MP4.

Finally, *the set of criteria I proposed is not definitive*. One can surely modify it (e.g., organize a criterion into sub-criteria or assemble several criteria into one criterion), even reject part of the criteria and propose new ones, according to one's personal interpretation of learning conditions fostering cognitive flexibility.

2.5 Criteria proposed by Jonnaert and Vander Borght

In the previous section, I transformed pedagogical principles underlying cognitive flexibility into operational criteria. In this section, I present a set of operational criteria proposed by Jonnaert and Vander Borght (2003) for the *concept* of constructivist learning in the school context, and I show that those criteria are complementary to the set of criteria for cognitive flexibility introduced earlier.

2.5.1 Socio-constructivist and interactive paradigm

In chapter 1, I explained that constructivism has many variations. Therefore, every constructivist author needs to clarify which paradigm(s) he or she follows before going into details his or her own pedagogical model. The learning theory I follow in this thesis is cognitive constructivism (see chapter 1). Jonnaert and Vander Borght follow a socio-constructivist and interactive approach in the school context (the SCI paradigm).

This paradigm primarily consists of three interdependent dimensions. In each dimension, the authors stated one or more postulates, as follows:

1. *Constructivist dimension*: (a) the individual construct his or her knowledge through his or her own activity, and (b) the object handled during this activity is strictly his or her own knowledge.
2. *"Socio" dimension related to social interactions*: The student personally constructs his or her knowledge through interactions with other people (i.e., the teacher and peers).
3. *Interactive dimension related to interactions with the environment*: The student learns concepts "anchored" in situations that are both source and criterion of knowledge. The situation with which the student is confronted is source of knowledge because it confronts his or her prior knowledge with situational demands. The situation is also criterion of knowledge because the student can be efficient in the situation, meaning that his or her knowledge is pertinent.

In my point of view, this approach underlines both individual development and social negotiation (see also section 1.1). This approach reflects my own beliefs on the concepts of knowledge and learning (see chapter 1).

2.5.2 Operational definition of learning and its application

To clarify the concept of learning consistent with their beliefs, Jonnaert and Vander Borght proposed an operational definition for this concept. This definition was formulated on the basis of a set of criteria. Table 2.2 shows the criteria proposed for the three dimensions identified earlier (notice that the concept of learning objects presented here has a more general meaning than the one in the e-Learning context presented in parts 2 and 3). I do not mention here other criteria proposed, for instance for several didactic constraints related to the school context the authors defined, because they are not related to my present work.

The authors showed the practicality of their criteria by using them to analyze a certain number of definitions of learning in the literature. Table 2.3 presents one example of those analyses.

Table 2.2. A main part of criteria proposed by Jonnaert and Vander Borght for the concept of constructivist learning in the school context

Dimensions	Criteria	Expected analysis
(1) Constructivist	(1.1) Who is the **actor** of the learning?	The student
	(1.2) Does the student learn on the basis of his **prior knowledge**?	Yes
	(1.3) Does learning have a **meaning** for the student?	Yes
(2) Socio	(2.1) Does the student learn through **interactions with peers**?	Yes
	(2.2) Does the student learn through **interactions with the teacher**?	Yes
	(2.3) Are the **zones of dialogue** defined to allow interactions among the students, the teacher, and the learning object?	Yes
(3) Interactive	(3.1) Does the student learn from **situations**?	Yes
	(3.2) Does the student have to discover the **learning object** in these situations?	Yes
	(3.3) Does the student have to **interact** with these situations and the learning object?	Yes
	(3.4) Does the environment permit to establish a **distinction** between the learning object and the student's knowledge?	Yes
	(3.5) Are there **interactions** between the learning object and the student's knowledge?	Yes

The point the authors made is that no available definition satisfies all of their criteria. Therefore, they proposed their own definition of the concept of constructivist learning satisfying all of their criteria. Here is its short version (see its full version on page 266 of their book):

> Learning in the school context is a dynamic process in which the student, through interactions with his or her peers and the teacher, interacts with the learning object in order to *construct new knowledge* adapted to the constraints and the resources of the situation the student confronts to use his or her new knowledge in non-didactic situations. (Jonnaert and Vander Borght, 2003, p. 266)

Table 2.3. Analysis of a constructivist definition of learning

Definition:
(…) Learning is a process; the modification of prior acquisition forms an integral part of this process; it is the learner who is the principal actor of the learning; learning in the school context must be anchored in a context that is meaningful to the student. (…) Learning occurs essentially through the management, the operations or the interventions of the student himself or herself; the learner carries out a certain number of steps to appropriate new management and representations of object, or to change some of them; learning is built on and with prior knowledge of the learner; the evoked learning process is strictly under responsibility of the student; the teacher manages only a part of the situation in which he or she places the student; the teacher is not a learning master of the student, the teacher simply manages certain conditions in which he or she places the student. (Jonnaert, 1995, p. 39; cited in Jonnaert and Vander Borght, 2003, p. 261)

Dimensions	**Criteria**	**Analysis**	**Comments**
(1) Constructivist	(1.1) Who is the **actor** of the learning?	**The student**	The student is the principal actor. In addition, it is him or her who manages the learning process.
	(1.2) Does the student learn on the basis of his **prior knowledge**?	**Yes**	This aspect is explicit in the definition.
	(1.3) Does learning have a **meaning** for the student?	**Yes**	This aspect is explicit in the definition.
(2) Socio	(2.1) Does the student learn through **interactions with peers**?	Not specified	
	(2.2) Does the student learn through **interactions with the teacher**?	Implicit	The teacher manages certain conditions in which he or she places the learner.
	(2.3) Are the **zones of dialogue** defined to allow interactions among the students, the teacher, and the learning object?	Not specified	
(3) Interactive	(3.1) Does the student learn from **situations**?	**Yes**	Learning must occur in meaningful contexts for the learner.
	(3.2) Does the student have to discover the **learning object** in these situations?	Not specified	
	(3.3) Does the student have to **interact** with these situations and the learning object?	Not specified	
	(3.4) Does the environment permit to establish a **distinction** between the learning object and the student's knowledge?	**Yes**	The logic of the student is dissociated from that of the learning object.
	(3.5) Are there **interactions** between the learning object and the student's knowledge?	Not specified	

On the basis of their operational definition of the concept of learning in the school context, the authors proposed their *own* learning conditions to foster learning in a manner consistent with the authors' SCI paradigm (see more details in chapter 6 of the authors' book).

2.5.3 Discussion

Jonnaert and Vander Borght's epistemic paradigm is different from mine. The learning context they are concerned with (the school context) is also different from mine (the e-Learning context). The way Jonnaert and Vander Borght and I exploited constructivism, however, is quite similar: We tried to use an *operational* approach to clarify what constructivist learning and instruction are. The main difference between Jonnaert and Vander Borght's work and mine is that they proposed a set of criteria for the *concept* of learning whereas I proposed a set of criteria for *conditions* of learning. I give here a simple analogy: Jonnaert and Vander Borght's criteria could be used to check whether or not an automobile can move forward, move backward, turn left, turn right, and so on. My criteria could be applied to verify whether or not an automobile has necessary materials and devices such as engine, fuel, steering wheel, gas pedal, break pedal to move forward, move backward, turn left, turn right, and so forth. That is why I claim that Jonnaert and Vander Borght's criteria and mine are *complementary*.

To consolidate the previous claim, in chapter 6 I show the practicality and the complement of the two sets of criteria by applying them to analyze the concept of learning and the conditions of learning involved in the COFALE learning environment.

2.6 Conclusion

In this chapter, I have argued that the set of *operational criteria* I described makes the process of instructional design more straightforward than do general guidelines suggested by educational theorists (e.g., Bourgeois & Nizet, 1999; Spiro & Jehng, 1990).I believe that the set of criteria is applicable to a great number of instructional situations (e.g., traditional instruction, computer-based instruction, and distance education). To consolidate this point, in chapter 3, I show how one can apply the set of criteria, as guidelines, to design a complete course for the instruction of complex concepts such as the recursion one in computing science.

The criteria for cognitive flexibility also facilitate the analysis of existing learning systems (see chapter 5). Indeed, the criteria may be used as a framework to identify which aspects of a learning system actually embody which pedagogical principles underlying cognitive flexibility.

CHAPTER 3

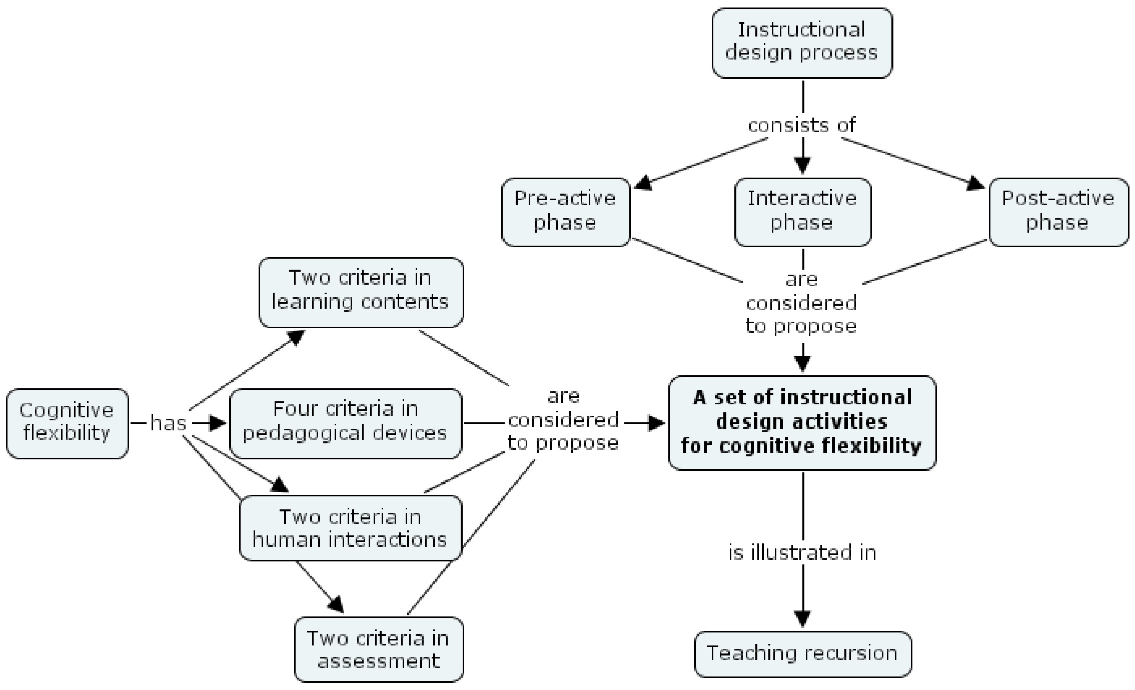
Instructional design with cognitive flexibility

"*I keep six honest serving people. They taught me everything I know. Their names are: What and Why and When and How and Where and Who.*"

Rudyard Kipling, English Novelist, 1865 – 1936 (cited in Santrock, 2001, p. 362)

(Reference to Appendices B & C)

In this chapter, on the basis of the set of operational criteria presented in chapter 2, I propose and justify an instructional design process for cognitive flexibility, and present an example of its use: The teaching of the concept of recursion in computing science. After reading this chapter, one *should* be able to propose and use one's own design process for the practice of one's own teaching.



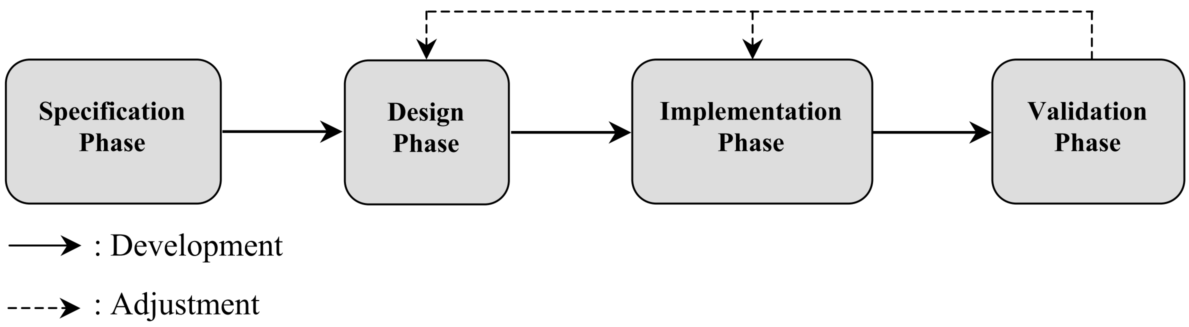
Instructional design process
consists of
Pre-active phase
Interactive phase
Post-active phase
are considered to propose
Two criteria in learning contents
Cognitive flexibility
has
Four criteria in pedagogical devices
are considered to propose
A set of instructional design activities for cognitive flexibility
Two criteria in human interactions
is illustrated in
Two criteria in assessment
Teaching recursion

Summary

3.1 Introduction

Chapter 2 presented a set of operational criteria for cognitive flexibility and a simple example of its use. To successfully create learning conditions satisfying all of these criteria for the instruction of complex concepts, however, I must develop a well-defined design process. This process, of course, is not normative: It is here as an evidence for indicating the *practicality* of the set of criteria for cognitive flexibility. It should, however, be both operational and open enough for the teacher to be able to find easily his or her own way to effectively fulfill all or part of the criteria for cognitive flexibility in his or her own teaching situations. Therefore, in this chapter I show how to use the set of criteria for cognitive flexibility as guidelines to design a complete course for a problem area presented in section 3.2: The learning of recursion, a complex concept in computing science (I will give an explanation of this concept for those who are not in the domain of computing science).

Figure 3.1. A development model of learning systems



The main objective of the thesis is to show how to construct ICT-based learning systems supporting cognitive flexibility. To do so, I use a metaphor in software engineering (Schach, 1999). Figure 3.1 illustrates a development model of learning systems. This model has been used in engineering of several ICT-based learning systems such as PETAL (Bhuiyan et al., 1994) and SimQuest (De Jong et al., 2004). This model consists of the following four phases:

1. *Specification*. This phase aims at specifying the context in which learning is expected to occur, for example to specify students' learning objectives.
2. *Design*. The main goal of this phase is to devise learning situations for students, for instance to prepare the learning contents and pedagogical devices.

3. *Implementation*. The main objective of this phase is to implement the learning conditions proposed in the design phase in an e-Learning context, for example to deliver the learning contents and pedagogical devices in a Web platform.
4. *Validation*. This phase aims to carry out experiments with actual students to formatively evaluate the learning system. The results of the experiments may be used to adjust the design and implementation phases.

In what follows, I first describe the learning context (specification phase). Then, I propose an instructional design process to systematically fulfill the set of 10 criteria for cognitive flexibility identified in chapter 2 (design phase). In chapters 6, 7, and 8, I show how to implement ICT-based learning conditions with respect to the design process proposed in the present chapter (implementation phase). And in chapter 9, I report on a study performed to evaluate those ICT-based learning conditions (validation phase). It should be noted that the set of criteria for cognitive flexibility is used as a useful pedagogical framework through the design phase, the implementation phase, and the validation phase.

3.2 Recursion and learning context

3.2.1 Recursion

« *To iterate is human. To recurse, divine.*”

Logout message on the Carnegie-Mellon CMUA computer (cited in Anderson et al., 1988, p. 163)

Recursion, in general, is the process of defining something in terms of itself (Lewis & Loftus, 2003). Recursive thinking is the ability of humans to solve a problem by reducing it to one or more sub-problems that are *identical* in structure to the original problem and somewhat *simpler* to solve (Robert, 1986). To illustrate recursive thinking, in the following paragraphs, I show a simple example (Kjell, 2003) and a classical and well-known game, named the Towers of Hanoi puzzle (Lewis & Loftus, 2003).

Example 1: Dividing a line

Say that Bob wish to divide a line into 16 equal pieces.

Firstly, he divides the line in half (see Figure 3.2); he has two equal lines. Then, for each of the two lines, he divides the line in half … He does not stop dividing until the number of pieces is 16.

Figure 3.2. Dividing the line in half

In the previous example, Bob divides the original problem into two sub-problems that are *identical* in structure to the original problem (to divide a line into a number of equal pieces) and *simpler* to solve (in the original problem, Bob needs to divide a line into *16* equal pieces, whereas in each of the two sub-problems, Bob needs to divide a line into only *8* equal pieces, and 8 < 16). That is *recursive thinking*.

Example 2: The Towers of Hanoi puzzle

The puzzle consists of three upright pegs (Figure 3.3). On the left peg (peg A), we place a number (`N`) of disks with holes in the middle so that they slide onto the peg (in Figure 3.3, `N=3`). Each disk has a different diameter, the largest disk resting at the bottom and the others getting smaller and smaller up to the top one.

The goal of the puzzle is to move all the disks from the left peg to the right one (peg C). We can use the "extra" peg (peg B) as a temporary place to put disks, but we must obey the following three rules:

1. We can move only one disk at a time.
2. We cannot place a large disk on top of a smaller disk.
3. At any time, all disks must be on some peg except for the disk in transit between pegs.

Figure 3.3. A solution to the three-disk Towers of Hanoi puzzle

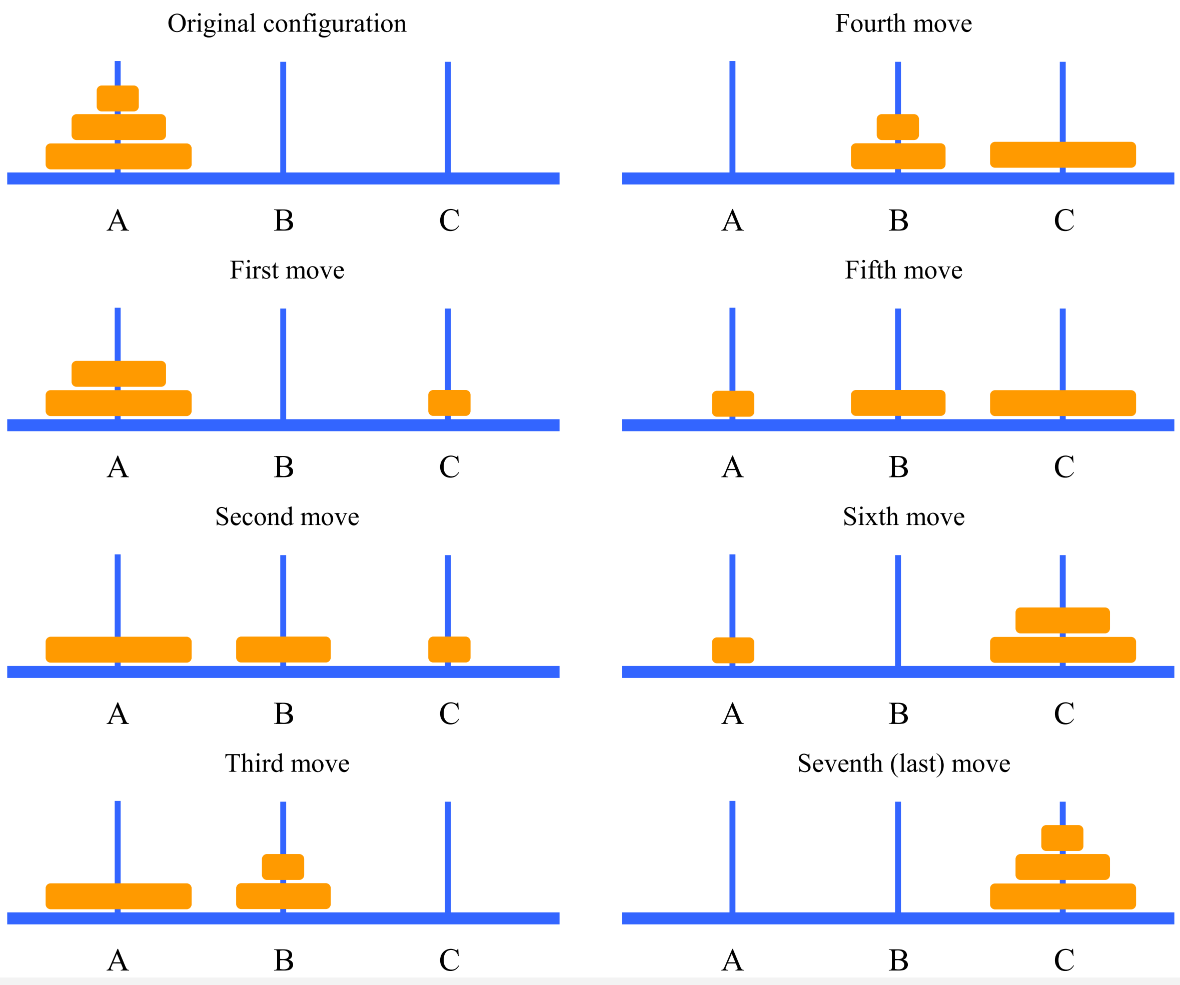


Figure 3.3 presents the step-by-step solution for the Towers of Hanoi puzzle using three disks. To ultimately move all three disks from peg A to peg C, we first have to get to the

point where the smaller two disks are out of the way on peg B so that we can move the largest disk from peg A to peg C.

We can consider the first three moves shown in Figure 3.3 as moving the smaller disks out of the way. The fourth move puts the largest disk in its final place. The last three moves then put the smaller disks to their final place on top of the largest one.

We can use the previous idea to form a general strategy. To move a stack of `N` disks from the original peg (peg A) to the destination peg (peg C):

1. Move the topmost `N-1` disks from the original peg (peg A) to the extra peg (peg B).
2. Move the largest disk from the original peg (peg A) to the destination peg (peg C).
3. Move the `N-1` disks from the extra peg (peg B) back to the destination peg (peg C).

In the example of the Towers of Hanoi puzzle, we reduce the problem of moving `N` disks to the problem of moving `N-1` disks. This sub-problem is *identical* in structure to the original problem: Moving a number of disks from the original peg to the destination peg using the "extra" peg as a temporary place to put disks. This sub-problem is *simpler* to solve than the original problem, because `N-1` < `N`. That is *recursive thinking*. Note that the problem of moving a "stack" that consists of only one disk is so easy: We can accomplish the task directly and without recursion.

The concept of recursion is very important in computing science (Henderson & Romero, 1989). To understand various aspects concerning the teaching and learning of recursion, I have analyzed a certain number of approaches in the literature (see Appendix B6). This analysis shows that many teachers consider that both teaching and learning recursion are difficult because of three main reasons, as follows:

1. The concept is unfamiliar: Students are induced to proceed by analogy from examples. That is, facing a new problem, learners often do not arrive at recursive solutions directly, but they must examine prior examples to find out an analogy to solve the problem recursively.
2. The concept is complex: It is hard for students to transfer from a pattern of recursion to a new one. That is, learners may still have difficulties to solve a new problem recursively, even though they have successfully built recursive solutions to a variety of problems.
3. Interference may arise from knowledge of other methods of solution (e.g. iterations). For example, "novice" learners, when constructing a recursive solution, try to adapt some part of an iterative structure, for instance the updating of loop index variables, to achieve recursion.

3.2.2 Learning context

I concentrate on the creation of learning conditions provided for students to learn recursion; so, for the purpose of the discussion, I assume in this chapter the following learning context:

- The targeted learners are first-year engineering students registered in an introductory course on object-oriented programming and Java at *Faculté des Sciences Appliquées, Université catholique de Louvain*. They have no knowledge of recursion.
- The learning objective, from the students' point of view, is to develop the ability to solve problems recursively.
- Students use a standard Java programming environment and the Internet to learn recursion.
- The course duration is 2 weeks (a 2-hour session in the presence of the teacher for each week). Between the two sessions, students work in groups to solve problems.
- There are not administrative constrains, (e.g., the evaluation of students' learning is formative).

In section 3.3, I show how to devise learning situations leading to cognitive flexibility so that the selected students can attain their learning objective effectively.

3.3 Instructional design process for cognitive flexibility

Because recursion is a complex concept, in this section I propose an instructional design process attempting to satisfy all of the criteria for cognitive flexibility defined in chapter 2. I start by considering the 10 criteria proposed for the four learning components: learning contents, pedagogical devices, human interactions, and assessment. From the teacher's point of view, the two main questions that should be considered in instructional design are:

1. What should be done to provide the learner with learning conditions in the four learning components in order to satisfy the 10 criteria for cognitive flexibility?
2. What should be done to evaluate students' learning behavior and the tutor's teaching behavior with respect to cognitive flexibility?

Educational researchers (e.g., Jonnaert & Vander Borght, 2003) have claimed that any instructional design process consists of three important phases: (a) *pre-active* phase (what should be done before the learning session), (b) *interactive* phase (what should be done during the learning session), and (c) *post-active* phase (what should be done after the learning session).

In what follows, in each of the three phases, I use the 10 operational criteria (see Table 3.1) for cognitive flexibility as guidelines to propose a set of instructional design activities. In the pre-active phase, I propose design activities that specify how the teacher should prepare the learning materials before the learning session. In the interactive phase, I suggest teaching activities specifying what the teacher should do, for instance, with the prepared learning materials, during the learning session. And in the post-active phase, I propose evaluation activities that specify how the teacher should evaluate both the tutor's teaching behavior and students' learning behavior, with respect to cognitive flexibility. For each instructional design activity, I declare which learning component the activity belongs to. I also justify each activity by showing which criteria for cognitive flexibility are the raison d'être of the activity, and I apply the activity to the instructional design for the recursion problem.

Table 3.1. Operational criteria for cognitive flexibility (MM = multiple modes, MP = multiple perspectives)

See sections 1.5.1 and 2.4 to know why the set of criteria is organized into four learning components.

Learning Contents
MM1: *The same learning content presenting concepts and their relationships is represented in different forms (e.g., text, images, audio, video, simulations).*
MP1: *The same abstract concept is explained, used, and applied systematically with other concepts in a diversity of examples of use, exercises, and case studies in complex, realistic, and relevant situations.*
Pedagogical Devices
MM2: *Learners are encouraged to study the same abstract concept for different purposes, at different times, by different methods including different activities (reading, exploring, knowledge reorganization, etc.).*
MP2: *When facing a new concept, learners are encouraged to explore the relationships between this concept and other ones as far as possible in complex, realistic, and relevant situations.*
MP3: *When facing a new concept, learners are encouraged to explore different interpretations of this concept (by other authors and by peers), to express their personal point of view on the new concept, and to give feedback on the points of view of other people.*
MP4: *When facing a new concept, learners are encouraged to examine, analyze, and synthesize a diversity of points of view on the new concept.*
Human Interactions
MM3: *The number of participants, the type of participant (learner, tutor, expert, etc.), the communication tools (e-mail, mailing lists, face to face, chat room, video conferencing, etc.), and the location (in the classroom, on campus, anywhere in the world, etc.) are varied.*
MP5: *During the discussion, learners are encouraged to diversify – as far as possible – the different points of view about the topic discussed.*
Assessment
MM4: *During the learning process, learners are encouraged to use different assessment methods and tools, at different times, and in different contexts for demonstrating their ability to solve different problems.*
MP6: *During the problem-solving process, learners are encouraged to confront multiple ways to solve the problem and multiple possible solutions to the problem.*

Key concepts:

The instructional design process is a set of instructional design activities.

An instructional design activity is one or more operations the teacher should perform in order to create or evaluate certain learning conditions for students.

3.3.1 Pre-active phase

This sub-section presents seven design activities, to be performed by the course designer.

Activity 1.1 (learning contents): *Prepare the learning* <u>*content*</u> *for the* <u>*underlying*</u> *concepts.*

This activity is not the result of examining any criterion for cognitive flexibility. Rather, it is the personal choice of the course designer for teaching a complex concept: We teach both abstract knowledge (concepts) and meaningful cases (learning situations), each one in the context of the other; learning is situated, but abstract knowledge is not ignored (Spiro & Nizet, 1990). However, because this activity is present, the teacher should consider criteria MM1 and MP1: Providing multiple examples and multiple forms such as text, images, and simulations.

In the recursion situation, I prepared the learning content for a number of underlying concepts of recursion, especially recursive thinking and DCG (divide, conquer, and glue) strategy (see Appendix B1). For each concept, I provided various simple examples explained by text, Java programs, and simulations, so that students can acquire the basic knowledge underlying recursion.

Activity 1.2 (learning contents): *Prepare a diversity of meaningful learning situations emphasizing the nature of the underlying concepts.*

This activity is directly guided by criteria MM1 and MP1, which emphasize the need for multiple interpretations and multiple forms of representation for the same concept.

In the recursion problem, I prepared several learning situations (see Appendix B2) to help students understand how to apply the concept of recursion in different contexts. For example, arithmetic expressions explain the use of recursion in binary trees in a natural way, and simple text search emphasizes the use of recursion in linked lists. For each situation, multiple forms of representation were systematically taken into account: textual definitions, images, simulations, and Java programs.

Note that the learning situations I prepared in Appendix B2 give rise to several *cognitive conflicts* in the learners' "mind", which help them master diverse aspects of recursion. For example, faced with arithmetic expressions and partition, students understand that it is very hard for people to solve the given problems with other methods than recursion (e.g., iterations), so they must think recursively. And when students compare recursive solutions with iterative ones in Fibonacci numbers or when students try to represent a large document in simple text search with other data types than linked lists (e.g., arrays), they understand that iterative solutions should not be appropriate to certain problems, they thus like to use recursion.

Activity 1.3 (learning contents): *Prepare learning contents for the concepts that are related to the underlying concepts.*

This activity is directly guided by criteria MP1 and MP2, which underline the learning in a complex landscape of interrelated concepts. Therefore, the teacher should devise the learning content for the related concepts as carefully as for the main concepts.

In the recursion problem, I applied Activities 1.1 and 1.2 to prepare the learning content for the concept of linked lists (see Appendices B3 and B4), which is strongly related to recursion because linked lists are a particular kind of recursive data structures. I did not prepare the learning content for the concepts related to linked lists because I supposed that the targeted students had mastered these concepts in the introductory course on object-oriented programming and Java.

Activity 1.4 (assessment): *Prepare* <u>*assessment situations*</u> *both for individual tests and for tests in groups. The nature of these situations should stimulate multiple points of view.*

This activity is proposed on the basis of criteria MM4 and MP6, which emphasize multiple assessment methods and multiple points of view on the solutions to the given problems.

In the recursion problem, I prepared the robot situation for individual tests (Appendix C) and the file management situation for tests in groups (Appendix C). In the robot situation, to compute the number of ways the robot can walk n meters, the learner could be encouraged to use and compare both the iterative method and the recursive one. In the file management, learners could be encouraged to confront and compare different solutions. For example, to list all files and sub-directories in a given directory, there are at least two solutions that print different results: (a) first list the files and sub-directories in the given directory, then in its sub-directories; and (b) first list the files and sub-directories in the sub-directories of the given directory, then in the given directory.

Activity 1.5 (human interactions): *Prepare* <u>*diverse*</u> *means for engaging the tutor, learners, and other people in exchanges.*

This activity is proposed on the basis of criterion MM3, which underlines the providing of communication tools to help students run diverse discussions. In the recursion problem, I prepared meeting rooms, mailing lists, and an online Java Q&A website (Java World, 2004).

Activity 1.6 (human interactions): *Prepare a list of* <u>*general discussion questions*</u> *and a list of* <u>*domain-specific discussion questions*</u>*.*

This activity is proposed on the basis of criterion MP5, which underlines the explicit tools helping students elicit peers' points of view. In the recursion problem, I made a list of general discussion questions and a list of discussion questions about recursion (see Appendix B5).

Activity 1.7 (pedagogical devices): *Prepare multiple* <u>*external resources*</u> *related to the underlying concepts.*

This activity is directly guided by criterion MP3, which encourages learners to explore different interpretations of the main concepts by other authors. In the recursion situation, I examined different resources and approaches for recursion (Appendix B6), I searched the Internet, and I chose the following resources:

- The books "Java software solutions" (Lewis & Loftus, 2003, chapter 11) and "Thinking recursively" (Roberts, 1986).
- The online Java tutorials (Eck, 2004; Kjell, 2003) in which the authors illustrate a great number of recursive examples.
- The online tutorials on Prolog (AMZI Inc., 1997) and on ML (Cumming, 1998) in which the authors treat recursion as a predominant concept and at the beginning of the course.

The external resources for linked lists should also be prepared in the same way.

Table 3.2 summarizes the seven design activities for the pre-active phase. In the next interactive phase, I explain what the teacher should do with the prepared learning materials.

Table 3.2. Design activities for cognitive flexibility in the pre-active phase

Activities	**Raison d'être**
Activity 1.1 (learning contents): Prepare the learning content for the underlying concepts.	The course designer's personal choice.
Activity 1.2 (learning contents): Prepare a diversity of meaningful learning situations emphasizing the nature of the underlying concepts.	Criteria MM1, MP1.
Activity 1.3 (learning contents): Prepare learning contents for the concepts that are related to the underlying concepts.	Criteria MP1, MP2.
Activity 1.4 (assessment): Prepare assessment situations both for individual tests and for tests in groups. The nature of these situations should stimulate multiple points of view.	Criteria MM4, MP6.
Activity 1.5 (human interactions): Prepare diverse means for engaging the tutor, learners, and other people in exchanges.	Criterion MM3.
Activity 1.6 (human interactions): Prepare a list of general discussion questions and a list of domain-specific discussion questions.	Criterion MP5.
Activity 1.7 (pedagogical devices): Prepare multiple external resources related to the underlying concepts.	Criterion MP3.

3.3.2 Interactive phase

This section shows nine teaching activities, to be performed by the tutor. The arrangement of these activities, according to the timetable of the 2-week learning session, is presented at the end of this section.

Activity 2.1 (pedagogical devices): *Engage learners explicitly in performing multiple learning activities related to the underlying concepts.*

This activity is proposed on the basis of part of criterion MM2, which emphasizes the studying of the same concept for different purposes by different methods including different activities. For instance, in the recursion situation, the tutor could engage learners in the following three learning activities:

1. At the beginning of the course, students are asked to examine the basic concepts underlying recursion and multiple learning situations, prepared in Activities 1.1 and 1.2. These learning contents could be delivered to students, for instance, in paper.
2. After that, students are asked to solve several of the exercises presented at the end of chapter 11 of the book "Java software solutions" (prepared in Activity 1.7) by writing and executing Java programs.
3. During the 2 weeks of the learning session, students are encouraged to find and discuss other interpretations about recursion in other programming languages such as Pascal, Prolog, and LISP (prepared in Activity 1.7).

Activity 2.2 (pedagogical devices): *Encourage learners explicitly to study the concepts that are related to the underlying concepts.*

This activity is directly guided by criterion MP2, which underlines the studying of the main concepts with related ones in meaningful situations. In the recursion problem, when learners study simple text search, the tutor could encourage them to examine the related concept "linked lists" (prepared in Activity 1.3). Similarly, while learners study this concept, the tutor encourages them to revise different aspects of the concept of recursion such as the definition of recursion, base cases, recursive part.

Activity 2.3 (pedagogical devices): *Encourage learners explicitly to examine different interpretations of the underlying concepts (by other authors and by peers), to express their personal points of view on the underlying concepts, and to give feedback on the points of view of other people.*

This activity is directly guided by criterion MP3 and part of criterion MM2. In the recursion situation, after exploring the course designer's interpretation about recursion, learners are engaged in four learning activities, as follows:

1. Add comments, for example to their notebook, on the learning content proposed by the teacher, for instance, reformulate the main points of the definition of recursion.
2. Explore other interpretations of other people about the concept of recursion (prepared in Activity 1.7).
3. Find and add other examples, exercises, and case studies in their own learning spaces, for instance, use online search tools and provided external resources to find recursive examples and add them to learners' notebook.
4. Explore peers' learning spaces to understand what they think and how they learn, for example, read peers' notebooks.

Activity 2.4 (pedagogical devices): *Stimulate learners explicitly to treat a diversity of points of view on the underlying concepts.*

Criterion MP4 and part of criterion MM2 are considered to propose this activity. In the recursion problem, after examining multiple points of view on recursion, learners are asked to produce syntheses on those points of view, for example, produce a table stating the learner's own definitions of recursion, recursive problem solving, and recursive methods, together with peers'. For each point the learner makes, he or she could be asked to provide the information source used to justify the point.

Activity 2.5 (human interactions): *Encourage learners explicitly to run a variety of discussions with other people in different contexts.*

This activity is proposed on the basis of criterion MM3. In the recursion situation, for instance, during 2 weeks of the learning session, the tutor could organize small groups of learners and provide them with: (a) meeting rooms and mailing lists so that they can discuss with peers to solve a problem about file management (see also Activity 2.7), and (b) a Q&A website so that they can ask experts questions about different aspects of recursion. These pedagogical devices were prepared in Activity 1.5.

Activity 2.6 (human interactions): *Make available tools so that learners can <u>actively express their personal points of view</u> and <u>stimulate those of other participants</u> during the discussion.*

This activity is proposed on the basis of criterion MP5. For example, in the recursion problem, the tutor could give a copy of general and domain-specific discussion questions (prepared in Activity 1.6) to each one of learners, and ask them to use this list often during the learning discussion.

Activity 2.7 (assessment): *Encourage groups of learners explicitly to do <u>assessment in groups</u>, to confront and compare multiple points of view.*

This activity, together with Activity 2.8, is directly guided by criteria MM4 and MP6. In the recursion situation, during 2 weeks of the learning session, the tutor could give the same programming problem about file management (prepared in Activity 1.4) to small groups of learners so that they can work collaboratively to solve the given problem (see also Activity 2.5).

Activity 2.8 (assessment): *Encourage learners explicitly to do <u>assessment individually</u>, to confront and compare multiple points of view.*

In the recursion situation, at the end of the learning session, the tutor could give the same tests in the robot situation (prepared in Activity 1.4) to learners, so that they can evaluate their understanding of the recursion concept.

Activity 2.9 (all four learning components): *During the learning session, observe and <u>evaluate</u> the learning behavior of each learner with respect to cognitive flexibility, so as to provide him or her with appropriate feedback.*

This activity is proposed on the basis of all the criteria for cognitive flexibility. Although all 10 criteria were satisfied in the previous activities, students may still not learn in a manner that truly stimulates cognitive flexibility. For example, students may not examine multiple learning situations or run discussions with peers. Therefore, the tutor should constitute a portfolio of the learning history for each individual learner (e.g., learning activities the learner has performed and duration of each activity), and evaluate this portfolio with respect to cognitive flexibility. Table 3.3 presents a checklist to help the tutor in this evaluation task. If the student's learning behavior does not respect a certain criterion, the tutor should encourage him or her to do learning activities to satisfy the criterion. For instance, if the learner explores only one learning situation for a particular concept, the tutor should explicitly ask him or her to examine multiple learning situations in order to try and transfer the concept in diverse contexts.

The evaluation of students' learning behavior is, of course, a difficult and time-consuming task, especially when the number of students is large. I discuss the need for using technology in order to automate this task in section 3.4.

It should be noted that the evaluation of students' learning behavior I mention here means that the tutor verifies whether the student follows the suggestions expected to be consistent with cognitive flexibility, *according to the tutor and the course designer's point of view.*

Evaluating whether the student actually exhibits the behavior of cognitive flexibility at a given time is another issue and will be discussed in chapter 9.

Table 3.3. A checklist for evaluating students' learning behavior with respect to cognitive flexibility

Criteria	**Questions**
MM1	Does the student examine *multiple representations* of the same concept *systematically*?
MP1	Does the student explore *multiple situations* prepared by the course designer for the same concept?
MM2	Does the student perform *multiple learning activities* suggested by the tutor for the same concept in different contexts?
MP2	Does the student study *related concepts* carefully when facing a new concept? Does the student *criss-cross* the learning landscape systematically?
MP3	Does the student express his or her *own points of view* on the new concept? Does the student examine *external resources* prepared by the course designer? Does the student examine *peers' learning space*s? Does the student give *feedback* on the points of view of other people?
MP4	Does the student produce *summaries* on the multiple points of view he or she has confronted?
MM3	Does the student use *communication tools* to run diverse discussions with peers and other people?
MP5	Does the student use the lists of *discussion questions systematically* to elicit peers' points of view?
MM4	Does the student do *individual tests* prepared by the course designer? Does the student *work with peers* to solve problems prepared by the course designer for groups?
MP6	Does the student confront and compare *different solutions* to a given problem systematically?

In the recursion problem, the tutor could ask learners to constitute themselves their own portfolio of learning history. At different points in time of the learning session, on the basis of the checklist shown in Table 3.3, the tutor could analyze students' portfolio and give them feedback, if necessary.

Table 3.4 summarizes the nine teaching activities for the interactive phase. I have been able to create column "Time" on the basis of an experiment presented in chapter 9. This column presents the time the teacher should plan for students' learning activities mentioned in the respective teaching activity.

Table 3.4. Teaching activities for cognitive flexibility in the interactive phase

Activities	**Raison d'être**	**Time**
Activity 2.1 (pedagogical devices): Engage learners explicitly in performing multiple learning activities related to the underlying concepts.	Criterion MM2.	Week 1, during 2 weeks.
Activity 2.2 (pedagogical devices): Encourage learners explicitly to study the concepts that are related to the underlying concepts.	Criterion MP2.	Week 1.
Activity 2.3 (pedagogical devices): Encourage learners explicitly to examine different interpretations of the underlying concepts (by other authors and by peers), to express their personal points of view on the underlying concepts, and to give feedback on the points of view of other people.	Criteria MM2, MP3.	Week 1.
Activity 2.4 (pedagogical devices): Stimulate learners explicitly to treat a diversity of points of view on the underlying concepts.	Criteria MM2, MP4.	Week 1.
Activity 2.5 (human interactions): Encourage learners explicitly to run a variety of discussions with other people in different contexts.	Criterion MM3.	During 2 weeks, week 2.
Activity 2.6 (human interactions): Make available tools so that learners can actively express their personal points of view and stimulate those of other participants during the discussion.	Criterion MP5.	During 2 weeks, week 2.
Activity 2.7 (assessment): Encourage groups of learners explicitly to do assessment in groups, to confront and compare multiple points of view.	Criteria MM4, MP6.	During 2 weeks, week 2.
Activity 2.8 (assessment): Encourage learners explicitly to do assessment individually, to confront and compare multiple points of view.	Criteria MM4, MP6.	Week 2.
Activity 2.9 (all four learning components): During the learning session, observe and evaluate the learning behavior of each learner with respect to cognitive flexibility, so as to provide him or her with appropriate feedback.	All criteria.	Weeks 1, 2.

3.3.3 Post-active phase

In this section, I show the last two evaluation activities in the post-active phase.

Activity 3.1 (all four learning components): *<u>Evaluate</u> the learning behavior and outcomes formatively for each learner, and communicate both the result of the analysis process and feedback explicitly to him or her, keeping a positive regard on his or her knowledge.*

This activity is directly guided by all the criteria for cognitive flexibility. It is similar to Activity 2.9; the essential difference is that it is done at the end of the learning session, and that it concerns a global analysis of students' learning process and outcomes. According to the constructivist point of view presented in chapters 1 and 2, the teacher should evaluate both *what* students learn (i.e., acquired knowledge of the taught subject) and *how* students learn (e.g., acquired knowledge of how to express, confront, and treat multiple points of view). So, in addition to analyzing students' portfolio of learning history (see Activity 2.9), the teacher should evaluate students' actual performance on the basis of their assessment (individual and in groups), communicate the result of the analysis process to each individual learner, and give appropriate feedback to him or her for future learning. Note that the teacher should refuse the failure in learning: In a constructivist point of view, there is neither success nor failure of learning (Jonnaert & Vander Borght, 2003).

In the recursion situation, the tutor should analyze the portfolio of learning history constituted by each student during the learning session (see also Activity 2.9). The tutor could analyze students' Java programs and justifications in the file management situation and students' tests in the robot situation. The tutor could also interview students about how they learn recursion and how they solve the problems in the given assessment situations. After the analysis process (including the interview one), the tutor should send each student a brief report indicating the results of analyzing his or her assessment (e.g., to show problems in his or her Java programs and tests) and the feedback on his or her learning (e.g., how to improve Java programs, ability to solve problems recursively, ability to work in groups).

Activity 3.2 (all four learning components): *Evaluate the teaching behavior with respect to cognitive flexibility.*

This activity is also proposed on the basis of all the criteria for cognitive flexibility. Al though the teacher effectively applies the set of design and teaching activities presented in sections 3.3.1 and 3.3.2, sometimes there would be a certain number of activities that need to be adjusted for future teaching, because they do not fit to the current learning context. For instance, there is a certain learning situation students do not like or the available examples cannot help students avoid several kinds of misconceptions about the underlying concepts. So, the teacher should keep a track of his or her design and teaching activities, and analyze them according to the set of criteria for cognitive flexibility. In Table 3.5, I show a checklist for this evaluation task.

In the recursion problem, for example, the course designer and the tutor should keep a track of every operation they did in the pre-active and interactive phases. Then they could use the checklist presented in Table 3.5 to analyze this track, for instance, to do an evaluation report indicating which design or teaching activities need to be adjusted, and how and why they should be modified. This report may help them improve teaching with respect to cognitive flexibility, not only for the recursion concept but also for other ones.

I believe that, in long-term learning sessions, the teacher should also do this evaluation activity in the interactive phase to adjust his or her teaching activities on the fly. I did not do so for the learning session I mentioned earlier because it is a short-term session (2 weeks).

Table 3.6 summarizes the two evaluation activities for the post-active phase.

Table 3.5. A checklist for evaluating the teaching behavior with respect to cognitive flexibility

Criteria	Questions
MM1	Is *every representation* the course designer prepared for a particular concept *useful* for students? Is it *necessary* to make *other representations* available for a particular concept for students?
MP1	Is *every example, exercise, situation etc.* the course designer prepared for a particular concept *useful* for students? Is it *necessary* to make *other examples, exercises, situations etc.* available for a particular concept for students?
MM2	Is *every learning activity* suggested by the tutor *useful* for students? Is it *necessary* to suggest *other learning activities* for students?
MP2	Is the way the tutor facilitates students' *criss-crossing* of the learning landscape *effective*?
MP3	Is the way the tutor encourages students to *express their own points of view effective*? Is *every external resource* prepared by the course designer *useful*? Is the way the tutor stimulates students to explore *peers' learning space*s *effective*? Is the way the tutor encourages students to give *feedback* on the points of view of other people *effec-tive*?
MP4	Is the way the tutor engages students in producing *summaries* on the multiple points of view they have confronted *effective*?
MM3	Is *every communication means* suggested by the tutor *useful* for students?
MP5	Are the lists of *discussion questions* prepared by the course designer *useful* for students?
MM4	Is *every individual test* prepared by the course designer *useful* for students? Is *every assessment in groups* prepared by the course designer is *useful* for students?
MP6	Does *every assessment situation* actually give rise to *multiple ways* to solve the given problem and *multiple solutions* to the given problem by students? Is the way the tutor encourages students to confront and compare *different solutions* for a given prob-lem *effective*?

Table 3.6. Evaluation activities for cognitive flexibility in the post-active phase

Activities	**Raison d'être**
Activity 3.1 (all four learning components): Evaluate the learning behavior and outcomes forma-tively for each learner, and communicate both the result of the analysis process and feedback explicitly to him or her, keeping a positive regard on his or her knowledge.	All criteria.
Activity 3.2 (all four learning components): Evaluate the teaching behavior with respect to cogni-tive flexibility.	All criteria.

Table 3.7 shows the pertinence of all 18 design, teaching, and evaluation activities to the 10 criteria for cognitive flexibility.

Table 3.7. Pertinence of the instructional design process to cognitive flexibility

		Operational criteria for cognitive flexibility									
		Learning contents		**Pedagogical devices**				**Human interactions**		**Assessment**	
Phases	**Activities**	**MM1**	**MP1**	**MM2**	**MP2**	**MP3**	**MP4**	**MM3**	**MP5**	**MM4**	**MP6**
Pre-active	**Activity 1.1**	The course designer's personal choice									
	Activity 1.2	X	X								
	Activity 1.3		X		X						
	Activity 1.4									X	X
	Activity 1.5							X			
	Activity 1.6								X		
	Activity 1.7					X					
Interactive	**Activity 2.1**			X							
	Activity 2.2				X						
	Activity 2.3			X		X					
	Activity 2.4			X			X				
	Activity 2.5							X			
	Activity 2.6								X		
	Activity 2.7									X	X
	Activity 2.8									X	X
	Activity 2.9	X	X	X	X	X	X	X	X	X	X
Post-active	**Activity 3.1**	X	X	X	X	X	X	X	X	X	X
	Activity 3.2	X	X	X	X	X	X	X	X	X	X

3.4 Discussion

In this chapter, I proposed an instructional design process (see Table 3.7) that fulfills all 10 criteria for cognitive flexibility introduced in section 2.4. This design process provides a clear evidence for the practicality of the set of operational criteria for cognitive flexibility: It may be used as a useful framework to facilitate the exploitation of pedagogical principles underlying cognitive flexibility for the practice of instruction.

As mentioned earlier, the practitioner should understand that the proposed design process is neither normative nor definitive. What I tried to do in this chapter is only to illustrate the usefulness of the set of criteria by showing *an example*. In practice, we may need to adjust certain operations to fit to situational demands of concrete contexts, because no single pedagogical model fits every teaching context (Jonnaert & Vander Borght, 2003; Spiro & Jehng, 1990). I believe, however, that the way I proposed the design process, on the basis of operational criteria, is applicable to a great number of instructional situations. The practitioner, after examining this chapter, should be able to propose his or her own design process to adapt to his or her personal teaching contexts. For example, one could modify the activities presented in section 3.3, reject part of them, or propose new activities, in such a way that one carefully takes the criteria for cognitive flexibility into account. The practitioner could also organize a certain activity into a number of operations to make the instructional design process more practical than does mine. The practitioner, however, should note that as the design process becomes more operational, it becomes more dependent on the context in which learning occurs. Another point is that it is not necessary to always satisfy all of the criteria for cognitive flexibility (see the discussion in section 2.4.5). The choice of criteria and the way to satisfy them depend on the concrete context in which learning occurs.

In this chapter, I concentrated on describing the teaching process. In chapter 6, I illustrate the learning process with support for cognitive flexibility in an ICT-based learning environment. Why and how should ICT concern cognitive flexibility?

The instructional design process presented in section 3.3 shows that ICT could be a very promising means by which to implement essential learning conditions exhibiting the pedagogical principles underlying cognitive flexibility. Many constructivist theorists have also advocated this point (Driscoll, 2000; Spiro & Jehng, 1990). For instance, regarding Activity 2.2 presented in section 3.3, hypermedia could provide significant help in implementing the complex landscape of interrelated concepts (Spiro & Jehng, 1990). Regarding Activity 2.5 shown in the previous section, the Internet and Web could provide various means of discussion such as mailing lists, chat rooms, forums (Milgrom et al., 1997). Considering Activities 2.9 and 3.1 presented earlier, the Web could also be used to automatically register part of the learning behavior of each individual learner, for example, the learning contents he or she has viewed and duration of each view (Adaptive Technology Resource Center, 2004; Milgrom et al., 1997).

In chapter 5, I go into details of the analysis of several ICT-based tools fostering cognitive flexibility. In chapter 7, I show how to exploit ICT to implement the learning conditions presented in section 3.3.

PART TWO: CONSTRUCTIVISM, ADAPTABILITY, AND ICT-BASED LEARNING ENVIRONMENTS

Some work presented in this part was reported in the following paper:

Chieu, V.M., & Milgrom, E. (2005). COFALE: An adaptive learning environment supporting cognitive flexibility. *The Twelfth International Conference on Artificial Intelligence in Education*, Amsterdam, The Netherlands, 491–498 (full paper).

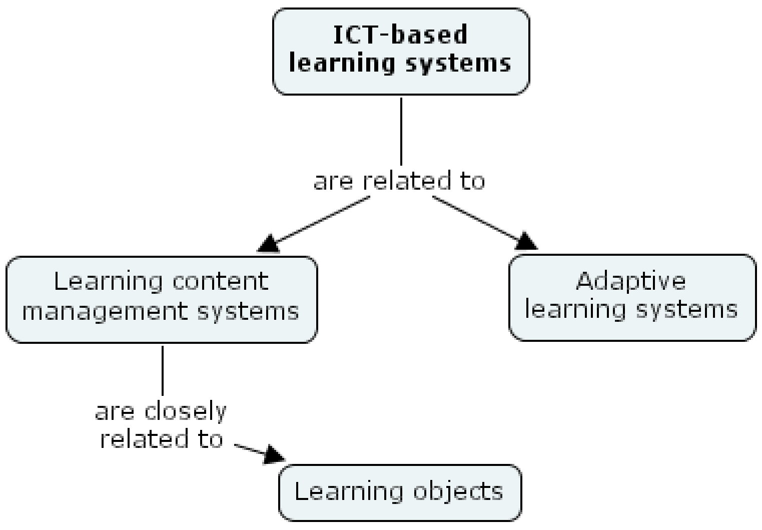
CHAPTER 4

Background

"*If I have seen further than the others, it is because I have stood on the shoulders of giants.*"

Isaac Newton, English Scientist, 1642 – 1727 (cited in Suomela, 2005)

In this chapter, I explain the following three concepts closely related to ICT-based learning systems: learning content management systems, learning objects, and adaptive learning systems. These concepts provide the reader with some background knowledge for understanding the next chapters, especially the use of the COFALE learning system shown in chapter 7.


ICT-based learning systems
are related to
Learning content management systems
Adaptive learning systems
are closely related to
Learning objects

Summary

4.1 Learning content management systems

In recent years, learning content management systems (LCMSs) have been widely adopted (Leslie, 2003), as evidenced by the appearance and the use of many e-Learning platforms such as ATutor (Adaptive Technology Resource Center, 2004), Claroline (De Praetere et al., 2004), Moodle (Dougiamas, 2004). As for many new terms in learning technology, there is no universal acceptance of the definition of a LCMS. The following definition seems to be adopted quite frequently by researchers in the field:

> A LCMS is a multi-user software application that enables content authors to manage the life-cycle of learning content by allowing them to create, register, store, assemble, re-use, and publish digital learning content for delivery via Web, print, CD, etc., within a central object repository (Masie Center, 2003).

Sometimes LCMSs may be confused with LMSs (learning management systems). A LMS is a high-level, strategic solution for planning, delivering, and managing all learning events within an organization, including online, virtual classroom, and instructor-led courses (Greenberg, 2002; Masie Center, 2003). The focus of LMSs is to manage learners, keeping track of their progress and performance across all types of training activities, whereas the focus of LCMSs is on learning content. See Greenberg's analysis (2002) for more details about the difference between LMSs and LCMSs.

LCMSs have been designed to enable subject matter experts, with little technology expertise, to design, create, deliver, and measure the results of e-Learning courses rapidly. Those systems play a key role in learning technology because they offer organizations one of the most effective and flexible means to deliver and manager just-in-time and up-to-date e-Learning courses (Masie Center, 2003; Robbins, 2002), meaning that the learner can get the most recent and right learning materials that meet his or her own needs at any time.

For example, ATutor is an open source, Web-based LCMS designed and maintained by ATRC (Adaptive Technology Resource Center, 2004). According to Greenberg (2002), ATutor is a good LCMS because of the following characteristics:

- *Various learning tools*. ATutor provides the learner with many learning tools. For example, learners can: (a) take tests, review test results, and keep track of their scores; (b) work in groups by using a virtual collaboration hyperspace; (c) use forums, chat rooms, and e-mail to exchange ideas; (d) introduce their own external resources and explore those of other people; (e) search a sharable database, for instance TILE (Adaptive Technology Resource Center, 2004), for learning objects; (f) review their navigation history; (g) set preferences for their course, for instance, screen layout, display of text and icons, display of navigation elements.

- *Simple adaptation support*. Learners can move through the learning content using global or hierarchical or sequential navigation tools. Navigation elements can be displayed as text or icons or both text and icons; they can also be hidden to simplify the learning environment.
- *Easy-to-use authoring tools*. A set of tools including a file manager, a content editor, and a visual editor enables the course designer to create content objects, information blocks, and learning objects, and to define associations among them easily (see section 7.2.1). A specific tool is offered to the course designer for introducing external resources (see section 7.2.1). And a number of other easy-to-use tools allow the course designer to specify many characteristics of a course such as the display of learning tools and navigation elements.
- *Course tracker*. The course designer and the teacher can review tool and content usage statistics to evaluate and adjust their teaching behavior. They can also analyze the learning behavior of a particular student to give him or her appropriate feedback (see sections 6.3.1 and 7.2.3).
- *IMS/SCORM content packaging*. The course designer finds it easy to export or import content objects, information blocks, learning objects, and even a complete course from or into ATutor as IMS/SCORM conformant content packages (Advanced Distributed Learning, 2004). One of the goals in the design of ATutor is to make it interchangeable with other conformant e-Learning systems.
- *Learning objects repository*. The course designer can export learning resources from ATutor to a sharable database, for instance TILE (Adaptive Technology Resource Center 2004), and search the database for appropriate learning resources related to a particular course.
- *Assessment tools*. The course designer can create several kinds of tests, evaluate students' tests, and give feedback to students (see sections 7.2.1 and 7.2.2).
- *Communication and collaboration tools*. The course designer can create forums, groups of students, and collaboration hyperspaces (see section 7.2.1). The teacher can participate in students' discussions via forums, mailing lists, and chat rooms (see section 6.3.1).
- *Effective administration tools*. Many tools are proposed to the administrator such as learner and instructor manager, course manager, backup manager, and language manager.
- *Automated installer and upgrade*. ATutor provides a fast and easy way to install or upgrade the system.
- *Information security*. Advanced techniques are used to protect users' data and learning contents of the system.

Several educational researchers (e.g., Dougiamas, 2004; Laanpere et al., 2004) have claimed that almost none of existing LCMSs support pedagogical principles explicitly in order to make them more attractive to the largest possible audience. Therefore, in practice, the course designer should apply pedagogical innovations to a LCMS to improve learning outcomes. For instance, I have exhibited the pedagogical principles underlying cognitive flexibility in the design and use of COFALE presented in chapters 6, 7, and 8.

The most important concept involved in LCMSs is learning objects (Masie Center, 2003). The next section explains this concept and its practicality.

4.2 Learning objects

Learning object is obviously a key concept in the field of learning technology, as it has been widely adopted by many organizations, from schools to enterprises (Advanced Distributed Learning, 2004; Masie Center, 2003). Here is a general definition of learning objects proposed by Masie Center (2003):

> A re-usable, media-independent chunk of information used as a modular building block for e-Learning content. Learning objects are most effective when organized by a metadata classification system and stored in a content repository such as a LCMS (p. 75).

Why learning objects are important and why we need to specify metadata for them. The next sub-sections explain these two critical issues.

4.2.1 Why are learning objects important?

In a traditional model of an e-Learning course (Figure 4.1), a course is organized as a single unit of instruction. It is a complete presentation of all the learning materials required to meet the defined course goal. Each lesson normally consists of a set of screens with information presented in multiple forms such as text, images, audio and video files. At the end of each lesson, there may be a set of traditional quiz interactions such as multiple-choice questions. The lessons are contained in a shell including navigation, which is usually a combination of back/next buttons and a course menu. At the end of the course, there may be a set of summary statements followed, for instance, by a multiple-choice test. The course is a complete unit with a single score that would be registered in a database or a LMS. Kjell's website (2003) is an example of the traditional model.

One of the characteristics of the traditional model is the ease for the course designer or the teacher to implement it, usually without intervention of the software developer. Critics, however, say that this model provides nothing more than electronic books, which are "standardized" for every learner. Moreover, it is hard for the course designer to repurpose a learning resource (Masie Center, 2003).

Now let me show how the course designer uses the learning object model (Figure 4.2) to organize the same information as in the previous example, regardless of pedagogical principles. Take into consideration the information from the first Lesson 01: Overview. The course designer first defines a measurable learning objective. Then, he or she defines an appropriate assessment to measure the competency of the learning objective. Finally, the course designer fills the learning object with assets, for instance a text and an image, required to meet the learning objective. The learning object "Recursion" presented in chapters 6 and 7 is an instance of this model.

Figure 4.1. Traditional course model (adapted from Masie Center, 2003)

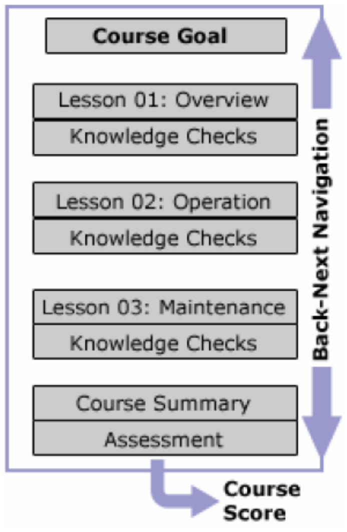
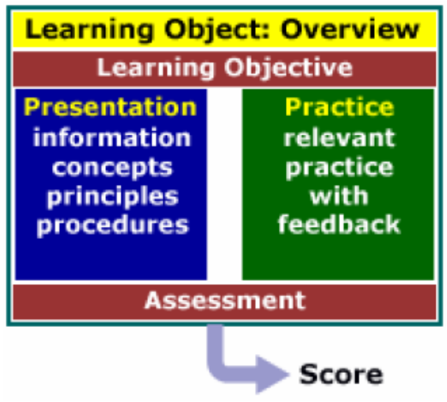


Figure 4.2. Learning object model (adapted from Masie Center, 2003)



The major difference between the learning object model and the traditional one is that each learning object is a stand-alone instructional unit with its own learning objective, learning content, assessment, and navigation mode. In my point of view, this flexibility has the following benefits:

- *For the learner*. The most promising benefit of learning objects for the learner is that personalized courses can be constructed to meet the needs of the individual learner (see examples in section 6.3.2).
- *For the course designer*. The most interesting benefit of learning objects for the course designer is that learning objects can be stored in a LCMS, so that the course designer can search for them and repurpose them in different contexts (see more details in section 7.2.1). It is also quite easy for the course designer to create a particular learning object that is tailored to the requirements of a particular student (see examples in section 7.3.2).
- *For the software developer*. Decomposing the learning content into fine-grained pieces of information could make the implementation of adaptive presentation of learning contents straightforward (see also chapter 8 and Appendix D).

According to experts in the field (Masie Center, 2003), however, adapting a learning object approach to content development and management may also encounter several obstacles. For instance, learners, teachers, and course designers have to change the way they learn, teach, and design, respectively (see chapters 6 and 7): the learner must be active in selecting learning objects and in navigating the content within the learning objects, the teacher must allow the learner to pursue his or her own learning objectives, the course designer needs to build many small objects, and the course designer may have to make a course by organizing several learning objects created by other people with different navigation modes.

4.2.2 Why and how do we specify metadata for learning objects?

Simply defined, metadata are the data that describe things. The purpose and usefulness of metadata in e-Learning are that they provide the ability to describe and identify a great number of learning contents so that we can find, assemble, and deliver the right content to the right person at the right time (Masie Center, 2003). Indeed, metadata allow learning objects to be stored, indexed, searched, and retrieved from a database. For example, providing a certain number of keywords for learning objects, the course designer allows himself or herself or someone else to search for a particular learning object that could be repurposed in different learning contexts.

To help people in the field ensure the six "-abilities" (interoperability, reusability, manageability, accessibility, durability, and scalability) within an organization and across multiple organizations, several learning technology groups such as IEEE (2005), IMS (2005), and ARIADNE (2005) have approved standards for learning object metadata (LOM). Table 4.1 shows the use of several LOM elements proposed by IEEE for describing learning resources.

The metadata implementer should be familiar with the "Metadata principles & practicalities" document (Duval et al., 2002) for a list of founding principles shared by almost all metadata groups.

Table 4.1. A part of LOM proposed by IEEE (URI = Universal Resource Identifier)

Nr	Property name	Explanation	Examples
1	**General**	This category groups the general information that describes this learning object as a whole.	-
1.1	**Identifier**	A globally unique label that identifies this learning object.	-
1.1.1	Catalog	The name or designator of the identification or cataloging scheme for this entry. A namespace scheme.	"URI"
1.1.2	Entry	The value of the identifier within the identification or cataloging scheme that designates or identifies this learning object. A namespace specific string.	"http://www.info.ucl.ac.be/learning_resources/recursion_vmc" for the learning object "Recursion" created by vmc "http://www.info.ucl.ac.be/learning_resources/linked_lists_vmc" for the learning object "Linked Lists" created by vmc
1.2	**Title**	Name given to this learning object.	"Recursion" for the learning object "Recursion" "Linked Lists" for the learning object "Linked Lists"
1.3	**Language**	The primary human language or languages used within this learning object to communicate to the intended user.	"en"
1.4	**Description**	A textual description of the content of this learning object.	"This learning object helps students develop the ability to solve problems recursively" for the learning object "Recursion" "This learning object helps students develop the ability to represent data by using linked lists" for the learning object "Linked Lists"
1.5	**Keyword**	A keyword or phrase describing the topic of this learning object.	"recursion", "recursive programming" for the learning object "Recursion" "linked lists", "dynamic data structures" for the learning object "Linked Lists"

4.2.3 Discussion

To look further into the concept of learning objects and metadata, one should examine the document made by Advanced Distributed Learning (2004) or Masie Center (2003) in which complete descriptions of learning objects and metadata were presented. To understand how educational researchers have applied various pedagogical principles to the design and use of learning objects, one should read Wiley's online book (2002). To understand more about the practice of learning objects, one should examine ATutor's "How To Course" (Adaptive Technology Resource Center, 2004).

In section 7.2.1, I look further into the practice of learning objects in COFALE: I explain how to design and use learning objects in COFALE to support cognitive flexibility and adaptability.

4.3 Adaptive learning systems

Adaptability is the ability of a learning system to provide each learner with appropriate learning conditions to facilitate his or her own process of knowledge construction and transformation. In an e-Learning context, especially in Web-based distance education, adaptability is important because of at least two main reasons (Brusilovsky, 1999; Brusilovsky & Peylo, 2003; Milgrom et al., 1997): (a) the student often learns without direct and personalized assistance of a human tutor or of peers; and (b) the e-Learning platform is often used by a variety of students (who have different personal profiles about, e.g., background knowledge and learning progress), so a learning experience that is designed with a particular class of learners in mind may not suit other learners.

Key concept: *Adaptability* is the ability of a learning system to provide a learning experience that is continuously tailored to the needs of the individual learner.

It should be noted that several authors (e.g., Brusilovsky & Peylo, 2003) use the term "adaptive and *intelligent* learning systems" to denote learning systems that support adaptability. They emphasize the qualifier "intelligent" because the learning systems with which they are concerned have, for example, the ability to analyze students' solutions during their problem-solving session in order to provide students with appropriate feedback (e.g., hints). In the present thesis, I use the term "adaptive learning systems" to denote all kinds of systems that provide students with support for adaptability, including intelligent support.

In section 1.5.4, I mentioned five adaptation techniques that are often used in adaptive learning systems: (a) adaptive presentation of learning contents, (b) adaptive use of pedagogical devices, (c) adaptive communication support, (d) adaptive problem-solving support, and (e) adaptive assessment. In section 5.3, I analyze several adaptive learning systems, and I show examples of the implementation of those five adaptation techniques. In chapters 6,7, and 8, I also explain the way to implement several basic adaptation techniques in the COFALE learning system.

4.4 Conclusion

In the present chapter, I do not contribute anything new. Rather, I provide the reader with several key concepts related to ICT-based learning systems so that the reader can easily understand the analysis of existing learning systems presented in the next chapter, and the design and use of the COFALE learning system shown in the third part of the thesis.

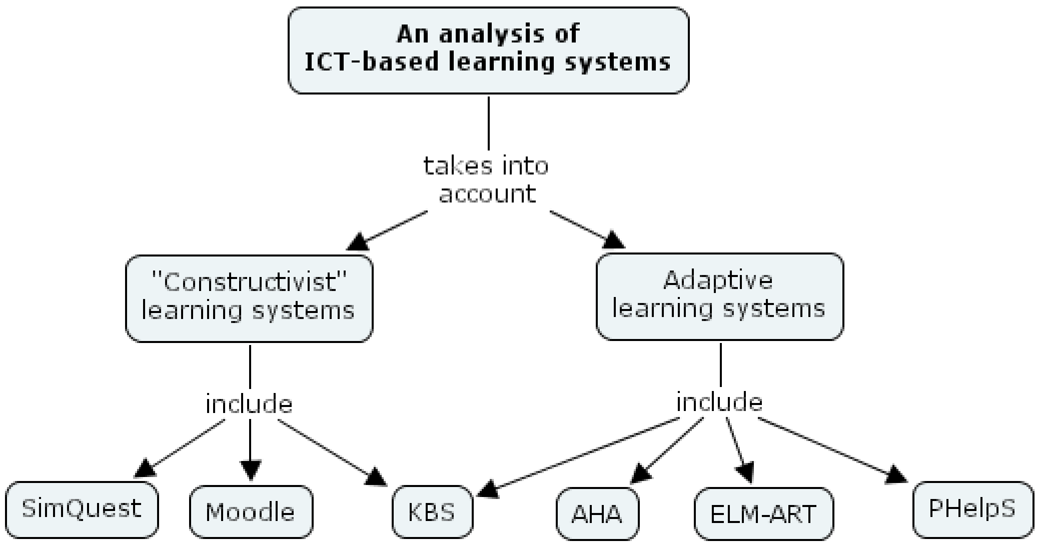
CHAPTER 5

State of the art

"*Every problem has a better solution when you start thinking [about] it differently than [in] the normal way.*"

Steve Wozniak, Co-designer of the Original Apple Computers, 1950 (cited in Kaplan, 2000, p. 94)

In this chapter, I first present an analysis of several "constructivist" learning systems. Then, I analyze a number of adaptive learning systems. The main purpose of the present chapter is to look into how researchers in the field have exploited available ICT to provide support for constructivism and support for adaptability.


An analysis of ICT-based learning systems
takes into account
"Constructivist" learning systems
Adaptive learning systems
include
include
SimQuest
Moodle
KBS
AHA
ELM-ART
PHelpS

Summary

5.1 Introduction

In recent years, constructivist beliefs and practices have been widely adopted, as evidenced by the appearance of several ICT-based "constructivist" learning systems (Kinshuk et al., 2004). During the past fifteen years, many interesting ICT-based adaptive learning systems have been developed and reported (Brusilovsky & Peylo, 2003).

The main goal of the present thesis is to help teachers designing truly ICT-based adaptive learning environments supporting cognitive flexibility, an important facet of constructivism. To do so, I must first analyze a number of existing learning systems with respect to the following two critical issues: how researchers have exploited available ICT (a) to foster constructivist learning, especially cognitive flexibility, and (b) to implement adaptation support. It should be noted that the word "to analyze" I use in this chapter means to examine, but not to evaluate, the conditions of learning provided by existing systems (I do not want to evaluate any values of the systems I analyze here).

5.2 "Constructivist" learning systems

In this section, I look into several examples handled by the following three systems that explicitly claim to support constructivism: **SimQuest** (De Jong et al., 2004), **Moodle** (Dougiamas, 2004), and **KBS** (Henze & Nejdl, 2001). I also look into **ATutor** (Adaptive Technology Resource Center, 2004), a learning content management system. Although ATutor does not support any pedagogical principle explicitly, I analyze it in this section because the COFALE learning system presented in the next chapters has been built on ATutor.

For each system, I first explain the educational approach of the authors. Then, I explore an example designed by the authors: I evaluate the conditions of learning shown in the example with respect to *my interpretation* of the pedagogical principles underlying cognitive flexibility introduced in chapter 2 (say the set of operational criteria for cognitive flexibility). Finally, I look into how the authors of the system ensured the learning conditions they created to be consistent with the educational approach they followed, and I also analyze, if possible, constructivist learning conditions provided in the example other than the learning conditions fostering cognitive flexibility.

5.2.1 SimQuest: Scientific discovery learning

Educational approach

The design of the SimQuest learning system is based on computer-based simulations and scientific discovery learning, a self-directed and constructivist form of learning (Klahr & Dunbar, 1988; Reimann, 1991; Van Joolingen & De Jong, 1997). Discovery learning can be summarized, as follows:

> Discovery learning is a type of learning where learners construct their own knowledge by experimenting with a domain, and inferring rules from the results of these experiments. The basic idea of this kind of learning is that because learners can design their own experiments in the domain and infer the rules of the domain themselves they are actually *constructing* their knowledge. Because of these constructive activities, it is assumed they will understand the domain at a higher level than when the necessary information is just presented by a teacher or an expository learning environment. (Van Joolingen, 1999, p. 386)

The authors of SimQuest state that students need to possess a certain number of scientific discovery skills for discovery of learning to be successful. These skills include hypothesis generation, experiment design, prediction, and data analysis. Lack of these skills can lead to ineffective discovery behavior such as drawing incorrect conclusions from collected data.

Therefore, the authors support a number of cognitive tools and guidelines (see the next sub-section) allowing learners to carry out scientific experiments, to formulate hypotheses, and to draw conclusions easily.

Analysis process

To analyze an example of the SimQuest system with respect to the criteria for cognitive flexibility, I played the role of the student to explore the course on "motion", which was designed to help learners mastering the effect of the acceleration on the vehicle speed and the distance covered and the influence of the mass, force, and friction on the acceleration.

Criterion MM1: *The same learning content presenting concepts and their relationships is represented in different forms (e.g., text, images, audio, video, simulations).*

This criterion is well satisfied in the example handled by SimQuest because it provides the combination of text, graphs, simulations, audios, and videos for the student to grasp diverse aspects of the concepts of acceleration, speed, etc. For instance, in Figure 5.1, the student is encouraged to run simulations to see relationships among the acceleration, the initial speed, and the distance covered.

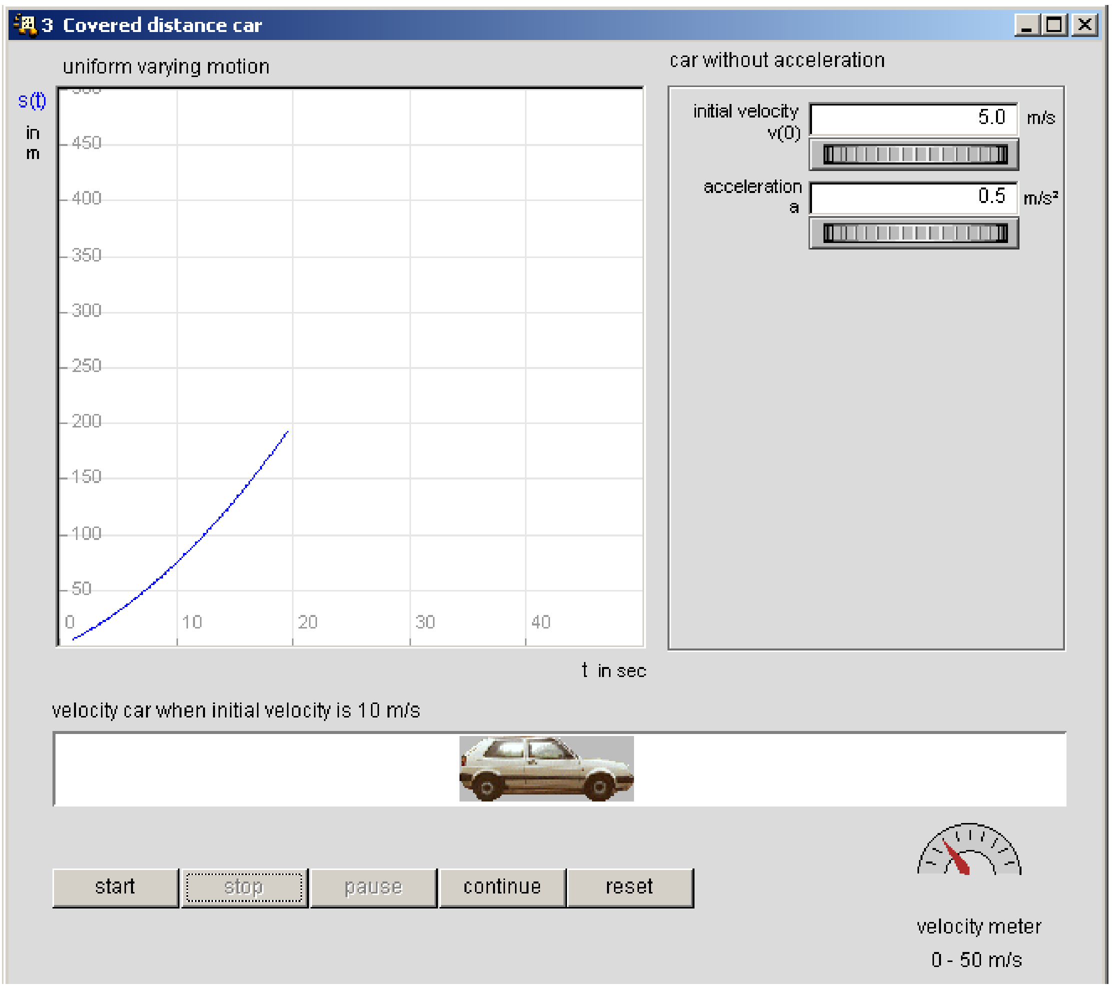
Criterion MP1: *The same abstract concept is explained, used, and applied systematically with other concepts in a diversity of examples of use, exercises, and case studies in complex, realistic, and relevant situations.*

In the course on motion, the course designer well prepared many kinds of experiments in the form of simulations. Each type of simulations involves several interrelated concepts in complex and relevant situations. For example, in Figure 5.1, the concepts of initial speed, acceleration, and distance are explained in the movement of an automobile. So, criterion MP1 is also well satisfied.

Criterion MM2: Learners are encouraged to study the same abstract concept for different purposes, at different times, by different methods including different activities (reading, exploring, discussion, knowledge reorganization, etc.).

In the course on motion, the learner is encouraged to examine the same concepts in different contexts, at different times, and by different methods. For instance, the learner can examine the concept of acceleration by reading its definition, doing multiple free experiments to see the effect of the acceleration with other concepts (e.g., Figure 5.1), and testing the learner's hypotheses through simulation exercises. Thus, this criterion is satisfied.

Figure 5.1. A part of the student's learning space in SimQuest



Criterion MP2: When facing a new concept, learners are encouraged to explore the relationships between this concept and other ones as far as possible in complex, realistic, and relevant situations.

This criterion is satisfied because in every simulation the student is always stimulated to examine multiple concepts systematically.

Criterion MP3: *When facing a new concept, learners are encouraged to explore different interpretations of this concept (by other authors and by peers), to express their personal point of view on the new concept, and to give feedback on the points of view of other people.*

This criterion is *not* satisfied here because the student is not encouraged to explore the points of view of other authors and peers. The learner can make his or her own hypotheses; however, he or she is not engaged in expressing them explicitly, for instance in a textbox or a table.

Criterion MP4: *When facing a new concept, learners are encouraged to examine, analyze, and synthesize a diversity of points of view on the new concept.*

This criterion is *not* satisfied because the learner reads summaries prepared in advance by the course designer.

Criterion MM3: *The number of participants, the type of participant (learner, tutor, expert, etc.), the communication tools (e-mail, mailing lists, face to face, chat rooms, video conferencing, etc.), and the location (in the classroom, on campus, anywhere in the world, etc.) are varied.*

The student learns individually with computer-based simulations: No communication tool is available for engaging students and the teacher in exchanges. Therefore, this criterion is *not* satisfied.

Criterion MP5: *During the discussion, learners are encouraged to diversify – as far as possible – the different points of view about the topic discussed.*

As with MM3, this criterion *fails* to be met.

Criterion MM4: *During the learning process, learners are encouraged to use different assessment methods and tools, at different times, and in different contexts for demonstrating their ability to solve different problems.*

This criterion is *not* satisfied because only individual exercises are provided for the student.

Criterion MP6: *During the problem-solving process, learners are encouraged to confront multiple ways to solve the problem and multiple possible solutions to the problem.*

In my point of view, this criterion is *not* satisfied because for every exercise, there is only one correct answer, and the student is expected to find it.

Discussion

There are no explicit validation means (or rather criteria) for one to know whether the conditions of learning the authors of SimQuest created for the motion course are consistent with the pedagogical principles they wanted to exhibit. Therefore, I shall only analyze the constructivist learning conditions fostering cognitive flexibility.

Although SimQuest's example satisfies only 4 among my 10 criteria for cognitive flexibility, I think those four criteria are satisfied in an appropriate manner because of the way

SimQuest exploits computer-based simulations for fostering constructivist learning. It may be the case that in an introductory course such as the course on motion, satisfying those four criteria is sufficient to help learners learn effectively. So, SimQuest's motion course could be an example indicating that the quality of satisfying the set of criteria for cognitive flexibility is more important than the number of satisfied criteria (see the discussion in section 2.4.5).

5.2.2 Moodle: Constructionist pedagogy

Educational approach

The design of the Moodle system is grounded in social constructionist pedagogy (Bonk & Cunningham, 1998; Jonassen, Peck, & Wilson, 1999). According to Dougiamas (2004), this pedagogical model consists of the following four main concepts:

1. *Constructivism*. The author believes that people *actively* construct new knowledge as they interact with their environment and that knowledge is strengthened if people can *actively* use it in the environment surrounding them.
2. *Constructionism*. The author assumes that learning is particularly effective if people construct artifacts for others to experience.
3. *Social constructivism*. Extending the idea of constructionism, the author thinks that learning is more effective if social group construct artifacts for one another and if people collaboratively create shared artifacts.
4. *Connected and separate*. Separate behavior is when a person tries to remain objective and tends to defend his or her own ideas using logic to find out holes in his or her opponent's ideas. Connected behavior is when a person accepts subjectivity, trying to listen and ask questions in an effort to understand other points of view. The author believes that a reasonable amount of connected behavior within a learning community is a powerful stimulant for learning, not only bringing learners closer together but promoting deeper reflection and re-examination of their existing beliefs.

The author claims that the current version of Moodle does not effectively support all pedagogical principles with which he is concerned. Further improvements in pedagogical support will be a principal direction for the development of Moodle.

Analysis process

To analyze an example of the Moodle system, I also played the role of the student to explore one of its demonstration courses: The course on Moodle features, which is designed to help people mastering the important features of Moodle.

<u>Criterion MM1</u>: *The same learning content presenting concepts and their relationships is represented in different forms (e.g., text, images, audio, video, simulations).*

This criterion is easily satisfied in the *Web-based* Moodle platform: The Moodle author explicitly encourages the course designer to prepare multiple forms for presenting the learning content.

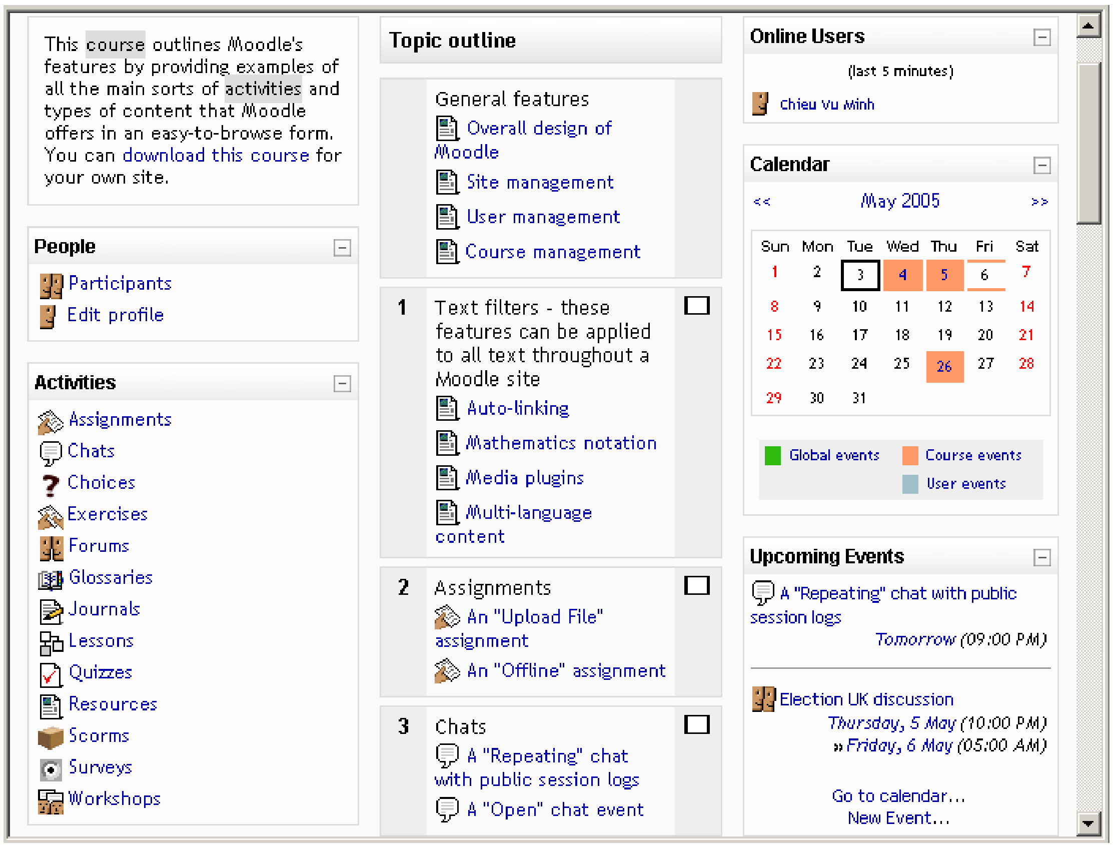
<u>Criterion MP1</u>: *The same abstract concept is explained, used, and applied systematically with other concepts in a diversity of examples of use, exercises, and case studies in complex, realistic, and relevant situations.*

Although this criterion should not be related to any learning platform, it seems that the Moodle author does not explicitly encourage the course designer to prepare diverse learning situations. Indeed, in the course on Moodle features, no situation is available for the student to look into abstract concepts such as learning activities and teaching activities. Thus, Moodle does <u>not</u> satisfy this criterion.

<u>Criterion MM2</u>: *Learners are encouraged to study the same abstract concept for different purposes, at different times, by different methods including different activities (reading, exploring, discussion, knowledge reorganization, etc.).*

The course on Moodle features obviously satisfies this criterion because the student is explicitly encouraged to do multiple learning activities at different times for mastering the same concepts; for instance, reading, writing, discussing, and testing (see the menu "Activities" on the bottom-left corner of Figure 5.2).

Figure 5.2. A part of the student's learning space in Moodle



This course outlines Moodle's features by providing examples of all the main sorts of activities and types of content that Moodle offers in an easy-to-browse form. You can download this course for your own site.

People

- Participants
- Edit profile

Activities

- Assignments
- Chats
- Choices
- Exercises
- Forums
- Glossaries
- Journals
- Lessons
- Quizzes
- Resources
- Scorms
- Surveys
- Workshops

Topic outline

General features
- Overall design of Moodle
- Site management
- User management
- Course management

1 Text filters - these features can be applied to all text throughout a Moodle site
- Auto-linking
- Mathematics notation
- Media plugins
- Multi-language content

2 Assignments
- An "Upload File" assignment
- An "Offline" assignment

3 Chats
- A "Repeating" chat with public session logs
- A "Open" chat event

Online Users

(last 5 minutes)

Chieu Vu Minh

Calendar

<< May 2005 >>

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

🟩 Global events 🟧 Course events
🟦 User events

Upcoming Events

A "Repeating" chat with public session logs
Tomorrow (09:00 PM)

Election UK discussion
Thursday, 5 May (10:00 PM)
» *Friday, 6 May* (05:00 AM)

Go to calendar...
New Event...

Criterion MP2: *When facing a new concept, learners are encouraged to explore the relationships between this concept and other ones as far as possible in complex, realistic, and relevant situations.*

In the features course, Moodle does not support explicit tools to engage the learner in criss-crossing the learning landscape. For instance, there is no explicit tool stimulating the student to explore concepts related to the one he or she is examining. Therefore, this criterion is *not* satisfied.

Criterion MP3: *When facing a new concept, learners are encouraged to explore different interpretations of this concept (by other authors and by peers), to express their personal point of view on the new concept, and to give feedback on the points of view of other people.*

In the features course, Moodle explicitly respects this criterion because it exhorts the students, for instance, to explore external resources, to assess themselves their work, and to assess peers' work (see the menu "Activities" on the bottom-left corner of Figure 5.2: Assignments, Resources, Workshops).

Criterion MP4: *When facing a new concept, learners are encouraged to examine, analyze, and synthesize a diversity of points of view on the new concept.*

Although students are encouraged to write reports on the course, they are not explicitly stimulated to produce summaries on multiple points of view they have met. That is why this criterion is *not* satisfied.

Criterion MM3: *The number of participants, the type of participant (learner, tutor, expert, etc.), the communication tools (e-mail, mailing lists, face to face, chat rooms, video conferencing, etc.), and the location (in the classroom, on campus, anywhere in the world, etc.) are varied.*

In the features course, Moodle satisfies this criterion quite well. For example, students can work in small groups, sometimes with the participation of the tutor, by using e-mail, chat rooms, forums (e.g., see the menu "Activities" on the bottom-left corner of Figure 5.2: Chats, Forums, Workshops).

Criterion MP5: *During the discussion, learners are encouraged to diversify – as far as possible – the different points of view about the topic discussed.*

This criterion is also well satisfied in the Moodle features course because it provides the student with a list of methodological tools such as how to read, write, and ask questions effectively.

Criterion MM4: *During the learning process, learners are encouraged to use different assessment methods and tools, at different times, and in different contexts for demonstrating their ability to solve different problems.*

In the features course, Moodle satisfies this criterion because it stimulates students to do diverse assessment activities at different points in time (e.g., see the menu "Activities" on the bottom-left corner of Figure 5.2: Assignments, Exercises, Quizzes, Workshops).

<u>Criterion MP6</u>: During the problem-solving process, learners are encouraged to confront multiple ways to solve the problem and multiple possible solutions to the problem.

In the features course, Moodle does *<u>not</u>* satisfy this criterion because it does not explicitly encourage the course designer to prepare assessment situations whose nature stimulates different points of view.

Discussion

As for SimQuest, the author of the Moodle learning system does not explicitly provide criteria for evaluating the conformity of the system with the educational approach applied for the design and use of the system.

The design and use of the course on Moodle features satisfy 6 of my 10 criteria for cognitive flexibility, distributed in all of the four learning components. I think Web technologies have been well exploited to satisfy those six criteria effectively. Thus, the Moodle features course could be another example of exploiting ICT to foster cognitive flexibility in introductory learning.

5.2.3 KBS: Constructivism in distance learning

Educational approach

The KBS system is built on constructivist models of learning and teaching (Duffy & Jonassen, 1992). In a context of distance learning, the authors argue for the needs of encouraging the student to learn *actively* and not just to read the information *passively*. The way the authors chose to stimulate students is to integrate problems or "real world tasks" in the curriculum of a "virtual course". This approach is more or less consistent with the constructivist facet of "reasoning, critical thinking, and problem solving" presented in section 1.4.1. Here are several key points the authors emphasize:

- The specification and integration of authentic and complex activities during the learning process are important elements in the design of constructivist learning environments (CLEs).
- The CLE simulates the problem context in which the student performs those authentic activities: The student has to decide how to structure and solve the problem, collect background information, develop solution strategies, and so forth.
- Authentic activities shift the responsibility for both selecting and performing tasks from the tutor to the student.
- Project-based and problem-based approach should be appropriate for designing such a CLE: Providing the learner with references, case studies, background, and related information as well as a working environment.

Analysis process

The example handled by KBS is an introductory course on object-oriented programming and Java, given to undergraduate students in electrical engineering and computer science. The main objective of the course is to help students develop object-oriented programming skills and master basic concepts of the Java language. Because the language used in the course is German that I do not know, and because I cannot access the course, I analyzed the example handled by the KBS system by examining the information about KBS presented by the authors in a journal article (Henze & Nejdl, 2001). Therefore, this analysis may not be fully pertinent.

Criterion MM1: The same learning content presenting concepts and their relationships is represented in different forms (e.g., text, images, audio, video, simulations).

The Java course has been built on a Web platform, so KBS easily satisfies this criterion. For example, several kinds of information are systematically presented to the student: text, Java programs, and images.

Criterion MP1: The same abstract concept is explained, used, and applied systematically with other concepts in a diversity of examples of use, exercises, and case studies in complex, realistic, and relevant situations.

The Java course has been built on a project-based and problem-based approach, so I would say KBS also satisfies this criterion. For instance, the student can explore a particular concept with other ones in multiple relevant problems and projects.

Criterion MM2: Learners are encouraged to study the same abstract concept for different purposes, at different times, by different methods including different activities (reading, exploring, discussion, knowledge reorganization, etc.).

This criterion is satisfied in KBS's Java course because the student can explore the same concept in different problems or projects by different activities such as reading, programming, testing.

Criterion MP2: When facing a new concept, learners are encouraged to explore the relationships between this concept and other ones as far as possible in complex, realistic, and relevant situations.

This criterion is explicitly considered in KBS's Java course because for each presentation of a concept, the system presents the student with a set of hyperlinks to related concepts, problems or projects.

Criterion MP3: When facing a new concept, learners are encouraged to explore different interpretations of this concept (by other authors and by peers), to express their personal point of view on the new concept, and to give feedback on the points of view of other people.

Although the student is engaged in exploring external resources such as the Sun Java tutorial, this criterion is *not* effectively satisfied because no tool is available for students to express their personal points of view and give feedback on those of other people.

Criterion MP4: When facing a new concept, learners are encouraged to examine, analyze, and synthesize a diversity of points of view on the new concept.

This criterion is *not* satisfied because the student is not encouraged to produce summaries after examining the course designer's interpretation and external resources.

Criterion MM3: The number of participants, the type of participant (learner, tutor, expert, etc.), the communication tools (e-mail, mailing lists, face to face, chat rooms, video conferencing, etc.), and the location (in the classroom, on campus, anywhere in the world, etc.) are varied.

Interactions among learners and between the learner and the teacher seem to be ignored in the KBS learning system: The criterion is *not* satisfied.

Criterion MP5: During the discussion, learners are encouraged to diversify – as far as possible – the different points of view about the topic discussed.

For the same reason as in the previous criterion, this one appears to be *not* satisfied.

Criterion MM4: During the learning process, learners are encouraged to use different assessment methods and tools, at different times, and in different contexts for demonstrating their ability to solve different problems.

In the Java course, KBS satisfies this criterion because it provides the learner with different assessment methods at different times. For instance, the student is asked to do different kinds of problems or projects, including programming tasks.

Criterion MP6: During the problem-solving process, learners are encouraged to confront multiple ways to solve the problem and multiple possible solutions to the problem.

No explicit information is available to assess this criterion in the design of KBS' Java course.

Discussion

As for SimQuest and Moodle, there are not explicit criteria for evaluating the conformity of the KBS learning system with the educational approach used to design the system.

Among my 10 criteria for cognitive flexibility, KBS's Java course satisfies five criteria effectively, on the basis of a problem-based and project-based approach. I could say that KBS's Java course is also an example of fostering cognitive flexibility in introductory learning by means of ICT-based learning conditions.

5.2.4 ATutor: A learning content management system

Educational approach

ATutor does not support any pedagogical principle explicitly.

Analysis process

Because of the pedagogical neutrality of the ATutor system, I did not analyze examples handled by the system. Instead, I looked into which criteria for cognitive flexibility can be satisfied if a course designer who is versed in the application of the set of criteria uses ATutor to deliver a course. For example, I examined which learning conditions shown in chapter 3 can be created using the ATutor system.

Criterion MM1: *The same learning content presenting concepts and their relationships is represented in different forms (e.g., text, images, audio, video, simulations).*

Built on a Web platform just as Moodle and KBS, ATutor easily satisfies this criterion.

Criterion MP1: *The same abstract concept is explained, used, and applied systematically with other concepts in a diversity of examples of use, exercises, and case studies in complex, realistic, and relevant situations.*

If the course designer understands this criterion, he or she would satisfy the criterion because this criterion is not related to the platform in which is delivered the course.

Criterion MM2: *Learners are encouraged to study the same abstract concept for different purposes, at different times, by different methods including different activities (reading, exploring, discussion, knowledge reorganization, etc.).*

This criterion can be satisfied using ATutor because the system supports many learning activities such as reading, testing, working in groups, and exploring external resources.

Criterion MP2: *When facing a new concept, learners are encouraged to explore the relationships between this concept and other ones as far as possible in complex, realistic, and relevant situations.*

The menu "Related topics" provided by ATutor allows the course designer to encourage learners to criss-cross the learning landscape. So, this criterion is satisfied.

Criterion MP3: *When facing a new concept, learners are encouraged to explore different interpretations of this concept (by other authors and by peers), to express their personal point of view on the new concept, and to give feedback on the points of view of other people.*

No explicit tool is available to satisfy this criterion. We should modify the ATutor system to implement explicit tools satisfying the criterion.

Criterion MP4: *When facing a new concept, learners are encouraged to examine, analyze, and synthesize a diversity of points of view on the new concept.*

As criterion MP3, this one is *not* satisfied.

Criterion MM3: *The number of participants, the type of participant (learner, tutor, expert, etc.), the communication tools (e-mail, mailing lists, face to face, chat rooms, video conferencing, etc.), and the location (in the classroom, on campus, anywhere in the world, etc.) are varied.*

The course designer can satisfy this criterion by using multiple communication tools provided by ATutor, for instance e-mail, chat rooms, forums.

Criterion MP5: During the discussion, learners are encouraged to diversify – as far as possible – the different points of view about the topic discussed.

As criteria MP3 and MP4, this one is *not* satisfied.

Criterion MM4: During the learning process, learners are encouraged to use different assessment methods and tools, at different times, and in different contexts for demonstrating their ability to solve different problems.

If the course designer is versed in the use of this criterion, he or she can satisfy it using ATutor, for example creating assessment situations both for the individual learner and for the groups of learners.

Criterion MP6: During the problem-solving process, learners are encouraged to confront multiple ways to solve the problem and multiple possible solutions to the problem.

As criterion MP1, this one can be satisfied if the course designer understands how to apply the criterion.

Discussion

Although ATutor is a learning content management system that does not explicitly support any pedagogical principle, it provides many promising tools for implementing learning conditions fostering cognitive flexibility. Indeed, if the course designer is versed in the use of the set of criteria for cognitive flexibility, he or she can exploit available tools of ATutor in order to satisfy many of those criteria.

5.2.5 Discussion of "constructivist" learning systems

SimQuest, Moodle, and KBS explicitly claim to support constructivism. Sometimes, it is hard to say to which facet(s) of the five ones identified in section 1.4 their educational approaches belong because they are often described in a general fashion. I believe, however, that the educational paradigms implied in those systems are more or less related to cognitive flexibility.

Table 5.1 summarizes the result of the previous analysis of the four learning systems, according to the information available to me about an example of their use (except for ATutor) and based on *my* set of criteria for *cognitive flexibility*. It should be noted that the limitation of my analysis is that it is based on my *personal interpretation* of the pedagogical principles underlying cognitive flexibility introduced in chapter 2. If one uses his or her own set of criteria to evaluate those systems, the result may be different from mine. The point I make here is that operational criteria are *practical* for evaluating the conformity of conditions of learning and pedagogical principles.

Indeed, SimQuest, Moodle, and KBS may have effectively implemented learning conditions fostering other facets of constructivism than cognitive flexibility. For instance, I believe that computer-based simulations provided by SimQuest could be very promising means to stimulate different aspects of constructivist learning (e.g., complex and realistic learning environments). It is, however, difficult to evaluate the conformity between the educational approach the authors followed and the learning conditions they implemented, because of *the lack of validation criteria*.

Table 5.1. Existing learning systems examples and support for cognitive flexibility

Existing learning systems	**Operational criteria for cognitive flexibility**									
	Learning contents		**Pedagogical devices**				**Human interactions**		**Assessment**	
	MM1	**MP1**	**MM2**	**MP2**	**MP3**	**MP4**	**MM3**	**MP5**	**MM4**	**MP6**
SimQuest (Motion course)	X	X	X	X						
Moodle (features course)	X		X		X		X	X	X	
KBS (Java course)	X	X	X	X					X	
ATutor (recursion course)	X	X	X	X			X		X	X

The analysis of the previous examples of use of learning systems may indicate that there are many different ways to create ICT-based learning conditions to foster cognitive flexibility and that the course designer should always take into account the quality of criteria satisfaction rather than only the number of satisfied criteria (see the discussion in section 2.4.5). In the next part, I show another way to foster cognitive flexibility in advanced learning such as mastering the concept of recursion in computing science.

From Table 5.1, we can see that criterion MP4 (for pedagogical devices) and criterion MP6 (for assessment) are absent in all of the examples I analyzed. In the next part, I show how to create ICT-based learning conditions satisfying those two criteria. Note that it is not surprising that the examples I analyzed do not satisfy all of the criteria for cognitive flexibility; maybe because the authors of those examples and of the underlying systems may have designed them without any explicit ideas of cognitive flexibility in mind.

Among the four analyzed learning systems, only Moodle and ATutor take into account the implementation of learning content management, and only ATutor takes into consideration the implementation of learning objects.

Except for KBS, none of the "constructivist" learning systems I looked at effectively implements adaptation support. The next section presents an analysis of several adaptive learning systems.

Main result: There have been various manners to exploit pedagogical principles underlying cognitive flexibility in the design and use of ICT-based learning systems. All 10 criteria for cognitive flexibility are seldom met.

5.3 Adaptive learning systems

In this section, I examine adaptation support in the following systems: **AHA** (De Bra & Calvi, 1998), **KBS** (Henze & Nejdl, 2001), **ELM-ART** (Weber & Brusilovsky, 2001), and **PHelpS** (Greer et al., 1998). ELM-ART can be accessible online, so in addition to reading papers describing the system I played the role of the learner to explore it. To analyze the other systems, I only examined published articles.

In the present thesis, I do not present any new technique for implementing adaptation support in ICT-based learning systems. Rather, I borrow several adaptation techniques from existing learning systems to implement adaptability in the COFALE learning system (shown in the next part). Therefore, I shall not go into details here about various adaptation techniques in those systems. Instead, for each system, I summarize the way the system models a course, the characteristics of the student the system takes into account, and the *adaptive features* the system offers to the student. The five main adaptation techniques (see section 1.5.4) I looked at are: (a) adaptive presentation of learning contents, (b) adaptive use of pedagogical devices, (c) adaptive communication support, (d) adaptive problem-solving support, and (e) adaptive assessment.

5.3.1 AHA: An open adaptive hypermedia architecture

This sub-section analyzes several examples handled by AHA about, for instance, the course on the subject of hypermedia, which is designed to help learners develop basic understanding of hypermedia structures and systems.

Course and learner modeling

Course modeling in AHA is very simple. Each course consists of a set of Web pages (HTML files). Each page is a text or a hypertext in which there are one or more hyperlinks to related pages.

Learner modeling in AHA is also simple. Each student is represented by a set of boolean variables. Each variable indicates *whether or not* the student, for example, knows a concept or fails a test or performs a certain learning activity (e.g., reading a page or completing a test).

At the beginning of the course, all variables are set to be "false" for every new learner. After the student reads a page or does a test, the system will update his or her learner model, for example to set the value of the variable corresponding to "the page is read" or "the test is done" to be "true".

In each HTML file, the course designer can insert several conditional fragments in order to adapt the content of the Web page to different kinds of students. This work, of course, requires the course designer's knowledge of HTML. Here is an instance of conditional content:

```
<!-- if not readme -->
  You must first read
  <a href="readme.html">the instructions</a>.
  These will explain how to use this course text.
<!-- else -->
  You have read
  <a href="readme.html">the instructions</a>.
  You can start studying this course.
<!-- endif -->
```

Adaptation support

Among the five adaptation techniques I mentioned previously, AHA effectively implements only adaptive presentation of learning contents. For example, if the student has not read the file `readme.html` yet, the system presents him or her with the following segment of hypertext (in a Web browser, the words "the instructions" appear as a link anchor):

```
You must first read the instructions.
These will explain how to use this course text.
```

Otherwise, the system presents the student with the following segment of hypertext (in a Web browser, the words "the instructions" appear as a link anchor):

```
You have read the instructions.
You can start studying this course.
```

AHA also supports adaptive presentation of hyperlinks (I consider this kind of adaptation to be a particular type of adaptive presentation of learning contents). Here are several examples:

- *Direct guidance*. At the end of each Web page, the system suggests the next "best" page to a particular student, according to his or her learner model.
- *Link hiding*. Sometimes the system does not show a hyperlink in a Web page for a particular student because the hyperlink is assumed to be irrelevant to him or her at that time.
- *Link annotation*. The system changes the color of a hyperlink for a particular student (according to the assumed learner model at a certain time) using the metaphor of traffic light, for instance, green for recommended links (at that time) and red for links that are not ready to explore (at that time). This is a good feature that many adaptive hypermedia systems take into account (Brusilovsky & Peylo, 2003; Brusilovsky, 1999).

Discussion

Although the course designer needs to know HTML to be able to implement adaptation support in the AHA system, it is straightforward to design a new course in any domain in AHA. The system, however, concentrates on adapting only learning content to different kinds of students, and student modeling using Boolean variables in AHA is quite superficial. I believe that AHA is effective for constructing general hypermedia systems, but it needs to be improved if used in the context of instruction and learning.

5.3.2 KBS: An open adaptive corpus hypermedia

In this sub-section, I look into an example designed in KBS about the introductory course on object-oriented programming and Java for undergraduate students in electrical engineering and computer science (see also section 5.2.3).

Course modeling

Course modeling in the KBS learning system is complex. In this sub-section, I give only an overview of several main points. More details are available in Henze and Nejdl's paper (2001).

A conceptual network is used to represent a course in KBS: Each node of the network may be an introduction of a learning concept, an example of a learning concept, a hyperlink to a Web resource, a glossary item, or a lecture grouping several introductions, examples, and Web resources, etc. Each link between two nodes of the network may be a "relevant" link (similar to "related" link in COFALE, see sections 6.3.1, 7.2.1), "prerequisite" link indicating that a particular learning concept must be mastered before another concept, etc. Problems and projects are also integrated into the conceptual network in the same way.

Learner modeling

Student modeling in the KBS system is also complex. The knowledge of the learner is modeled as a knowledge vector (a multi-layered overlay model). Each component of the vector is a conditional probability, describing the system's estimation of the fact that the learner has knowledge about a knowledge item (i.e., a learning concept such as `if` or `while` or `classes` in the course on Java). In KBS, the authors distinguish five knowledge levels a student may have about a knowledge item: "excellently known", "well known", "known", "partly known", and "not known".

At the beginning of the course, the system sets a default model for every new student: The "not known" knowledge level for every knowledge item of the course. At a certain time during the learning process, the tutor evaluates the student's work on projects and updates his or her knowledge level on knowledge items. For instance, after evaluating a student's work on a project concerning the concept `while`, the tutor could diagnose that a student has "expert's knowledge" on this concept (excellently known).

On the basis of the project evaluation provided by the tutor about a particular student, the system will estimate again the student's knowledge of every knowledge item of the current course. For example, if the student's knowledge about a concept is assumed to be "known", then the student's knowledge about all "prerequisite" knowledge items of this one is also supposed to be "known". After updating the information about the student's knowledge, the system adapts the learning materials to that new learner model. In KBS, a Bayesian network engine is used to calculate the probability of the fact that the student has a certain knowledge level about a concept (see Henze & Nejdl, 2001).

Adaptation support

To let the KBS system perform adaptation support, every information resource (Web pages, examples, projects, etc.) needs to be indexed. To do so, the course designer examines the content of each information resource and introduces a set of knowledge items related to this resource.

Among the five adaptation techniques identified earlier, KBS effectively implements the following three ones.

Adaptive presentation of learning contents. The KBS system provides students with appropriate information resources while they are performing their projects, depending on the student's current knowledge. For instance, if the student lacks some "prerequisite" knowledge to solve a problem, the system will present appropriate information units for him or her to "fill the gaps" before solving the problem.

Similarly to the AHA system, KBS also supports adaptive navigation for the student, for example, by changing the color of hyperlinks to indicate whether a Web page is ready for reading or suggesting the next reasonable learning step, according to the student's current knowledge. Following is an example showing the way the KBS system detects whether a HTML page is ready for reading for a particular student:

- Let H to be the HTML page, HKI to be the set of knowledge items related to H (according to the indexing described earlier), and PHKI to be the set of all "prerequisite" knowledge items of the ones in HKI.
- H is recommended for reading for the student if his or her knowledge level (estimated by the system) on every knowledge item in PHKI is "known" or "well known" or "excellently known". This expression could mean that all prerequisites required to understand page H are at least known to the student, so page H is ready for reading for him or her.

Adaptive use of pedagogical devices. While performing a certain project, the student is encouraged to do appropriate learning activities, depending on his or her present knowledge. For instance, for a particular student, the system may suggest him or her to review his or her prior successful examples related to the project he or she is working on.

Adaptive assessment. The KBS system does not propose the same projects for every student. Rather, it suggests suitable problems to the individual student, depending on his or her current knowledge. For instance, a student who is interested in learning simple control structures in Java will have difficulties with a project that applies control structures to construct a graphical user interface, provided that he or she possesses only a beginner's knowledge about graphical user interfaces. So, a project with no graphical features should be appropriate to him or her.

Discussion

KBS effectively provides students with several kinds of adaptation. It is also a domain-independent platform (Henze & Nejdl, 2001). The system, however, has not provided the course designer with a set of authoring tools yet.

5.3.3 ELM-ART: A Web-based adaptive versatile system

To analyze the Web-based ELM-ART learning system, I logged, as a learner, into its online introductory course on LISP programming, which is designed to help undergraduate students develop programming skills in the LISP language. The 10-year-research ELM-ART system is very complex. I would say that my understanding about the workings of this system, especially various artificial intelligence techniques implemented in it, is incomplete. I can only show here a few general points of the system and the major results it offers. To look further into the system, one should explore multiple resources (e.g., Weber & Brusilovsky, 2001; Weber & Specht, 1997; Weber, 1996) describing it.

Course modeling

Course modeling in ELM-ART consists of two parts: (a) domain knowledge or conceptual knowledge, which is declarative, consists of all the predicates, functions, and symbols that are required to solve problems in the given domain (i.e., LISP programming), and (b) reasoning knowledge, which is procedural, consists of the knowledge about problem solving such as plans, rules in the given domain (i.e., in LISP).

The representation of conceptual knowledge in ELM-ART is more or less similar to the one in the KBS system. The conceptual network, claimed by the authors to be *domain independent*, is hierarchically organized into lessons, sections, sub-sections, and terminal pages (units). Each unit in the conceptual network is represented as an object containing slots for the content (e.g., text) of the page and the information for relating this page to other units, information required for interactive tests, and for programming problems.

The representation of reasoning knowledge in the system, which is *domain dependent*, is an important part to support problem solving by the student, explained in the next sub-sections. Problem-solving knowledge is represented as a complex network of concepts, plans, and rules, created by experts in the LISP programming language. The network contains information about plan transformations leading to semantically equivalent solutions and about rules describing different ways to solve a particular kind of problems. Additionally, there are bug rules describing both errors anticipated by experts in the domain and errors observed from students' interactions.

Learner modeling

Learner modeling in ELM-ART is also organized into two parts: (a) a multi-layered overlay model that stores the individual student's knowledge about learning concepts and terminal pages, and (b) an episodic learner model that registers all problem-solving sessions of the individual learner.

The overlay model in the ELM-ART system is similar to the one in the KBS system. The main difference is that in KBS the teacher updates the model for every learner whereas in ELM-ART there are two kinds of evaluations that may be performed during the learning process: (a) self-evaluation (the system presents the learner with his or her learning history and asks him or her to evaluate his or her knowledge about, e.g., a learning concept), and (b) evaluation by the system (after the student finishes a test, the system will update his or her knowledge about the learning concepts and terminal pages to which the test is related).

The episodic model consists of a collection of episodes that are descriptions of how programming problems have been solved by a particular learner. To construct the episodic model for the individual student, the code produced by the student is analyzed in terms of the domain knowledge on the one hand and a task description on the other hand. This diagnosis process results in a tree of concepts and rules the student might have used to solve the corresponding problem. These concepts and rules are instantiations of units from the knowledge base of the system. They are used to provide the learner with intelligent support showed in the next sub-section.

Adaptation support

ELM-ART effectively implements the next four adaptation techniques.

Adaptive presentation of learning contents. Similarly to the KBS system, this aspect is explicit in ELM-ART. The structure of the course (curriculum sequencing) and the navigation are adaptive to the student's current knowledge about the learning concepts and terminal pages of the given domain. The traffic-light metaphor is also applied in the system for adaptive navigation support.

Adaptive use of pedagogical devices. This aspect is also explicit in the system, as in KBS. The system encourages the learner to reuse the code of previously analyzed examples when solving a new problem. To support problem solving by a particular learner, the system can select the most helpful examples (for the current problem) from his or her learning history, sort them corresponding to their relevance, and present them to the learner as an ordered list of hypertext links (see "show example" at the bottom of Figure 5.3). The authors also consider this feature as support for problem solving by the student.

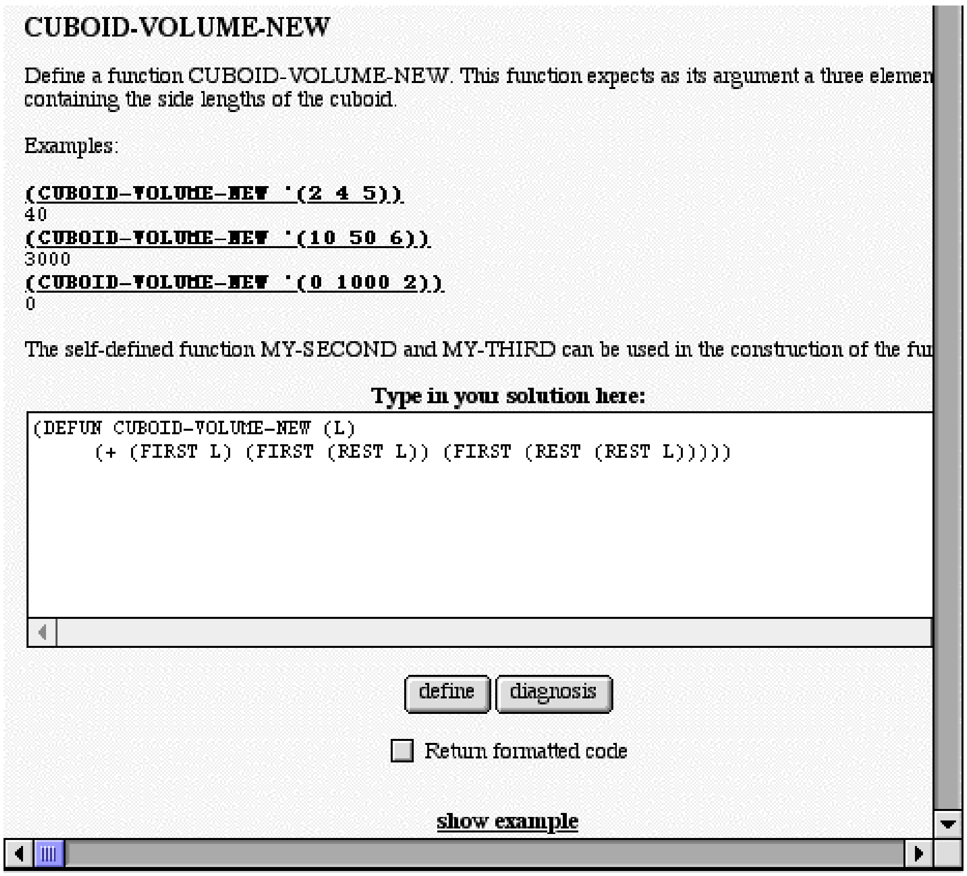
Adaptive problem-solving support. This should be the most interesting support of the system, as the authors claim that students may solve the problem without the help of the human teacher. Here is an example of interactions between a student (Alice) and the system during the session of solving a problem seen on the top of Figure 5.3:

- Alice introduces her solution to the given problem (see "Type in your solution here" on the middle of Figure 5.3).
- Alice clicks on the button "Define", showed on the bottom of Figure 5.3, to select an example call of her function with typical arguments. The evaluator window opens and

shows the result of the function call. Because the result is not as expected, Alice is encouraged to try to find out the error on her own.

- Because Alice cannot find out why her solution is incorrect, she clicks on the button "diagnosis", seen on the bottom of Figure 5.3, to ask the system to detect the error for her. The system uses its knowledge base including problem-solving knowledge and the episodic model constructed for Alice to analyze her LISP code: A wrong operator is found (+ instead of *). Then, the system formulates a sequence of help messages with increasingly detailed explanation of the error (see "Messages" on the middle of Figure 5.4), and the system sends it back to Alice. The sequence of messages starts with a very vague hint on what is wrong and ends with a code-level suggestion of how to correct the error.
- After examining one or several hints in the feedback provided by the system, Alice can correct and check again the solution.
- Alice can use this kind of help as many times as required to solve the problem correctly.

Figure 5.3. A page with a programming problem in ELM-ART



CUBOID-VOLUME-NEW

Define a function CUBOID-VOLUME-NEW. This function expects as its argument a three elemen
containing the side lengths of the cuboid.

Examples:

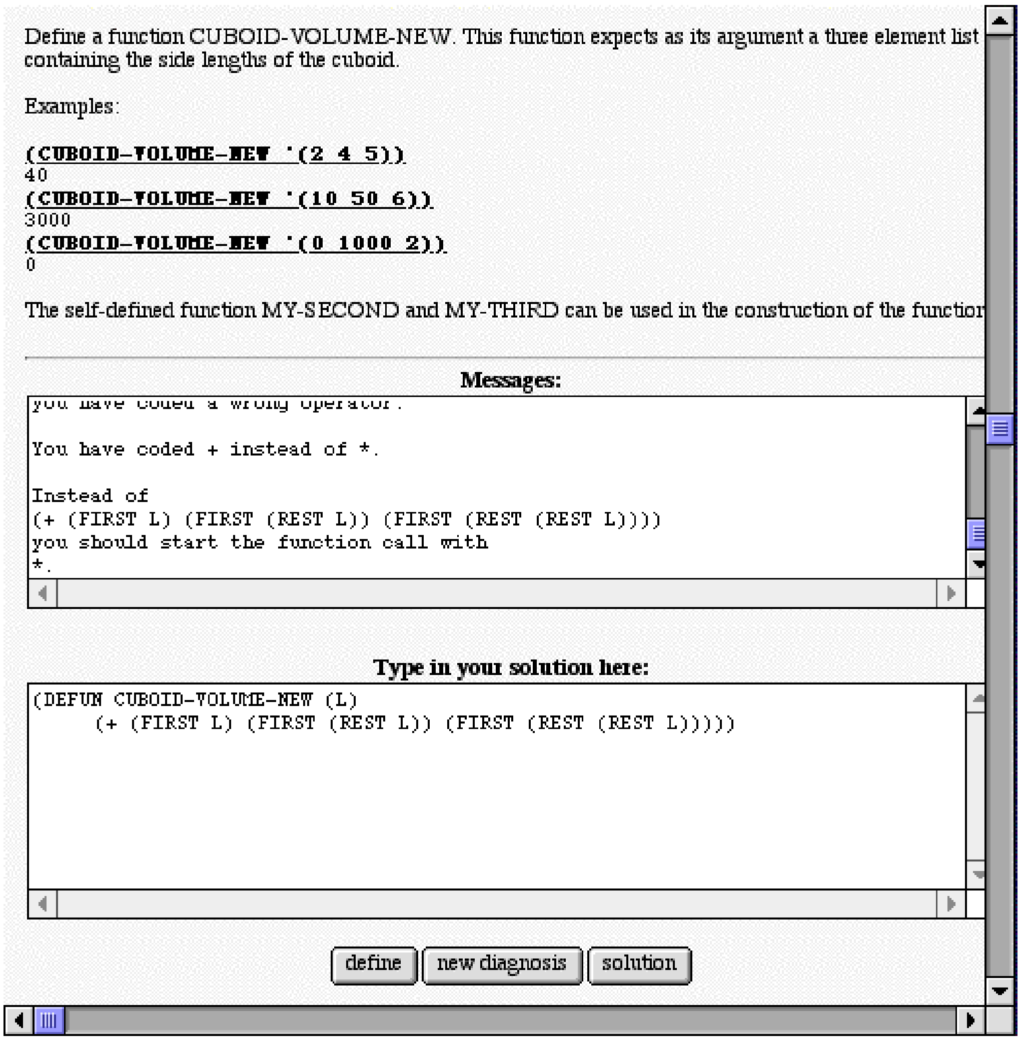
```
(CUBOID-VOLUME-NEW '(2 4 5))
40
(CUBOID-VOLUME-NEW '(10 50 6))
3000
(CUBOID-VOLUME-NEW '(0 1000 2))
0
```

The self-defined function MY-SECOND and MY-THIRD can be used in the construction of the fu

Type in your solution here:

```
(DEFUN CUBOID-VOLUME-NEW (L)
     (+ (FIRST L) (FIRST (REST L)) (FIRST (REST (REST L)))))
```

Figure 5.4. A diagnosis of an incorrect solution in ELM-ART


Define a function CUBOID-VOLUME-NEW. This function expects as its argument a three element list containing the side lengths of the cuboid.

Examples:

```
(CUBOID-VOLUME-NEW '(2 4 5))
40
(CUBOID-VOLUME-NEW '(10 50 6))
3000
(CUBOID-VOLUME-NEW '(0 1000 2))
0
```

The self-defined function MY-SECOND and MY-THIRD can be used in the construction of the functio

Messages:

```
you have coded a wrong operator.

You have coded + instead of *.

Instead of
(+ (FIRST L) (FIRST (REST L)) (FIRST (REST (REST L))))
you should start the function call with
*.
```

Type in your solution here:

```
(DEFUN CUBOID-VOLUME-NEW (L)
     (+ (FIRST L) (FIRST (REST L)) (FIRST (REST (REST L)))))
```

Adaptive assessment. This kind of adaptation is explicit in ELM-ART. At the end of each terminal page, sub-section, and section, and at the end of the course, the system presents the student with suitable tests, depending on his or her current knowledge about the domain as well as about problem-solving skills. For example, at the end of a terminal page, the system starts with showing a test item with medium difficulty. In case of an error made by the student, the system will randomly select another test with lower difficulty and present it to the student. In case of no error, the system will randomly select two tests with higher difficulty and show them to the student. When the system has enough "evidence" that the student has mastered the current learning concept, no further tests will be suggested to him or her. The learner, however, can continue working on tests by clicking on the link "more exercises" displayed with the feedback to the last answers.

Discussion

Several experiments (e.g., Weber & Brusilovsky, 2001) shows that ELM-ART is interesting in terms of its adaptability, especially the support for problem solving by the learner. ELM-ART has two main parts: a domain-dependent part and a domain-independent part. The latter is similar to the KBS's and could be reused. The former concerns knowledge for problem solving in the LISP language. *It took about 10 years for the authors to develop a powerful knowledge base for the support of problem solving in LISP. One must, however, start from scratch to build a similar knowledge base for other subjects than LISP programming.*

5.3.4 PHelpS: Adaptive peer help and collaboration

The PHelpS system has been developed, tested, and deployed in the context of the Correctional Services of Canada as a part of a staff training initiative. Almost 11000 workers in 281 different locations have been expected to make significant use of the system in their everyday activities. One of the goals of PHelpS is to facilitate peer help while workers do their tasks.

Course and learner modeling

"Course" modeling in the PHelpS system is simple. The main concept in the "course" provided by the system is tasks. Each task commonly undertaken in the system is represented as a hierarchical set of steps or subtasks.

User modeling in the system consists of two parts: (a) a personal profile describing personal characteristics of the worker such as age, gender, login status, linguistic fluency, the number of times the worker has provided help for the others, etc.; and (b) a multi-layered overlay model containing information that shows the tasks the worker can perform and the level of capability in carrying out each coarse or fine-grained step in these tasks. The system's belief about the worker's skill on a task or subtask is based on the number of times he or she has completed the task or subtask recently, the number of times he or she has given help (for other workers) on the task or subtask, the number of times this help was useful to the worker requesting help, the number of times this help was not useful, and so forth.

At the beginning of the course, for each worker, the system constitutes a personal profile and sets a default overlay knowledge model about the domain tasks (e.g., the worker cannot perform any task). During the training session, the user model is updated, as follows:

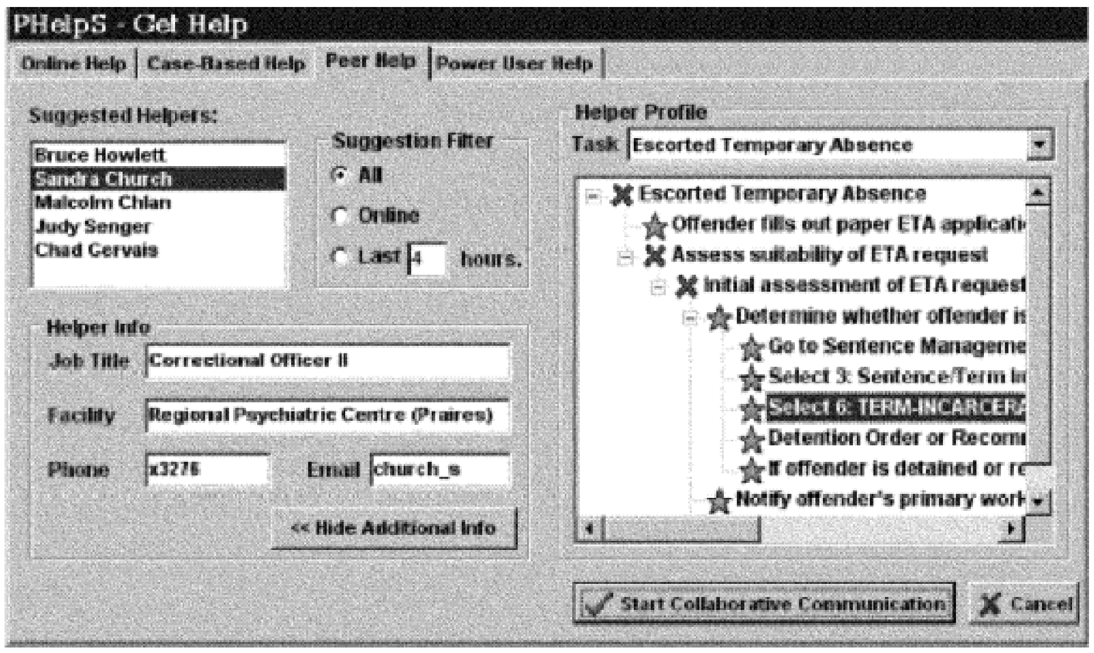
- For some personal information of the worker such as login status and the number of times the worker has provided help, the system can update it automatically.
- For the worker's knowledge about tasks, two kinds of evaluations may be performed: evaluation by the system and self-evaluation. Firstly, when the worker does a certain task, the system presents him or her with a checklist and encourages him or her to use this checklist while performing the task. The checklist contains every subtask of the task. On the basis of analyzing this checklist, the system can update the tasks or subtasks the worker can perform. The system can also automatically update several other features such that the number of times the worker has given help on a particular task or subtask. Secondly, workers can inspect and maintain themselves their own user model. For instance, for each step in each task, the worker can specify whether or not he or she can help on the step.

Adaptation support

The PHelpS system effectively implements adaptive communication support. To illustrate this kind of adaptation offered by the system, the authors showed the following simple working scenario:

- Assume a worker (Bob) using PHelpS reaches an impasse at a task step.
- Bob requests a peer helper (by clicking on a button provided by the system).
- The system searches a knowledge base (i.e., user models) for a set of potential peer helpers within the organization who: (a) are knowledgeable about the problem area of the specific task, (b) are available to provide help in the time frame required, (c) have not been overburdened with other help requests in the recent past, and (d) have other characteristics critical to a successful peer help session, for instance they speak the same language as Bob. The help request and these criteria form the inputs to a constraint solver embedded in the system. The solver produces a set of candidate peer helpers (see "Suggested Helpers" on the top right of Figure 5.5) ordered according to their suitability on these criteria.
- Bob selects his preferred peer helper from the candidate list (maybe after examining some information in the profile of the potential peer helpers). Once the helper is selected, a dialogue between Bob and the helper is begun (e.g., through telephone).

Figure 5.5. Peer helper suggestions for Bob by PHelpS



Suggested Helpers:
Bruce Howlett
Sandra Church
Malcolm Chlan
Judy Senger
Chad Gervais

Suggestion Filter: [✓ All] Online Last 4 hours.

Helper Info
Job Title: Correctional Officer II
Facility: Regional Psychiatric Centre (Praires)
Phone: x3276
Email: church_s

Helper Profile
Task: Escorted Temporary Absence

- Escorted Temporary Absence
  - Offender fills out paper ETA applicati
  - Assess suitability of ETA request
    - Initial assessment of ETA request
      - Determine whether offender is
        - Go to Sentence Manageme
        - Select 3: Sentence/Term In
        - Select 6: TERM-INCARCER
        - Detention Order or Recom
        - If offender is detained or r
      - Notify offender's primary work

Discussion

PHelpS successfully implements a kind of adaptation that the previous three systems do not: Adaptive peer help. A critical characteristic of PHelpS is that it has been tested in a real work place and that experimental result is encouraging. The adaptation techniques used in PHelpS could also be applied in any domain and other contexts (e.g., high schools). The system, however, concentrates on modeling only learning tasks. I believe that other aspects such as information resources, learning tools should also be taken into consideration, even in the context of work training.

5.3.5 Discussion of adaptive learning systems

Table 5.2 summarizes the previous analysis of adaptation support offered by several existing learning systems. From this table, I may conclude that none of these systems effectively implements all of the adaptation techniques I am concerned with in the present thesis. ELM-ART seems to be the most interesting learning system in terms of adaptation support, especially the adaptive problem-solving support it offers. It is worth, however, to note that the adaptation technique for problem-solving support is domain-dependent (the other four techniques can be implemented independently of the teaching domain, see Weber & Brusilovsky, 2001).

Table 5.2. Existing learning systems examples and support for adaptability

Existing learning systems	**Presentation of learningcon-tents**	**Use of pedagogical devices**	**Communication support**	**Problem-solving support**	**Assessment**
AHA (hypermedia course)	X				
KBS (Java course)	X	X			X
ELM-ART (LISP course)	X	X		X	X
PHelpS (work environment)			X		

Among the four analyzed systems, only KBS explicitly claims to support a learning theory (constructivism). It appears that few of existing learning systems effectively take into account both pedagogical principles implied from learning theories and adaptation techniques. I believe that the careful consideration of both aspects could be more effective in terms of learning outcomes than that of only one of them (Henze & Nejdl, 2001).

One should examine other analyses (e.g., Henze & Nejdl, 2001; Brusilovsky & Peylo, 2003) for more information about the adaptability issue.

Main result: Existing learning systems effectively implement the various adaptation techniques but none of them takes into account all techniques.

5.4 Conclusion

In this chapter, I presented two important analyses: an analysis of "constructivist" learning systems and an analysis of adaptive learning systems. The main objective of the two analyses is to know how researchers in the field have exploited ICT to foster cognitive flexibility and to implement adaptation support. Those analyses are useful for the construction of a new ICT-based learning environment (COFALE) presented in the next part, in which I try to satisfy all of the criteria for cognitive flexibility and to implement several basic adaptation techniques borrowed from existing systems.

Starting the construction of a complex learning system from scratch, however, should be a very hard work (Adaptive Technology Resource Center, 2004). Therefore, I decided to build the COFALE system on an existing learning content management system (LCMS). Among many open-source LCMSs, I selected ATutor (Adaptive Technology Resource Center, 2004) mainly because it is a good LCMS (see section 4.1) and it apparently makes it easy to create learning conditions exhibiting many desired characteristics of cognitive flexibility (see the evaluation of ATutor presented in section 5.2.4). I look further into the reasons for choosing ATutor in section 8.3.

PART THREE: COFALE: AN ADAPTIVE LEARNING ENVIRONMENT SUPPORTING COGNITIVE FLEXIBILITY

This part presents the extended work of the following paper:

Chieu, V.M., & Milgrom, E. (2005). COFALE: An adaptive learning environment supporting cognitive flexibility. *The Twelfth International Conference on Artificial Intelligence in Education*, Amsterdam, The Netherlands, 491–498 (full paper).

CHAPTER 6

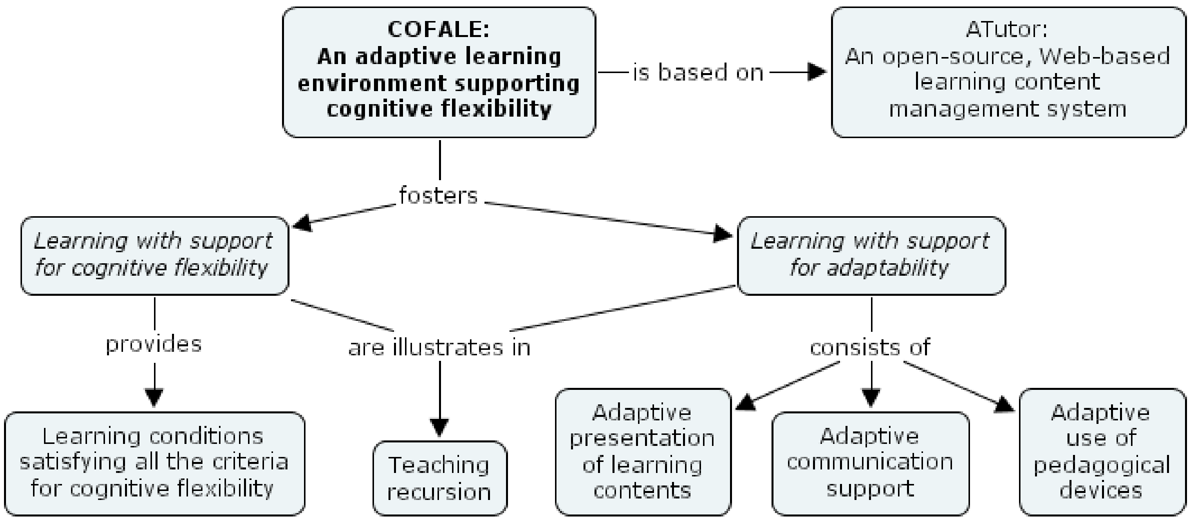
COFALE: Conditions of learning

"*In science the credit goes to the man who convinces the world, not to the man to whom the idea first occurs.*"

Francis Darwin, English Botanist, 1848 – 1925 (cited in Suomela, 2005)

(Reference to Appendices B & C)

In this chapter, I present COFALE, a new domain-independent e-Learning platform, and an example of its use. I argue that COFALE truly provides learners with personalized learning experiences that extensively facilitate and stimulate cognitive flexibility. After examining the presentation of COFALE in this chapter, one *could* understand that the operational criteria introduced in chapter 2 may be used as means of validation for the design of learning situations exhibiting the desired characteristics of cognitive flexibility.


COFALE: An adaptive learning environment supporting cognitive flexibility
is based on
ATutor: An open-source, Web-based learning content management system
fosters
Learning with support for cognitive flexibility
Learning with support for adaptability
provides
are illustrates in
consists of
Learning conditions satisfying all the criteria for cognitive flexibility
Teaching recursion
Adaptive presentation of learning contents
Adaptive communication support
Adaptive use of pedagogical devices

Summary

6.1 Introduction

The objective of this chapter and chapters 7 and 8 is to describe a new domain-independent e-Learning platform, named COFALE (cognitive flexibility in adaptive learning environments), in which I provide every learner with personalized learning situations that extensively support cognitive flexibility.

On the one hand, this chapter illustrates, in an ICT-based learning context, the learning conditions identified in chapter 3 for the learning of the concept of recursion (in the next chapter, I show how to implement those learning conditions in COFALE). Through demonstrating the learning process of a learner, I show that all of the criteria identified in chapter 2 are satisfied by means of learning situations proposed to the learner.

On the other hand, this chapter shows the adaptability of COFALE. A constructivist point of view on adaptation support and scaffolding (a particular kind of adaptability) was presented in section 1.5.4. To illustrate adaptation support in COFALE (i.e. scaffolding), I show how the course designer uses COFALE to provide different learning experiences for different kinds of students. The next chapter describes authoring tools provided for the course designer by COFALE to implement adaptation support.

In the following sections, I first identify different kinds of students in the context of learning recursion; then, I demonstrate students' learning process with support for cognitive flexibility and with support for adaptability in COFALE.

6.2 Mental models of recursion and adaptability

6.2.1 Mental models of recursion

In a constructivist point of view presented in section 1.5.3, I explained mental models (i.e., a mental representation or knowledge structure) as a critical characteristic of the learner. I also explained the need for providing appropriate conditions of learning, according to learners' mental models on the taught subject.

Several researchers (Anderson et al., 1988; Bhuiyan et al., 1994; Götschi et al., 2003) interviewed many students and analyzed students' tests on the subject of recursion. They distinguished four approaches that students try to apply to generate recursive solutions to a given problem:

1. *Loop model.* “Novice” students, when constructing a recursive solution, try to adapt some part of an iterative structure, for example the updating of loop index variables, in order to achieve recursion. That is why they often produce incorrect recursive solutions to a given problem. For instance, it may be impossible for this kind of students to solve the Towers of Hanoi puzzle presented in section 3.2.1.
2. *Syntactic model.* Students consider recursion as a template consisting of a base case and a recursive part. Although they may not fully understand the functionality of the recursive part, they are able to solve simple problems by filling the condition part and the action part of the base case and the recursive part. It should be easy for this type of learners to find out the base case of the Towers of Hanoi puzzle (where the number of disks is equal to 1). It may be, however, very difficult for them to find out the condition part and the action part of the recursive part for this problem.
3. *Analytic model.* Students consider recursion as a problem-solving technique. They analyze diverse cases of a given problem; then, for each case, they determine input conditions and output actions; finally, they write recursive code. Although these students try to analyze different cases of the Towers of Hanoi puzzle, they may not arrive at a recursive solution because they may not see recursion in this problem.
4. *Analysis-synthesis model.* “Expert” students, in addition to the ability implied by the analytic model, are able to apply the DCG (Divide, Conquer, and Glue) strategy to solve problems recursively: They break a large problem into one or more sub-problems that are identical in structure to the original problem and somewhat simpler to solve. It should be quite easy for these "expert" learners to build a recursive solution to the Towers of Hanoi puzzle by applying recursive thinking in this case.

The researchers also identified one approach that learners try to apply to verify the correctness of recursive solutions to a given problem: The *trace model* (students create mental images of control flow, particularly unraveling of recursive programs). It must be very hard for people to trace the recursive solution to the Towers of Hanoi puzzle.

From my point of view, each of these methods may be seen as defining the mental model of a learner getting acquainted with the concept and applications of recursion. For the purpose of the discussion in this part of the thesis, hereafter I distinguish only four kinds of mental models in learning recursion: loop, syntactic, analytic, and analysis-synthesis.

6.2.2 Adaptability

In chapter 1, I presented five principal adaptation techniques: (a) adaptive presentation of learning contents, (b) adaptive use of pedagogical devices, (c) adaptive communication support, (d) adaptive assessment, and (e) adaptive problem-solving support.

In section 5.3.5, I explained that only the first four adaptation techniques are domain-independent. In addition, the way to implement adaptive assessment is similar to the one to perform adaptive presentation of learning contents (I discuss this issue further in section 7.3.2). So, in the next section, I show how COFALE adapts the learning contents, pedagogical devices, and communication support to the different kinds of learners identified previously, in a manner consistent with the constructivist point of view presented in chapter 1.

6.3 COFALE as a learning environment

COFALE is an adaptive learning environment supporting cognitive flexibility. COFALE is based on ATutor, an open-source, Web-based learning content management system designed and maintained by ATRC (Adaptive Technology Resource Center, 2004).

For the purpose of the discussion, I shall assume that a “novice” learner (Bob), familiar with "traditional" programming (say, in the Java language) and thus with the concept of iterations, uses COFALE to learn recursion (i.e. to develop the ability to solve problems recursively); a tutor and a number of other learners (peers) also participate in the same learning experience.

In section 6.3.1, I show for each criterion for cognitive flexibility identified in chapter 2, how the course designer uses COFALE to present Bob with learning situations satisfying the corresponding criterion. Section 6.3.2 explains how the course designer uses COFALE to provide Bob and his peers with adaptation support.

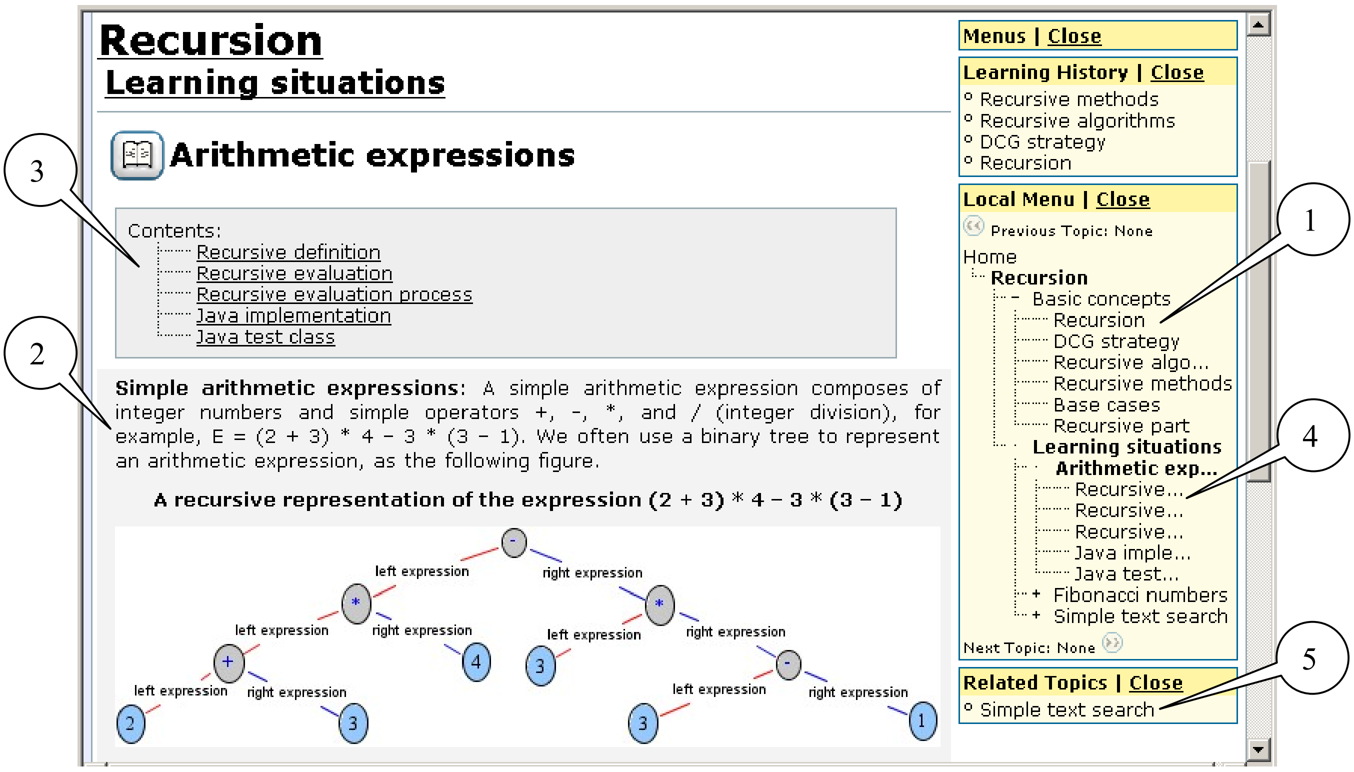
6.3.1 Learning with support for cognitive flexibility

Bob needs to develop his capacity to implement recursive solutions for a variety of problems. Navigating the "Local Menu" seen on the right hand side of Figure 6.1, Bob reads the definition and examples of the main concepts such as recursion, DCG strategy, recursive algorithms, and recursive methods (Figure 6.1: Area 1, see how the course designer prepared these main concepts in Activity 1.1, chapter 3). After that, Bob is encouraged to explore a situation about arithmetic expressions (Figure 6.1: Area 2). I show below, in the presentation for criterion MM2, how Bob is encouraged, in COFALE, to explore situations.

Criterion MM1: *The same learning content presenting concepts and their relationships is represented in different forms (e.g., text, images, audio, video, simulations).*

In the arithmetic expressions situation, the course designer induces Bob to examine multiple representations of recursion through the use of hyperlinks presented in Area 3 or in Area 4 of Figure 6.1: a textual definition, two simulations, and a Java implementation. For instance, Figure 6.1a shows the textual definition and Figure 6.1b illustrates the textual simulation.

Figure 6.1. A part of Bob's learning hyperspace in COFALE



Recursion
Learning situations

Arithmetic expressions

Contents:
- Recursive definition
- Recursive evaluation
- Recursive evaluation process
- Java implementation
- Java test class

Simple arithmetic expressions: A simple arithmetic expression composes of integer numbers and simple operators +, -, *, and / (integer division), for example, E = (2 + 3) * 4 – 3 * (3 – 1). We often use a binary tree to represent an arithmetic expression, as the following figure.

A recursive representation of the expression (2 + 3) * 4 – 3 * (3 – 1)

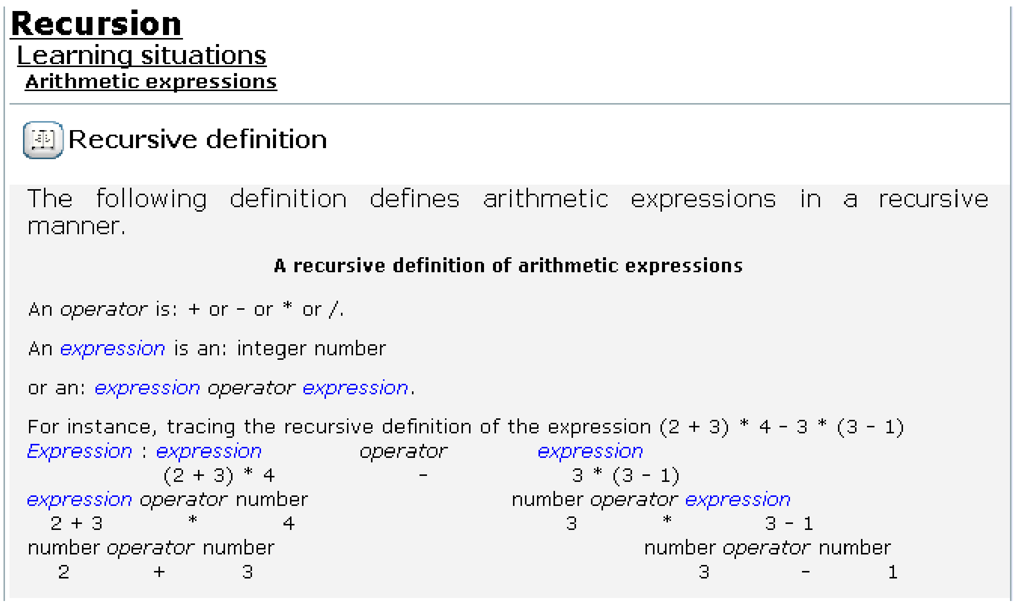
Menus | Close

Learning History | Close
- Recursive methods
- Recursive algorithms
- DCG strategy
- Recursion

Local Menu | Close
Previous Topic: None
```
Home
 Recursion
   - Basic concepts
       Recursion
       DCG strategy
       Recursive algo...
       Recursive methods
       Base cases
       Recursive part
     Learning situations
       Arithmetic exp...
         Recursive...
         Recursive...
         Recursive...
         Java imple...
         Java test...
       + Fibonacci numbers
       + Simple text search
```
Next Topic: None

Related Topics | Close
- Simple text search

Figure 6.1a. A textual definition of arithmetic expressions



Recursion
Learning situations
Arithmetic expressions

Recursive definition

The following definition defines arithmetic expressions in a recursive manner.

A recursive definition of arithmetic expressions

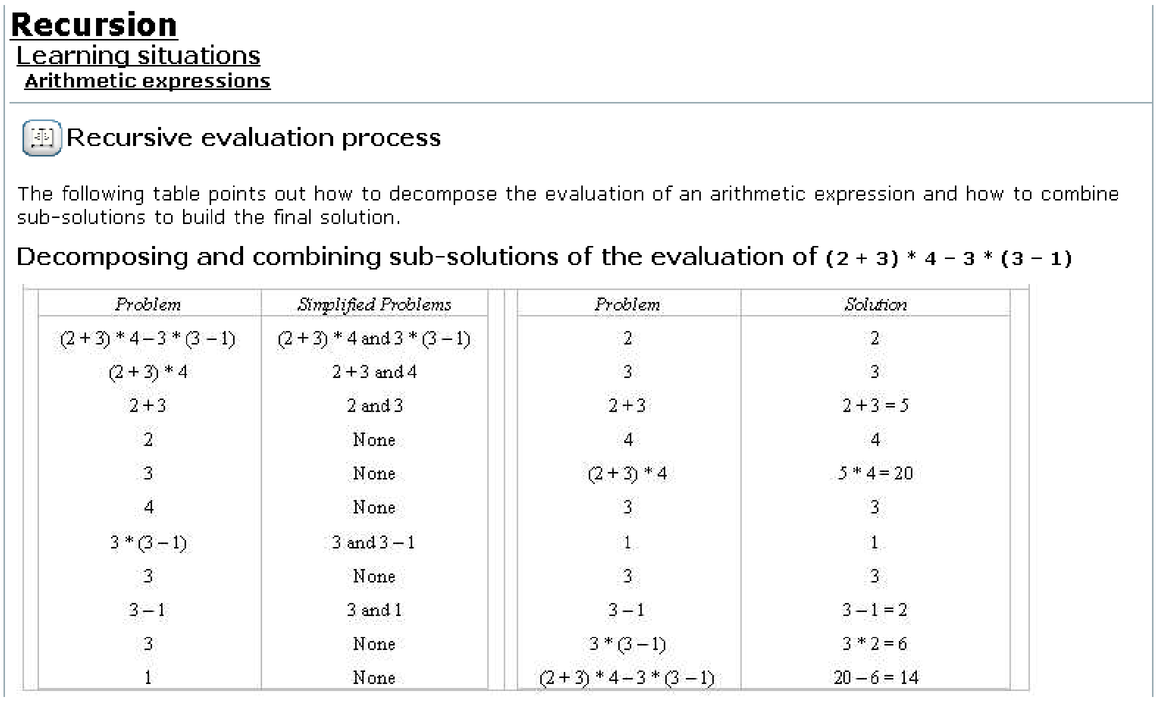
An *operator* is: + or - or * or /.

An *expression* is an: integer number

or an: *expression operator expression*.

For instance, tracing the recursive definition of the expression (2 + 3) * 4 - 3 * (3 - 1)
```
Expression : expression          operator          expression
                (2 + 3) * 4          -                3 * (3 - 1)
expression operator number                     number operator expression
   2 + 3        *       4                         3        *        3 - 1
number operator number                                  number operator number
    2         +       3                                    3          -        1
```

Figure 6.1b. A textual simulation for arithmetic expressions



Recursion

Learning situations

Arithmetic expressions

Recursive evaluation process

The following table points out how to decompose the evaluation of an arithmetic expression and how to combine sub-solutions to build the final solution.

Decomposing and combining sub-solutions of the evaluation of (2 + 3) * 4 – 3 * (3 – 1)

Problem	*Simplified Problems*	*Problem*	*Solution*
(2 + 3) * 4 – 3 * (3 – 1)	(2 + 3) * 4 and 3 * (3 – 1)	2	2
(2 + 3) * 4	2 + 3 and 4	3	3
2 + 3	2 and 3	2 + 3	2 + 3 = 5
2	None	4	4
3	None	(2 + 3) * 4	5 * 4 = 20
4	None	3	3
3 * (3 – 1)	3 and 3 – 1	1	1
3	None	3	3
3 – 1	3 and 1	3 – 1	3 – 1 = 2
3	None	3 * (3 – 1)	3 * 2 = 6
1	None	(2 + 3) * 4 – 3 * (3 – 1)	20 – 6 = 14

<u>Criterion MP1</u>: *The same abstract concept is explained, used, and applied systematically with other concepts in a diversity of examples of use, exercises, and case studies in complex, realistic, and relevant situations*.

After exploring the first situation (i.e., arithmetic expressions), Bob is encouraged to explore another one: "Simple text search" through the menu "Related Topics" offered by ATutor, thus also by COFALE (Figure 6.1: Area 5). In this situation, Bob sees how to apply recursion to represent a text (i.e. a list of words) as a linked list and to look up a phrase in a document.

The reader should refer back to Activities 1.2 and 2.1 in chapter 3 to understand how and why the course designer created arithmetic expressions and simple text search and presented them to the learner. In section 7.2.1, I illustrate a set of instructor tools for creating the learning content in hypermedia form.

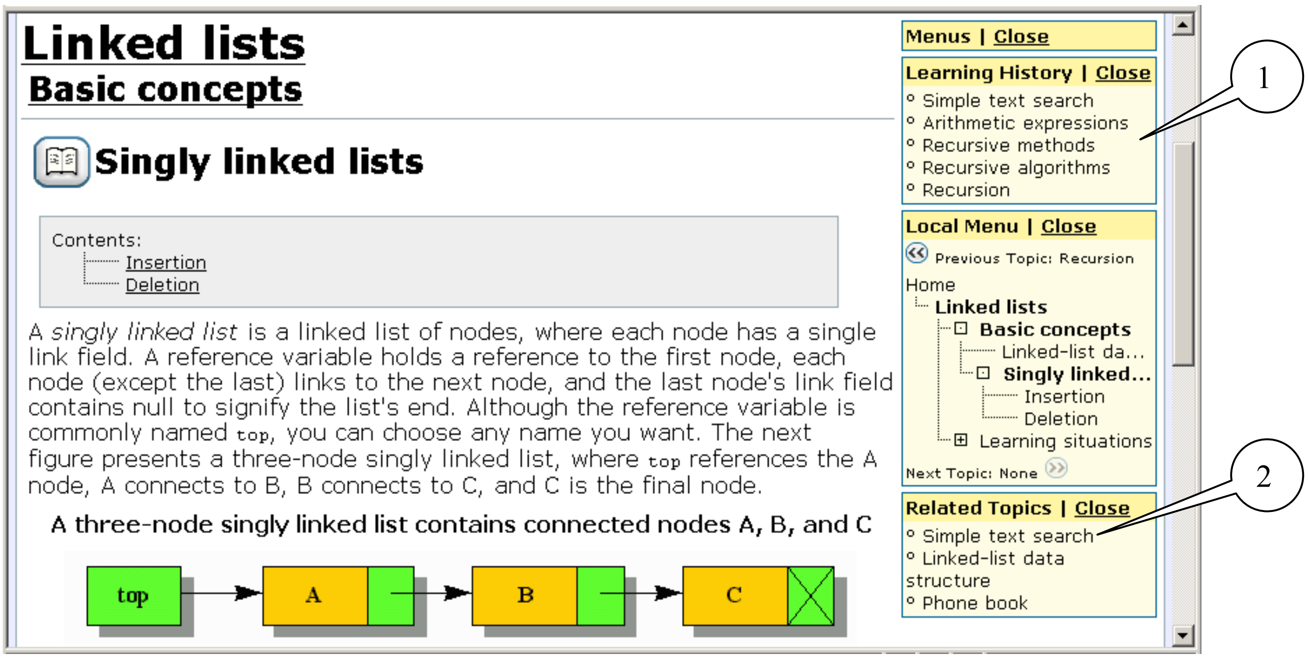
<u>Criterion MP2</u>: *When facing a new concept, learners are encouraged to explore the relationships between this concept and other ones as far as possible in complex, realistic, and relevant situations*.

When Bob explores simple text search, COFALE presents a hyperlink encouraging Bob to examine the *related* concept "linked lists". This concept is related to the concept of recursion because linked lists are a particular type of recursive data structures (see also Appendices B3 and B4). Similarly, while exploring this concept (Figure 6.2), Bob could return to the recursion hyperspace by using one of the hyperlinks presented in "Related Topics" (Figure 6.2: Area 2) and "Learning History" (Figure 6.2: Area 1). The latter contains the hyperlinks of Bob's recently visited content pages, those hyperlinks are automatically generated by

COFALE. The two menus (Figure 6.2: Related Topics and Learning History) also help Bob navigate intelligently to avoid getting lost in the learning hyperspace.

Activities 1.3 and 2.2 in chapter 3 explained how the course designer satisfied criterion MP2. In addition, in COFALE the course designer has had to define, for every discrete piece of learning content (page), the other pages related to that one (section 7.2.1 presents an authoring tool to do so); for example, simple text search related to arithmetic expressions, linked lists related to simple text search. On the basis of those associations, COFALE automatically generates the hyperlinks in "Related Topics" (Figures 6.1 and 6.2).

Figure 6.2. Bob's learning hyperspace about linked lists in COFALE



Linked lists

Basic concepts

Singly linked lists

Contents:
- Insertion
- Deletion

A *singly linked list* is a linked list of nodes, where each node has a single link field. A reference variable holds a reference to the first node, each node (except the last) links to the next node, and the last node's link field contains null to signify the list's end. Although the reference variable is commonly named `top`, you can choose any name you want. The next figure presents a three-node singly linked list, where `top` references the A node, A connects to B, B connects to C, and C is the final node.

A three-node singly linked list contains connected nodes A, B, and C

Menus | Close

Learning History | Close
- Simple text search
- Arithmetic expressions
- Recursive methods
- Recursive algorithms
- Recursion

Local Menu | Close

Previous Topic: Recursion

Home
- Linked lists
  - Basic concepts
    - Linked-list da...
    - Singly linked...
      - Insertion
      - Deletion
  - Learning situations

Next Topic: None

Related Topics | Close
- Simple text search
- Linked-list data structure
- Phone book

Criterion MM2: *Learners are encouraged to study the same abstract concept for different purposes, at different times, by different methods including different activities (reading, exploring, discussion, knowledge reorganization, etc.).*

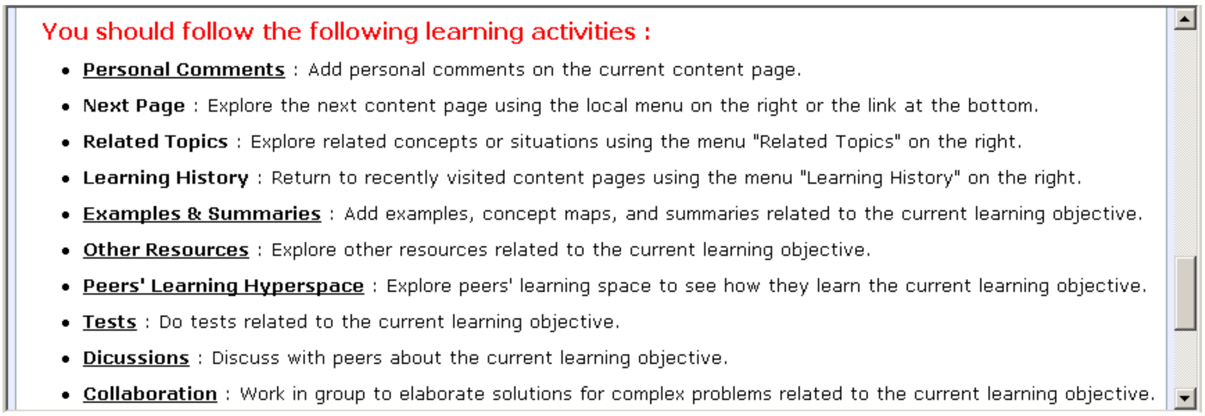
At the bottom of each content page, COFALE presents Bob with learning activities to guide and encourage him in the exploration of the learning hyperspace. For instance, after exploring arithmetic expressions, Bob is led to multiple activities in different contexts to look further into recursion (Figure 6.3).

To satisfy criterion MM2, the course designer has defined, for each content page (e.g. "Java test class", the last item of arithmetic expressions in Figure 6.1), the learning activities related to that content page (e.g. the 10 activities shown in Figure 6.3). To help the course designer in this work, COFALE supports a set of predefined learning activities and an authoring tool (see section 7.2.2). One should also see Activity 2.1 (shown in chapter 3), which provided a similar example of this teaching activity.

Note that presenting 10 activities (Figure 6.3) is only an example of use in COFALE. If one takes into account cognitive load in instructional design (Kirsh, 2000; Sweller, 2005), one should present the student with only several activities at a given time (showing 10 activities at the same time may be too much).

Criterion MP3: When facing a new concept, learners are encouraged to explore different interpretations of this concept (by other authors and by peers), to express their personal point of view on the new concept, and to give feedback on the points of view of other people.

Figure 6.3. Learning activities proposed to Bob by COFALE



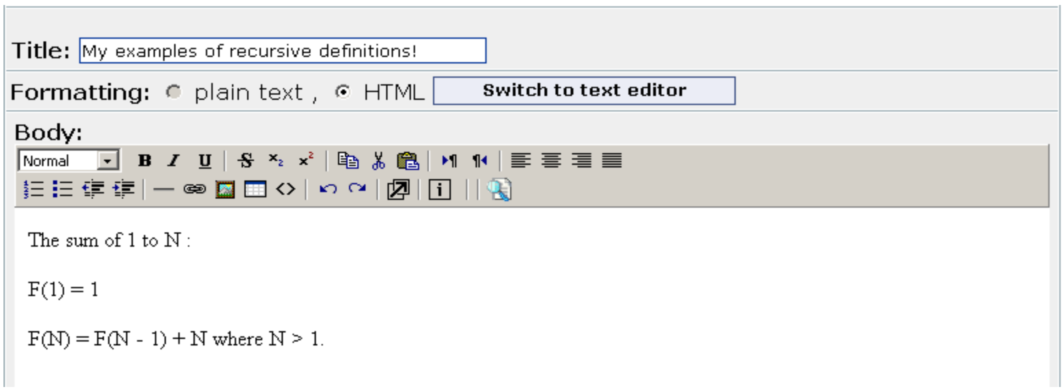
You should follow the following learning activities :

- **Personal Comments** : Add personal comments on the current content page.
- **Next Page** : Explore the next content page using the local menu on the right or the link at the bottom.
- **Related Topics** : Explore related concepts or situations using the menu "Related Topics" on the right.
- **Learning History** : Return to recently visited content pages using the menu "Learning History" on the right.
- **Examples & Summaries** : Add examples, concept maps, and summaries related to the current learning objective.
- **Other Resources** : Explore other resources related to the current learning objective.
- **Peers' Learning Hyperspace** : Explore peers' learning space to see how they learn the current learning objective.
- **Tests** : Do tests related to the current learning objective.
- **Dicussions** : Discuss with peers about the current learning objective.
- **Collaboration** : Work in group to elaborate solutions for complex problems related to the current learning objective.

To satisfy this criterion, COFALE engages Bob in four learning activities:

1. Add comments on the learning content proposed by the course designer, for example reformulate the main points of the definition of recursion (Figure 6.3: Personal Comments).
2. Add his own examples, for instance a recursive phenomenon in his life (Figure 6.3: Examples & Summaries). Figure 6.4 shows a Web tool allowing Bob to add his recursive examples in the form of HTML to his learning hyperspace.
3. Explore external resources, for instance the online Java tutorials (Eck, 2004; Kjell, 2003) in which the author illustrates a great number of recursive examples (Figure 6.3: Other Resources). Bob could also ask the tutor to approve a new resource he has found (see section 7.2.1).
4. Explore peers' learning spaces, for example log into the learning hyperspace of an "expert" to see her own recursive examples (Figure 6.3: Peers' Learning Hyperspace).

Figure 6.4. Tool provided by COFALE for Bob to add his own examples



Title: My examples of recursive definitions!

Formatting: plain text , [✓ HTML] Switch to text editor

Body:

The sum of 1 to N :

F(1) = 1

F(N) = F(N - 1) + N where N > 1.

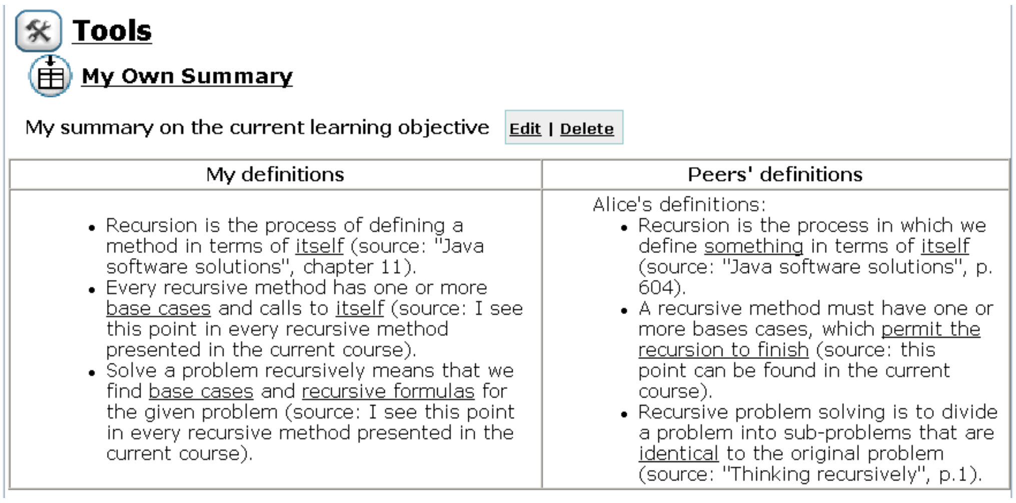
To support the third activity, the course designer has had to prepare external resources (see Activity 1.7 in chapter 3); section 7.2.1 explains how COFALE helps the course designer introduce those resources. The other three activities are supported by COFALE without explicit intervention of the course designer. Those learning activities are proposed to the learner on the basis of Activity 2.3 presented in chapter 3.

There would be a problem of privacy in the last activity. The current version of COFALE allows Bob only to explore (not modify) the learning hyperspace (not the personal profile) of peers. For future research, one can allow the learner to decide whether or not a peer can visit his or her learning hyperspace, if yes, which kinds of information could be public.

<u>Criterion MP4</u>: *When facing a new concept, learners are encouraged to examine, analyze, and synthesize a diversity of points of view on the new concept.*

To satisfy this criterion, COFALE engages Bob to produce summaries of the points of view of other sources and peers (Figure 6.3: Examples & Summaries). For instance, COFALE provides Bob with an empty table so that he can state his own definitions of recursion, recursive methods, and recursive problem solving, together with peers' (Figure 6.5). COFALE supports this activity without intervention of the course designer. This activity is proposed to the leaner by considering Activity 2.4, shown in chapter 3.

Figure 6.5. Tool provided by COFALE for Bob to produce summaries



My summary on the current learning objective

My definitions	Peers' definitions
• Recursion is the process of defining a method in terms of itself (source: "Java software solutions", chapter 11). • Every recursive method has one or more base cases and calls to itself (source: I see this point in every recursive method presented in the current course). • Solve a problem recursively means that we find base cases and recursive formulas for the given problem (source: I see this point in every recursive method presented in the current course).	Alice's definitions: • Recursion is the process in which we define something in terms of itself (source: "Java software solutions", p. 604). • A recursive method must have one or more bases cases, which permit the recursion to finish (source: this point can be found in the current course). • Recursive problem solving is to divide a problem into sub-problems that are identical to the original problem (source: "Thinking recursively", p.1).

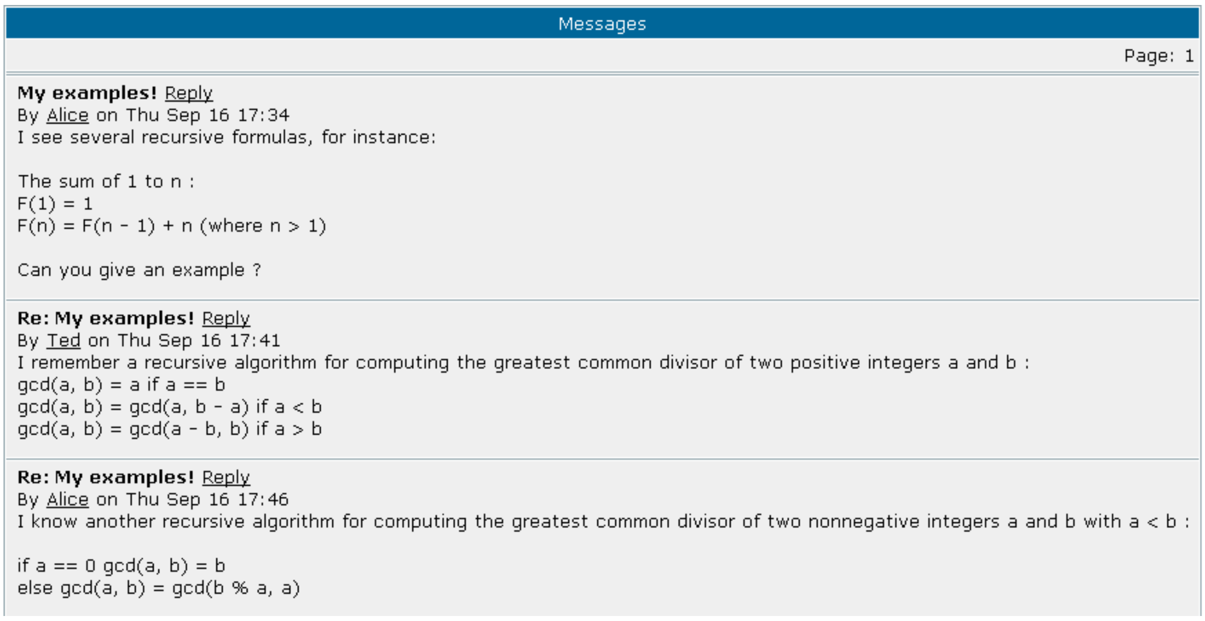
<u>Criterion MM3</u>: *The number of participants, the type of participant (learner, tutor, expert, etc.), the communication tools (e-mail, mailing lists, face to face, chat rooms, video conferencing, etc.), and the location (in the classroom, on campus, anywhere in the world, etc.) are varied.*

To satisfy this criterion, COFALE encourages Bob to work with others (Figure 6.3: Discussions), sometimes with the participation of the tutor, by using multiple communication tools supported by ATutor – thus also by COFALE – such as e-mail, forums, chat rooms. COFALE also incites Bob to use a Q&A website (Java World, 2004) introduced by the course designer (see Activity 1.5 in chapter 3) to ask experts questions about recursion. The platform supports multiple communication tools, but to engage learners to use them, the course designer has created a forum and invited Bob and his peers to confront and discuss their recursive examples that they have encountered in their everyday life (Figure 6.6a). Activity 2.5 in chapter 3 is taken into account to propose those learning activities to the student.

Criterion MP5: During the discussion, learners are encouraged to diversify – as far as possible – the different points of view about the topic discussed.

To satisfy this criterion, COFALE presents two dropdown lists of general and domain-specific questions (Figure 6.6b: Areas 1 & 2) that Bob could use to elicit peers' point of view. For instance, when Bob sees an example or solution proposed by a peer, Bob can select the question "What was your source of information?" from the list (Figure 6.6b: Area 1) to ask the peer to justify the solution (in Figure 6.6b, the question Bob selects is automatically inserted in textbox "Body").

Figure 6.6a. A learning forum created by the course designer in COFALE


Messages

Page: 1

My examples! Reply
By Alice on Thu Sep 16 17:34
I see several recursive formulas, for instance:

The sum of 1 to n :
F(1) = 1
F(n) = F(n - 1) + n (where n > 1)

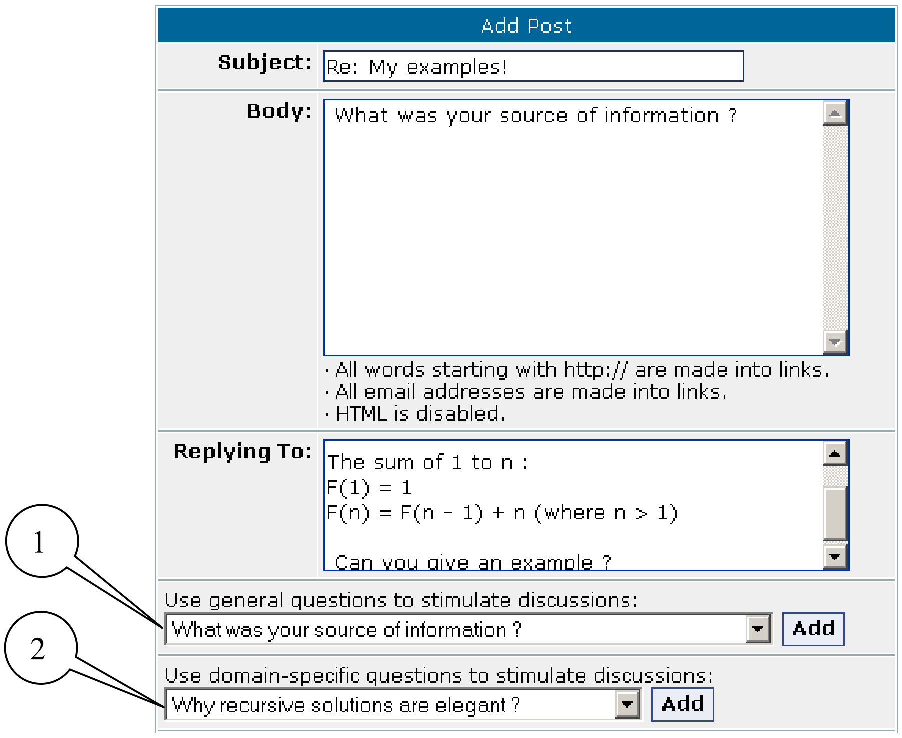
Can you give an example ?

Re: My examples! Reply
By Ted on Thu Sep 16 17:41
I remember a recursive algorithm for computing the greatest common divisor of two positive integers a and b :
gcd(a, b) = a if a == b
gcd(a, b) = gcd(a, b - a) if a < b
gcd(a, b) = gcd(a - b, b) if a > b

Re: My examples! Reply
By Alice on Thu Sep 16 17:46
I know another recursive algorithm for computing the greatest common divisor of two nonnegative integers a and b with a < b :

if a == 0 gcd(a, b) = b
else gcd(a, b) = gcd(b % a, a)

Figure 6.6b. Tool provided by COFALE for Bob to reply peers' messages


Add Post

Subject: Re: My examples!

Body: What was your source of information ?

· All words starting with http:// are made into links.
· All email addresses are made into links.
· HTML is disabled.

Replying To:
The sum of 1 to n :
F(1) = 1
F(n) = F(n - 1) + n (where n > 1)

Can you give an example ?

Use general questions to stimulate discussions:
What was your source of information ? Add

Use domain-specific questions to stimulate discussions:
Why recursive solutions are elegant ? Add

The course designer is asked to prepare a list of general questions and a list of domain-specific questions (see Activity 1.6 in chapter 3). COFALE supports a list of predefined general questions proposed by researchers in pedagogy (Appendix B5). Section 7.2.1 describes how the course designer uses COFALE to make those lists available for students. Activity 2.6 in chapter 3 is considered to propose those tools to the learner.

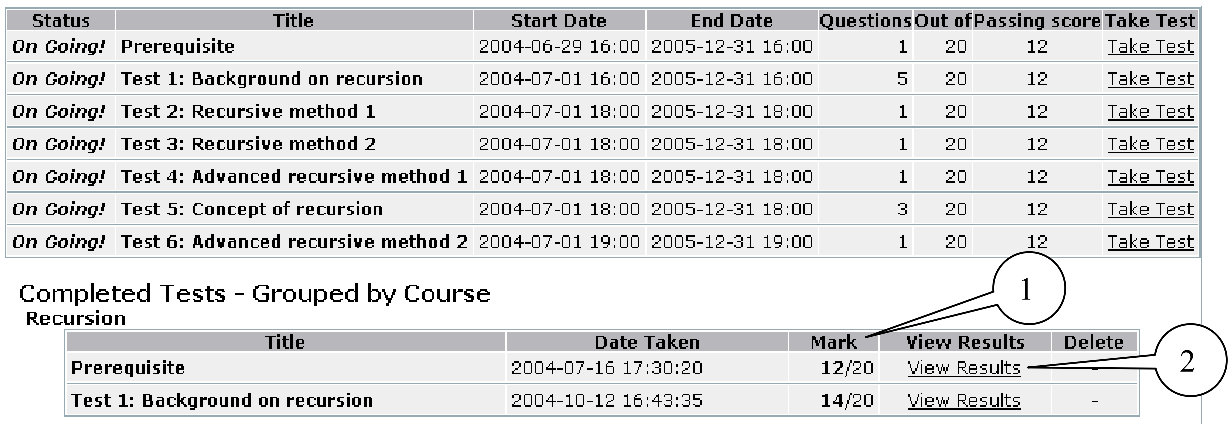
Criterion MM4: *During the learning process, learners are encouraged to use different assessment methods and tools, at different times, and in different contexts for demonstrating their ability to solve different problems.*

At different points in time, for example, after exploring multiple learning situations or after discussing with peers, Bob is engaged in two assessment activities:

1. *Individual tests (Figure 6.3: Tests)*. Bob confronts a robot situation (Appendix C) in which the course designer has Bob take a number of tests (Figure 6.7) such that computing the number of ways the robot can walk n meters and listing all the ways the robot can walk n meters where n is a positive integer. In Figure 6.7, the upper table presents the tests prepared by the course designer, and the lower one presents Bob's taken tests and the scores marked by the tutor for each of these taken tests (Figure 6.7: Area 1). Clicking on the hyperlink "View Results" attached to each taken test (Figure 6.7: Area 2), Bob can see the feedback provided by the tutor for the corresponding taken test.
2. *Work in small group (Figure 6.3: Collaboration)*. The course designer engages Bob in a small group (2 or 3 learners) to solve complex problems in a tree-structured file system (Appendix C), for instance listing all files and sub-directories in a given directory. To support this assessment activity, ATutor (thus also COFALE) provides Bob and his peers with a collaboration hyperspace (Figure 6.8): The learners can use the resources provided by the course designer (Figure 6.8: Area 1) to solve the given problem. They can submit their own solutions to a shared place (Figure 6.8: Area 2). They can use multiple communication tools, shown on the right hand side of Figure 6.8, to run diverse discussions with members in their group.

Activities 1.4, 2.7, and 2.8 presented in chapter 3 explained how the course designer satisfied criterion MM4. In section 7.2.1, I describe the test manager and the collaboration manager provided for the course designer by COFALE.

Figure 6.7. Individual tests proposed to Bob by COFALE



Status	Title	Start Date	End Date	Questions	Out of	Passing score	Take Test
On Going!	Prerequisite	2004-06-29 16:00	2005-12-31 16:00	1	20	12	Take Test
On Going!	Test 1: Background on recursion	2004-07-01 16:00	2005-12-31 16:00	5	20	12	Take Test
On Going!	Test 2: Recursive method 1	2004-07-01 18:00	2005-12-31 18:00	1	20	12	Take Test
On Going!	Test 3: Recursive method 2	2004-07-01 18:00	2005-12-31 18:00	1	20	12	Take Test
On Going!	Test 4: Advanced recursive method 1	2004-07-01 18:00	2005-12-31 18:00	1	20	12	Take Test
On Going!	Test 5: Concept of recursion	2004-07-01 18:00	2005-12-31 18:00	3	20	12	Take Test
On Going!	Test 6: Advanced recursive method 2	2004-07-01 19:00	2005-12-31 19:00	1	20	12	Take Test

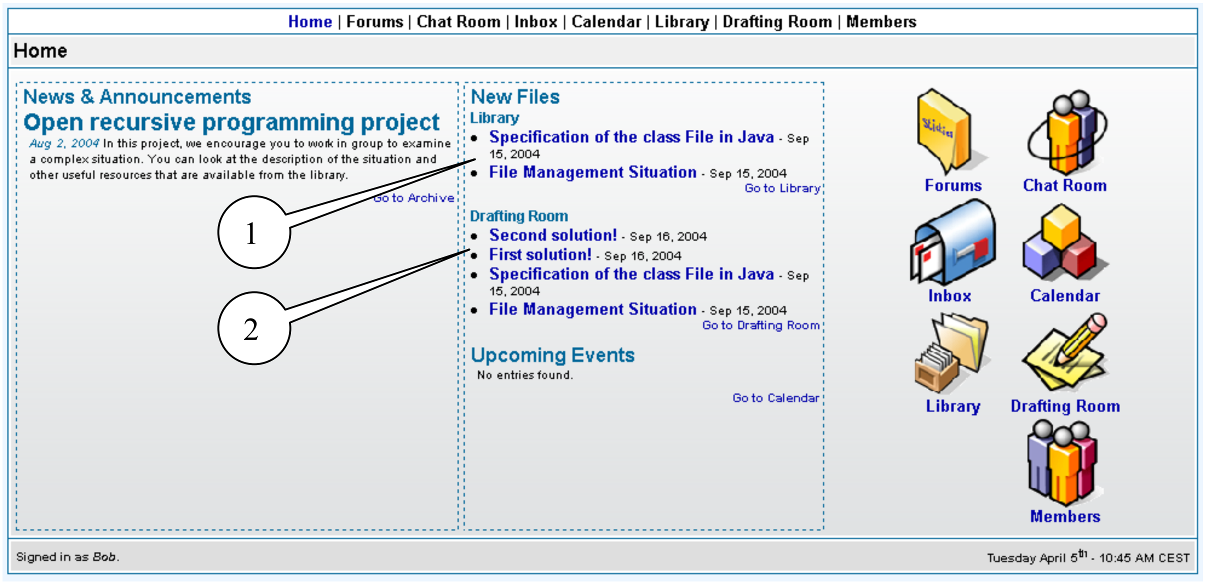
Completed Tests - Grouped by Course

Recursion

Title	Date Taken	Mark	View Results	Delete
Prerequisite	2004-07-16 17:30:20	**12**/20	View Results	-
Test 1: Background on recursion	2004-10-12 16:43:35	**14**/20	View Results	-

<u>Criterion MP6</u>: *During the problem-solving process, learners are encouraged to confront multiple ways to solve the problem and multiple possible solutions to the problem.*

Figure 6.8. Collaboration hyperspace proposed to Bob and his peers by COFALE



Home | Forums | Chat Room | Inbox | Calendar | Library | Drafting Room | Members

Home

News & Announcements

Open recursive programming project

Aug 2, 2004 In this project, we encourage you to work in group to examine a complex situation. You can look at the description of the situation and other useful resources that are available from the library.

Go to Archive

New Files

Library

- **Specification of the class File in Java** - Sep 15, 2004
- **File Management Situation** - Sep 15, 2004

Go to Library

Drafting Room

- **Second solution!** - Sep 16, 2004
- **First solution!** - Sep 16, 2004
- **Specification of the class File in Java** - Sep 15, 2004
- **File Management Situation** - Sep 15, 2004

Go to Drafting Room

Upcoming Events

No entries found.

Go to Calendar

Signed in as *Bob*.

Tuesday April 5th - 10:45 AM CEST

In the robot situation, to compute the number of ways the robot can walk n meters (see Appendix C), Bob is encouraged to use and compare both the iterative method and the recursive one. In the file management (see Appendix C), the course designer exhorts Bob and his peers to confront and compare different solutions. For example, in the "Drafting Room" (Figure 6.8: Area 2), Bob and Alice propose two different solutions to the given problems: Bob first lists the files and sub-directories in the given directory, then in its sub-directories, and Alice first lists the files and sub-directories in the sub-directories of the given directory, then in the given directory. They can use a domain-specific tool, JDiff in jEdit (2005), to help them find out the difference between the two Java implementations. JDiff is rather simple, it helps detecting the textual difference rather than the actual difference between two Java programs; so, for future research, one can search for tools that are more effective than JDiff.

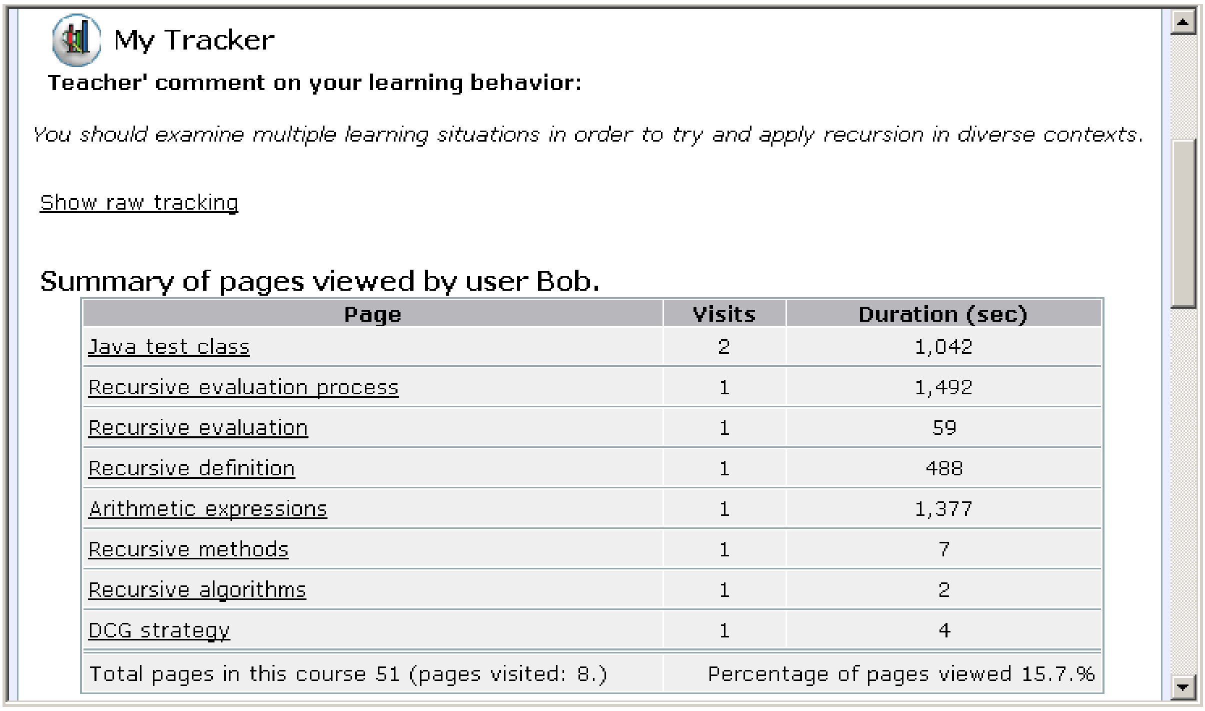
Activity 1.4 shown in chapter 3 explained how and why the course designer prepared the previous assessment situations to satisfy criterion MP6.

In addition to the previous learning situations proposed to Bob, at any time Bob may review his learning behavior or navigation history, supported by ATutor, thus also by COFALE (Figure 6.9). For instance, after exploring arithmetic expressions, Bob can look again at the content pages he has viewed, the number of visits for each content page, and the total time he has used for each content page (see the table shown on Figure 6.9).

Bob can also see the tutor's feedback (seen on the top of Figure 6.9) on his learning behavior with respect to cognitive flexibility. To give Bob feedback on his learning behavior, the tutor first logs into his learning hyperspace as a peer (Figure 6.3: Peers' Learning Hyperspace). Then, the tutor examines Bob' navigation history (Figure 6.9) to see how Bob has learned the concept of recursion. Finally, the tutor uses a simple text editor supported by

COFALE to give comments to Bob. Activities 2.9 and 3.1 in chapter 3 are considered for this evaluation activity.

Figure 6.9. Bob's navigation history registered by COFALE



My Tracker

Teacher' comment on your learning behavior:

You should examine multiple learning situations in order to try and apply recursion in diverse contexts.

Show raw tracking

Summary of pages viewed by user Bob.

Page	Visits	Duration (sec)
Java test class	2	1,042
Recursive evaluation process	1	1,492
Recursive evaluation	1	59
Recursive definition	1	488
Arithmetic expressions	1	1,377
Recursive methods	1	7
Recursive algorithms	1	2
DCG strategy	1	4
Total pages in this course 51 (pages visited: 8.)	Percentage of pages viewed 15.7.%	

Main result: COFALE provides the learner with learning conditions satisfying all the criteria for cognitive flexibility.

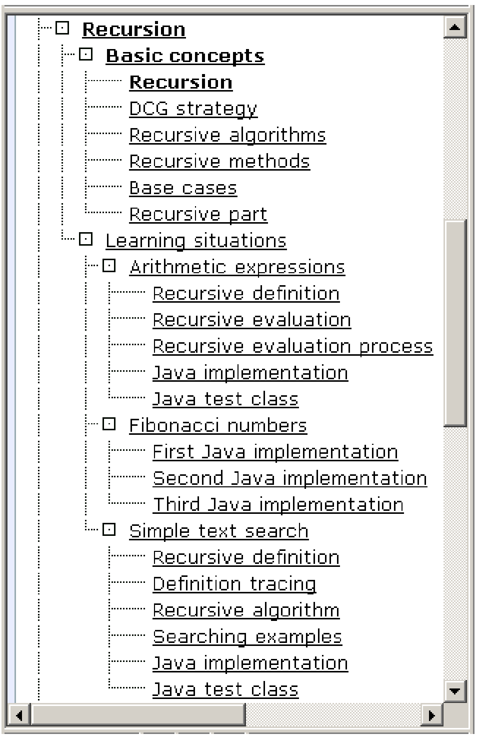
6.3.2 Learning with support for adaptability

I shall now assume that two other learners (Ted and Alice, both at the "expert" level) are active in the course: They are well versed in the use of COFALE and they have reached the analysis-synthesis model of the recursion concept. I now describe how COFALE adapts the learning contents, pedagogical devices, and communication support to the specific needs of Bob, Ted, and Alice.

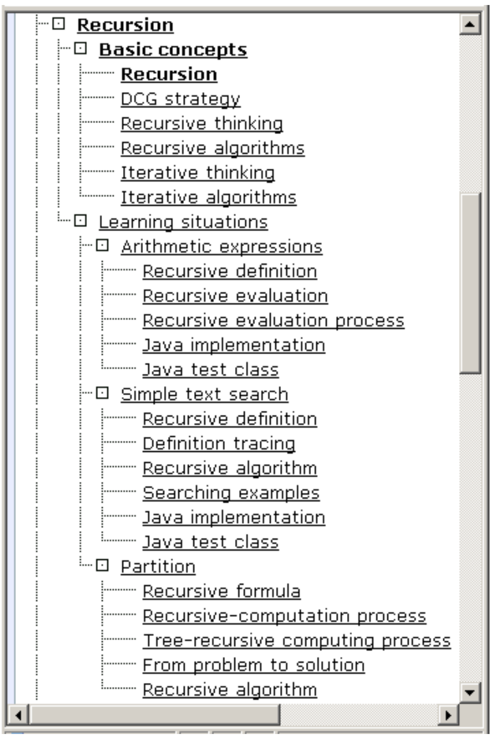
Learning contents. COFALE presents each learner with different content pages (Figure 6.10). For example, because Bob is "novice" and Alice is "expert", COFALE introduces simpler concepts and propose simpler situations to Bob (e.g., Appendix B1: recursive methods, base cases, recursive part; Appendix B2: Fibonacci numbers) than to Alice (e.g., Appendix B1: recursive thinking, iterative thinking; Appendix B2: partition).

To allow COFALE to perform this adaptation, the course designer has first decomposed the learning content into short content pages; then, the appropriate content pages are selected for each kind of learner, according to their mental models regarding recursion. This, of course, is a step in which the teacher's understanding of the various mental models among learners is essential. For example, Fibonacci numbers (Appendix B2) are proposed to the students with the loop model on recursion because this situation may help them master the difference between recursion and iteration, and partition (Appendix B2) is suggested to the students with the analysis-synthesis model on recursion because this situation may help them understand how to build recursive solutions to complex problems (see also section 3.3.1). Section 7.3.2 presents an authoring tool in COFALE enabling the course designer to define those associations (mental models – learning contents).

Figure 6.10. Part of the learning content proposed to Bob (left) and Alice (right) by COFALE



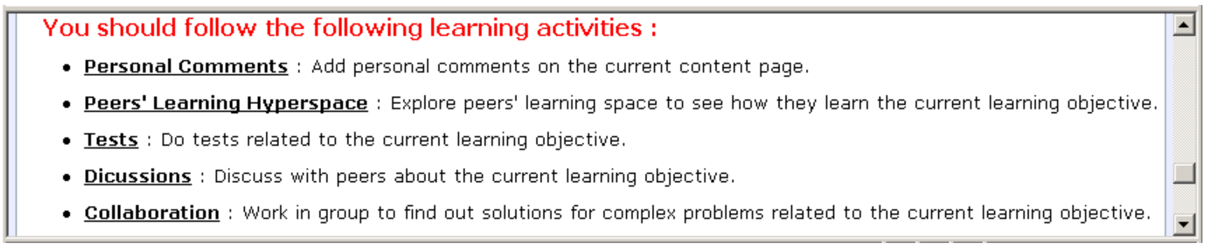
- Recursion
  - Basic concepts
    - Recursion
    - DCG strategy
    - Recursive algorithms
    - Recursive methods
    - Base cases
    - Recursive part
  - Learning situations
    - Arithmetic expressions
      - Recursive definition
      - Recursive evaluation
      - Recursive evaluation process
      - Java implementation
      - Java test class
    - Fibonacci numbers
      - First Java implementation
      - Second Java implementation
      - Third Java implementation
    - Simple text search
      - Recursive definition
      - Definition tracing
      - Recursive algorithm
      - Searching examples
      - Java implementation
      - Java test class



- Recursion
  - Basic concepts
    - Recursion
    - DCG strategy
    - Recursive thinking
    - Recursive algorithms
    - Iterative thinking
    - Iterative algorithms
  - Learning situations
    - Arithmetic expressions
      - Recursive definition
      - Recursive evaluation
      - Recursive evaluation process
      - Java implementation
      - Java test class
    - Simple text search
      - Recursive definition
      - Definition tracing
      - Recursive algorithm
      - Searching examples
      - Java implementation
      - Java test class
    - Partition
      - Recursive formula
      - Recursive-computation process
      - Tree-recursive computing process
      - From problem to solution
      - Recursive algorithm

Pedagogical devices. Because Bob is a “novice” and Ted and Alice are “experts”, we must guide and encourage Bob much more than Ted and Alice in the learning process. For instance, COFALE suggests 10 activities (Figure 6.3) to Bob but only 5 "advanced" tasks to Ted and Alice (Figure 6.11): Alice and Ted are versed in the use of COFALE, so COFALE does not present them with the three learning activities "Next Page", "Related Topics", and "Learning History" (Figure 6.11).

Figure 6.11. Learning activities proposed to Alice and Ted by COFALE



You should follow the following learning activities :

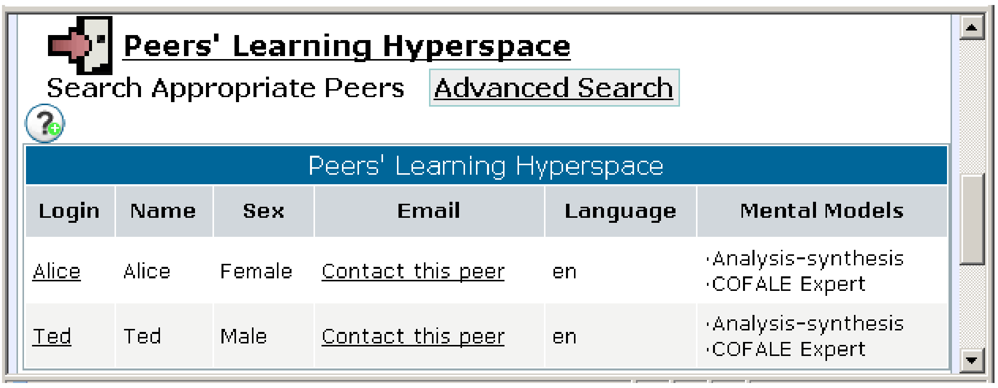
- **Personal Comments** : Add personal comments on the current content page.
- **Peers' Learning Hyperspace** : Explore peers' learning space to see how they learn the current learning objective.
- **Tests** : Do tests related to the current learning objective.
- **Dicussions** : Discuss with peers about the current learning objective.
- **Collaboration** : Work in group to find out solutions for complex problems related to the current learning objective.

To make this possible, COFALE provides the course designer with a specific tool (see section 7.2.2) so that he or she can define, for each content page, the appropriate learning activities for each type of learner.

Communication support. While learning with COFALE, learners can use a tool to search for peers who could help them overcome difficulties about acquiring the concept of recursion;

COFALE may, for instance, suggest Ted and Alice to Bob (Figure 6.12) so that he can ask them questions about simple problems; COFALE may suggest Ted to Alice so that they can exchange ideas about advanced recursive techniques. Moreover, students could use "Advanced Search" (seen at the top-middle of Figure 6.12) to find particular peers by introducing, for example, their name, gender, mental models.

Figure 6.12. Appropriate peers proposed to Bob by COFALE



Peers' Learning Hyperspace

Login	Name	Sex	Email	Language	Mental Models
Alice	Alice	Female	Contact this peer	en	·Analysis-synthesis ·COFALE Expert
Ted	Ted	Male	Contact this peer	en	·Analysis-synthesis ·COFALE Expert

The course designer needs to define, for each kind of learner (according to the assumed mental model), the appropriate peers (e.g. learners with more advanced mental models for learners with less advanced ones). COFALE also supports an instructor tool for this purpose (see section 7.3.2).

In addition to the previous three types of adaptation support, COFALE also supports goal-based learning (Henze & Nejdl, 2001; Masie Center, 2003): It adapts the learning hyperspace to the learner's current learning objective. For example, moving from the learning hyperspace of recursion to the one of linked lists (or vice versa), the learner is presented with the learning situations, learning activities, and tests that are exclusively designed for linked lists (or recursion).

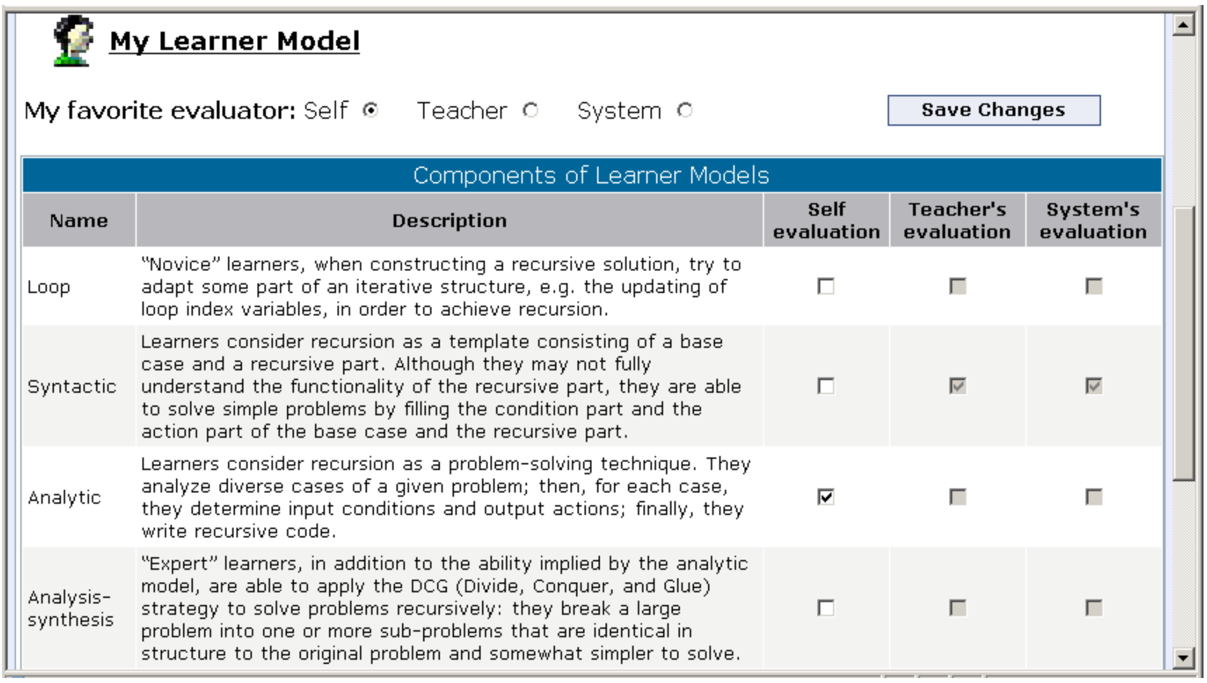
It should be noted that, at the beginning of the course, the course designer sets a default model for every new learner (e.g., the loop model in the case of recursion). During the learning process, three kinds of evaluations of mental models may be performed:

1. *Self-evaluation*. For instance, after exploring situations and doing tests, Bob could identify that he possesses the analytic model (Figure 6.13).
2. *Evaluation by the tutor*. For example, after evaluating Bob's tests and learning behavior, the tutor could diagnose that Bob possesses the syntactic model (Figure 6.13).
3. *Evaluation by COFALE*. For instance, on the basis of Bob's test results provided by the tutor, COFALE could detect that Bob possesses the syntactic model (Figure 6.13).

At certain times, for example after a test, learners may be asked to update the information about their mental model and choose one of the three kinds of evaluation they prefer. Bob, for instance, decides to always rely on his own evaluation (Figure 6.13: My favorite evaluator). COFALE will immediately adapt the learning contents, pedagogical devices, and communication support to the learners' new mental model.

In section 7.3.1, I detail the way and the tools provided for the course designer to manage learners' mental models. Note that the previous three kinds of evaluation have been used in a number of adaptive learning systems (Brusilovsky, 1999; Henze & Nejdl, 2001; Weber & Brusilovsky, 2001).

Figure 6.13. Learner model manager proposed to Bob by COFALE



My Learner Model

My favorite evaluator: [✓ Self] Teacher System

Components of Learner Models

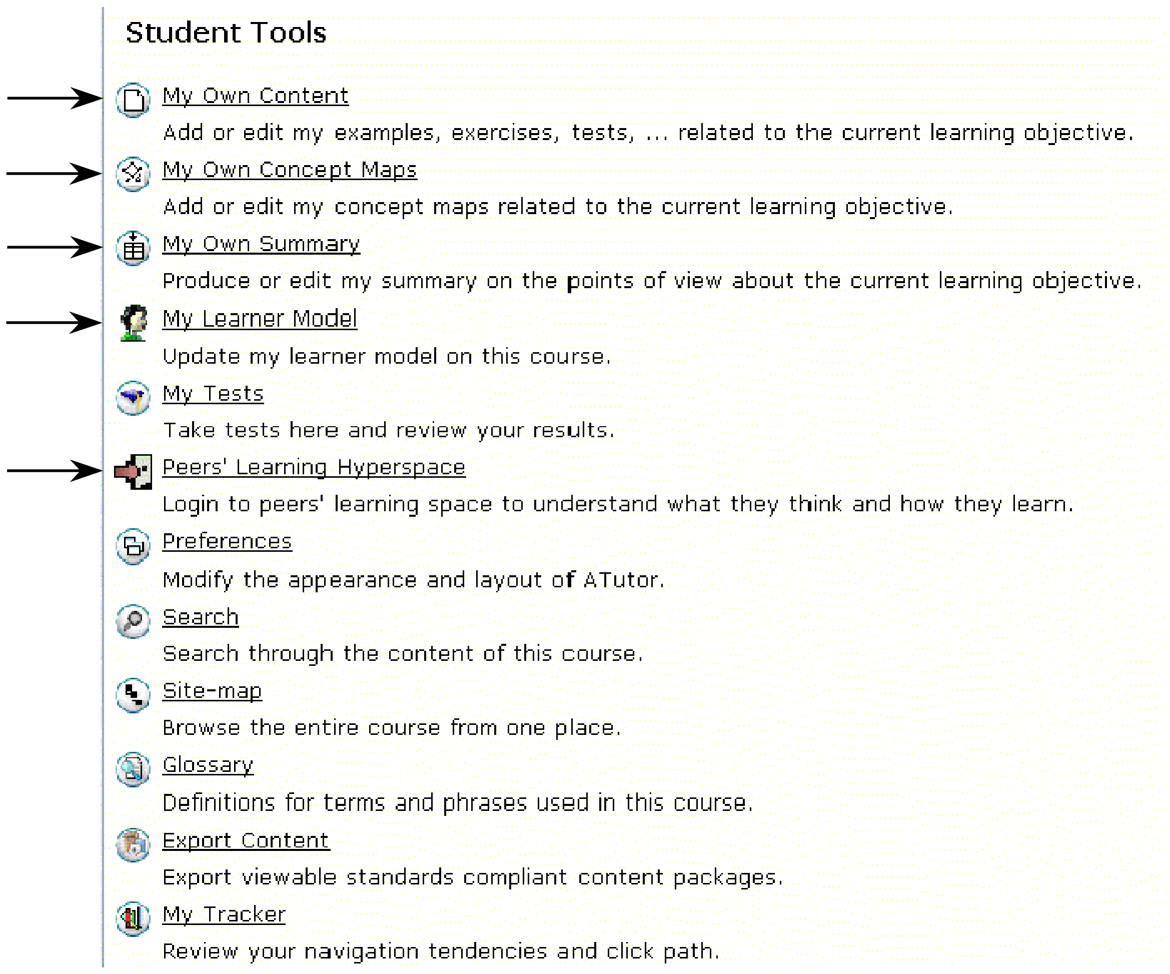
Name	Description	Self evaluation	Teacher's evaluation	System's evaluation
Loop	"Novice" learners, when constructing a recursive solution, try to adapt some part of an iterative structure, e.g. the updating of loop index variables, in order to achieve recursion.	☐	☐	☐
Syntactic	Learners consider recursion as a template consisting of a base case and a recursive part. Although they may not fully understand the functionality of the recursive part, they are able to solve simple problems by filling the condition part and the action part of the base case and the recursive part.	☐	✓	✓
Analytic	Learners consider recursion as a problem-solving technique. They analyze diverse cases of a given problem; then, for each case, they determine input conditions and output actions; finally, they write recursive code.	✓	☐	☐
Analysis-synthesis	"Expert" learners, in addition to the ability implied by the analytic model, are able to apply the DCG (Divide, Conquer, and Glue) strategy to solve problems recursively: they break a large problem into one or more sub-problems that are identical in structure to the original problem and somewhat simpler to solve.	☐	☐	☐

Main result: COFALE adapts the learning contents, pedagogical devices, and communication support to different kinds of students, according to their mental models.

6.3.3 Other learner tools

A part from the previous learning conditions proposed for Bob, COFALE also provides him with many other learning tools (Figure 6.14). In Figure 6.14, many tools are originally supported by the ATutor system, only the tools next to the arrows are COFALE's.

Figure 6.14. A subset of learner tools proposed to Bob by COFALE


Student Tools

My Own Content
Add or edit my examples, exercises, tests, ... related to the current learning objective.
My Own Concept Maps
Add or edit my concept maps related to the current learning objective.
My Own Summary
Produce or edit my summary on the points of view about the current learning objective.
My Learner Model
Update my learner model on this course.
My Tests
Take tests here and review your results.
Peers' Learning Hyperspace
Login to peers' learning space to understand what they think and how they learn.
Preferences
Modify the appearance and layout of ATutor.
Search
Search through the content of this course.
Site-map
Browse the entire course from one place.
Glossary
Definitions for terms and phrases used in this course.
Export Content
Export viewable standards compliant content packages.
My Tracker
Review your navigation tendencies and click path.

6.4 COFALE and criteria of Jonnaert and Vander Borght

In this section, I use the set of criteria proposed by Jonnaert and Vander Borght for the concept of learning, according to their socio-constructivist and interactive (SCI) model (see section 2.5), to analyze the concept of learning that was described through Bob's learning process illustrated in section 6.3.1. Table 6.1 shows that the concept of learning I described earlier in COFALE satisfies all the criteria proposed by Jonnaert and Vander Borght for the three dimensions in their SCI model. So, I may conclude that my educational approach is consistent with this SCI model.

Table 6.1. Conformity of the definition of learning in COFALE with Jonnaert and Vander Borght's criteria

Dimensions	Criteria	Analysis	Comments
(1) Constructivist	(1.1) Who is the **actor** of the learning?	**The student**	This aspect is explicit: Bob is the main actor and he actively manages his own learning.
	(1.2) Does the student learn on the basis of his **prior knowledge**?	**Yes**	Bob learns recursion on the basis of his prior knowledge such as iterations and linked lists.
	(1.3) Does learning have a **meaning** for the student?	**Yes**	Recursion is a very important concept in computing science.
(2) Socio	(2.1) Does the student learn through **interactions with peers**?	**Yes**	Bob runs discussion with Alice and Ted through e-mail, forums, chat rooms.
	(2.2) Does the student learn through **interactions with the teacher**?	**Yes**	The teacher participates in students' discussion to facilitate their learning.
	(2.3) Are the **zones of dialogue** defined to allow interactions among the students, the teacher, and the learning object?	**Yes**	Forums, chat rooms, collaboration hyperspaces related to the current learning object.
(3) Interactive	(3.1) Does the student learn from **situations**?	**Yes**	Bob learns recursion in arithmetic expressions, simple text search, etc.
	(3.2) Does the student have to discover the **learning object** in these situations?	**Yes**	Bob has to examine multiple learning situations to be able to master the recursion concept.
	(3.3) Does the student have to **interact** with these situations and the learning object?	**Yes**	Bob is exhorted to do many learning activities at the end of each situation, for instance to add personal comments.
	(3.4) Does the environment permit to establish a **distinction** between the learning object and the student's knowledge?	**Yes**	Bob adds his own comments, examples, and produces summaries related to the current learning object.
	(3.5) Are there **interactions** between the learning object and the student's knowledge?	**Yes**	While adding recursive examples, Bob is encouraged to establish the links between his prior knowledge with new knowledge.

6.5 Discussion

The previous presentation of COFALE's conditions of learning shows, *not surprisingly*, that the recursion course in COFALE satisfies all 10 criteria for cognitive flexibility. It should be noted that there is not any comparison, either implicit or explicit, between an example of use of COFALE, which satisfies all of 10 criteria, and the examples of use of learning systems I analyzed in section 5.2, which satisfy about a half of the criteria. The point I make here is that we can exploit ICT to satisfy all of the criteria for cognitive flexibility and that the importance should be in the quality rather than in the quantity (see the discussion in section 2.4.5).

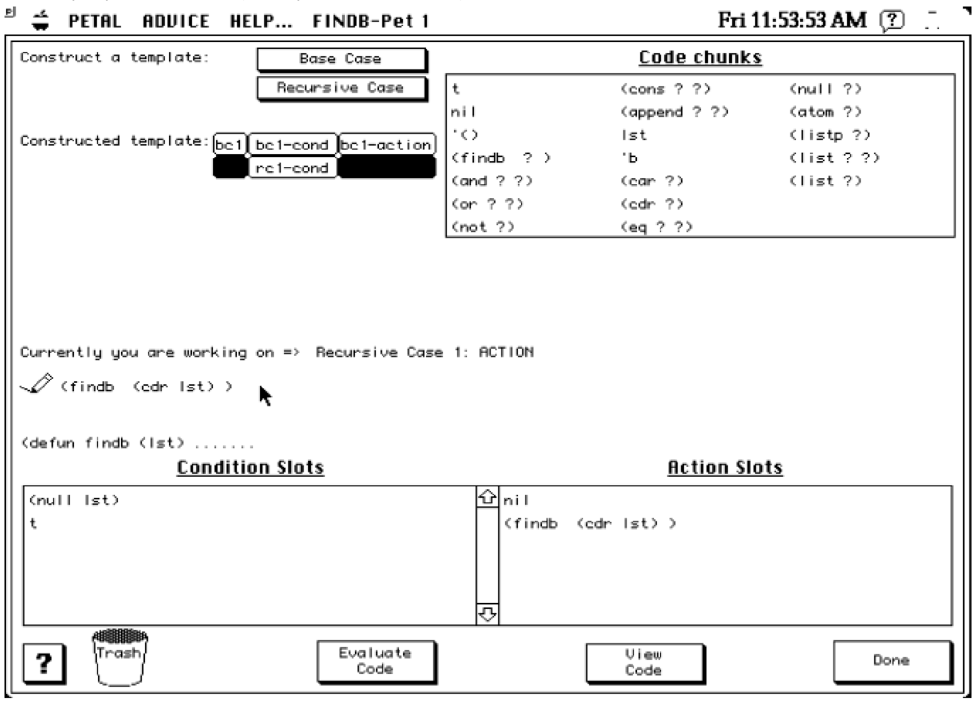
If the student's learning process occurs in a similar way as described earlier, we would draw two principal conclusions. Firstly, the student's process of knowledge construction could be *visible*, that is, learners actively construct their own knowledge through their own learning activities. For example, Bob actively give comments on the learning content provided by the course designer, Bob actively produces summaries on multiple points of view by peers and by other people. Secondly, the student is expected to learn both *what* (e.g., to apply recursive techniques to solve diverse problems) and *how* (e.g., to express personal points of view, elicit peers' points of view, produce summaries on different points of view). That is why I would say the pedagogical principles underlying cognitive flexibility reflect the basic characteristics of constructivism, as Spiro and colleagues (1988, 1990, 1991) stated. I discuss this issue further in the evaluation of COFALE presented in chapter 9.

Apart from exploiting the desired characteristics of cognitive flexibility, COFALE also supports adaptation for different kinds of learners. Although COFALE implements several simple adaptation techniques borrowed from other adaptive learning systems (see section 5.3), it provides learners with scaffolding: It adapts the learning contents, pedagogical devices, and communication support to the mental model of each individual student. For example, when a particular learner is "novice", COFALE presents him or her with simple examples, situations, and tasks. And when the learner develops a higher mental model, COFALE provides him or her with more complex examples, situations, and tasks.

In section 1.4, I presented other facets of constructivism than cognitive flexibility (e.g., problem solving) and other adaptation techniques than the ones demonstrated in section 6.3.2 (e.g., adaptive problem-solving support). To make learning environments such as COFALE more completely constructivist and adaptive, we should also exploit other constructivist facets and other adaptation techniques. For example, we should integrate specific tools similar to the ones provided by the PETAL system (Bhuiyan et al., 1994) into COFALE to support recursive problem solving by learners. PETAL supports three programming environment tools (PETs): the Syntactic PET (Figure 6.15), the Analytic PET, and the A/S PET that externalize the problem-solving process of the syntactic method, the analytic method, and the analysis-synthesis method (see section 6.2.1), respectively. Such externalization helps learners concentrate on recursive problem solving, and therefore significantly improve the ability to solve problems recursively.

Figure 6.15. Syntactic PET proposed to learners by PETAL.

> "[A learner wants to write a recursive function in LISP, Findb, which returns true if a B is present on the top level of a list. The] learner using this PET constructs a recursive template consisting of base case(s) and recursive case(s). Next he or she fills in the case slots with problem-specific code chunks. In order to assist the learner, the Syntactic PET provides a menu of available code chunks specific to the selected problem. The code chunks for a particular problem are created in advance by the domain expert. Several extra code chunks are included in the PET as distractors. Once the selected code chunks have been placed into the template by the learner, the corresponding LISP code can be generated automatically by the PET." (Bhuiyan et al., 1994)



Code chunks:

```
t            (cons ? ?)     (null ?)
nil          (append ? ?)   (atom ?)
'()          lst            (listp ?)
(findb ? )   'b             (list ? ?)
(and ? ?)    (car ?)        (list ?)
(or ? ?)     (cdr ?)
(not ?)      (eq ? ?)
```

Currently you are working on => Recursive Case 1: ACTION

```
(findb (cdr lst) )

(defun findb (lst) .......
```

Condition Slots	Action Slots
(null lst)	nil
t	(findb (cdr lst))

In the next chapter, I describe a set of authoring tools in COFALE allowing one to design learning environments such as the one I presented in this chapter. I show that COFALE is a domain-independent learning platform, meaning that it may be used for teaching a large number of various subjects, from mathematics and sciences to economics and literature.

Note: One can explore COFALE's demonstration course on recursion at the following address: `http://renoir.info.ucl.ac.be/elearning/Cofale/login.php`. In section 8.4, I provide information about how to download and use the COFALE system.

CHAPTER 7

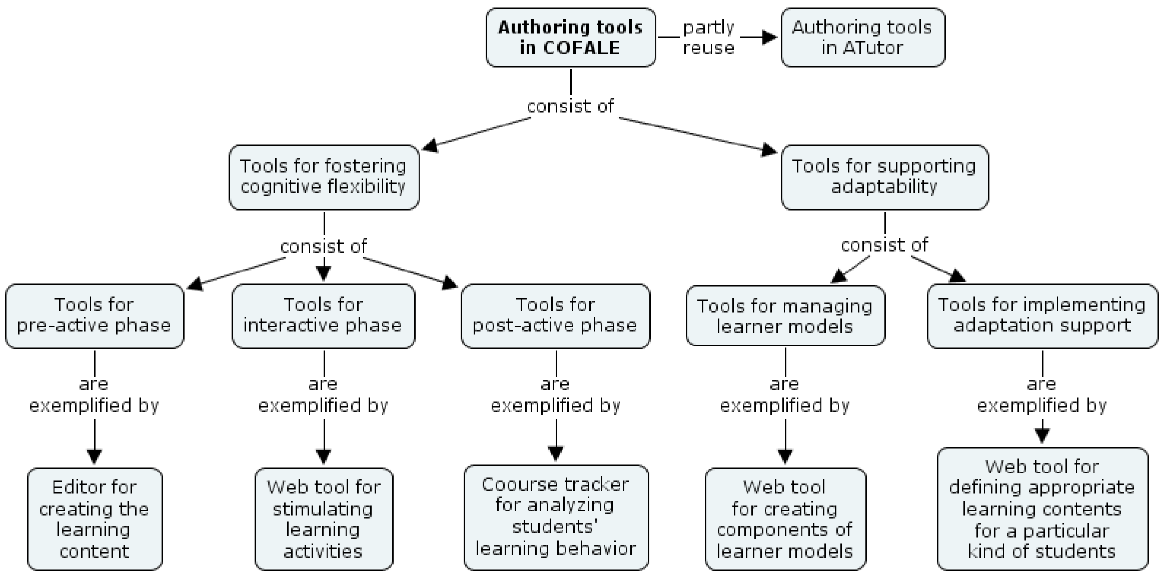
COFALE: Instructional design tools

"*Give me a long enough stick and a place to stand and I will move the world.*"

Archimedes, Greek Scientist, 287(?) – 212 B.C. (cited in Suomela, 2005)

(Reference to Appendices B & C)

In this chapter, I present a set of instructor tools provided by COFALE for the course designer to devise adaptive learning situations leading to cognitive flexibility. After reading the present chapter, one *should* be able to design and use adaptive learning environments supporting cognitive flexibility. One *could* also understand that operational criteria for cognitive flexibility may be effectively used as guidelines for devising learning situations fostering cognitive flexibility.


Authoring tools in COFALE
partly reuse
Authoring tools in ATutor
consist of
Tools for fostering cognitive flexibility
Tools for supporting adaptability
consist of
consist of
Tools for pre-active phase
Tools for interactive phase
Tools for post-active phase
Tools for managing learner models
Tools for implementing adaptation support
are exemplified by
are exemplified by
are exemplified by
are exemplified by
are exemplified by
Editor for creating the learning content
Web tool for stimulating learning activities
Coourse tracker for analyzing students' learning behavior
Web tool for creating components of learner models
Web tool for defining appropriate learning contents for a particular kind of students

Summary

7.1 Introduction

Chapter 3 presented an instructional design process for creating learning conditions fostering cognitive flexibility. Chapter 6 illustrated, in an ICT-based learning environment (COFALE), a process of learning with support for cognitive flexibility and with support for adaptability. The present chapter aims at describing a set of authoring tools in COFALE that allows one to design learning environments such that the one shown in chapter 6 by following the design process explained in chapter 3.

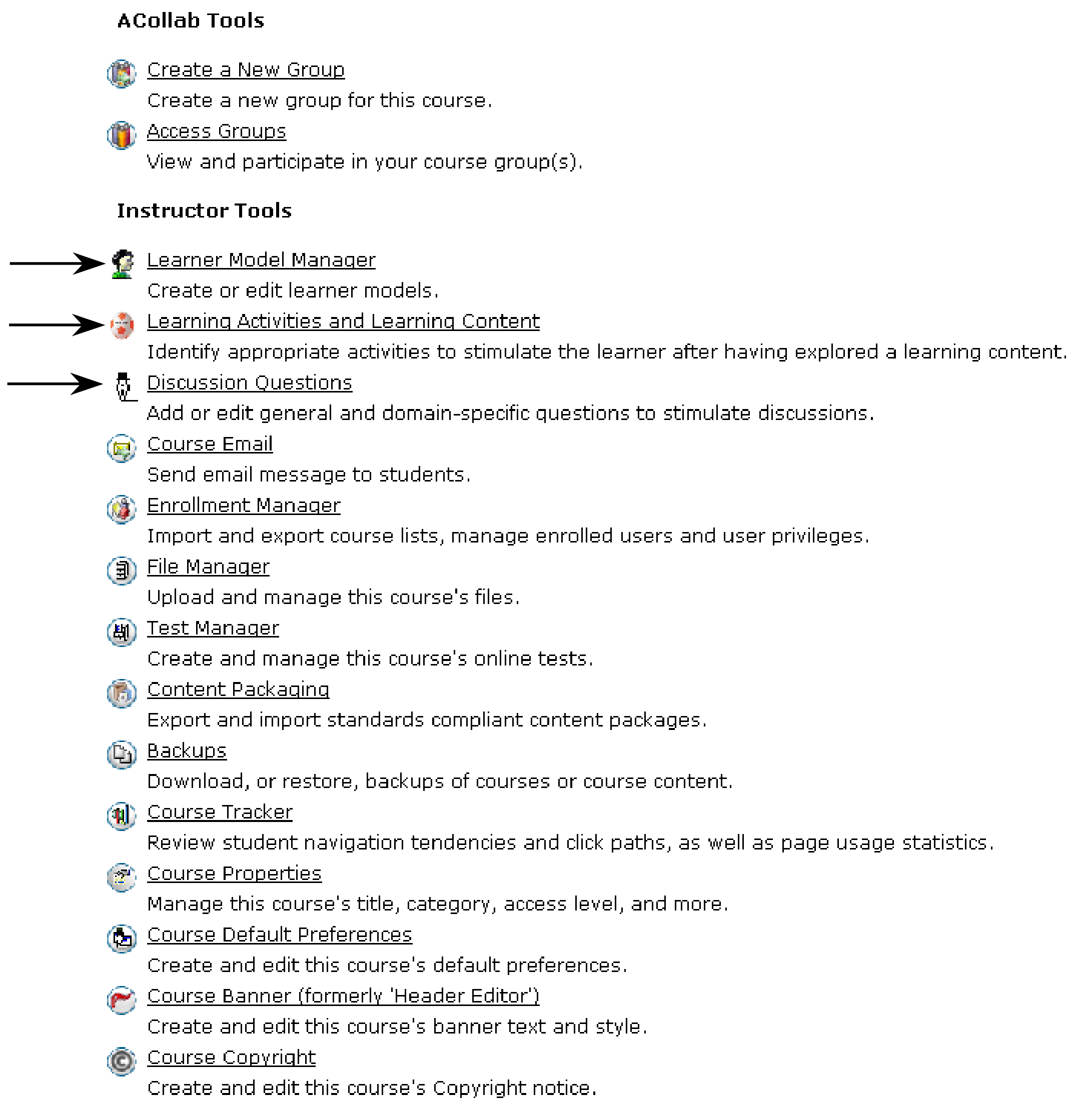
COFALE supports many instructor tools for creating and managing courses. In this chapter, however, I describe only the essential tools for implementing the learning conditions demonstrated in section 6.3. In Figure 7.1, the three tools next to the three arrows are COFALE's, and the other tools are ATutor's. Hereafter, except for those three tools, when I say a tool is supported by COFALE, the tool is originally provided by ATutor.

To understand more about all of the tools provided by COFALE, one should read ATutor's "How To Course" (Adaptive Technology Resource Center, 2004). Note that the tools provided by COFALE facilitate the design of goal-based learning; that is, organizing the learning materials concentrated around specific learning objectives (Henze & Nejdl, 2001; Masie Center, 2003).

To help the reader understanding this chapter, I clarify the following concepts:

- The *course designer* is the person who makes a course available in COFALE, for example to create the learning content.
- The *tutor* or *teacher* is the person who participates in COFALE to facilitate students' process of knowledge construction and transformation, for instance to evaluate students' tests and learning behavior and to give them feedback.
- The *instructor* is either the course designer or the tutor or the teacher.
- The *software developer* is the person who develops COFALE, for example to add a software component to COFALE.

Figure 7.1. Subset of instructor tools proposed to the course designer by COFALE


ACollab Tools

Create a New Group
Create a new group for this course.

Access Groups
View and participate in your course group(s).

Instructor Tools

Learner Model Manager
Create or edit learner models.

Learning Activities and Learning Content
Identify appropriate activities to stimulate the learner after having explored a learning content.

Discussion Questions
Add or edit general and domain-specific questions to stimulate discussions.

Course Email
Send email message to students.

Enrollment Manager
Import and export course lists, manage enrolled users and user privileges.

File Manager
Upload and manage this course's files.

Test Manager
Create and manage this course's online tests.

Content Packaging
Export and import standards compliant content packages.

Backups
Download, or restore, backups of courses or course content.

Course Tracker
Review student navigation tendencies and click paths, as well as page usage statistics.

Course Properties
Manage this course's title, category, access level, and more.

Course Default Preferences
Create and edit this course's default preferences.

Course Banner (formerly 'Header Editor')
Create and edit this course's banner text and style.

Course Copyright
Create and edit this course's Copyright notice.

7.2 Authoring tools for supporting cognitive flexibility

In this section, I describe a certain number of instructor tools provided by COFALE allowing the course designer to create the learning conditions presented in section 6.3.1 for fostering cognitive flexibility. To present those authoring tools in a systematic manner that is consistent with the instructional design process shown in section 3.3, I organize this section into three sub-sections: (a) tools for the pre-active phase, (b) tools for the interactive phase, and (c) tools for the post-active phase. In each of the three phases, and for each activity in the phase (see Table 7.1), I show the tools helping the construction of learning conditions mentioned in the activity.

Table 7.1. Instructional design process for cognitive flexibility

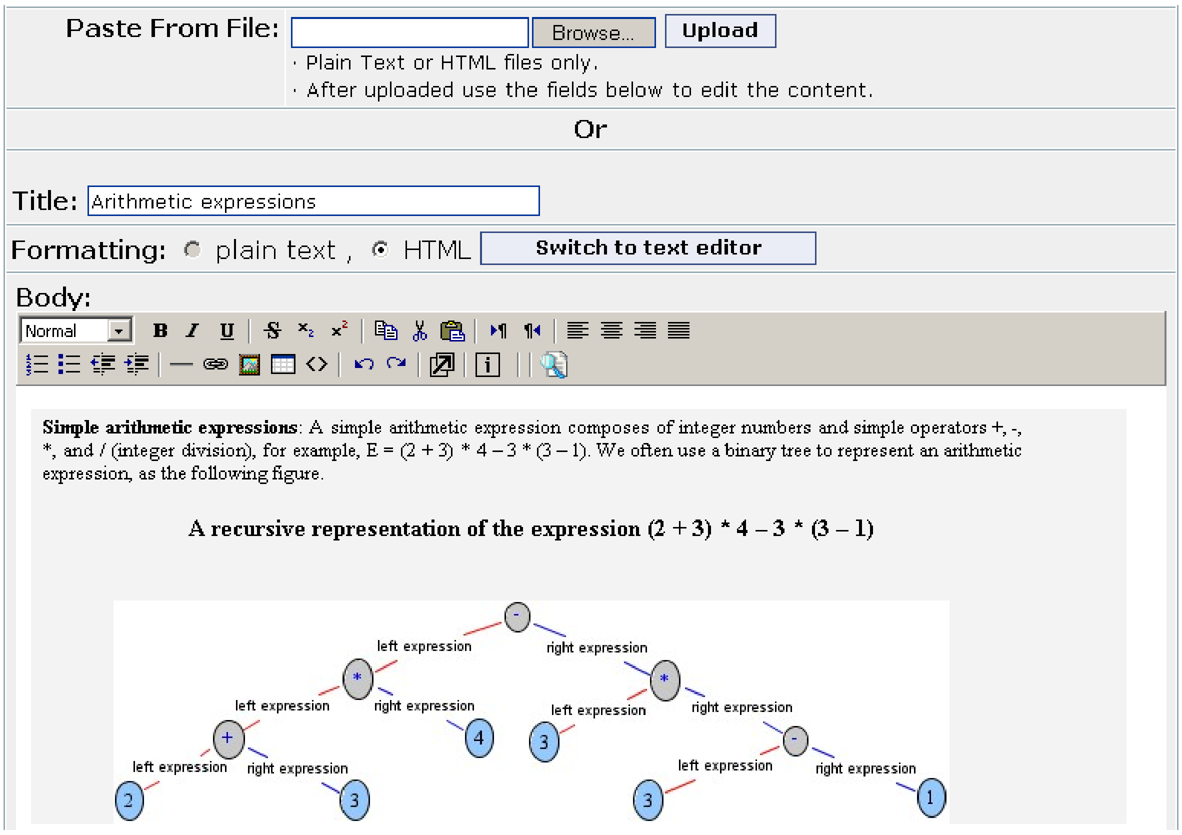
See section 3.3 to understand why the design process consists of three phases.

Pre-active phase
Activity 1.1: Prepare the learning content for the underlying concepts.
Activity 1.2: Prepare a diversity of meaningful learning situations emphasizing the nature of the underlying concepts.
Activity 1.3: Prepare learning contents for the concepts that are related to the underlying concepts.
Activity 1.4: Prepare assessment situations both for individual tests and for tests in groups. The nature of these situations should stimulate multiple points of view.
Activity 1.5: Prepare diverse means for engaging the tutor, learners, and other people in exchanges.
Activity 1.6: Prepare a list of general discussion questions and a list of domain-specific discussion questions.
Activity 1.7: Prepare multiple external resources related to the underlying concepts.
Interactive phase
Activity 2.1: Engage learners explicitly in performing multiple learning activities related to the underlying concepts.
Activity 2.2: Encourage learners explicitly to study the concepts that are related to the underlying concepts.
Activity 2.3: Encourage learners explicitly to examine different interpretations of the underlying concepts (by other authors and by peers), to express their personal points of view on the underlying concepts, and to give feedback on the points of view of other people.
Activity 2.4: Stimulate learners explicitly to treat a diversity of points of view on the underlying concepts.
Activity 2.5: Encourage learners explicitly to run a variety of discussions with other people in different contexts.
Activity 2.6: Make available tools so that learners can actively express their personal points of view and stimulate those of other participants during the discussion.
Activity 2.7: Encourage groups of learners explicitly to do assessment in groups, to confront and compare multiple points of view.
Activity 2.8: Encourage learners explicitly to do assessment individually, to confront and compare multiple points of view.
Activity 2.9: During the learning session, observe and evaluate the learning behavior of each learner with respect to cognitive flexibility, so as to provide him or her with appropriate feedback.
Post-active phase
Activity 3.1: Evaluate the learning behavior and outcomes formatively for each learner, and communicate both the result of the analysis process and feedback explicitly to him or her, keeping a positive regard on his or her knowledge.
Activity 3.2: Evaluate the teaching behavior with respect to cognitive flexibility.

7.2.1 Tools for the pre-active phase

Activities 1.1 and 1.2. To help the course designer in these two activities, COFALE provides a hypermedia editor (Figure 7.2). This editor allows the course designer to create different types of discrete pieces of learning content or hypermedia pages. With the help of the editor, the course designer finds it easy, for instance, to format a hypermedia page and to insert a hyperlink to a resource such as a website, an image, an audio or video file into a hypermedia page (see the formatting toolbars on the middle of Figure 7.2). The course designer can also import (see the button "Upload" on the top of Figure 7.2) a HTML file he or she creates with a specific Web design tool such as MSWord or Netscape Composer. If the course designer is versed in HTML, he or she can switch the editor to a text mode (see the button "Switch to text editor" above the formatting toolbars in Figure 7.2) and then modify the structure as well as the content of the HTML file, for example, insert a simulation written in Javascript into the HTML file (see ATutor's "How To Course" for more details).

Figure 7.2. A Web tool provided by COFALE for the course designer to create learning contents


Paste From File:

· Plain Text or HTML files only.

· After uploaded use the fields below to edit the content.

Or

Title: Arithmetic expressions

Formatting: plain text , [✓ HTML]

Body:

Simple arithmetic expressions: A simple arithmetic expression composes of integer numbers and simple operators +, -, *, and / (integer division), for example, E = (2 + 3) * 4 – 3 * (3 – 1). We often use a binary tree to represent an arithmetic expression, as the following figure.

A recursive representation of the expression (2 + 3) * 4 – 3 * (3 – 1)

In the design of the learning content, particularly in the context of e-Learning, the course designer should take into account the concept of learning objects introduced in section 4.2. There is no universal acceptance among educational technologists of the definition of learning objects. So, I present here my definition that is close to the one proposed by Masie Center (2003):

- An *asset* is learning content in its most basic form such as electronic media, text, images, and sound. The learning content shown in Figure 7.2, for instance, would have two assets: a paragraph of text and an image.
- A *sharable content object* is the lowest level of granularity of learning content that can be tracked by a learning content management system such as ATutor, for example, the learning content presented in Figure 7.2 as being a hypermedia page.

- An *information block* is a set of sharable content objects organized to present concepts, learning situations, and so on. In Figure 6.1, "Arithmetic expressions" consisting of a textual definition, two simulations, and a Java implementation, "Basic concepts" regrouping the concepts underlying recursion, and "Learning situations" that regroups different learning situations for recursion are examples of information blocks.
- A *learning object* is a set of information blocks or sharable content objects organized to meet a particular learning objective. For example, "Recursion" shown in Figure 6.1 is assembled by "Basic concepts" and "Learning situations" to help students develop the ability to solve problems recursively. "Linked lists" illustrated in Figure 6.2 is constituted by "Basic concepts" and "Learning situations" to help students develop the ability to use linked lists to represent different kinds of information data.

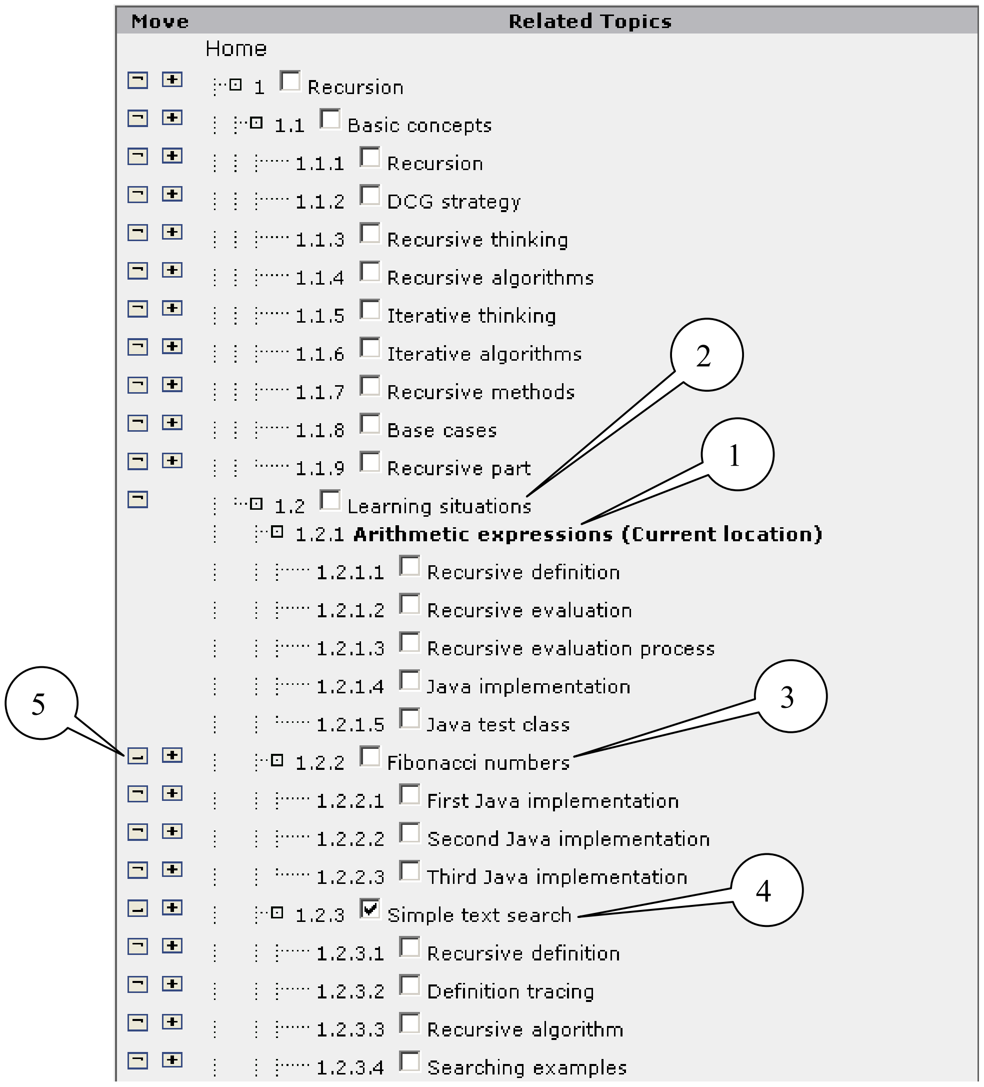
The careful decomposition of the learning content into sharable content objects is important because it may take several easy-to-see advantages (Masie Center, 2003), as follows:

- *Design of goal-based learning*. Because of the flexibility of fine-grained content objects, the course designer finds it easy to assemble and reassemble them to meet a certain learning objective. For instance, it is easy for him or her to modify the structure as well as the content of the learning object "Recursion" shown in Figure 6.1 or "Linked lists" shown in Figure 6.2.
- *Personalization of learning contents*. It is also straightforward for the course designer to personalize the content within a learning object for different kinds of students (see also section 7.3.2).
- *Reusability*. The course designer can reuse certain content objects designed for an existing learning object to create a new one. For instance, he or she can reuse the situation about simple text search (Appendix B2) designed for recursion in the design of linked lists.

It is worth noting that COFALE provides a number of tools (see ATutor's "How To Course") for the management of content objects that are compliant with the IMS/SCORM standard (Advanced Distributed Learning, 2004). For example, COFALE supports a specific tool allowing the course designer to introduce the metadata for the content object such as keywords and release date, which are compliant with the learning object metadata proposed by IMS (Advanced Distributed Learning, 2004). More discussions about the concepts of learning objects and metadata, and theirs usefulness are presented in section 4.2.

COFALE also provides "Move" tools, shown on the left hand side of Figure 7.3, to help the course designer rearrange the order in which content objects are presented to students; for instance, to specify that the "Arithmetic expressions" content object (Figure 7.3: Area 1) is a sub-topic of the "Learning situations" one (Figure 7.3: Area 2), the "Arithmetic expressions" content object must precede the "Fibonacci numbers" and "Simple text search" ones (Figure 7.3: Areas 3 & 4) in the list of content objects provided for the learner (see also Figure 6.1), and so on. Now, if the course designer wants to place "Arithmetic expressions" after "Fibonacci numbers", he or she should click on the small icon shown at Area 5 of Figure 7.3.

Figure 7.3. A Web tool supported by COFALE for the course designer to define relationships among content objects


Move | Related Topics

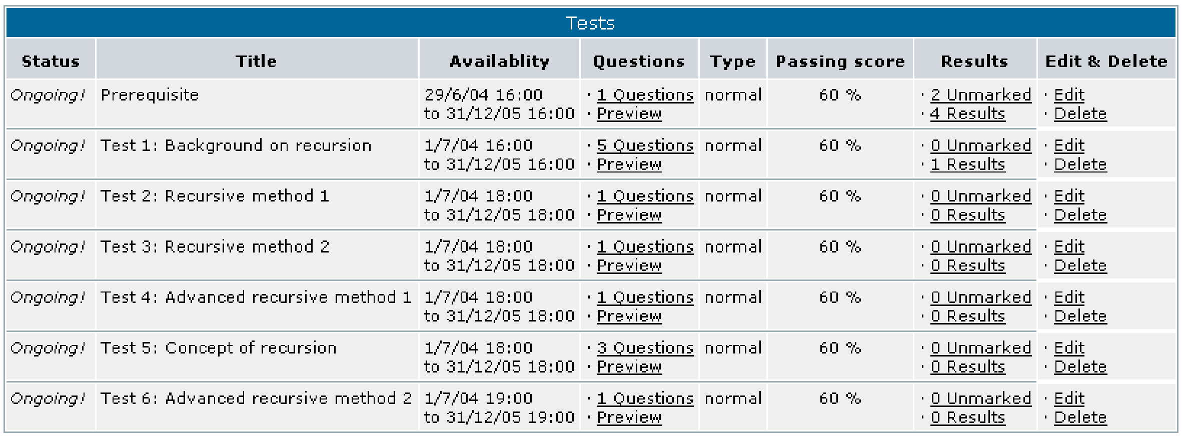
Home
1 ☐ Recursion
1.1 ☐ Basic concepts
1.1.1 ☐ Recursion
1.1.2 ☐ DCG strategy
1.1.3 ☐ Recursive thinking
1.1.4 ☐ Recursive algorithms
1.1.5 ☐ Iterative thinking
1.1.6 ☐ Iterative algorithms
1.1.7 ☐ Recursive methods
1.1.8 ☐ Base cases
1.1.9 ☐ Recursive part
1.2 ☐ Learning situations
1.2.1 **Arithmetic expressions (Current location)**
1.2.1.1 ☐ Recursive definition
1.2.1.2 ☐ Recursive evaluation
1.2.1.3 ☐ Recursive evaluation process
1.2.1.4 ☐ Java implementation
1.2.1.5 ☐ Java test class
1.2.2 ☐ Fibonacci numbers
1.2.2.1 ☐ First Java implementation
1.2.2.2 ☐ Second Java implementation
1.2.2.3 ☐ Third Java implementation
1.2.3 ☑ Simple text search
1.2.3.1 ☐ Recursive definition
1.2.3.2 ☐ Definition tracing
1.2.3.3 ☐ Recursive algorithm
1.2.3.4 ☐ Searching examples

Activity 1.3. In addition to the previous tools proposed to the course designer for preparing the learning content of related concepts, COFALE provides him or her with a tool "Related Topics" (seen on the right hand side of Figure 7.3) to define the content objects that are related to the one the course designer is editing. For example, to associate the introduction of "simple text search" (Figure 7.3: Area 4) with the introduction of "arithmetic expressions" (Figure 7.3: Area 1), the course designer selects the checkboxes next to the content object whose title is "simple text search" (Figure 7.3: Area 4). On the basis of this association, COFALE presents the hyperlink in the menu "Related Topics" (see Figure 6.1).

Presently, the relation "Related Topics" is symmetric in COFALE, meaning that when the course designer associates content object A with content object B, COFALE automatically associates content object B with content object A. Sometimes, this automatic association may be inconvenient, for instance, if the course designer wants a hyperlink to the definition of recursion to be presented in the menu "Related Topics" when COFALE shows the introduction of simple text search but not vice versa. For future research, one can modify COFALE so that the course designer can specify whether an association is symmetric or not.

Activity 1.4. In support of this activity, COFALE provides the course designer with a test manager (Figure 7.4). To create a new individual test, the course designer introduces an assessment situation (e.g., the robot situation presented in Appendix C), a passing score (e.g., 12/20 or 60%), one or more questions (e.g., 4 questions of Test 1 in Appendix C), and some other information such as start and end dates. At the present, COFALE supports three types of questions: multiple-choice questions, true or false questions, and open-ended questions whose answer size is one word or one sentence or one short paragraph (e.g., questions of Tests 1 and 5, Appendix C) or one page (e.g., questions of Tests 2, 3, 4, 6 in Appendix C). For future research, one can add other kinds of quizzes to COFALE such as matching questions (e.g., in the learning of languages, we ask the learner to find appropriate sentences in a column to match sentences in another column) and fill-in-the-blank questions (e.g., in the learning of languages, we ask the learner to complete sentences).

Figure 7.4. Test manager proposed to the course designer by COFALE

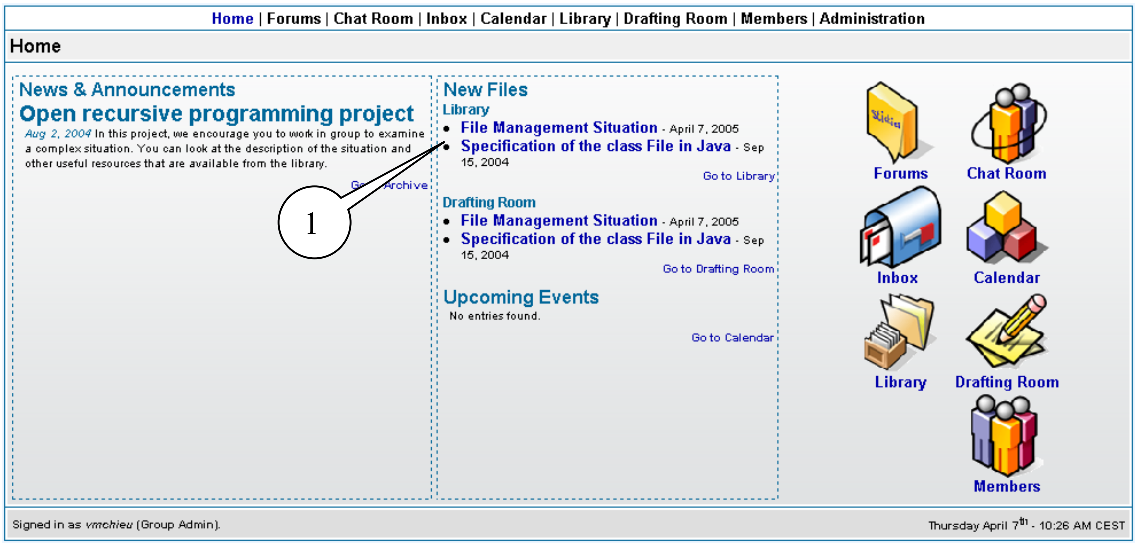


Tests

Status	Title	Availablity	Questions	Type	Passing score	Results	Edit & Delete
Ongoing!	Prerequisite	29/6/04 16:00 to 31/12/05 16:00	· 1 Questions · Preview	normal	60 %	· 2 Unmarked · 4 Results	· Edit · Delete
Ongoing!	Test 1: Background on recursion	1/7/04 16:00 to 31/12/05 16:00	· 5 Questions · Preview	normal	60 %	· 0 Unmarked · 1 Results	· Edit · Delete
Ongoing!	Test 2: Recursive method 1	1/7/04 18:00 to 31/12/05 18:00	· 1 Questions · Preview	normal	60 %	· 0 Unmarked · 0 Results	· Edit · Delete
Ongoing!	Test 3: Recursive method 2	1/7/04 18:00 to 31/12/05 18:00	· 1 Questions · Preview	normal	60 %	· 0 Unmarked · 0 Results	· Edit · Delete
Ongoing!	Test 4: Advanced recursive method 1	1/7/04 18:00 to 31/12/05 18:00	· 1 Questions · Preview	normal	60 %	· 0 Unmarked · 0 Results	· Edit · Delete
Ongoing!	Test 5: Concept of recursion	1/7/04 18:00 to 31/12/05 18:00	· 3 Questions · Preview	normal	60 %	· 0 Unmarked · 0 Results	· Edit · Delete
Ongoing!	Test 6: Advanced recursive method 2	1/7/04 19:00 to 31/12/05 19:00	· 1 Questions · Preview	normal	60 %	· 0 Unmarked · 0 Results	· Edit · Delete

COFALE also supports the tools (Figure 7.5) enabling the course designer to create assessment situations in groups. For example, the course designer can constitute different groups of learners, and present them (see "Library" on Area 1 of Figure 7.5) with certain problems in the situation about file management and a brief specification of the class File in Java, which could be useful for them to solve the given problems (see Appendix C).

Figure 7.5. A Web tool provided by COFALE for the course designer to create collaboration hyperspaces



Home | Forums | Chat Room | Inbox | Calendar | Library | Drafting Room | Members | Administration

Home

News & Announcements

Open recursive programming project

Aug 2, 2004 In this project, we encourage you to work in group to examine a complex situation. You can look at the description of the situation and other useful resources that are available from the library.

Go to Archive

New Files

Library

- File Management Situation - April 7, 2005
- Specification of the class File in Java - Sep 15, 2004

Go to Library

Drafting Room

- File Management Situation - April 7, 2005
- Specification of the class File in Java - Sep 15, 2004

Go to Drafting Room

Upcoming Events

No entries found.

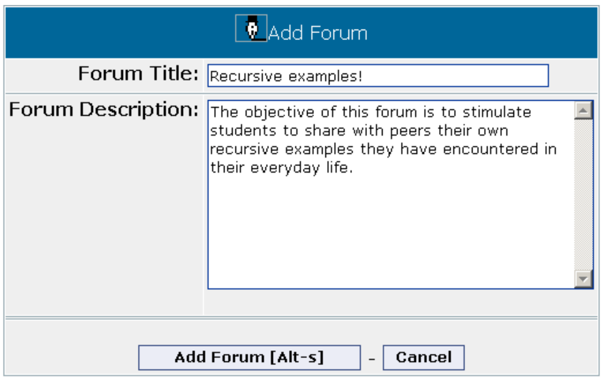
Go to Calendar

Signed in as *vmchieu* (Group Admin).

Thursday April 7th - 10:26 AM CEST

Activity 1.5. In support of this activity, COFALE supports a certain number of communication tools: e-mail, forums, and chat rooms. The course designer, however, should prepare several forums in advance. For instance, in Figure 7.6, the course designer creates a forum to engage students in the confrontation of their own recursive examples.

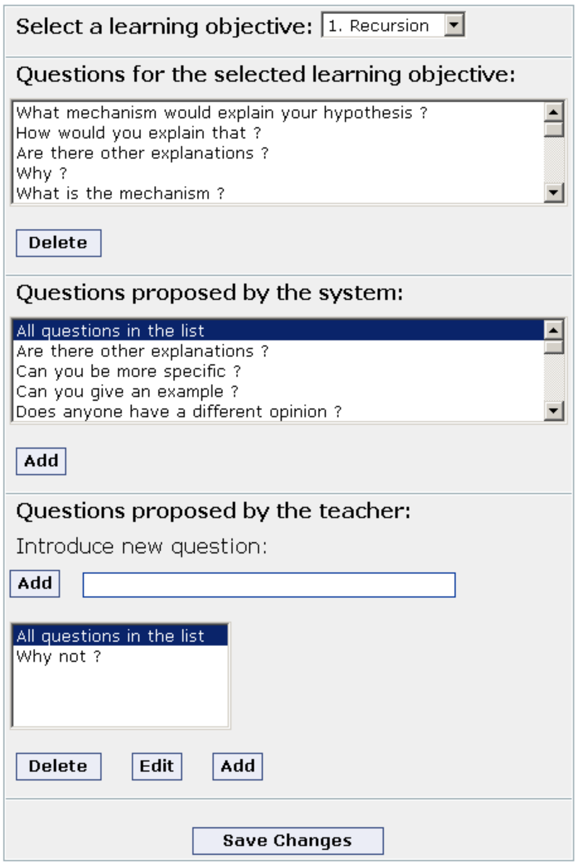
Figure 7.6. A Web tool provided by COFALE for the course designer to add a forum



Forum Title: Recursive examples!

Forum Description: The objective of this forum is to stimulate students to share with peers their own recursive examples they have encountered in their everyday life.

Figure 7.7. Discussion questions manager proposed to the course designer by COFALE



Select a learning objective: 1. Recursion

Questions for the selected learning objective:

What mechanism would explain your hypothesis ?
How would you explain that ?
Are there other explanations ?
Why ?
What is the mechanism ?

Questions proposed by the system:

All questions in the list
Are there other explanations ?
Can you be more specific ?
Can you give an example ?
Does anyone have a different opinion ?

Questions proposed by the teacher:

Introduce new question:

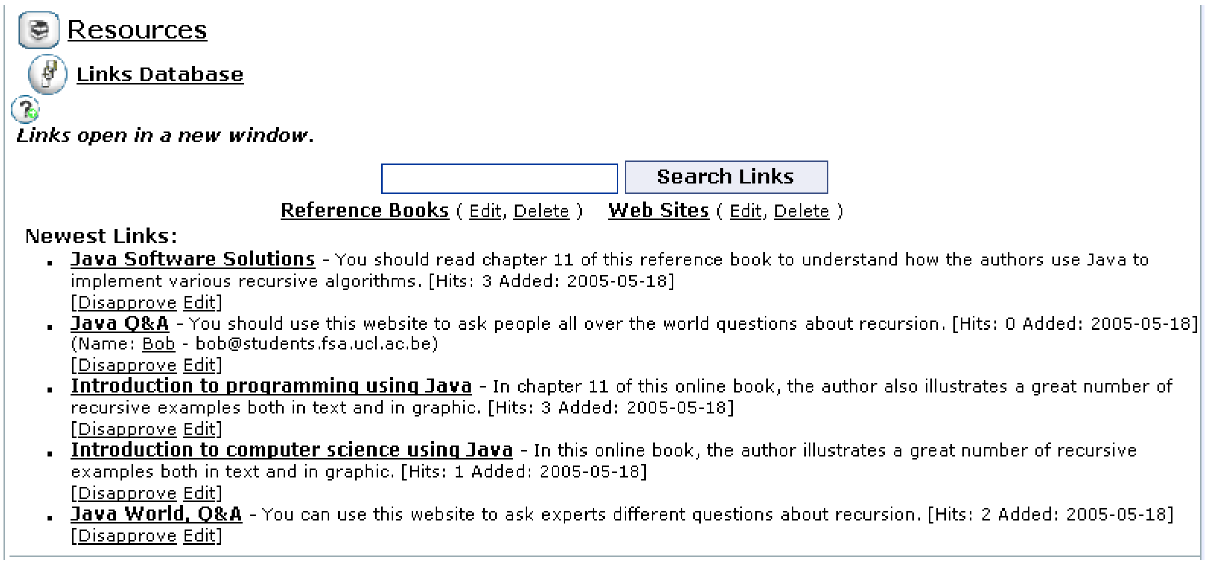
All questions in the list
Why not ?

Activity 1.6. To help the course designer in this activity, COFALE supports a built-in list of general questions (shown in the middle of Figure 7.7) proposed by educational theorists (see Appendix B5). The course designer and the tutor may also propose their own list of questions by using the tools seen on the bottom of Figure 7.7. To create a particular list of questions for a certain learning objective, the course designer first selects the learning objective from a dropdown list (e.g., recursion seen at the top of Figure 7.7), then selects the appropriate questions in the list proposed by COFALE as well as in his or her own list, and adds them to the list of questions seen on the top of Figure 7.7 for the selected learning objective.

To help the course designer make a list of domain-specific questions for a particular learning objective, COFALE also provides him or her with a tool similar to the one presented in Figure 7.7 (except that COFALE cannot support predefined domain-specific questions).

Activity 1.7. COFALE offers several tools (Figure 7.8) allowing the course designer to introduce the learner to external resources. The course designer can create different categories of external resources, for example, "References Books" and "Web Sites". In each category, he or she can add one or more references or hyperlinks, together with their descriptions, for instance, the first link in the category "Reference Books" and the last three links in the category "Web Sites" (Figure 7.8). The course designer can also approve (or disapprove) references or hyperlinks suggested by the learner, for example, the second link (Figure 7.8) proposed by the student Bob.

Figure 7.8. External resources manager proposed to the course designer by COFALE



Links open in a new window.

Reference Books (Edit, Delete) Web Sites (Edit, Delete)

Newest Links:

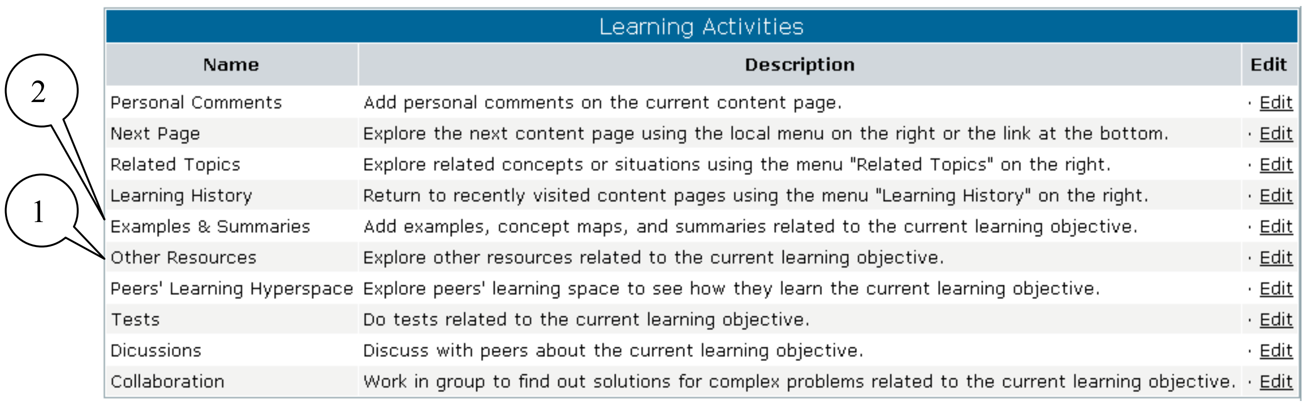
- Java Software Solutions - You should read chapter 11 of this reference book to understand how the authors use Java to implement various recursive algorithms. [Hits: 3 Added: 2005-05-18]
  [Disapprove Edit]
- Java Q&A - You should use this website to ask people all over the world questions about recursion. [Hits: 0 Added: 2005-05-18]
  (Name: Bob - bob@students.fsa.ucl.ac.be)
  [Disapprove Edit]
- Introduction to programming using Java - In chapter 11 of this online book, the author also illustrates a great number of recursive examples both in text and in graphic. [Hits: 3 Added: 2005-05-18]
  [Disapprove Edit]
- Introduction to computer science using Java - In this online book, the author illustrates a great number of recursive examples both in text and in graphic. [Hits: 1 Added: 2005-05-18]
  [Disapprove Edit]
- Java World, Q&A - You can use this website to ask experts different questions about recursion. [Hits: 2 Added: 2005-05-18]
  [Disapprove Edit]

7.2.2 Tools for the interactive phase

Activity 2.1. COFALE supports a set of predefined learning activities (Figure 7.9). Most activities are associated with a hyperlink, so that when an activity is presented to learners (see Figure 6.3), they can go directly to the pedagogical device(s) corresponding to the activity through this hyperlink. For example, the activity "Other Resources" (Figure 7.9: Area 1) is associated with the hyperlink to the pedagogical device "Resources" (Figure 7.8), so that learners can access this pedagogical device directly while they are exploring arithmetic expressions (see Figure 6.3). The software developer stores those hyperlinks in a particular database in COFALE in advance.

To define, for each type of learner and for each learning activity, the appropriate content pages (objects) to which the activity is related, the course designer first clicks on the command "Edit" next to the activity, for example "Examples & Summaries" (Figure 7.9: Area 2). Then, the course designer selects a particular kind of learner or all kinds of learners from a dropdown list, for instance the loop model shown at the top-right of Figure 7.10. Then, the course designer selects a learning objective to show all the content objects designed for the learning objective (Figure 7.10). Finally, the course designer selects the checkboxes next to the content pages he or she wants to associate with the learning activity "Examples & Summaries" (Figure 7.10). COFALE will present the students possessing the selected model ("Loop") with the selected learning activity ("Examples & Summaries") at the bottom of the selected content pages, for example "Java test class", the last item of arithmetic expressions (see Figures 7.10, 6.3).

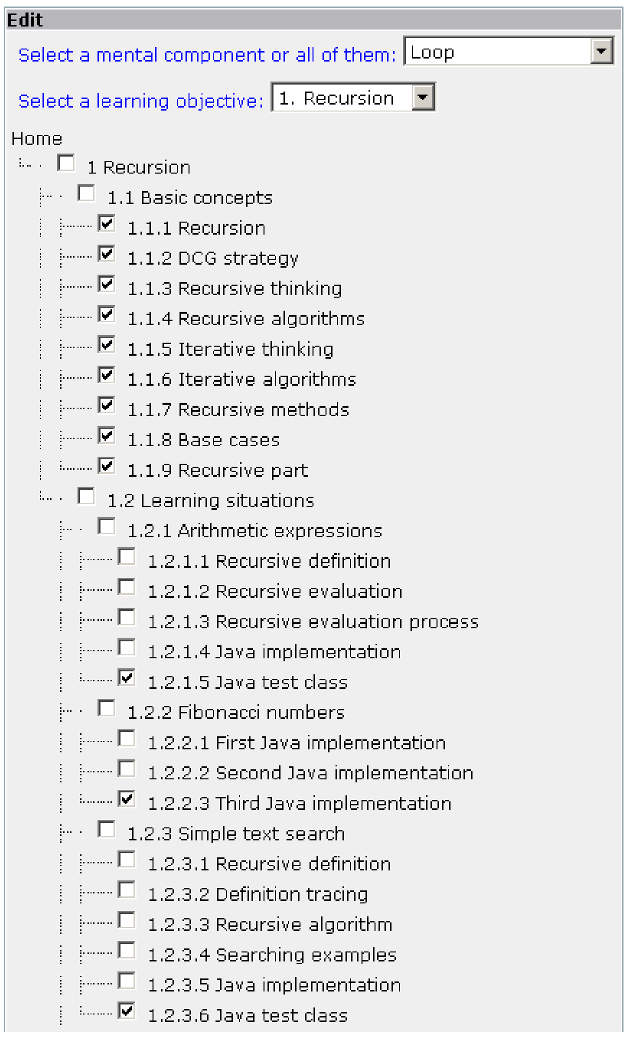
Figure 7.9. Predefined learning activities proposed to the course designer by COFALE



Learning Activities

Name	Description	Edit
Personal Comments	Add personal comments on the current content page.	· Edit
Next Page	Explore the next content page using the local menu on the right or the link at the bottom.	· Edit
Related Topics	Explore related concepts or situations using the menu "Related Topics" on the right.	· Edit
Learning History	Return to recently visited content pages using the menu "Learning History" on the right.	· Edit
Examples & Summaries	Add examples, concept maps, and summaries related to the current learning objective.	· Edit
Other Resources	Explore other resources related to the current learning objective.	· Edit
Peers' Learning Hyperspace	Explore peers' learning space to see how they learn the current learning objective.	· Edit
Tests	Do tests related to the current learning objective.	· Edit
Dicussions	Discuss with peers about the current learning objective.	· Edit
Collaboration	Work in group to find out solutions for complex problems related to the current learning objective.	· Edit

Figure 7.10. A Web tool provided by COFALE for the course designer to define, for the students with the loop model on recursion, the content objects to which the learning activity "Examples & Summaries" is related



Edit

Select a mental component or all of them: Loop

Select a learning objective: 1. Recursion

```
Home
☐ 1 Recursion
  ☐ 1.1 Basic concepts
    ☑ 1.1.1 Recursion
    ☑ 1.1.2 DCG strategy
    ☑ 1.1.3 Recursive thinking
    ☑ 1.1.4 Recursive algorithms
    ☑ 1.1.5 Iterative thinking
    ☑ 1.1.6 Iterative algorithms
    ☑ 1.1.7 Recursive methods
    ☑ 1.1.8 Base cases
    ☑ 1.1.9 Recursive part
  ☐ 1.2 Learning situations
    ☐ 1.2.1 Arithmetic expressions
      ☐ 1.2.1.1 Recursive definition
      ☐ 1.2.1.2 Recursive evaluation
      ☐ 1.2.1.3 Recursive evaluation process
      ☐ 1.2.1.4 Java implementation
      ☑ 1.2.1.5 Java test class
    ☐ 1.2.2 Fibonacci numbers
      ☐ 1.2.2.1 First Java implementation
      ☐ 1.2.2.2 Second Java implementation
      ☑ 1.2.2.3 Third Java implementation
    ☐ 1.2.3 Simple text search
      ☐ 1.2.3.1 Recursive definition
      ☐ 1.2.3.2 Definition tracing
      ☐ 1.2.3.3 Recursive algorithm
      ☐ 1.2.3.4 Searching examples
      ☐ 1.2.3.5 Java implementation
      ☑ 1.2.3.6 Java test class
```

The previous design activity must be done before the student's learning session (say, in the pre-active phase). The previous tools, however, are related to Activity 2.1; so, I consider them as tools for the interactive phase. Note that in a virtual learning environment such as COFALE, the human tutor is not always present during the student's learning process, as in a traditional class, to suggest to the student what to do next. Therefore, the course designer should prepare learning activities in advance, as I described earlier, so that COFALE can present them to the learner at appropriate points in time to encourage him or her (see Figure 6.3).

Activities 2.2 – 2.8. After the course designer has carefully prepared different learning materials and has defined relationships among them, as described earlier, COFALE can help the student execute the learning tasks mentioned in Activities 2.2 – 2.8 (see section 6.3.1). For instance, for Activity 2.2, on the basis of the associations among content objects defined by the course designer, COFALE automatically generates the hyperlinks in the menu "Related Topics" (Figures 6.1 and 6.2) to stimulate the learner to explore related concepts or situations. During the student's learning process, however, the tutor should be active and use the available tools to facilitate the student's learning. For example:

- According to Activity 2.3, the teacher should often visit the hyperspace "Resources" (Figure 7.8) to approve (disapprove) the learner's proposals for external resources.
- Regarding Activities 2.5 and 2.6, sometimes the tutor should participate in students' forums and chat rooms to facilitate and stimulate students' discussions.
- According to Activity 2.7, the tutor should monitor the work of each group to give students appropriate feedback via e-mail.
- Regarding Activity 2.8, the teacher should often visit the test manager (Figure 7.4) to evaluate students' individual tests and provide them with feedback through e-mail.

Activity 2.9. In support of this activity, COFALE presents the teacher with a special tool "Course Tracker" (see ATutor's "How To Course") allowing him or her to see how a particular student uses COFALE (see also Figure 6.9). By analyzing these data about the student's navigation history, for instance by using the checklist shown in Table 3.3, the teacher knows whether or not the student explores COFALE in a manner consistent with cognitive flexibility (as described in section 6.3.1), so as to give him or her appropriate feedback, for instance via e-mail.

Furthermore, the teacher could log onto the student's learning space (Figure 6.3: Peers' Learning Hyperspace) to understand more about how the student learns, for example, to see comments, examples, summaries, tests, work in groups, discussions the student made or done during his or her learning process. The teacher could also provide the student with feedback directly in his or her own space "My Tracker" (see Figure 6.9).

For future research, one can modify COFALE so that the teacher can give feedback to a *particular student* by adjusting the student's learning hyperspace directly, for example to insert a particular hyperlink in the student's learning hyperspace to exhort him or her to do a certain learning activity.

Also for future research, it should be useful to detect students' *learning methods* (or strategies) during their learning process in order to adjust teaching and to give feedback to students. For example, analyzing students' learning behavior, the teacher knows that most students tend to find and do exercises and tests before exploring the learning content (see section 9.4.3). Therefore, it may be useful to make explicit hyperlinks to learning situations in the assessment hyperspace and vice versa.

7.2.3 Tools for the post-active phase

Activity 3.1. The tutor can use here the same tools as provided for Activity 2.9. At the end of the learning session of a particular student, however, the tutor should carefully analyze the student's overall learning process (i.e., his or her navigation history, personal examples, comments, summaries, tests, etc.), and communicate the result of the analysis process to the student.

The evaluation work that is individualized for each student is hard work for the teacher, especially if the number of students is large. Therefore, for future research, one can develop several specific tools for COFALE to alleviate the teacher's workload. For example, one can integrate an automatic evaluation tool into COFALE to detect a particular student's "level" of exploring COFALE with respect to cognitive flexibility, according to the data of his or her navigation history that COFALE collects. This tool should be able to identify which criterion or criteria for cognitive flexibility the learner does not satisfy in order to give him or her appropriate feedback or adjust the learning hyperspace for him or her, in an automatic manner.

Activity 3.2. Presently, the tool "Course Tracker" (see ATutor's "How To Course") enables the course designer to review two statistics:

1. "Tools Usage" that shows, for each tool (e.g., resources, chat, forum), the number of hits and average duration of all hits. For each tool and for each hit, it also indicates the date, the duration, the learner, and the originating source (e.g., the content page) from which the learner goes to the tool.
2. "Content Tracking" that shows, for each content page (e.g., "Arithmetic expressions", "Java test class"), the number of hits and average duration of all hits. In addition, for each content page and for each hit, it indicates the date, the duration, the learner, and the access method, for example, via "Local Menu" or "Related Topics".

On the basis of those data, the course designer, the educational researcher, and the software developer may know the effectiveness of a particular tool or content page, so as to improve the current course as well as COFALE. For instance, the course designer can detect that every learner explores a certain content page within a very short time, maybe because it is not compelling; this is a signal for the course designer to improve the content of that page.

For future research, it is useful to provide the course designer, the educational researcher, and the software developer with two specific tools that can help them measure and compare the "level" of support for cognitive flexibility of different courses, according to the set of criteria for cognitive flexibility (see also the checklist presented in Table 3.5). The first tool will help them view statistics indicating the "level" of support for a certain criterion for cognitive flexibility of the learning conditions they prepare. For instance, for criterion MM1, how often text, images, simulations, and Java programs are prepared for different concepts. The second tool will use and analyze the data collected by "Course Tracking", as described previously, to present them with statistics illustrating the effectiveness of the learning conditions provided for learners, regarding the set of criteria for cognitive flexibility. For example, for each criterion for cognitive flexibility (e.g. MP2), the tool should be able to show how many percent of the students respect the criterion (e.g., 80 percent of the students explore linked lists) and how the students satisfy the criterion (e.g., the students explore linked lists several times via the menu "Related Topics"); so, the course designer can conclude that the menu "Related Topics" is useful. The reader should refer back to the checklists presented in Tables 3.3 and 3.5 for this issue.

Main result: COFALE provides the course designer with a set of authoring tools for designing and using learning environments supporting cognitive flexibility.

7.3 Authoring tools for supporting adaptability

To implement adaptation support in COFALE, as in any adaptive learning system, the course designer should take into account learner models carefully (Brusilovsky, 1999; Brusilovsky & Peylo, 2003; Murray, 1999). The learner model of a particular student represents, for instance, the student's personal information and preferences, learning history, test results (Henze & Nejdl, 2001; Weber & Brusilovsky, 2001). Because one of the goals for designing COFALE is to make it as domain-independent as possible, learner modeling in COFALE is rather primitive. In the next paragraphs, I present several tools to help the course designer manage learner models and set up adaptation support for different kinds of students.

Key concept: *Learner model* is a representation of the student's personal information and preferences, learning history, test results, and so on.

7.3.1 General tools for managing learner models

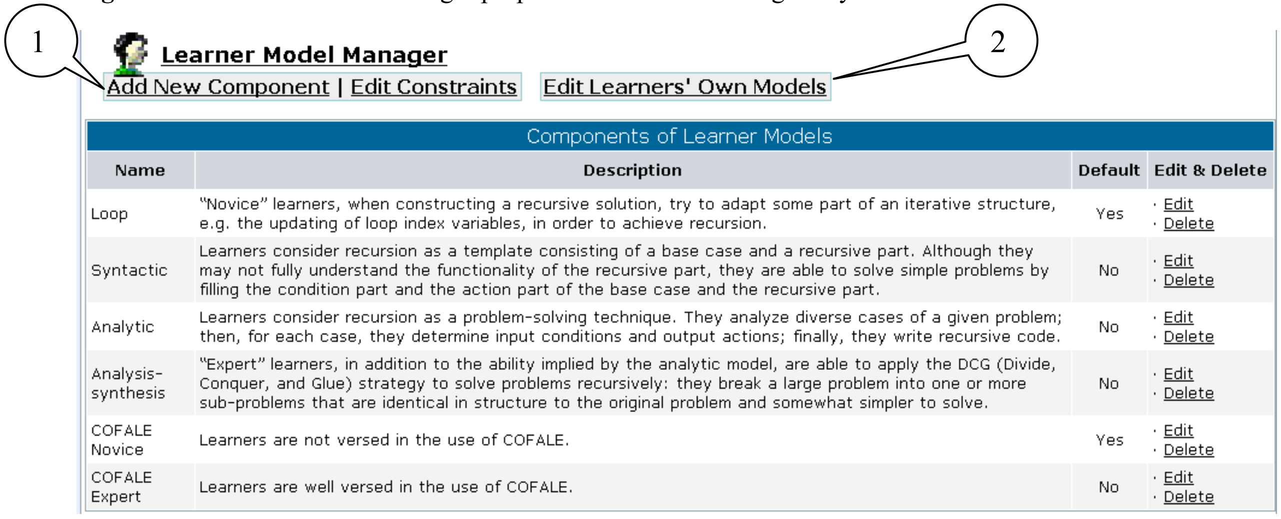
Modeling the learner. In the COFALE environment, I represent each student by a learner model. Each learner model consists of several parts. Each part represents certain characteristics of the student. One should explore ATutor's "How To Course" to understand how COFALE manages the student's personal information and preferences, learning behavior (or navigation history), and test results. In this section, to explain how to implement the adaptation support described in section 6.3.2, I present the course designer with the tools and guidelines for manage two important parts of learner modeling, as follows:

1. *Mental models*. I explained this characteristic of the student in sections 1.5.3 and 6.2.1.
2. *Skill in using COFALE to explore the current learning objective*. I distinguish two levels: "novice" and "expert". The "novice" level indicates that the student is not yet familiar with the way COFALE presents the current learning objective (e.g., recursion), and the "expert" level states that the student understands quite well how to explore the current learning objective in COFALE.

Creating components of learner models. In the learner model manager provided by COFALE for the course designer and the tutor (Figure 7.11), I define a component of learner models to be a mental model or a skill level of using COFALE to explore the current learning objective.

COFALE supports a tool "Add New Component" (Figure 7.11: Area 1) allowing the course designer to add a new component of learner models. For instance, in Figure 7.12, the course designer introduces the description of the loop model on recursion and sets it as default for the course on recursion. When a component is set as default, COFALE will automatically assign it for every new learner (see also section 6.3.2). After creating four mental models on recursion and two skill levels of using COFALE to explore the concept of recursion, the course designer obtains a set of six components of learner models on the particular course on recursion (Figure 7.11).

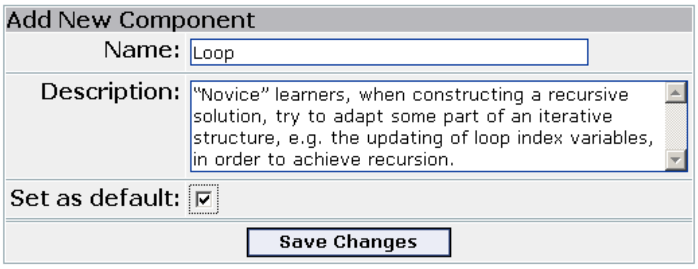
Figure 7.11. Learner model manager proposed to the course designer by COFALE


1
2
Learner Model Manager
Add New Component | Edit Constraints
Edit Learners' Own Models

Components of Learner Models

Name	Description	Default	Edit & Delete
Loop	"Novice" learners, when constructing a recursive solution, try to adapt some part of an iterative structure, e.g. the updating of loop index variables, in order to achieve recursion.	Yes	· Edit · Delete
Syntactic	Learners consider recursion as a template consisting of a base case and a recursive part. Although they may not fully understand the functionality of the recursive part, they are able to solve simple problems by filling the condition part and the action part of the base case and the recursive part.	No	· Edit · Delete
Analytic	Learners consider recursion as a problem-solving technique. They analyze diverse cases of a given problem; then, for each case, they determine input conditions and output actions; finally, they write recursive code.	No	· Edit · Delete
Analysis-synthesis	"Expert" learners, in addition to the ability implied by the analytic model, are able to apply the DCG (Divide, Conquer, and Glue) strategy to solve problems recursively: they break a large problem into one or more sub-problems that are identical in structure to the original problem and somewhat simpler to solve.	No	· Edit · Delete
COFALE Novice	Learners are not versed in the use of COFALE.	Yes	· Edit · Delete
COFALE Expert	Learners are well versed in the use of COFALE.	No	· Edit · Delete

Figure 7.12. A Web tool provided by COFALE for the course designer to add a new component of learner models

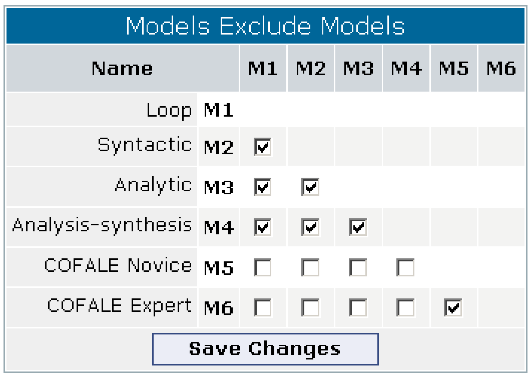

Add New Component

Name:	Loop
Description:	"Novice" learners, when constructing a recursive solution, try to adapt some part of an iterative structure, e.g. the updating of loop index variables, in order to achieve recursion.
Set as default:	✓

Defining constraints of components of learner models. To keep consistency of learner models, COFALE presents the course designer with an explicit tool (Figure 7.13) to define exclusion relations among components of learner models: "C_1 R C_2" means that a particular student cannot possess C_1 and C_2 at the same time; this relation is symmetric ("C_1 R C_2" => "C_2 R C_1"), but not reflective ("C R C") and not necessarily transitive ("C_1 R C_2" and "C_2 R C_3" => "C_1 R C_3"). In Figure 7.13, to define M_i excludes M_j, the course designer selects the checkbox found at the cell (i,j): the four mental models on recursion exclude each other, and the two skill levels of using COFALE to explore the recursion concept exclude each other. Those definitions help COFALE to prevent the user (the student or the tutor) from selecting, for instance by accident, two or more mutually exclusive components of learner models for a certain student, so that COFALE can perform adaptation support accurately.

Updating the learner model for a particular student. At a given moment in the process of learning a particular concept (e.g., recursion), there are two components associated with a particular student: a skill level of using COFALE to explore the current learning objective and a mental model (I assumed in section 6.2.1 that each student possesses one mental model on recursion at a given point in time). For instance, Bob, a new learner of the COFALE environment, possesses the loop model on recursion and the "novice" level of using COFALE to explore the course on recursion provided by COFALE. During the learning process of a certain student, the tutor can update the student's learner model by using a tool "Edit Learners' Own Models" (Figure 7.11: Area 2). For example, after evaluating Bob's tests and learning behavior, the tutor could diagnose that he has reached the syntactic model on recursion and the "expert" level of using COFALE, so the tutor updates his or her evaluation of Bob's learner model (see also section 6.3.2).

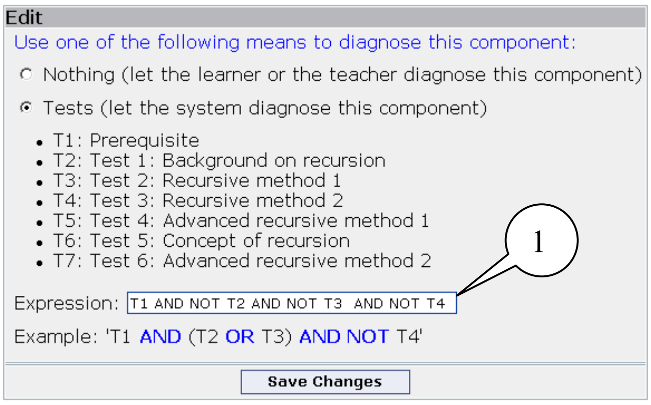
Figure 7.13. A Web tool supported by COFALE for the course designer to define the exclusion relations among components of learner models



Models Exclude Models

Name		M1	M2	M3	M4	M5	M6
Loop	M1						
Syntactic	M2	☑					
Analytic	M3	☑	☑				
Analysis-synthesis	M4	☑	☑	☑			
COFALE Novice	M5	☐	☐	☐	☐		
COFALE Expert	M6	☐	☐	☐	☐	☑	

Figure 7.14. A Web tool provided by COFALE for the course designer to define the logic expression for the automatic diagnosis of the loop model on recursion



Edit

Use one of the following means to diagnose this component:

○ Nothing (let the learner or the teacher diagnose this component)

● Tests (let the system diagnose this component)

- T1: Prerequisite
- T2: Test 1: Background on recursion
- T3: Test 2: Recursive method 1
- T4: Test 3: Recursive method 2
- T5: Test 4: Advanced recursive method 1
- T6: Test 5: Concept of recursion
- T7: Test 6: Advanced recursive method 2

Expression: T1 AND NOT T2 AND NOT T3 AND NOT T4

Example: 'T1 AND (T2 OR T3) AND NOT T4'

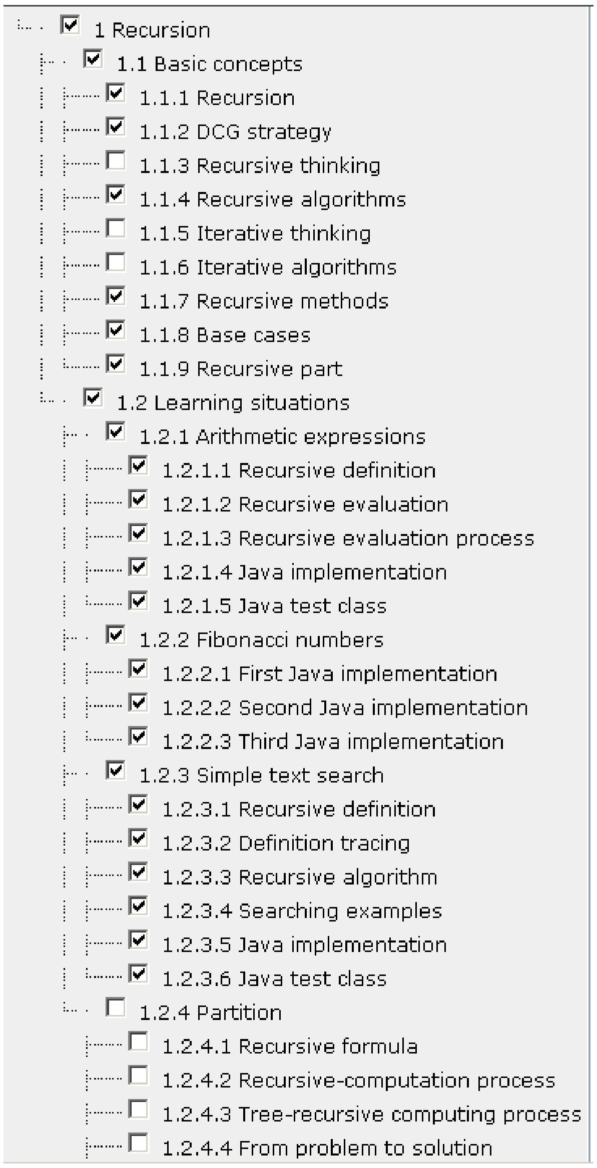
To help COFALE automatically detect the student's mental model, for each mental model, the course designer must define a logic expression in which the variables are the student's results of individual tests. In a textbox of Figure 7.14 (Area 1), for example, the course designer introduces an expression to allow COFALE to automatically detect the loop model on recursion of a particular student: If the student passes test T1 but not tests T2, T3, and T4, then he or she possesses the loop model on recursion.

Note that students can also evaluate their learner model by themselves (see section 6.3.2). Therefore, at any time, there are three evaluations for a particular student's learner model: self-evaluation, evaluation by the tutor, and evaluation by COFALE. At a given time, however, the system takes into account only one evaluation to perform adaptation. This evaluation is selected either by the student (see section 6.3.2) or by the tutor when he or she edits the student's learner model. In a constructivist point of view, I think, the tutor should respect the student's choice.

7.3.2 Tools for implementing adaptation support

Adaptive presentation of learning contents. As mentioned in section 6.3.2, to allow COFALE to perform this adaptation, the course designer must decompose the learning content into fine-grained content objects or pages (see also section 7.2.1); then, the course designer selects the appropriate content objects for each kind of students, according to their mental models. COFALE provides the course designer with a specific tool (Figure 7.15) to help him or her in this work. For instance, in Figure 7.15, the course designer selects the checkboxes next to the content objects he or she wants to present to the students with the loop model on the recursion concept (see also section 6.3.2). Note that although there are two values associated to a particular student (a mental model on the current concept and a skill level of using COFALE to explore the current learning objective), only the student' mental model is considered to perform adaptive presentation of learning contents because there would be no relationship between the second value and this kind of adaptation (the same learning content should be presented to both students with "novice" level and students with "expert" level of using COFALE).

Figure 7.15. A Web tool provided by COFALE for the course designer to define appropriate learning contents for a particular kind of learners



- [✓] 1 Recursion
  - [✓] 1.1 Basic concepts
    - [✓] 1.1.1 Recursion
    - [✓] 1.1.2 DCG strategy
    - [ ] 1.1.3 Recursive thinking
    - [✓] 1.1.4 Recursive algorithms
    - [ ] 1.1.5 Iterative thinking
    - [ ] 1.1.6 Iterative algorithms
    - [✓] 1.1.7 Recursive methods
    - [✓] 1.1.8 Base cases
    - [✓] 1.1.9 Recursive part
  - [✓] 1.2 Learning situations
    - [✓] 1.2.1 Arithmetic expressions
      - [✓] 1.2.1.1 Recursive definition
      - [✓] 1.2.1.2 Recursive evaluation
      - [✓] 1.2.1.3 Recursive evaluation process
      - [✓] 1.2.1.4 Java implementation
      - [✓] 1.2.1.5 Java test class
    - [✓] 1.2.2 Fibonacci numbers
      - [✓] 1.2.2.1 First Java implementation
      - [✓] 1.2.2.2 Second Java implementation
      - [✓] 1.2.2.3 Third Java implementation
    - [✓] 1.2.3 Simple text search
      - [✓] 1.2.3.1 Recursive definition
      - [✓] 1.2.3.2 Definition tracing
      - [✓] 1.2.3.3 Recursive algorithm
      - [✓] 1.2.3.4 Searching examples
      - [✓] 1.2.3.5 Java implementation
      - [✓] 1.2.3.6 Java test class
    - [ ] 1.2.4 Partition
      - [ ] 1.2.4.1 Recursive formula
      - [ ] 1.2.4.2 Recursive-computation process
      - [ ] 1.2.4.3 Tree-recursive computing process
      - [ ] 1.2.4.4 From problem to solution

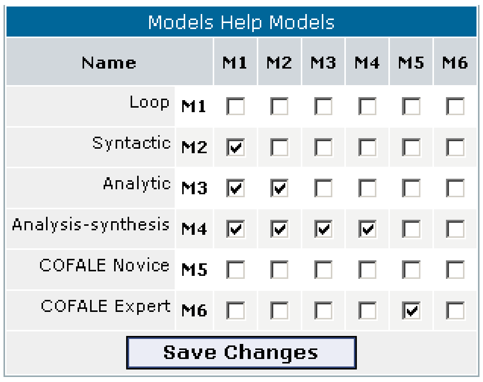
The current version of COFALE does not adapt the order of presenting learning contents to different kinds of students, for example to present arithmetic expressions before simple text search for students with the loop model on recursion but simple text search before arithmetic expressions for students with the syntactic model on recursion. For future research, it should be useful to take into consideration this type of adaptation. Presently, the course designer defines the order of presenting content pages to students (see the presentation for Activities 1&2 in section 7.2.1).

In COFALE, the organization of tests is more or less identical to the one of learning contents. Therefore, for future research, one can provide the course designer with a tool similar to the one shown in Figure 7.15, so that the course designer can implement adaptive assessment in the same way he or she does for adaptive presentation of learning contents.

Adaptive use of pedagogical devices. The reader should return to the presentation for Activity 2.1 in section 7.2.2 to understand the tools provided by COFALE for the course designer to implement this type of adaptation.

It is worth to note that there are two values associated to a particular student: a mental model on the current concept and a skill level of using COFALE to explore the current learning objective. Thus, the learning activities proposed to a particular student are the union of the learning activities the course designer defines for his or her mental model with the ones the course designer selects for his or her skill level of using COFALE. For example, for the "Java test class" (the last item of arithmetic expressions shown in Figure 7.10), and for the loop model on recursion, the course designer defines 7 activities related to the learning of recursion (Figure 7.9: Personal Comments, Examples & Summaries, Other Resources, Peers' Learning Hyperspace, Tests, Discussions, and Collaboration). For the same page "Java test class" and for the "novice" level of using COFALE to explore the recursion concept, the course designer selects 3 activities related to the using of COFALE (Figure 7.9: Next Page, Related Topics, and Learning History). So, at the bottom of the page "Java test class" and for the student Bob with the loop model and the "novice" level, COFALE presents him with 10 activities (see also section 6.3.2).

Figure 7.16. A Web tool supported by COFALE for the course designer to define appropriate peers for a particular kind of learners



Models Help Models

Name		M1	M2	M3	M4	M5	M6
Loop	M1	☐	☐	☐	☐	☐	☐
Syntactic	M2	☑	☐	☐	☐	☐	☐
Analytic	M3	☑	☑	☐	☐	☐	☐
Analysis-synthesis	M4	☑	☑	☑	☑	☐	☐
COFALE Novice	M5	☐	☐	☐	☐	☐	☐
COFALE Expert	M6	☐	☐	☐	☐	☑	☐

Adaptive communication support. To help the course designer in implementing this kind of adaptation, COFALE presents a tool (Figure 7.16) allowing him or her to define help relations among components of learner models: "C_1 R C_2" means that students possessing C_1 can help students possessing C_2 overcome difficulties about acquiring the current learning objective; this relation may be reflective, but not necessarily symmetric and transitive. In Figure 7.16, to define M_i helps M_j, the course designer selects the checkbox found at the cell (i,j): M_2 helps M_1; M_3 helps M_2 and M_1; M_4 helps M_4, M_3, M_2, and M_1; M_6 helps M_5. In general, students with more advanced models could help students with less advanced ones (see also section 6.3.2). On the basis of those associations, COFALE can automatically find and suggest a list of appropriate peers to a particular student (see Figure 6.12).

Main result: COFALE provides the course designer with a set of easy-to-use tools for implementing certain adaptation support.

7.4 Discussion

It is worth noting that the design and use of COFALE are based on a constructivist or learner-centered approach. Thus, many teacher-centered issues (e.g., representation of teaching methods) are not discussed here.

COFALE provides the course designer and the tutor with a set of Web-based, easy-to-use tools for designing and using adaptive learning environments supporting cognitive flexibility. The course designer's workload for making a course available in COFALE is not particularly high (about 8 person-hours for the course on recursion), because COFALE supports many learning activities without intervention of the course designer. The tutor's workload, however, is significantly high in the case of evaluating learning behavior and tests of a large number of students (see chapter 9 to know why). Therefore, for future research, it is necessary to improve COFALE to automate several processes such as evaluating students' tests and evaluating students' learning behavior with respect to cognitive flexibility. Various techniques have been proposed for automatic assessment (Ala-Mutka, 2003; Malmi & Korhonen, 2004). It is also necessary to provide students with additional tools stimulating and facilitating them to review each other's work (peer review). Several tools (e.g., SWoRD by Cho & Schunn, 2004) and studies (e.g., Cho & Schunn, 2003) have showed that peer collaboration is effective and peer grading is reliable and valid.

In this chapter, I explained guidelines and tools for the teacher to be able to satisfy all of the criteria for cognitive flexibility. In practice, however, it is not necessary to always satisfy *all* of the criteria. The choice of the teacher depends on his or her own teaching context (see the discussion in section 2.4.5).

COFALE is a ***domain-independent*** platform, meaning that it can be used to design "courses" for many domains, from sciences and mathematics to economics and literature. Indeed, COFALE is based on ATutor, claimed to be domain-independent (Adaptive Technology Resource Center, 2004). Additionally, the features COFALE has added on to the ATutor system are also domain-independent. For example, the pedagogical device proposed for the learner to produce summaries on multiple points of view (see Figure 6.5) is an empty table; so, it could allow the learner to produce summaries in table form in any domain. The learner model manager (see section 7.3.1) could allow the course designer to manage a number of components of learner models in any subject (e.g. see Figure 7.12). The adaptation implementer (see section 7.3.2) could also enable the course designer to personalize learning contents, learning activities, and communication facilities for the student in any subject (e.g. see Figure 7.15).

I believe that teachers, even with limited knowledge of computing science (e.g., without programming skill), can use COFALE to design his or her own "courses". Indeed, ATutor has been using by many teachers in the domains different from computing science (Adaptive Technology Resource Center, 2004). Moreover, the tools provided by COFALE is as easy to use as the ones with which people are familiar, for instance, MSWord, Web browsers. The present chapter, however, does not support complete documentation for the use of the COFALE system. I think it is useful to provide practitioners with such documentation so that they can make their own courses available in COFALE in a manner similar to the one described in this chapter. It is also useful to collect feedback from practitioners in order to improve both the learning tools and the authoring tools of the COFALE learning system.

In section 7.2.1, I showed that it is quite straightforward for the course designer to add and manage content objects. It is, however, hard for the course designer to add a pedagogical device to COFALE without intervention of the software developer. For instance, if the course designer wants to provide students with a concept-mapping tool, for example IHMC's CmapTools (Institute for Human and Machine Cognition, 2005), permitting students to (re)structure their own knowledge, the course designer must ask the software developer to insert a hyperlink to IHMC's website and its description to COFALE's student tools. If the course designer desires to integrate a specific tool (see section 6.5) into COFALE to support recursive problem solving by students, the course designer must ask the software developer to modify the source code of COFALE to add such a complex tool to COFALE's student tools. *How to integrate a specific pedagogical device (e.g., a software component) into an existing learning system (e.g., COFALE) without intervention of the software developer should be an exciting but difficult problem for educational technologists*.

One of the objectives in the design of COFALE is to make it domain-independent so that teachers in diverse domains can benefice it. Thus, the treatment of learner modeling and support for adaptability is technically superficial. Indeed, in COFALE, the course designer specifies different kinds of students in advance, and what and how learning materials are presented to a particular kind of students. I believe, however, that this approach is effective because in principle the course designer is well versed in the subject he or she is designing.

In comparison with the adaptive learning systems analyzed in section 5.3, COFALE effectively supports adaptive presentation of learning contents, adaptive use of pedagogical devices, and adaptive communication support. In addition, I believe that it is possible to implement adaptive assessment in COFALE in the same way adaptive presentation of learning contents has been carried out in COFALE (see section 7.3.2). Thus, I would say adaptation support in COFALE is *more or less comparable* to that present in the systems analyzed in section 5.3.

Several researchers (e.g., Brusilovsky & Peylo, 2003) have claimed that a more adaptive learning system improves learning outcomes for students. So, for future research, it should be useful to improve COFALE's adaptability. For instance, one can modify COFALE so that the teacher can add a particular software component to COFALE to support problem solving by students adaptively. One can also improve COFALE to support adaptation in a course in which many learning objects are interrelated, meaning that when the student is exploring a certain learning object, the system should personalize the learning materials for him or her, taking into account his or her knowledge of the learning objects related to the learning object being explored. For example, in an introductory course on object-oriented programming and Java in which classes, interfaces, objects, methods, variables, control flow, and so on are related one another, when the learner is exploring control flow, the learning situations the system proposes for a learner with little experience in variables should be different from the learning situations the system proposes for a learner with much experience in variables (because the concept of control flow is closely related to the concept of variables).

Note: It is possible to explore the COFALE authoring environment as a teacher at the following address: `http://renoir.info.ucl.ac.be/elearning/Cofale/login.php`; one can ask me for an instructor account.

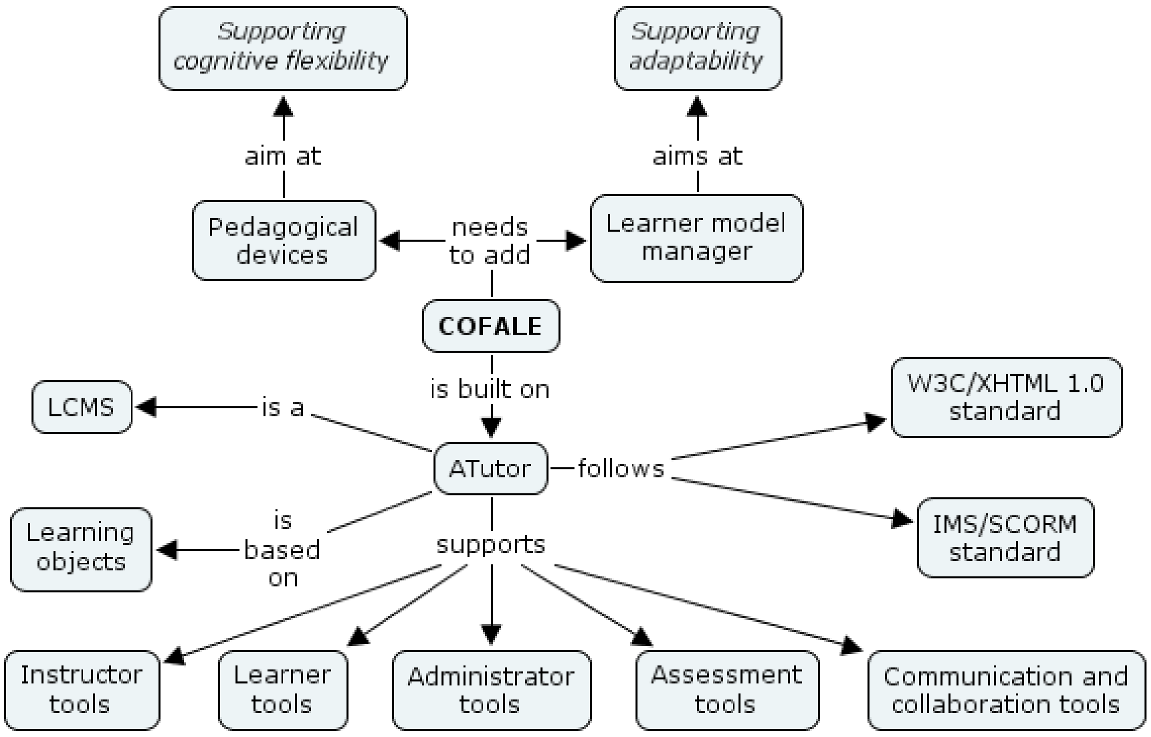
CHAPTER 8

COFALE: Implementation

"*The devil is in the details.*"

English Idiom (cited in Masie Center, 2003, p. 34)

This chapter reports the main points in the development of the COFALE learning environment. COFALE is based on ATutor, an open-source, Web-based learning content management system. The contribution of COFALE to ATutor is about 20 percent of the source code, meaning 5000 lines of PHP code and 1500 person-hours of programming work. After reading this chapter (and Appendix D), one *should* be able to modify the source code of ATutor or COFALE to build on a new learning platform, if he or she has good programming skill.



Summary

8.1 Introduction

The implementation of COFALE, based on ATutor, was a complex process. I exploited the *available* ICT to build COFALE, meaning that the present dissertation does not contribute new and important techniques in the domain of computing science. Therefore, in the present chapter, I show only the key points about how COFALE was developed. For instance, what were the main difficulties to build COFALE on the ATutor existing system? How to overcome these difficulties?

8.2 Implementation of ATutor

ATutor is an *open-source*, Web-based learning content management system (LCMS) designed and maintained by the ATRC group (Adaptive Technology Resource Center 2004). Presently (April, 2005), the ATRC group supports good documentation of "How To Course" for every released version of ATutor. It also provides preliminary ATutor developer documentation.

Figure 8.1. General architecture of ATutor

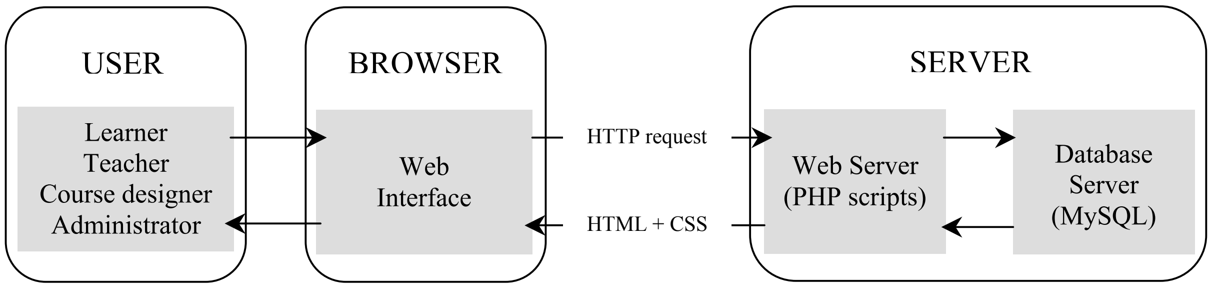


Figure 8.1 shows a very schematic architecture of the ATutor system. In general, the functionality of the system can be explained, as follows:

- The user uses a Web browser to log into the system and make a request. For example, the learner asks the system to present the definition of the recursion concept.
- Taking into account the user's request, the browser sends a HTTP request to the Web server in which a set of PHP scripts was installed. For instance, the browser sends the Web server several parameters with the name of the PHP script that is responsible for presenting the definition of the concept of recursion.

- Depending upon the kind of the HTTP request, the Web server will create new data or update existing data or retrieve existing data by connecting the MySQL database server in which all data of the user, learning content, tests, forums, and so on are stored and indexed. Then, the Web server formulates a HTML file including a CSS format and sends it back to the browser. For example, the responsible PHP script uses the parameters provided by the HTTP request to retrieve the information of the recursion definition from the database and build a HTML file containing the definition of the concept of recursion.
- On the basis of the HTML file and the CSS format received from the Web server, the browser will create a Web page and present the user with the created page. For instance, the browser shows the learner with the Web page presenting the recursion definition.

The structure of the database in ATutor is simple: There are totally about 30 tables organized in small groups. In Appendix D, I present two examples so that the reader can have a global view about the way ATutor represents information such as the user, learning content, and tests. ATutor's source code is organized into a tree-structured file system, which contains a variety of short PHP scripts.

The reader should look at the ATutor developer documentation (Adaptive Technology Resource Center 2004) for more information about how the system was developed, including how learning objects were implemented.

8.3 Implementation of COFALE

At the beginning of the development of COFALE, I searched the Internet for several LCMSs and I made a preliminary evaluation of each system, as follows:

- If I were to use the system to design a course, which criteria for cognitive flexibility would be satisfied without modifying the system's source code?
- How easy is it for me to modify the system's source code to satisfy all the criteria for cognitive flexibility, as I described in section 6.3.1?
- How easy is it for me to modify the system's source code to add the learner model manager and implement adaptation support, as I described in sections 6.3.2?

The previous analyses (sections 5.1, 5.2.4, and 8.2) showed that ATutor is a good LCMS; besides, its internal organization is not particularly complex. In addition, if the course designer is versed in the use of the set of criteria for cognitive flexibility presented in chapter 2, it is possible for him or her to implement in ATutor learning conditions satisfying about two thirds of the set of criteria (see section 5.2.4). Therefore, it seemed to me that ATutor was a good base to implement learning conditions satisfying all the criteria for cognitive flexibility and supporting adaptability.

The main difficulty to build on COFALE from ATutor is to understand the internals of the ATutor system. This process took time and effort because, when I first used ATutor, there was nothing more than its source code available to me.

To overcome this difficulty, I carefully examined the tables in ATutor's database and many important PHP scripts in its source code. I also tried to modify a number of PHP files and saw the effects of the modification on a Web browser. Once I understood the main functionality of the ATutor system, it was easy for me to modify it and to add software and database components to it in order to achieve COFALE's requirements.

In the following paragraphs, I show three examples explaining how to add software and database components to ATutor in order to achieve certain requirements of COFALE. The reader should refer to Appendix D to understand more about how I added the learner model manager to ATutor, and how I implemented adaptation support in ATutor.

Example 1. To satisfy a part of criterion MP2 and to help the learner navigate intelligently to avoid getting lost in the learning hyperspace (see section 6.3.1), I created the menu "Learning History" (see Figure 6.1), as follows.

For the database. Examining ATutor's database, I discovered that I can use the data in the existing table `g_click_data` to create the menu "Learning History", because this table contains the information about the navigation history of every learner. Here is its description:

g_click_data	
FK2	member_id
FK3	course_id
FK4	from_cid
FK5	to_cid
FK1	g
	timestamp
	duration

`course_id`: The identity of the course the learner is exploring.
`member_id`: The identity of the learner.
`from_cid`: The identity of the content object from which the learner passes to the content object identified by `to_cid` (see Appendix D).
Note: FK stands for foreign key.

For the source code. In principle, constructing the menu "Learning History" and constructing the menu "Related Topics" (see Figure 6.1) is similar. Therefore, I searched ATutor's PHP file system for all segments of code related to the implementation of the menu "Related Topics". Then, I carefully examined those segments of code. Finally, I implemented the menu "Learning History" in the same way "Related Topics" had been created.

Example 2. To satisfy criterion MP4, I had to provide students with an explicit tool (Figure 6.14: My Own Summary) allowing them to synthesize multiple points of views. Here is the process I used to do so.

For the database. To register students' summaries, on the basis of ATutor's table `content` (see Appendix D: Figure D.1), I created table `content_of_learners`, as follows:

content_of_learners	
PK	content_of_learners_id
FK1	course_id
FK2	learning_object_id
FK3	member_id
	type
	ordering
	last_modified
	revision
	formatting
	release_date
	keywords
	content_path
	title
	text
	inherit_release_date

`course_id`: The identity of the course the learner is exploring.

`learning_object_id`: The identity of the learning object the learner is exploring (this attribute corresponds to `content_id` in table `content`, see Appendix D: Figure D.1).

`member_id`: The identity of the learner.

`type`: The kind of the learner's work (an example or a concept map or a summary).

`title`: The title of the learner's work.

`text`: The content of the learner's work.

…

Note: PK stands for primary key, FK for foreign key.

For the source code. Here is the process:

1. I logged into the recursion course designed in COFALE as a learner.
2. I selected the menu "Tools" to know the PHP file needed to modify in order to add the desired learning tool: `tools/index.php` was found.
3. I examined this file to understand how learner tools had been implemented.
4. I modified this file to display the desired learner tool (Figure 6.14: My Own Summary). This tool is linked to a file `learners/my_own_content.php` created by me. This file leads the student to the actual tool for constructing his or her own summary. The techniques used to write this file are similar to the ones used to write ATutor's PHP file that helps the course designer add learning content: a part to display Web interface for the user and another part to save the data submitted by the user to the database (Figure 7.2).

Example 3. To satisfy a part of criterion MP3, I created a specific tool (Figure 6.14: Peers' Learning Hyperspace) for the student. To construct this tool, I needed to modify only the source code, in the same way I did in the previous example. Note that ATutor supports automatic login using cookies. So, it is straightforward to allow the student to log into peers' learning space without knowing their password: When the student selects a peer's login, the system will set the value of variables `login` and `password` (stored in the system's cookie) to the peer's login and password; then, the system will call a PHP file serving for automatic login to lead the student to the peer's learning hyperspace. Similarly, when the student selects a hyperlink to return to his or her own learning hyperspace, the system will reset the value of variables `login` and `password` to the student' login and password and make an automatic login to bring the student back to his or her homepage.

8.4 Discussion

In this chapter, I have shown the way to build on a new e-Learning platform (COFALE) from an existing LCMS (ATutor). The point I make here is *the possibility to modify the source code of a LCMS in order to create guidelines and models for effective pedagogical principles implied from a relevant learning theory* (i.e., cognitive flexibility, one important facet of constructivism). One more time, the set of operational criteria has been applied as guidelines to the development of the new learning platform. I have also illustrated a simple way to add the adaptability on to the LCMS.

My contribution to the ATutor system is approximately 20 percent of the source code, meaning 5000 lines of PHP code (1500 person-hours of programming work). Although COFALE's source code has not been optimized yet, it seemed to work well when I carried out an experiment with actual students (see chapter 9). For future work, it would be useful to optimize the source code of COFALE to make it more stable and powerful.

To respect the GNU General Public License (1991), which encourages the free use, modification, and distribution of open-source projects such as ATutor and COFALE, I have created a website for the open-source COFALE project, available at the following address: `http://renoir.info.ucl.ac.be/elearning/COFALE.html`, from which one can download both the source code and the documentation for the development and use of COFALE.

PART FOUR: EVALUATION

CHAPTER 9

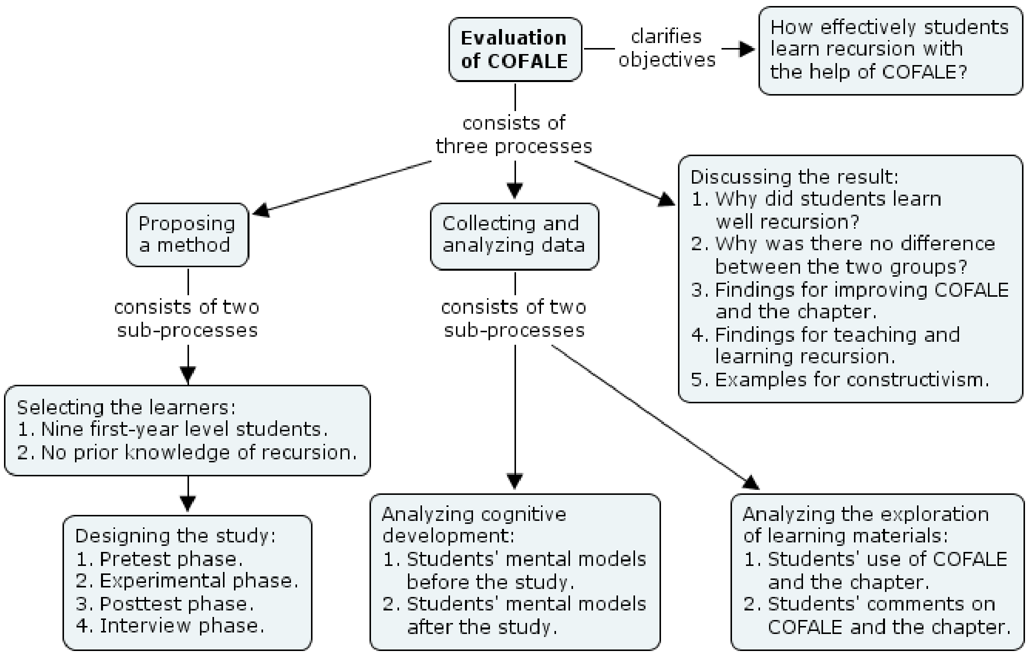
A preliminary evaluation of COFALE

"*No amount of experimentation can ever prove me right; a single experiment can prove me wrong.*"

Albert Einstein, German Scientist, 1879 – 1955 (cited in Suomela, 2005)

(Reference to Appendix C)

A 2-week-long survey was carried out to evaluate the COFALE learning environment. Nine first-year engineering students were selected for the study: four learners in the COFALE group and five learners in a traditional approach. Both groups were given the same 45-minute-long lecture and 2-hour-long programming homework about recursion. In addition, within 1 hour, the COFALE group was exposed to explore COFALE and the traditional group a chapter of a reference book. After the learning session, both groups of learners developed the ability to build recursive solutions to a variety of problems, and were interested in learning with the help of COFALE or the chapter. The survey did not provide clear evidence for the difference of *learning outcomes* between the COFALE group and the traditional one. The main differences were *learning motivation* and cognitive flexibility behavior: Students had more motivation for learning with COFALE than for learning with the book chapter, and the COFALE group's learning behavior seems to be somewhat more consistent with cognitive flexibility than the traditional group's. I claim that, to evaluate the complete effectiveness of ICT-based learning conditions exhibiting the desired characteristics of cognitive flexibility, we should perform long-term experiments with a significant number of students.


Evaluation of COFALE
clarifies objectives
How effectively students learn recursion with the help of COFALE?
consists of three processes
Proposing a method
Collecting and analyzing data
Discussing the result:
1. Why did students learn well recursion?
2. Why was there no difference between the two groups?
3. Findings for improving COFALE and the chapter.
4. Findings for teaching and learning recursion.
5. Examples for constructivism.
consists of two sub-processes
consists of two sub-processes
Selecting the learners:
1. Nine first-year level students.
2. No prior knowledge of recursion.
Designing the study:
1. Pretest phase.
2. Experimental phase.
3. Posttest phase.
4. Interview phase.
Analyzing cognitive development:
1. Students' mental models before the study.
2. Students' mental models after the study.
Analyzing the exploration of learning materials:
1. Students' use of COFALE and the chapter.
2. Students' comments on COFALE and the chapter.

Summary

9.1 Introduction

In the previous chapters, I showed a useful approach based on operational criteria for the design and use of COFALE, an ICT-based adaptive learning environment truly exhibiting the desired characteristics of cognitive flexibility, according to educational theorists (e.g., Bourgeois & Nizet, 1999; Driscoll, 2000; Spiro & Jehng, 1990). A number of studies showed positive results that pedagogical models proposed for cognitive flexibility helped students in advanced knowledge acquisition (Spiro & Jehng, 1990). The implementation of learning conditions fostering cognitive flexibility in an e-Learning platform (COFALE), however, is relatively new. Thus, it is necessary to carry out a certain number of surveys to evaluate various aspects of the COFALE systems.

In a constructivist point of view, to evaluate COFALE, as a new adaptive learning system supporting cognitive flexibility, I take into account, for instance, the following questions (see also Reeves & Okey, 1996; Wilson, 1997; Wilson et al., 1995):

- *Do learning conditions provided by COFALE foster students' cognitive flexibility effectively*? The questions for this kind of studies include: After learning with the help of COFALE, do learners take into account multiple aspects of a situation or problem systematically? Do learners express their personal points of view clearly, give pertinent feedback to peers' points of view, and produce good summaries on multiple points of view? Does this meta-cognitive knowledge of students help them mastering other subjects faster and more effectively?
- *What are outcomes of learning a particular concept with the assistance of COFALE*? For example, how effectively do students learn the concept of recursion with the help of the COFALE learning environment? Why?
- Do students follow suggestions proposed by COFALE, for example to explore related concepts and do learning activities presented at the bottom of each page? More specifically, do their learning processes respect all of the criteria for cognitive flexibility?
- Does COFALE provide a learning experience that is really tailored to the needs of the individual student? Does learning with adaptation support help students attain the learning objective faster and more effectively than learning without support for adaptability?

- What are students' reactions to the COFALE learning environment? Do they like or dislike the way COFALE presents them with learning experiences?

In my point of view, to persuade teachers to use COFALE, it is necessary to show them at least two evidences. Firstly, we need to show that the conditions of learning proposed by COFALE truly facilitate and stimulate cognitive flexibility in students. Secondly, we need to convince that COFALE really helps improving students' learning outcomes, for instance to master a particular concept to a significant degree.

To provide the first evidence, however, it is necessary to carry out a *long-term* experiment with a significant number of students because the characteristics of cognitive flexibility are complex (Spiro et Jehng, 1990). It is also necessary to construct an *effective means* (e.g. a set of criteria for the assessment of cognitive flexibility) in order to perform this kind of experiments effectively.

In the context of this PhD thesis, I did not have the means to completely answer all of the questions I mentioned earlier, particularly the question about whether COFALE fosters cognitive flexibility effectively. Therefore, this chapter describes a *short-term* study (a *preliminary* evaluation) whose main purpose is to evaluate how students master a particular concept (i.e. the changes of mental models about the concept of recursion identified in chapter 6) with the help of COFALE (an e-Learning approach) in comparison with the help of a chapter of a reference book (a traditional approach). I decided to choose a chapter that was designed for teaching recursion by other authors because this choice could make the experiment as objective as possible. I also analyze students' feedback on learning with the help of COFALE or the book chapter, and when possible, I look into the difference between the two groups, regarding their cognitive flexibility behavior.

In the study, I used a method borrowed from a successful survey of teaching recursion by Bhuiyan and associates (Bhuiyan et al., 1994). An important characteristic of this method is to provide learners with a realistic learning situation. In the study, I attempted to simulate such a situation for both groups of students to see the effects of COFALE in real environments. The study involved two groups of learners: the COFALE group (four students) and the traditional group (five students). The selected students had no knowledge of recursion, and their learning objective was to develop the ability to solve problems recursively. Both groups participated in the same 45-minute-long lecture and were asked to do the same 2-hour-long homework. The main difference was that the COFALE group explored the COFALE learning environment within 1 hour and the traditional group explored a chapter about recursion in a reference book "Java software solutions" (Lewis & Loftus, 2003) within 1 hour as well.

The next sections show and discuss the evidences I found in the survey.

9.2 Method

I first describe the learners selected for the study. Then I present the materials and the protocol used to perform the experiment.

9.2.1 Selecting the learners

Because recursion is a high-level programming concept, the study was targeted at students having knowledge of programming and no knowledge of recursion. So, I decided to select first-year engineering students in FSA (*Faculté des sciences appliquées, Université catholique de Louvain*). Those students had been registered in an introductory course on object-oriented programming and Java before the study. To engage students in participating in the study, I sent them an e-mail in which I showed the elegance of recursive solutions and the usefulness of recursion in drawing fractals, and I offered each participant two movie tickets. To make the result of the study as convincing as possible, I wanted about 20 students for the study; however, only 11 volunteer students were initially registered. Two students dropped out of the study leaving four members in the COFALE group, and five learners in the traditional one. The two groups were randomly constituted. The COFALE group learners were labeled C1 through C4, and the traditional group learners were labeled T1 through T5.

In the introductory course on object-oriented programming and Java, the selected students were taught in a PBL (problem-based learning) approach (Pedagogy Group at FSA/UCL, 2005), and they were able to learn the concept of linked lists within 1 week (about 8 hours). The main reference book for this course was "Java software solutions" (Lewis & Loftus, 2003). Table 9.1 presents the grade of the final exam of those students in this course (the average grade of all students in this course was 11.40/20). Table 9.1 shows that: (a) although the COFALE group was more "homogeneous" than was the traditional one, the two groups were more or less equal in academic capacity; and (b) the selected students were highly graded among about 280 students in this course.

Table 9.1. The grade of the final exam of the selected students in the introductory course on object-oriented programming and Java

COFALE group		Traditional group	
Label	**Grade (/20)**	**Label**	**Grade (/20)**
C1	17	T1	12
C2	13	T2	19
C3	16	T3	17
C4	13	T4	15
		T5	11
Average	**14.75**	**Average**	**14.80**
Standard deviation	**2.06**	**Standard deviation**	**3.35**

9.2.2 Design of the study

In carrying out an evaluation, we must always avoid bias, meaning that we should not, for example, do the best for the COFALE group and do the worst for the traditional one. Therefore, in the study, I tried to provide the two groups of learners with the same learning conditions except for COFALE and the chapter of the reference book, two tools that I wanted to compare. I organized the study into four phases: the pretest phase, the experimental phase, the posttest phase, and the interview phase.

Pretest phase

The pretest phase aimed at identifying students' mental models on the recursion concept before the learning session. All of the selected students took the same paper-and-pencil pretest (see Appendix C) within *15 minutes*. The first two questions in the pretest were borrowed from a study of Götschi and colleagues (Götschi et al., 2003). These authors claimed that those two questions could be used to determine students' certain mental models on recursion. I also proposed the last three questions in the pretest to gather more information about what students had really thought about recursion before the learning session. Students' comprehension about recursion at this point of time was also verified one more time in the interview phase.

Experimental phase

The objective of this phase was to provide each group of learners with a realistic learning situation for developing the ability to solve problems recursively. So, I decided to integrate COFALE or the chapter of the reference book as a part of the learning materials provided for students. This phase was divided into three sessions:

1. *Lecture*. I gave the same lecture on recursion within *45 minutes* to both groups of students, although in separate sessions because of students' constraints of time. In the lecture, I first introduced the DCG (Divide, Conquer, and Glue) strategy in problem solving with several examples. Then, I explained the concept of recursion used in the Java programming language. Finally, I showed how to apply recursion and Java to solve a diversity of simple problems and a problem about drawing a snowflake. The content of the lecture was essentially based on a chapter about recursion written by Kjell (2003).
2. *Exploration*. After the lecture, each group immediately explored the tool I provided within *1 hour*. The learners in the COFALE group explored the COFALE learning environment. Because the robot situation presented in the test section of COFALE was reserved for the posttest of the study, I replaced it with several exercises presented at the end of chapter 11 of the reference book (Lewis& Loftus, 2003). The learners in the traditional group explored chapter 11 of the reference book (Lewis& Loftus, 2003) with which they had been familiar in the introductory course on object-oriented programming and Java (the content of this chapter was summarized in Appendix B6).

   With the COFALE group, I let students explore themselves the COFALE environment; however, to simulate a realistic learning situation, I was active online to answer students' questions and to give suggestions to students through communication tools such that e-mail, forums, chat rooms. With the traditional group, I let students read themselves the chapter of the reference book. I also observed and guided their reading and answered their questions.
3. *Homework*. At the end of the exploration session, I gave the same programming homework (see Appendix C) to the two groups of students. I estimated the time for students to finish the homework to be about *2 hours*. I explicitly encouraged the learners in the COFALE group to use COFALE and the learners in the traditional group to use the chap-

ter of the reference book to help solving the problems presented in the homework. I asked both groups to send me both their Java program and their brief report through e-mail.

Posttest phase

The purpose of the posttest phase was to identify students' mental models on recursion after the learning session. All of the selected students took the same paper-and-pencil posttest (see Appendix C) within about *1 hour*. In the posttest, I had students confront a complex situation whose nature made them to think recursively to be able to solve the problems. I also asked them the same three questions presented in the pretest to see the changes in their understanding of recursion. To facilitate the analysis process, I collected students' tests as well as draft sheets. The posttest analysis was also verified in the interview phase.

Interview phase

The interview phase aimed to gather additional information to identify students' changes of mental models on recursion more precisely. It also helped me understand the type of difficulties students had encountered with recursion and the way they had used the available learning materials in the learning session. Both groups of learners were asked the same questions (see Appendix C) within *15 minutes*. All of the interviews were recorded in WAV format using an audio-recorded computer program and saved on two CD-ROMs.

9.3 Result

On the basis of the collected data, in this section I analyze two important issues: (a) how students developed the ability to solve problems recursively, and (b) how students used the available learning materials during the learning session.

9.3.1 Cognitive development

To see students' cognitive development, I analyze their mental models on recursion *before* and *after* the learning session.

Before the learning session

The data used to identify learners' mental models on recursion before the learning session were their pretest and a part of their interview.

Table 9.2 summarizes the evaluation of learners' pretest (see the pretest in Appendix C). For the question 1, "incomplete" means that learners did not consider the base case of the recursive method (i.e. their answer was "hheelloo"). For the question 2, I marked "correct" if learners were able to trace the computation process of the recursive method (although the final answer several learners provided was incorrect because of miscalculations). The evaluation of the first two questions showed that the learners in the traditional group traced recursive methods significantly better than did the COFALE group learners. The evaluation of the last three questions showed that both groups of learners had some basic understanding of recursion but more or less deduced from iteration.

Table 9.2. Pretest analysis

Learners	Question 1	Question 2	Question 3	Question 4	Question 5
C1	Incomplete	Correct	Call a code many times	Repeat the same instruction	Solve the sub-problems
C2	Incomplete	Give up	An alternative of loop	Give up	Give up
C3	Correct	Correct	Call itself	Problem solving technique	Use the same method to obtain a solution
C4	Incomplete	Correct	Call itself	Something repeated many times	Do the same action many times
T1	Correct	Correct	Call itself, have an exit of loop	Method that calls itself	Recursion = a type of iteration in itself
T2	Correct	Correct	Call itself, have an exit of loop	Method that calls itself	Recursion = repetition
T3	Correct	Correct	Call itself, have an exit of loop	Find the result according to previous results	Solve problems by iteration, simplification
T4	Correct	Correct	Call itself, have an exit of loop	Find the result according to previous results	Solve problems based on previous results
T5	Correct	Correct	Call itself, have an exit of loop	Loop with the same tool (method)	Find and solve different cases

The pretest analysis also provided the evidence that students demonstrated different levels about the trace model. I distinguished the following five levels:

1. *Novice*. Students cannot trace any recursive method.
2. *Beginner*. Students can trace several simple cases but the result is sometimes incomplete.
3. *Intermediate*. Students can trace correctly a variety of simple cases.
4. *Advanced*. Students can trace correctly a diversity of cases, simple or complex.
5. *Expert*. Students, in addition to the ability implied by the advanced level, actively use the trace approach to verify the correctness of their recursive solutions.

Table 9.3. Interview analysis on learners' mental models on recursion before the learning session

Learners	Interview analysis
C1	"I was familiar with the loop concept [for, while] to solve problems."
C2	"I had no idea about what recursion is and for what it is used."
C3	"I saw recursion in the snowflake but I did not imagine how it works."
C4	"I knew a method calls itself but I did not see how it works, it has base cases, and how to build it."
T1	"I had no idea about the concept of recursion."
T2	"I did not know the concept of recursion."
T3	"I only saw recursion as a recursive series: we cannot find a term without previous terms […] I thought we use iteration rather than recursion."
T4	"I heard about recursion but did not see what it is and for what it is used."
T5	"I saw recursive formulas in math […] I knew a method calls itself but I did not know base cases."

Table 9.3 present the analysis of learners' interview (see Appendix C for the interview question), I believe that all of them possessed the loop model on recursion before the learning session because they claimed they could not build recursive methods. Table 9.4 summarizes the mental representations about recursion of both groups of learners before the learning session.

Note that I did not prepare any materials to evaluate students' cognitive flexibility behavior before the learning session. Because students were randomly organized into the two groups, I assume that, before the learning session, the two groups have the same initial "level" regarding cognitive flexibility behavior.

Table 9.4. Learners' mental models on recursion before the learning session

Note that students with intermediate level of the trace model may possess higher levels of this model (they traced the two answers of the pretest correctly, so they may trace more complex tests correctly).

COFALE group		Traditional group	
Learners	**Mental models**	**Learners**	**Mental models**
C1	Loop model, trace model (beginner)	**T1**	Loop model, trace model (intermediate)
C2	Loop model, trace model (novice)	**T2**	Loop model, trace model (intermediate)
C3	Loop model, trace model (intermediate)	**T3**	Loop model, trace model (intermediate)
C4	Loop model, trace model (beginner)	**T4**	Loop model, trace model (intermediate)
		T5	Loop model, trace model (intermediate)

Main result: Both groups of learners possessed the loop model before the learning session. The traditional group had somewhat better skills in tracing recursive methods than did the COFALE group.

After the learning session

The data used to identify learners' mental models on recursion after the learning session were their homework and posttest, and a part of their interview (see Appendix C).

Table 9.5. Homework analysis

Learners	Program	Used resources	Justification	Time
C1	Good	Not specified	Not specified	Not specified
C2	Problem with the class File, recursion	File specification, the reference book, Internet	Not specified	4h30
C3	Good	Not specified	Not specified	Not specified
C4	Good	File specification	Good analysis, being interested in the problem	45 minutes
T1	Problem with recursion	Not specified	Not specified	Not specified
T2	Good	Not specified	Not specified	Not specified
T3	Good	API of Java	Good analysis	1h
T4	Good	API of Java, Java tutorial of Sun, the reference book	Good analysis	2h30
T5	Good	API of Java	Not good analysis	2h30

The homework analysis (Table 9.5) showed that both groups of students were more or less equal in the ability to solve the programming problem presented in the homework. Most students solved the problem about file management well because the nature of the problem enabled them to see recursion easily: Only one (C2) among the four students of the COFALE group and one (T1) among the five students of the traditional group could not see how to build recursive solutions to the given problem. It appears that the specification of the class File was sufficient enough for the students to solve the given problem. Most students who submitted the report analyzed correctly the advantage of using recursion to solve the given problem. The time to finish the homework varied widely from one student to another; the mean was approximately 2 hours.

Can we see the difference of the two groups regarding their cognitive flexibility behavior from Table 9.5? I think it is hard to answer this question by analyzing, for instance students' Java programs or justification reports. It is necessary to define a set of operational criteria to do so. This work is of course difficult, and because I am not a cognition specialist, I am not able to propose such criteria here.

Table 9.6a. Posttest grade (SD = Standard Deviation)

Learners	Test 1 (/4)	Test 2 (/4)	Test 3 (/4)	Test 4 (/5)	Test 5 (/4)	Test 6 (/4)	Total grade (/25)	Total time (minutes)
COFALE group								
C1	4	3	4	3	4	3	21	75
C2	3	1	2	1	3	0	10	85
C3	0	3	4	3	3	4	17	90
C4	4	4	4	1	4	0	17	75
Average	**2.75**	**2.75**	**3.50**	**2.00**	**3.50**	**1.75**	**16.25**	**81.25**
SD	**1.89**	**1.26**	**1.00**	**1.15**	**0.58**	**2.06**	**4.57**	**7.50**
Traditional group								
T1	4	4	3	0	3	0	14	75
T2	4	4	4	5	3	4	24	60
T3	4	4	3	4	4	0	19	90
T4	1	1	1	2	3	0	8	95
T5	4	1	2	3	3	0	13	90
Average	**3.40**	**2.80**	**2.60**	**2.80**	**3.20**	**0.80**	**15.60**	**82.00**
SD	**1.34**	**1.64**	**1.14**	**1.92**	**0.45**	**1.79**	**6.11**	**14.40**

Table 9.6a displays students' posttest results and Table 9.6b presents a qualitative analysis of students' tests. Posttest solutions to recursive problems were graded on a 5-point scale (except for test 4 on a 6-point scale): 0 through 4 with 4 indicating a correct solution (see Appendix C for the detailed scale of each test). The evaluation of the posttest was concentrated on recursive solutions; therefore, I did not consider students' syntactic errors such as parenthesis closing, forgetting semicolon; I also took into account students' interview to evaluate their posttest more accurately. It appears that the COFALE group progressed a little better than did the traditional group, and the COFALE group's posttest results were more "homogeneous" than were the traditional group's. The time to finish the posttest of both groups was more or less equal. In the COFALE group, C1, C3, and C4 demonstrated both the under-

standing of the recursion concept and the ability to apply it to solve a diversity of problems, and C2 had serious problems with building recursive solutions. In the traditional group, T2 and T3 (especially T2) showed good understanding of recursion as well as good ability to solve problems recursively, T4 had serious problems and T5 had several problems on the paper-and-pencil test (although they did the programming homework well), and T1 showed good understanding of recursion in several simple cases but had certain problems with the Java programming language. Students' posttest grades presented in Table 9.6a is more or less consistent with students' final exam grade presented in Table 9.1: The students who had better programming skill and Java knowledge learned recursion better. Table 9.6b indicates that students often did not take into account all the base cases of a recursive solution to complex problems.

Table 9.6b. Analysis on students' solutions

Tests	COFALE group	Traditional group
Test 1	C1 and C4 arrived at a correct solution, C2 could not answer the fourth question, C3 could not see recursion.	T4 could not see recursion, the others arrived at a correct solution.
Test 2	C4 arrived at a correct solution, C1 and C3 did not consider all the base cases, C2 could not build a recursive solution.	T4 and T5 could not build a recursive solution, the others arrived at a correct solution.
Test 3	C2 did not consider all the base cases, the others arrived at a correct solution.	T2 arrived at a correct solution, T1 and T3 did not consider all the base cases, T4 and T5 could not build a recursive solution.
Test 4	C1 and C3 did not construct the complete recursive part, C2 and C4 could not see recursion.	T2 arrived at a correct solution, T3 and T5 did not consider all the base cases, T4 did not construct base cases correctly, T1 could not see recursion.
Test 5	C1 and C4 gave adequate definitions, C2 and C3 gave incomplete definitions.	T3 gave adequate definitions, the others gave incomplete definitions.
Test 6	C3 arrived at a correct solution, C1 did not consider all the base cases, C2 and C4 could not build a recursive solution.	T2 arrived at a correct solution, the others could not see recursion.

Table 9.7 presents the mental models students tried to use to generate or verify recursive solutions to the problems presented in the posttest. It was hard for me to know exactly how students solved the problems because they found it difficult to explain how they had built their recursive solutions to the given problems. So, I think we need more ***practical techniques*** to help students express what they have actually learned. I created Table 9.7 on the basis of the following signs:

- *Iterative*. Students tried to generate an iterative solution, for example count all the ways the robot can walk 8 meters in test 1 or propose an iterative solution in test 3.
- *Loop*. Students tried to make a recursive call in a *for* loop although they did not see the recursive formula in test 2.
- *Syntactic*. Students tried to fill the condition and the action part of the base case and the recursive part; they did not analyze different cases explicitly, for example in draft sheets.
- *Analytic*. Students tried to analyze different cases explicitly for a given problem, for example in draft sheets.

- *Analysis-synthesis*. Students tried to think recursively first of all to identify the sub-problems that are identical in structure to the original problem.
- *Trace*. Students tried to trace their recursive solutions after writing the code to verify the correctness of those solutions.
- *Problems with TernaryTree*. Students did not understand the data structure TernaryTree in tests 4 and 6.

Table 9.7. Interview analysis on the mental approaches learners used in the posttest

Learners	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
C1	Analysis-synthesis	Analytic, analysis-synthesis	Analytic	Analytic, analysis-synthesis	Analysis-synthesis	Analytic, analysis-synthesis
C2	Analysis-synthesis	Syntactic	Syntactic	Problem with TernaryTree	Analysis-synthesis	Give up
C3	Iterative, not see recursion	Analytic	Iterative, analytic	Analytic	Analytic	Analytic
C4	Analysis-synthesis	Analytic, analysis-synthesis	Analytic	Problem with TernaryTree	Analysis-synthesis	Problem with TernaryTree
T1	Analysis-synthesis	Analytic, analysis-synthesis	Syntactic	Problem with TernaryTree	Analysis-synthesis	Give up
T2	Analysis-synthesis	Syntactic, analysis-synthesis	Iterative, syntactic	Syntactic, analysis-synthesis	Syntactic, analysis-synthesis	Analysis-synthesis
T3	Analysis-synthesis	Analytic, analysis-synthesis, trace	Iterative, analytic	Analysis-synthesis	Analysis-synthesis	Analytic
T4	Iterative, not see recursion	Loop, syntactic	Iterative, syntactic	Analytic	Analysis-synthesis	Give up
T5	Iterative, analysis-synthesis	Syntactic	Iterative	Syntactic	Analysis-synthesis	Give up

Table 9.7 provides the evidence for the following four points: (a) at a given time, students may possess different mental models on the concept of recursion; (b) they tried to use different mental approaches to generate recursive solutions to a given problem; (c) although they tried to use different mental approaches, they may not arrive at correct solutions; and (d) most students did not use the trace model to verify the correctness of their recursive solutions.

Can we see the difference of the two groups regarding their cognitive flexibility behavior from Tables 9.6a, 9.6b, and 9.7? I believe C1, C3, and C4 in the COFALE group and T2 and T3 in the traditional group demonstrated behavior of cognitive flexibility during the problem-solving process in the posttest. For instance, they tried to use the analytic method or the analytic-synthesis method systematically to solve the given problems. I think this behavior is consistent with cognitive flexibility because students tried to activate their prior knowledge, in different ways, in order to analyze different aspects of a new problem and to propose a solution as complete as possible. Because of limited space, I do not show here all of the data I collected from students' posttests (see also Appendix C).

Table 9.8 presents students' mental representations about recursion discovered in the interview session. It seems that the students in the COFALE group expressed the definition of the concept of recursion more clearly than the students in the traditional group. For instance, all the COFALE students mentioned either "base cases and recursive part" or "dividing a problem into identical and smaller sub-problems" or both of them, whereas it seemed that only T3 of the traditional groups mentioned "dividing a problem into identical and smaller sub-problems".

Can we see the difference of the two groups regarding their cognitive flexibility behavior from Table 9.8? Although most students understood the concept of recursion, it seems that, in the interview session, students in the COFALE group tried to define the concept of recursion more clearly and accurately than did students in the traditional group. This may be an indicator of cognitive flexibility behavior. Why? I am not able to answer this question because I am not a cognition specialist; it is only my personal remark.

Table 9.8. Interview analysis on learners' mental models on recursion after the learning session

Learners	Interview analysis
C1	"Now I see recursion as dividing a large problem into identical pieces that are simpler to solve […] Opposed to the loop concept, it is easy to read a recursive method but not easy to write it."
C2	"Now I see recursion as dividing a problem into sub-problems in the same type […] I see for what recursion is used but I do not understand well how to apply it."
C3	"Now it is clear, recursion allows us to write methods with one or more base cases and recursive part […] I know how to build recursive methods."
C4	"Now I know a recursive method has base cases and recursive part […] I know how it works and how to divide a problem into the same sub-problems."
T1	"Now I know for what recursion is used […] recursion is to reduce a large problem into sub-problems until we arrive at base problems."
T2	"Now recursion is to repeat something many times by calling the same method […] I would say in trees where I see things that are regular in the form."
T3	"Now I see recursion as dividing a problem into sub-problems that are identical and simpler to solve."
T4	"Now recursion is to divide a problem into sub-problems that are equivalent to each other […] it changes the way to solve problem […] it is useful for manipulating trees and fractals."
T5	"Now I know recursion is to divide a problem into small entities until we arrive at the base cases."

It is worth to note that oral expressions sometimes do not completely reflect students' mastering of the recursion concept. For example, C2 seemed to define the concept of recursion well, but she could not arrive at any correct recursive solution in the posttest. T2 seemed to express the concept of recursion unclearly, but he correctly solved most problems in the posttest. Once more, I think we need more ***practical techniques*** to measure students' understanding accurately.

The posttest and interview analyses also showed that students demonstrated different levels for the syntactic, analytic, and analysis-synthesis models (Table 9.9). I distinguished the following five levels:

1. *Novice*. Students do not know or use the model.
2. *Beginner*. Students know how to apply the model but in several simple cases, and their solutions are often incorrect or incomplete.

3. *Intermediate*. Students know how to apply the model in a variety of cases, and their solutions are sometimes incomplete or incorrect.
4. *Advanced*. Students know how to apply the model in a diversity of cases, and their solutions are hardly incomplete or incorrect.
5. *Expert*. Students, in addition to the ability implied by the advanced level, know when and why apply the model to solve problems.

Table 9.9. Some evidences for different levels of mental models on recursion

Levels	Syntactic	Analytic	Analysis-synthesis
Novice	Students' mental models before the learning session.	Students' mental models before the learning session.	Students' mental models before the learning session.
Beginner	T4 tried to apply this model in Tests 2, 3, but could not arrive at a correct solution.	T4 tried to use this model in Test 4, but could not arrive at a correct solution.	C2 tried to use this model in Test 1, she correctly answered the first three questions, but not the fourth one.
Intermediate	C2, T1 applied this model in Test 3, their solutions were approximately correct.	No evidence was found.	T1 applied this model to do Test 1 correctly, but his solution to Test 2 was incomplete.
Advanced	No evidence was found.	C4 applied this model successfully in Tests 1, 2, 3.	C1 and T3 applied this model well to do Tests 1, 2, 4.
Expert	T2 used this model to do Tests 2, 3, 4 correctly. He knew that this model is not appropriate to do Test 6.	No evidence was found.	T2 used this model to do Tests 1, 6 correctly. He knew that this model is appropriate to do these tests.

On the basis of the previous criteria and *students' interview*, I constructed Table 9.10, which summarizes students' mental models on the concept of recursion after the learning session (except for T3, there were no explicit data for analyzing students' changes of the trace model). Three (C1, C2, and C4) among four students of the COFALE group and two (T2 and T3) among five students of the traditional group developed the ability to solve problems recursively to a relatively high level. Two other students (T1 and T5) of the traditional group also made significant progress in learning recursion. We cannot distinguish considerable differences of learning outcomes between the two groups: Both groups of learners seem to be more or less equal in mastering the recursion concept.

Table 9.10. Learners' mental models on recursion after the learning session

Learners	Loop model	Syntactic model	Analytic model	Analysis-synthesis model	Trace model
C1	No	Not used	Advanced	Advanced	Beginner
C2	No	Intermediate	Not used	Beginner	Novice
C3	No	Not used	Advanced	Not used	Intermediate
C4	No	Not used	Advanced	Intermediate	Beginner
T1	No	Intermediate	Advanced	Intermediate	Intermediate
T2	No	Expert	Not used	Expert	Intermediate
T3	No	Not used	Advanced	Advanced	Advanced
T4	Yes	Beginner	Beginner	Beginner	Intermediate
T5	No	Intermediate	Not used	Intermediate	Intermediate

When asked the question "Did you have particular difficulties when you were brought to work on this concept [recursion]?", most students said (Table 9.11) that they understood the concept of recursion quite well but found it *difficult to see recursion in concrete problems*, particularly in complex situations such as the ones in the posttest of the study.

Table 9.11. Learners' difficulties about learning recursion

Learners	Difficulties
C1	Recursive thinking: how to write a recursive method for a given problem.
C2	Not see how to build recursive methods for a given problem.
C3	It is not easy to see recursion in different problems.
C4	Application to diverse problems.
T1	What are base cases? How to build recursive solutions to a given problem.
T2	No difficulties.
T3	Recursion is a counter-natural concept. Problem is to identify recursive elements in a given situation.
T4	See how to solve problem recursively and where is recursion in concrete situations such as trees.
T5	It is difficult to apply recursion in concrete cases.

Main result:

Most students in both groups developed the ability to solve problems recursively to a certain degree but there was no clear difference of learning outcomes between the two groups of learners.

The COFALE group's behavior seemed to be more consistent with cognitive flexibility than the traditional group's behavior, more analysis is needed to verify this claim.

All the students in both groups had difficulty to apply recursion in concrete problems, particularly complex problems different from the ones with which they were confronted.

9.3.2 Exploration of learning materials

I now analyze students' exploration of and feedback on the provided learning materials (the lecture, the homework, and the COFALE environment or the chapter of the reference book). The analysis could explain why students learned recursion and why there was no difference of learning outcomes between the two groups. The data used for this analysis were a part of students' interview and my observation about students' learning behavior during the learning session.

On the basis of students' navigation history registered in COFALE (see chapters 6 and 7) and students' interviews, I created Table 9.12. From this table, I may deduce that, *within 1 hour*, all of students followed a certain number of suggestions fostering cognitive flexibility offered by COFALE. Indeed, the three main learning activities that students performed were: (a) exploring multiple representations and learning situations related to both recursion and linked lists (e.g., arithmetic expressions, simple text search, and phone book); (b) doing all of three exercises presented in the test section of COFALE; and (c) asking me via e-mail to give feedback on their exercises.

Table 9.12. COFALE group learners' exploration of COFALE with respect to criteria for cognitive flexibility

Criteria for cognitive flexibility	**Students' learning behavior**
MM1: *The same learning content presenting concepts and their relationships is represented in different forms (e.g., text, images, audio, video, simulations).*	All learners explored multiple representations.
MP1: *The same abstract concept is explained, used, and applied systematically with other concepts in a diversity of examples of use, exercises, and case studies in complex, realistic, and relevant situations.*	All learners explored multiple situations.
MM2: *Learners are encouraged to study the same abstract concept for different purposes, at different times, by different methods including different activities (reading, exploring, knowledge reorganization, etc.).*	All learners performed multiple learning activities at different times.
MP2: *When facing a new concept, learners are encouraged to explore the relationships between this concept and other ones as far as possible in complex, realistic, and relevant situations.*	All learners examined related concepts.
MP3: *When facing a new concept, learners are encouraged to explore different interpretations of this concept (by other authors and by peers), to express their personal point of view on the new concept, and to give feedback on the points of view of other people.*	Only C4 explored external resources. No one examined peers' learning spaces.
MP4: *When facing a new concept, learners are encouraged to examine, analyze, and synthesize a diversity of points of view on the new concept.*	No one produced summaries.
MM3: *The number of participants, the type of participant (learner, tutor, expert, etc.), the communication tools (e-mail, mailing lists, face to face, chat room, video conferencing, etc.), and the location (in the classroom, on campus, anywhere in the world, etc.) are varied.*	All learners used e-mail to ask questions to the tutor. Only C3 participated in the forum created in advance by the tutor.
MP5: *During the discussion, learners are encouraged to diversify – as far as possible – the different points of view about the topic discussed.*	No one used the list of predefined discussion questions.
MM4: *During the learning process, learners are encouraged to use different assessment methods and tools, at different times, and in different contexts for demonstrating their ability to solve different problems.*	All learners did tests and homework individually.
MP6: *During the problem-solving process, learners are encouraged to confront multiple ways to solve the problem and multiple possible solutions to the problem.*	No work in groups was recorded.

Why did COFALE students' learning behavior satisfy only about a half of the set of criteria? *Why did their learning behavior satisfy those criteria but not other ones*? Here is what I observed: At the beginning of COFALE exploration, all students tried to find tests to do. They arrived at learner tools (see section 6.3.3), they found tests and did them. They had questions and they tried to find communication tools. They arrived at discussion tools (it is easy to see them in COFALE). They used e-mail to communicate with the tutor. The tutor suggested them, via e-mail, to explore learning situations before doing tests. They followed the tutor's suggestions, they explored different learning situations. They saw the menu "Related topics", they explored related topics (linked lists). Then, they returned to do tests and to ask the tutor questions. C3 and C4 also explored other learner tools (external resources, forums). And time (1 hour) was up. From this observation, I derive three conclusions:

- One hour may be too short for students to explore every condition of learning provided by the recursion course in COFALE. It is, however, not necessary to always examine *all* of the learning conditions in the environment.
- Although COFALE has been designed to guide students automatically in the exploration of the learning hyperspace, it is important that the tutor be active to help students explore the learning environment effectively, especially at the beginning of the learning process, when students are still "novice" in the use of COFALE.
- In an e-Learning context such as COFALE, students like to find and do exercises and tests more than explore learning contents. I discuss this issue further in section 9.4.3.

There was no clear evidence registered for the effect of adaptability supported by COFALE, because the exploration time was too short (about 1 hour during the experimental phase, in the interview session, most COFALE students claimed that they did not use COFALE after the experimental session and before the posttest and interview ones).

On the basis of students' interviews, I created Table 9.13. This table shows that T2 and T3 had good reading skills, whereas T1, T4, and T5 did not examine the content of the chapter well enough. My observation of students' reading also confirmed this claim: T2 and T3 carefully read each example presented in the chapter, whereas T1, T4, and T5 had a quick glance at the content of the chapter. This claim may explain the traditional group students' grades both in the final exam of the introductory course on object-oriented programming and Java and in the posttest of this study.

Tables 9.12 and 9.13 and my observation of students' learning behavior showed that the COFALE group had more <u>*learning motivation*</u> than did the traditional one. Indeed, all of the four COFALE group students did *all* three exercises presented in COFALE in comparison with one or two among many exercises presented at the end of the book chapter did four of the five traditional group students. This *behavior of persistence* of students (i.e. to maintain attention to the learning environment) is an important indicator in a theory of motivation (Huitt, 2001). In addition, during the exploration session, I received 20 questions of the COFALE group students via e-mail compared to only 3 face-to-face questions of the students in the traditional group (the questions principally concerned the feedback on students' exercises). This *meta-cognitive strategy* of students (i.e. to ask questions) is also a critical indicator in a theory of motivation (Huitt, 2001).

Table 9.13. Traditional group learners' exploration of the chapter of the reference book

Learners	Exploration
T1	"I had a quick glance at the chapter […] I did one or two exercises at the end of the chapter."
T2	"I examined all the examples well […] I did not do exercises at the end of the chapter."
T3	"I read only the points that are interesting to me, the examples particularly […] I also did several exercises at the end of the chapter."
T4	"I rarely read the comments […] I read only the source code in examples […] I did one or two exercises at the end of the chapter."
T5	"I had a quick glance at the examples […] I did one or two exercises at the end of the chapter."

Table 9.14 displays students' feedback on the provided learning materials. Almost all of students in both groups claimed that the lecture was good because it provided many simple examples for the understanding of the recursion concept. All students in the COFALE group were satisfied with the way COFALE provided learning situations and activities; especially, C3 and C4 were very interested in learning with COFALE. Most students in the traditional group were also satisfied with the chapter of the reference book, except for T4 who did not like the way the chapter presented the Java code and for T5 who wanted the chapter would explain the way to arrive at recursive solutions. Students also liked the homework (its analysis was presented in Table 9.5) because it helped them see the usefulness of recursion well.

When asked the question "Were two weeks sufficient for this course?", except for C2, students said that the time duration and the learning materials were sufficient for learning the concept of recursion. They claimed, however, that they needed more time to apply recursion in a diversity of complex situations by themselves. Thus, it is necessary to prepare conditions of learning to help students transfer their own knowledge in future contexts, especially outside the school contexts (Jonnaert & Vander Borght, 2003).

Table 9.14. Learners' feedback on the provided learning materials

Learners	Comments	Difficulties
C1	"I had not much time to examine COFALE […] it is useful but what happens if the tutor has to correct the exercises of 200 students."	Not understand several learner tools
C2	"I need more lectures to be able to master recursion […] COFALE is good […] but I need feedback immediately after I submit my homework."	No difficulties
C3	"The lecture was good […] COFALE is interesting, instructive […] examples with graphics are clear, very well explained […] I like the way to navigate freely […] COFALE can replace the lecture […] it should be presented in French."	No difficulties
C4	"The lecture was good […] COFALE is good, personalized […] we can work anywhere, submit exercises online […] there is not much in one page […] many examples, they are clear and well explained, in each example we do not give the solution immediately, there is one page to explain how to think, one page to explain how to build the solution, and one page to present the solution […] but I need more exercises."	Not easy to find exercises in COFALE
T1	"The lecture was very clear […] Examples in the chapter are very clear, very good, and interesting."	No difficulties
T2	"The lecture provided simples examples […] The chapter is good but should also provide several simple examples."	Sometimes English
T3	"The lecture was good but I need more homework […] The chapter is well done […] examples are clear and well explained except for the first several ones, and examples are presented with increasing difficulties."	No difficulties
T4	"Examples and exercises in the chapter are good […] but the source code is too long, so I must turn back pages when reading."	Not see how to find out recursion, how it works
T5	"The lecture was good […] The chapter is clear but it should provide complex exercises and explain how to arrive at recursive solutions."	Not see how to build recursive solutions

Main result: Most students were satisfied with the lecture, the homework, and the COFALE environment or the book chapter; the students in the COFALE group, however, had more learning motivation than did the students in the traditional one.

9.4 Discussion

After the 4-hour-long learning session, most students in both groups developed the ability to solve problems recursively to a significant degree, and were interested in and satisfied with COFALE or the book chapter. The COFALE group had *more learning motivation* than did the traditional one: It appears that all of the COFALE group students followed COFALE's suggestions fostering cognitive flexibility, whereas about half of the traditional group students followed the chapter's suggestions. It seems that the COFALE group progressed a little better than did the traditional one in learning the concept of recursion. There was, however, *no clear evidence for the difference of learning outcomes* in terms of the concept of recursion between the two groups. For example, the posttest grade, the quality of solutions to the problems of the posttest, and the change of mental models on recursion of the two groups are more or less comparable.

I also found out some evidence that indicate that the COFALE group demonstrated *more cognitive flexibility behavior* than did the traditional group.

In what follows, on the basis of my findings, I discuss the following five issues: (a) reasons for the fact that both groups of students learned the concept of recursion to a certain degree, (b) reasons for the fact that there was no difference of learning outcomes between the two groups of students, (c) useful findings for improving COFALE, (d) useful findings for the teaching and learning of the recursion concept, and (e) examples of students' knowledge construction consistent with the constructivist point of view shown in chapter 1.

9.4.1 Why did students learn recursion to a significant degree?

In this study, among nine first-year engineering students at UCL, five learners mastered reasonably well and two learners mastered to a high degree both the concept of recursion and its application to solve a diversity of problems in a complex situation. It is worth to note that the selected students learned recursion with the Java programming language within about 4 hours during 2 weeks in comparison, for example, with 12 hours during 3 weeks Bhuiyan's nine first-year science students mastered recursion with the Lisp programming language (Bhuiyan et al., 1994). Because there was no difference of learning outcomes between the two groups of students, the main reasons for that success may be among of the following ones:

1. *Students' programming skills were good before the study*. The four evidences for this claim are: (a) in the introductory course on object-oriented programming and Java, students mastered both programming and problem solving techniques and Java well through a problem-based learning approach (Pedagogy Group at FSA/UCL, 2005), a kind of active learning; (b) in the introductory course on object-oriented programming and Java, students were able to learn the concept of linked lists, which is related to recursion, within 8 hours during 1 week; (c) most selected students were among the best ones of about 280 students in this course; and (d) most selected students were able to trace one or two recursive methods successfully although they had no knowledge of recursion.

2. *The lecture was good*. The main evidence for this claim is that in the interview session, most selected students advocated that the lecture was very clear with many simple examples that helped them mastering the basic knowledge about the recursion concept. Further more, the explicit teaching of the DCG (Divide, Conquer, and Glue) strategy may also be an important factor for the success (Turbak et al., 1999).
3. *COFALE and the chapter were good*. The main indication for this affirmation is that in the interview session, most students said (see Table 9.14) that they were satisfied with COFALE (or the chapter) because it provided a variety of compelling examples that helped them see recursion in different situations.

   … (it may have more reasons than that present here)

9.4.2 Why was there no significant difference between the two groups?

This study did not provide clear evidence for the difference of learning outcomes between the COFALE group learners and the traditional group learners. One or more reasons of the following ones could explain this affirmation:

1. *The time for the study was too short to see the difference*. Although the time was sufficient enough for most students to learn recursion, it may be too short to see the difference of learning outcomes between the two groups of students. Indeed, within 1 hour the COFALE group students could not benefit all of the learning materials supported by the COFALE learning environment such as forums, learning tools for adding students' personal examples and producing students' own summaries. More long-term experiments should thus be performed.
2. *The number of participants was not sufficient enough to see the difference*. Among four COFALE group students, three learners mastered recursion well and one learner had serious problems with it. And among five traditional group students, two learners mastered recursion well, two learners had a little difficulty in applying it to solve complex problems, and one learner had serious difficulty in the application of recursion. These factors, however, cannot provide significantly statistical evidence for the difference of learning outcomes between the two groups because the population of the study was too small. So, more experiments with more participants (e.g. 20 students) should be carried out.
3. *The selected students had good enough programming skills and the lecture was good enough for students to learn recursion*. Therefore, after the lecture two students (C3 in the COFALE group and T2 in the traditional one) declared that they had mastered the recursion concept. They also stated that COFALE (or the chapter) was useful but they considered it as an additional resource for seeing how to apply recursion in various examples.
4. *The chapter of the reference book was as good as was COFALE for learning recursion*. Most students in both groups advocated that COFALE or the chapter clearly presented a variety of interesting examples. Moreover, the traditional group students were versed in

the exploration of the book in their introductory course on object-oriented programming and Java.

5. *The means (the posttest and interview questions) and criteria used to assess students' learning outcomes were not pertinent*. I think that it is necessary to propose finer (rather operational) criteria for the assessment of students' learning outcomes. For instance, a set of operational criteria should be useful for evaluating students' posttest and interview.
6. *The learning conditions provided by COFALE did not foster cognitive flexibility effectively*. Although COFALE's learning conditions presented in chapter 6 were consistent with the pedagogical principles underlying cognitive flexibility, no experiment was carried out with actual students to verify whether those conditions of learning truly foster cognitive flexibility. It is thus important to perform one or several surveys for this issue.
7. *The selected students had done unanticipated learning activities before the posttest session*. After the experimental session and before the posttest one, the selected students had been asked to do the homework (see section 9.2.2). During that time, however, the two groups could use other pedagogical devices than COFALE or the book chapter to learn the concept of recursion.

   … (it may have more reasons than that present here)

9.4.3 Findings for improving COFALE and the book chapter

The experiment provided evidence that may be useful for improving the COFALE system and the chapter of the reference book, as follows:

- In the interview session, several students claimed that the chapter should present more simple examples, and especially explain how to go from problem specifications to recursive solutions. This claim is consistent with the comments presented in Appendix B6.
- In virtual learning environments such as COFALE, students tend to find (e.g., through learner tools in COFALE) and do exercises and tests first. The current version of COFALE presents assessment activities at the bottom of certain content pages (see section 6.3.1), but not vice versa. So, for future research, for each exercise or test in COFALE, it is useful to associate appropriate concepts and learning situations so that students can examine them while doing the exercise or the test. Incorporating assessment into students' learning process, possibly at its beginning, is an aspect consistent with constructivism (Lajoie & Lesgold, 1992; Wilson et al., 1995).
- When exploring learner tools in COFALE at the beginning of the learning session, students may not understand several tools such as "Export Content", "My Tracker" (see ATutor's "How To Course": Adaptive Technology Resource Center, 2004). Thus, the hyperspace of learner tools should be adaptive to students' skill level in exploring COFALE: When students are not familiar with using COFALE, we should present them with simple tools. When they are versed in using COFALE, we should provide them with more "advanced" tools.

- During he learning session, students ask the tutor similar questions, for instance about tests. Therefore, the tutor should create one or more Q&A forums to reduce his or her workload.
- Students like to learn a new concept in their native language. So, COFALE's interface and the learning content should be presented in students' native tongue, for example in French for students at FSA/UCL.

9.4.4 Findings for teaching and learning recursion

The survey provided a certain numbers of findings that may be useful both for teaching and for learning recursion. Here are several essential findings:

- Explicit teaching of the DCG strategy may help students applying recursion flexibly (i.e. they can use different mental approaches to solve problems recursively), and therefore grasping the recursion concept well. Turbak and colleagues (Turbak et al., 1999) also claimed this finding.
- Recursion is really unfamiliar and complex because: (a) it is hard for students to see recursion in a given problem, for example, in the posttest, C3 was able to solve the complex problem in test 6 but was not able to see recursion in test 1, whereas several other students saw recursion in test 1 but were not able to solve the problem in test 6; (b) sometimes, students express the concept of recursion correctly but cannot apply it in concrete situations (e.g. C2 and T4) and vice versa (e.g. T2); and (c) to construct complete and correct recursive solutions to complex problems, students often have to use different mental approaches such as analytic and analysis-synthesis – using only the analysis-synthesis approach often leads to incomplete solutions. Several researchers on recursion (e.g., Anderson et al., 1988; Bhuiyan et al, 1994) also drew this conclusion. Therefore, the tutor should make a great diversity of situations available and encourage learners to apply recursion to those situations.
- If the nature of problems does not force students to think recursively, it seems that they prefer to think iteratively and propose iterative solutions (of course if they have prior knowledge of iteration), for instance, in the posttest, several students preferred an iterative solution to a recursive one to the problem in test 3, several students tried to count the number of ways the robot can walk 8 meters in test 1, but most students tried to think recursively to solve the problems in test 4 and in test 6.
- Although students have no knowledge of recursion, they can trace correctly several recursive methods if they have good programming skills (see the pretest analysis in section 9.3.1).
- To personalize learning materials for different students in the recursion course, we need to take into consideration the two following factors: (a) students often use different mental approaches to solve problems recursively – several researchers on recursion (e.g. Bhuiyan et al., 1994) also drew this conclusion; and (b) the same mental approach, for

example analytic, may have different levels, from "novice" to "expert" (see section 9.3.1) – it appears that there has not been considerable research attention devoted to this issue. This finding suggests that it is necessary to consider many more kinds of mental models on recursion than the four ones I assumed in section 6.2.1 in the design of COFALE. Thus, for future research, one can improve COFALE's learner model manager, for example, to provide an explicit tool allowing the course designer to specify the level of each mental model on the learning object being designed.

- Students' results of the posttest may reflect more or less their mental models on the concept of recursion. It isn't, however, always exact to identify students' mental models automatically if we rely only on their test results (passed or failed), as I proposed in section 7.3.1. For example, in the posttest, C3 successfully used the analytic model (but not the analysis-synthesis one) to solve the problem in test 6 (in the design of COFALE, I assumed that if students pass this test then they have reached the analysis-synthesis model). Sometimes, we need to interview students in order to know exactly their mental representations about recursion. Further research should thus be done for the problem of detecting learners' mental models automatically. In addition, it is worth to encourage the learner to carefully examine the following three methods of evaluation of his or her learner model (see section 6.3.2): self-evaluation, evaluation by the tutor, and evaluation by the system.

9.4.5 Examples for constructivism

While evaluating students' posttest, I found the following two examples, which are frequent in computing science:

1. *Different ways to solve the same problem*. Students proposed three different solutions to the problem in test 2: one approach with two base cases, one approach with three base cases, and one approach with four base cases (none of them resembles the solution presented in Appendix C). All of those solutions are correct and produce the same output for the same input. This example means that given the same problem, individuals use their own way to arrive at their own solution, which is not necessarily the tutor's.
2. *Different solutions to the same problem*. In test 3, because there was no precondition for the method "isZigzag", students expressed two different points of view for the given problem: (a) one student thought that if a list is null then return "false", and (b) several other students supposed that list is not null. These points of view are different from (even opposed to) the tutor's (see Appendix C: If a list is null then return "true"), although they should be all acceptable. This example explains that, given the same situation, people construct their personal understanding, which may be different from each other's.

The previous examples may suggest that in the design of COFALE, we should systematically provide students with problems whose nature must give rise directly to different ways to solve the problems and to different solutions to the problems. This claim is consistent with criteria MP6 shown in section 2.4.4.

9.5 Conclusion

In the present chapter, I have reported a ***preliminary*** evaluation of the COFALE learning environment. Although the survey provided several encouraging results for fostering cognitive flexibility by means of ICT-based learning conditions, there are still a number of important questions needed to be taken into account, especially the question about whether COFALE's learning conditions truly facilitate and stimulate students' cognitive flexibility.

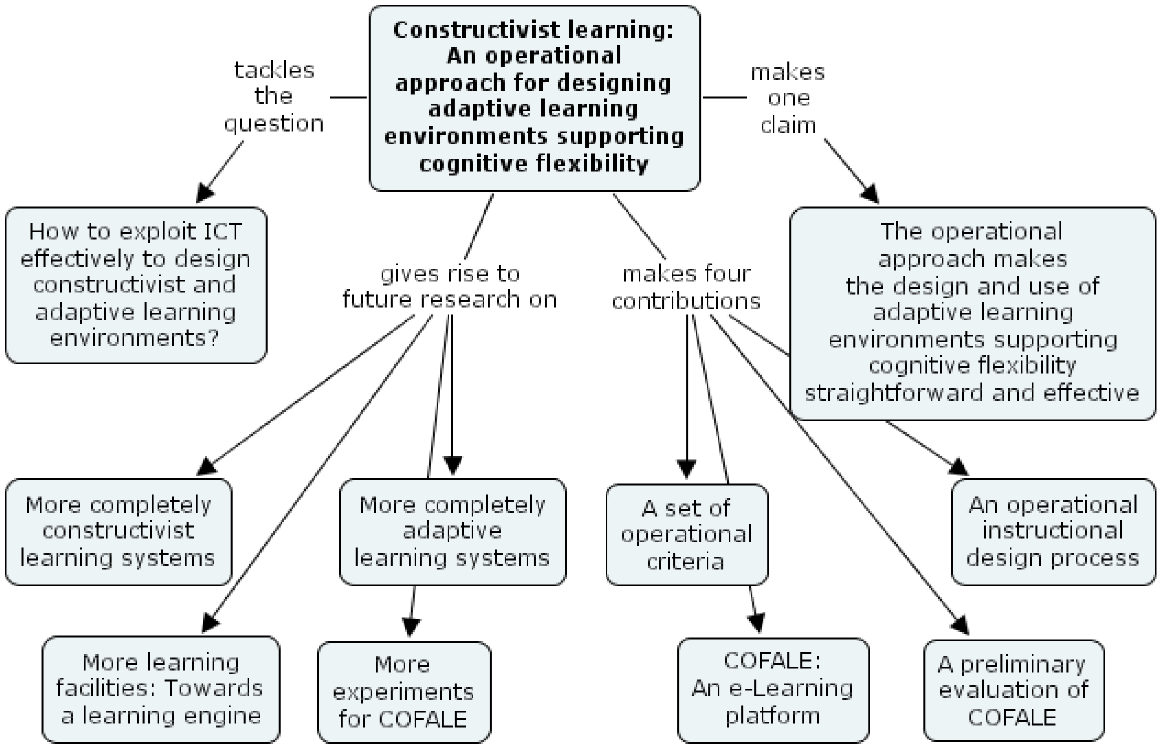
For future research, I claim that to gather more evidences indicating the full effectiveness of learning with the assistance of COFALE, it is necessary to carry out a number of long-term experiments with a significant number of learners, for instance, to integrate COFALE into long-term official courses of students at FSA/UCL.

Conclusions and future work

"*The journey of a thousand miles begins with a single step.*"

Lao Tsu, Chinese Philosopher, 6th Century B.C. (cited in WorldPeace, 1997)

I summarize in this conclusion section the aim of the dissertation, the main results of the present work, the directions for future research, and several concluding remarks.


Constructivist learning: An operational approach for designing adaptive learning environments supporting cognitive flexibility
tackles the question
How to exploit ICT effectively to design constructivist and adaptive learning environments?
makes one claim
The operational approach makes the design and use of adaptive learning environments supporting cognitive flexibility straightforward and effective
gives rise to future research on
More completely constructivist learning systems
More completely adaptive learning systems
More learning facilities: Towards a learning engine
More experiments for COFALE
makes four contributions
A set of operational criteria
COFALE: An e-Learning platform
A preliminary evaluation of COFALE
An operational instructional design process

Aim of the thesis

The main objective of the dissertation is to investigate on how to exploit ICT effectively to design constructivist and adaptive learning environments. I have tackled two important questions of ICT-based constructivist and adaptive learning systems by means of an *operational* approach:

1. How to exploit ICT to provide the individual student with *appropriate* learning conditions that truly facilitate and stimulate *constructivist* learning?
2. How to help the teacher design ICT-based *adaptive* learning systems from a *constructivist* point of view?

Contributions

Among many facets of constructivism, cognitive flexibility is often mentioned as being strongly relevant to the constructivist learning theory. The important claim I make in this thesis is that: *The operational approach proposed in this thesis makes the design and use of adaptive learning systems supporting cognitive flexibility straightforward and effective*.

More specifically, this dissertation makes the four main contributions, as follows:

1. *A set of operational criteria for cognitive flexibility*. In chapter 2, I proposed a set of operational criteria for cognitive flexibility (stressing the qualifier "operational"). Showing an example of their application, I argued that operational criteria make the process of instructional design more straightforward than do general indications suggested by educational theorists. The set of criteria was then applied as a useful framework for designing and evaluating learning conditions fostering cognitive flexibility: it is used, for example, (a) as guidelines for proposing an instructional design process in chapter 3, (b) as means of validation for demonstrating the COFALE learning environment in chapter 6, and (c) as means of validation for analyzing several "constructivist" learning environments in chapter 5.

   I believe that the *usefulness* of the set of operational criteria is in the *way* I have proposed it. After reading chapter 2, one should be able to propose and use one's own operational criteria for *any* pedagogical principle other than cognitive flexibility.

2. *An operational instructional design process for cognitive flexibility*. In chapter 3, on the basis of the set of criteria for cognitive flexibility, I proposed an instructional design process and I showed an example of its application for the instruction of the complex concept of recursion in computing science. The set of instructional design activities was then applied in chapter 7 to guide the course designer in using authoring tools offered by COFALE to devise learning conditions that facilitate and stimulate cognitive flexibility. I argued that the process of instructional design I proposed is useful in the practice of in-

struction because the set of instructional design activities was used as a practical framework (or as guidelines) for creating conditions of learning.

In my point of view, the *usefulness* of the instructional design process is in the *manner* I have exploited the pedagogical principles exhibiting cognitive flexibility in order to make them practical in instruction. After reading chapter 3, one should be able to propose one's own instructional design process for the practice of one's own teaching.

3. *A domain-independent adaptive e-Learning platform supporting cognitive flexibility*. In chapters 6 and 7, I showed how to exploit available learning technologies to design learning environments that facilitate and stimulate cognitive flexibility. The design and use of COFALE learning environment provides *clear evidence* for the usefulness of my approach because: (a) using operational criteria as means of validation, it is quite easy to illustrate that COFALE provides the individual student with learning conditions fostering cognitive flexibility (see chapter 6); and (b) the operational instructional design process makes it straightforward for the course designer to use authoring tools provided by COFALE to design and use learning systems supporting cognitive flexibility (see chapter 7).

   COFALE also supports several adaptation techniques borrowed from other adaptive learning systems. It adapts the learning contents, pedagogical devices, and communication support to different kinds of students (see section 6.3.2). The two adaptation techniques COFALE has not implemented yet are adaptive problem-solving support and adaptive assessment (see sections 1.5.4 and 5.3).

   The *usefulness* of the design and use of COFALE I described is in the *way* I have exploited available ICT to implement learning conditions exhibiting the desired characteristics of a relevant pedagogical principle (i.e., cognitive flexibility). I think that the COFALE system is ready to be used in a great number of domains, for instance, in mathematics, in sciences, in economics (see section 7.4), and that it is open enough for one to modify it to exploit *any* pedagogical principle. COFALE may also be used as a "*test bed*" for educational researchers to carry out various experiments related to learning technologies and cognitive flexibility. Of course, the system needs to be tested by different people (teachers, course designers, educational researchers, software developers) in different domains, so that we can draw definitive conclusions.

4. *A preliminary evaluation of ICT-based learning conditions fostering cognitive flexibility*. A number of studies have been carried out for evaluating how learning conditions fostering cognitive flexibility affect how students learn (Spiro & Jehng, 1990); these experiments have showed that such learning conditions significantly help students in acquiring knowledge, especially in advanced learning. In chapter 9, I also carried out a preliminary experiment to evaluate the COFALE system: How students learn the complex concept of recursion in computing science with the help of COFALE? Although the short-term study did not provide evidence for the difference of outcomes between learning with the assistance of COFALE and learning with the help of a book chapter, several encouraging results from learning with the help of COFALE were reported (e.g., students were satisfied

with and interested in the way COFALE supports learning, students gained cognitive flexibility behavior to some level). Also, the study provided a certain number of findings that were useful to improve the design and use of COFALE as well as the creation of learning situations for the particular concept of recursion.

It is important, however, to emphasize the need for carrying out long-term experiments in order to verify whether the learning conditions provided by COFALE truly foster students' cognitive flexibility.

Future directions

In the previous chapters, I discussed a number of ideas for future research. Here, I summarize a certain number of critical issues.

More completely constructivist learning environments

Operational criteria for constructivism. Jonnaert and Vander Borght (2003) proposed a set of operational criteria for the *concept of learning* in a constructivist point of view (see also section 2.5). In chapter 2, I proposed a set of operational criteria for *learning conditions* fostering cognitive flexibility, an important facet of constructivism.

In chapter 1, however, I explained that constructivism has various paradigms and many facets related to instructional design. The educational paradigm Jonnaert and Vander Borght followed is only one of many constructivist variations. Cognitive flexibility I exploited is only one of many constructivist facets. In addition, in section 9.4.2, I claimed that is it necessary to propose criteria for constructivist *assessment*. I believe that the operational approach used by Jonnaert and Vander Borght and in this thesis could also be used to make constructivist learning, instruction, and assessment clearer than what we have done.

For example, for the facet of problem solving related to instructional design, one first needs to propose operational criteria for this facet in the same way I proposed such criteria for cognitive flexibility in chapter 2: One should identify learning conditions facilitating problem solving by students and propose criteria for these conditions of learning in each of the four learning components identified in section 1.5.1. Then, on the basis of those criteria, one should be able to propose an instructional design process (see chapter 3).

While exploiting other facets of constructivism than cognitive flexibility, one may encounter several difficulties. For instance, facing the problem-solving facet, one should note that problem solving is domain-dependent (Spiro & Jehng, 90; Weber & Brusilovsky, 2001). Thus, it must be hard to propose a general framework that covers different domains.

Learner and instructor tools for constructivism. In chapters 6, 7, and 8, I introduced a new and open-source e-Learning platform, named COFALE, in which ICT-based learning conditions fostering cognitive flexibility were created. For future research, one can build various tools for fostering not only cognitive flexibility but also other facets of constructivism. Here are several examples:

- To alleviate the teacher's workload, it should be necessary to build a tool allowing the automatic evaluation of students' *learning behavior* with respect to the pedagogical principles underlying cognitive flexibility (see also section 7.2.3).
- To help teachers use COFALE effectively, we may need to provide them with specific tools to analyze their *teaching behavior* with respect to the desired characteristics of cognitive flexibility (see also section 7.2.3). It should also be necessary to build a tool allowing the teacher to "deduce" *learning methods* (or strategies) learners use during their learning process, so that the teacher can adjust the conditions of learning he or she has created for learners (see also section 7.2.2).
- It should be useful to present teachers with a specific tool allowing them to exhort a specific student to follow a *teaching method* (see section 7.2.2).
- It should be important to make COFALE open enough so that one can integrate specific pedagogical devices into COFALE without intervention of the software developer, for instance *domain-specific tools* to support problem solving by students (see also sections 6.5 and 7.4).
- It should be useful to ameliorate a certain number of COFALE's existing learner tools such as the tool for exploring peers' learning hyperspace (see section 6.3.1).

More completely adaptive learning systems

In section 1.5.4, I described five main adaptation techniques that can be implemented in a learning system: adaptive presentation of learning contents, adaptive use of pedagogical devices, adaptive communication support, adaptive problem-solving support, and adaptive assessment. In section 6.3.2, I showed that COFALE takes into account the first three techniques mentioned earlier, and in section 7.3.2, I explained how to implement those techniques. I believe that COFALE' adaptability can be improved. Here are several examples:

- *Adaptive assessment* can be implemented in the same way I implemented adaptive presentation of learning contents (see section 7.3.2).
- *Adaptive problem-solving support* should be a very useful technique (Weber & Brusilovsky, 2001). The implementation of such a domain-dependent technique in a domain-independent platform such as COFALE must be difficult (Weber & Brusilovsky, 2001). One possible way is to make the platform open enough so that one can easily integrate domain-specific tools into the system to support problem solving by learners (see also sections 6.5 and 7.4).
- *Adaptive presentation of learning contents* can be improved (see also section 7.3.2). For instance, we can implement adaptive navigation support and adaptive curriculum sequencing (see section 5.3) in COFALE.

- Adaptive features of COFALE, designed for separate learning objects, can be improved for designing "complex" courses in which many learning objects are interrelated (see also section 7.4).

More experiments for COFALE

In chapter 9, I reported a *preliminary* evaluation of the COFALE learning environment. The survey is preliminary because it was carried out in a short term and with a small number of students. Cognitive flexibility is important in instruction (Driscoll, 2000; Spiro & Jehng, 1990). Therefore, it should be useful to conduct several kinds of *long-term* studies to know the full extent of how learning conditions fostering cognitive flexibility affect how students learn, especially in an e-Learning context. Those studies could also help ameliorating the design and use of COFALE. Here are several instances (see also section 9.1):

- To verify whether COFALE's conditions of learning *truly foster students' cognitive flexibility*. To do so, I have claimed that we need a set of operational criteria.
- To know whether students follow every suggestion proposed by COFALE.
- To know the impact of COFALE's adaptability on students' learning.
- To analyze the impact of COFALE on students' learning when it is integrated in their long-term courses, for instance the course on object-oriented programming and Java given to the first-year engineering students. The questions for this kind of experiments include: Does COFALE help students master those concepts better than, for instance, a traditional approach? Are students satisfied with and interested in learning with the assistance of COFALE? Can COFALE replace a part of traditional pedagogical devices, for instance, the tutor's exercise sessions, students' meeting rooms?
- To know teachers and educational researchers' feedback on the design and use of COFALE in their own teaching and research, respectively.

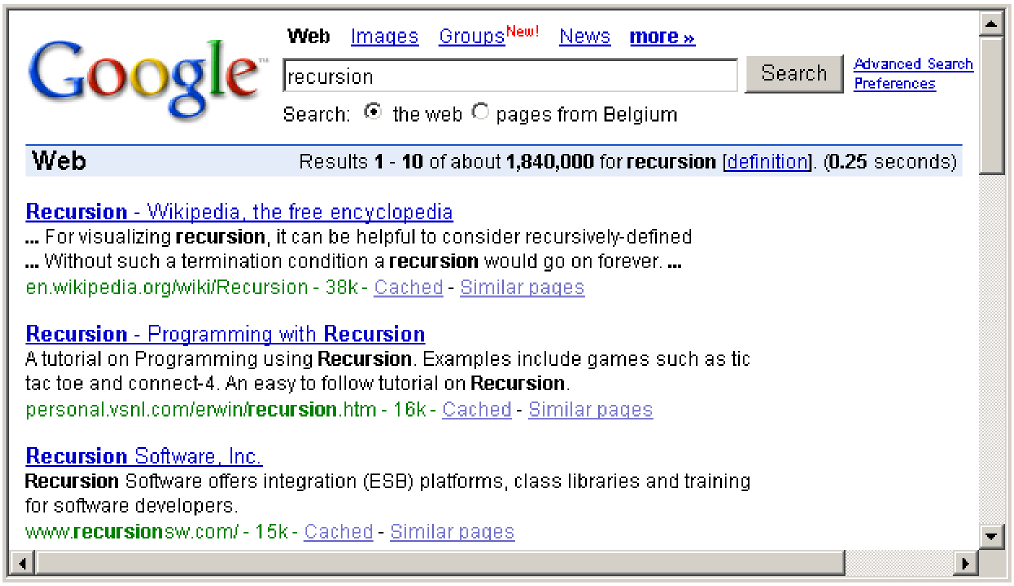
More learning facilities: Towards a learning engine

<u>Context</u>. For the purpose of the discussion, I shall assume that a teacher (Tom) uses COFALE to design a certain learning object, for instance linked lists, and that a computing-science student (Alice) is using COFALE to explore this learning object. While exploring linked lists, by accident Alice discovers that linked lists are closely related to the concept of recursion. Thus, she desires to master this concept. Tom, however, did not anticipate this situation, so he did not make the course on recursion available in COFALE.

Alice uses a search engine, for instance Google (2005), to search for websites that teach the concept of recursion. After introducing a keyword "recursion" in a small textbox and clicking on a button "Search", she receives a list of Web resources related to the keyword "recursion" (Figure 10.1). It is obvious that this list includes a great number of hyperlinks that are not suitable to Alice's objective. Even with a website (Figure 10.1: Recursion – Programming with Recursion) that is actually related to Alice's objective, it seems to be hard for Alice to attain her own learning goal because the website is an electronic book rather than a learning environment. So, Alice is not satisfied with the results provided by the search engine.

COFALE provides Alice with an online *learning engine* (integrated in COFALE). After introducing the keyword "recursion" (e.g., Figure 10.2), she is led to a learning hyperspace of the concept of recursion designed by another teacher (not Tom), as described in chapter 6. This kind of learning is called "right", anytime, and anywhere learning (Masie Center, 2003). That is, to provide the learner with the right learning materials, at the right time, in the right context, in the right amount, and so forth. I believe that *this learning facility reinforces the pedagogical principles underlying cognitive flexibility* because it provides the student with a very useful learning activity for exploring related concepts or topics on the demand.

Figure 10.1. Search results provided for Alice by Google



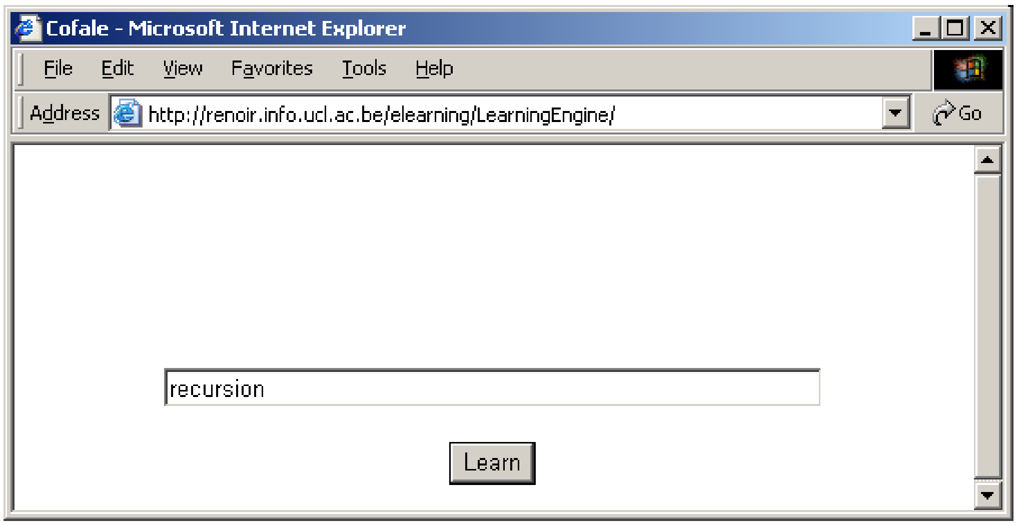
Results **1** - **10** of about **1,840,000** for **recursion** [definition]. (**0.25** seconds)

Recursion - Wikipedia, the free encyclopedia
... For visualizing **recursion**, it can be helpful to consider recursively-defined ... Without such a termination condition a **recursion** would go on forever. ...
en.wikipedia.org/wiki/Recursion - 38k - Cached - Similar pages

Recursion - Programming with **Recursion**
A tutorial on Programming using **Recursion**. Examples include games such as tic tac toe and connect-4. An easy to follow tutorial on **Recursion**.
personal.vsnl.com/erwin/**recursion**.htm - 16k - Cached - Similar pages

Recursion Software, Inc.
Recursion Software offers integration (ESB) platforms, class libraries and training for software developers.
www.**recursion**sw.com/ - 15k - Cached - Similar pages

Figure 10.2. A learning engine proposed for Alice


Cofale - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://renoir.info.ucl.ac.be/elearning/LearningEngine/
Go
recursion
Learn

Problems. Providing people with right, anytime, and anywhere learning is obviously *important* because it will change the way students and workers learn, as online search engines have been changing the fashion people search for information. In my personal point of view, the research on building learning engines gives rise to a number of critical problems in computing science, education, artificial intelligence, and learning psychology.

For example, a difficult problem is to create a very large repository of learning resources well described by metadata. If we want a learning engine be able to serve for a great diversify of learners studying in many different subjects, we may have to construct a very great number of learning objects in a variety of domains. Note that the Google robust search engine (2005) has recently indexed about *eight billion* Web links available in the Internet.

A second problem is to construct a database of learner profiles so that the engine can automatically provide each individual learner with appropriate learning materials in a given context. For example, in the previous scenario, if the engine has information about which domain Alice is studying and her prior knowledge, it will present her with the learning object on recursion and the Java programming language (but not the one on recursion and the LISP programming language she does not know).

Other problems include how to automate the process of constructing a learning object meeting a particular learning objective of the learner, how to index a large number of content objects and learners so that the engine can easily find the right objects for the right learners at the right time and in the right context.

Feasibility. I believe that, to develop a robust learning engine, it requires much effort of both the researcher and the practitioner in computing science, education, artificial intelligence, and learning psychology.

For example, for the repository of learning objects, several organizations of learning technologies have recently proposed standards and models such as IMS/SCORM (Advanced Distributed Learning, 2004) for practitioners in the field to build sharable content object databases (see also section 4.1). Therefore, in the coming years, I think a significant number of sharable learning objects will be available.

A possible way to enrich the database of content objects is to encourage learners to introduce learning resources they have found somewhere, together with several elements of metadata such as keywords for describing those resources (see also section 7.2.1).

Another way to create a large repository of learning resources is to crawl the Web, analyze Web resources, and define metadata for them. This method has been successfully used in various search engines such as Google (2005), Yahoo (2005). I think, however, the automatic process of analyzing the content of a Web page and defining learning metadata for it is very hard. For instance, it is very difficult for a system to automatically detect which topic(s) a Web page concerns, whether it is actually related to a particular learning concept. This issue concerns the field of artificial intelligence.

It is worth to note that the COFALE learning environment supports many learning activities without intervention of the course designer (see chapters 6 and 7). Using COFALE, the main work of the course designer is: (a) to create assets and content objects; (b) to construct learner models; and (c) to organize assets and content objects into information blocks and learning objects for different kinds of learners. Therefore, once we have a database of learning resources well described by metadata and a database of learner profiles (e.g., we ask learners to create and update it), we can use automated techniques, for example constraint satisfaction (Mitrovic, 2005; Deville et al., 1999; Van Hentenryck et al., 1998), to create the right learning object for the right learner in the right context.

More discussions related to the right, anytime, and anywhere learning are presented in Masie Center's "Making sense of learning specifications and standards" (2003).

Concluding remarks

I claim that the *approach* I used to exploit ICT and pedagogical principles in this thesis is more important than the results I obtained. I believe that clarifying what pedagogical principles entail in an *operational* manner is effective in the practice of instruction, because it provides a useful framework for the course designer, the teacher, and the software developer to design, use, and develop learning systems exhibiting the desired characteristics of the pedagogical principles.

By claiming constructivism for my educational approach in this thesis, as a personal choice, I am not saying that I dispute the widely reported findings of other learning theories. In practice of instruction, many educational approaches have been successfully used, and no one can say that constructivism is the best (Driscoll, 2000; Santrock, 2001). Similarly, by claiming domain-independence for my operational approach, I do not state that I dispute a variety of domain-dependent findings in cognitive science, for example findings about problem solving. Indeed, my dissertation is mainly concerned with cognitive flexibility, a domain-independent facet of constructivism (Spiro & Jehng, 1990).

A final point: What the present research attempts to do is to contribute a part of making learning and instruction, in a constructivist point of view, as easy as possible.

APPENDIX A: Implementation for the collection of compact discs

Two possible implementations for classes CD and CDCollection

Implementation 1 (successfully tested)

```
/* SPECIFICATION
 * Representation of a compact disc.
 */

import java.text.NumberFormat;

public class CD {

  // Instance variables
  private String title, artist;
  private double cost;
  private int tracks;

  // Constructor
  public CD(String title, String artist, double cost, int tracks) {
    this.title = title;
    this.artist = artist;
    this.cost = cost;
    this.tracks = tracks;
  }

  // Method
/* PRECOND
 * POSTCOND
 * Return a description of this CD.
 */
  public String toString() {
    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    return fmt.format(cost) + "\t" + tracks + "\t" + title + "\t" + artist;
  }
}

/* SPECIFICATION
 * Representation of a collection of compact discs.
 */

import java.text.NumberFormat;

public class CDCollection {

  // Instance variables
  private CD[] collection;
  private int count;
  private double totalCost;

  // Constructor
  public CDCollection() {
    collection = new CD[200];
    count = 0;
    totalCost = 0.0;
  }
  // Methods
/* PRECOND
 * POSTCOND
 * Adds a CD to the collection.
 */
  public void addCD(String title, String artist, double cost, int tracks) {
    if (count >= collection.length) return;

    collection[count] = new CD(title, artist, cost, tracks);
    totalCost += cost;
    count++;
  }

/* PRECOND
 * POSTCOND
 * Return a report describing the CD collection: the number of CDs, the total cost
 * and the average cost of all CDs, and the list of CDs.
 */
  public String toString() {
    if (count == 0) return "My CD Collection is empty.\n";

    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    String report = "My CD Collection:\n\n";
    report += "Number of CDs: " + count + "\n";
    report += "Total cost: " + fmt.format(totalCost) + "\n";
    report += "Average cost: " + fmt.format(totalCost/count) + "\n\n";
    report += "CD List:\n\n";
    for (int i = 0; i < count; i++)
      report += collection[i].toString() + "\n";

    return report;
  }
}
```

Implementation 2 (successfully tested)
The difference between Implementation 1 and Implementation 2 is marked by vertical lines.

```
/* SPECIFICATION
 * Representation of a compact disc.
 */
import java.text.NumberFormat;

public class CD {

  // Instance variables
  private String title, artist;
  private double cost;
  private int tracks;

  // Constructor
  public CD(String title, String artist, double cost, int tracks) {
    this.title = title;
    this.artist = artist;
    this.cost = cost;
    this.tracks = tracks;
  }

  // Methods
/* PRECOND
 * POSTCOND
 * Return the cost of this CD.
 */
  public double getCost() {
    return cost;
  }

/* PRECOND
```

```
 * POSTCOND
 * Return a description of this CD.
 */
  public String toString() {
    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    return fmt.format(cost) + "\t" + tracks + "\t" + title + "\t" + artist;
  }
}

/* SPECIFICATION
 * Representation of a collection of compact discs.
 */
import java.text.NumberFormat;

public class CDCollection {

  // Instance variables
  private CD[] collection;
  private int count;

  // Constructor
  public CDCollection() {
    collection = new CD[200];
    count = 0;
  }

  // Methods
/* PRECOND
 * POSTCOND
 * Adds a CD to the collection.
 */
  public void addCD(String title, String artist, double cost, int tracks) {
    if (count >= collection.length) return;

    collection[count] = new CD(title, artist, cost, tracks);
    count++;
  }

/* PRECOND
 * POSTCOND
 * Return a report describing the CD collection: the number of CDs, the total cost
 * and the average cost of all CDs, and the list of CDs.
 */
  public String toString() {
    if (count == 0) return "My CD Collection is empty.\n";

    String report = "My CD Collection:\n\n";

    report += "CD List:\n\n";
    double totalCost = 0.0;
    for (int i = 0; i < count; i++) {
      report += collection[i].toString() + "\n";
      totalCost += collection[i].getCost();
    }

    report += "\nNumber of CDs: " + count + "\n";
    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    report += "Total cost: " + fmt.format(totalCost) + "\n";
    report += "Average cost: " + fmt.format(totalCost/count) + "\n";

    return report;
  }
}
```

APPENDIX B: Teaching recursion

This appendix is a reference for the learning resources used for the design of learning materials presented in chapter 3. Note that all Java implementations presented here were successfully tested. I have organized this appendix into the next six sections:

1. B1: Basic concepts related to recursion.
2. B2: Learning situations for recursion.
3. B3: Basics concepts related to linked lists.
4. B4: Learning situations for linked lists.
5. B5: General discussion questions.
6. B6: Teaching recursion in the literature.

Appendix B1: Basic concepts related to recursion

Recursion

Definition: Recursion is the process of defining something in terms of itself.

Example 1: Definition of the factorial function
1! = 1
N! = N x **(N – 1)!** (N > 1)

Example 2: Definition of a list of one or more numbers
A **List** is a: number
or a: number comma **List**

For instance, tracing the recursive definition of the list 2, 3, 5, 7:

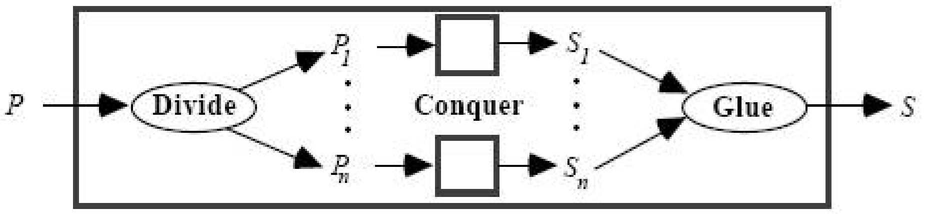
```
List: number comma List
         2      ,   3, 5, 7
                    number comma List
                       3      ,   5, 7
                                  number comma List
                                     5      ,   7
                                                number
                                                   7
```

DCG Strategy

DCG stands for *divide*, *conquer*, and *glue*. In problem solving, this strategy means (see also the following figure):

- *Divide* a problem P into sub-problems P_1, P_2, … P_n.
- *Conquer* these sub-problems by solving them and yielding sub-solutions S_1, S_2, … S_n.
- *Glue* these sub-solutions together into the solution S to the whole problem.

DCG strategy in problem solving (Turbak et al., 1999)



Recursive thinking

Definition: Recursive thinking is the ability of humans to solve a problem by reducing it to one or more sub-problems that are (1) **identical** in structure to the original problem and (2) somewhat **simpler** to solve.
Note: We would say this technique of problem solving is top down.

Example1: Computing 2^{15}
STEP 1: since $2^{15} = 2^7 * 2^8$, we compute 2^7 and 2^8
STEP 2: since $2^7 = 2^3 * 2^4$ and $2^8 = 2^4 * 2^4$, we compute 2^3 and 2^4
STEP 3: since $2^3 = 2 * 2^2$ and $2^4 = 2^2 * 2^2$, we compute 2^2
STEP 4: we know $2^2 = 2 * 2 = 4$, so $2^3 = 2 * 2^2 = 8$, $2^4 = 2^2 * 2^2 = 16$, $2^7 = 2^3 * 2^4 = 128$, $2^8 = 2^4 * 2^4 = 256$, and $2^{15} = 2^7 * 2^8 =$ **32768**
Example2: Look up a word in a dictionary, for instance, "recursion"
STEP 1: we look up the words that start with 'r'

STEP 2: within these words, we look up the words that begin with “re”
…

Recursive algorithms

Definition: A recursive algorithm to solve a problem is a set of steps in each step we see one or more sub-problems that are either solved or **identical** in structure to and **smaller** than the problem(s) of the previous step.
Notes: We use the term “**base case**” to denote a simple case in a recursive algorithm, and we use the term “**recursive part**” to denote the references to the problems that are identical in structure to the original problem.

Example: A recursive procedure for factorials

```
procedure factorial (n : positive integer)
/* BASE CASE */
if n = 1 then
  factorial (n) : = 1
/* RECURSIVE PART */
else
  factorial (n) : = n * factorial (n - 1)
```

For instance, tracing the computation of 5! :
STEP 1: since 5! = 5 * 4!, we compute 4! (identical to and smaller than 5!)
STEP 2: since 4! = 4 * 3!, we compute 3! (identical to and smaller than 4!)
STEP 3: since 3! = 3 * 2!, we compute 2! (identical to and smaller than 3!)
STEP 4: since 2! = 2 * 1!, we compute 1! (identical to and smaller than 2!)
STEP 5: we know 1! = 1, so 2! = 2 *1 = 2, 3! = 3 * 2! = 6, 4! = 4 * 3! = 24, and 5! = 5 * 4! = **120**

Iterative thinking

Definition: Iterative thinking is the ability of humans to **combine** a number of **small** and **well-known** problems to solve a larger problem.
Note: We would say this technique of problem solving is bottom up.

Example1: Computing 2^{15}
STEP 1: we compute $2^2 = 2 * 2 = 4$
STEP 2: we compute $2^3 = 2 * 2^2 = 8$
…
STEP 14: we compute $2^{15} = 2 * 2^{14} = 2 * 16384 =$ **32768**

Example 2: Look up a word in a dictionary, for instance, “recursion”
No one looks up a word from the beginning of the dictionary because all the words in the dictionary are ordered alphabetically.

Iterative algorithms

Definition: An iterative algorithm to solve a problem is a set of steps in each step we see one or more problems that are **solved** either independently or by combining the solutions of the problems in previous steps.
Note: Iterative thinking often produces an iterative algorithm.

Example: An iterative procedure for factorials

```
procedure factorial (n : positive integer)
f : = 1
for i : = 2 to n
  f : = i * f
/* f is n! */
```

For instance, tracing the computation of 5! :
STEP 1: we compute 1 * 2 = 2
STEP 2: we compute 2 * 3 = 6
STEP 3: we compute 6 * 4 = 24
STEP 4: we compute 24 * 5 = **120**

Recursive methods

Definition: In programming, a recursive method is a method that calls **itself**.
Notes: Other possible names for recursive methods are recursive functions, recursive procedures.

Example: A Java method for computing n!
```
/* PRECOND
 * n is a positive integer
 * POSTCOND
 * Return n!
 */
public int factorial (int n) {
   /* BASE CASE */
   if (n == 1) return 1;
   /* RECURSIVE PART */
   else return n * factorial (n - 1); // this method calls itself
}
```

Base cases

Definition: The base cases of a recursive definition (e.g., a recursive algorithm or a recursive data structure) are the **non-recursive part** of the definition; this non-recursive part permits the recursion to eventually end.
Notes: We find out the base case(s) of a recursive algorithm by examining its simplest case(s).

Example: A Java method for computing n!
```
/* PRECOND
 * n is a positive integer
 * POSTCOND
 * Return n!
 */
public int factorial (int n) {
   /* BASE CASE */
   if (n == 1) return 1; // this command line shows the simplest case of n!
   /* RECURSIVE PART */
   else return n * factorial (n - 1);
}
```

Recursive part

Definition: The recursive part of a recursive definition (e.g., a recursive algorithm or a recursive data structure) is the part (of the definition) in which one or more **self-references** occur.
Notes: We find out the recursive part of a recursive algorithm by examining the self-reference(s).

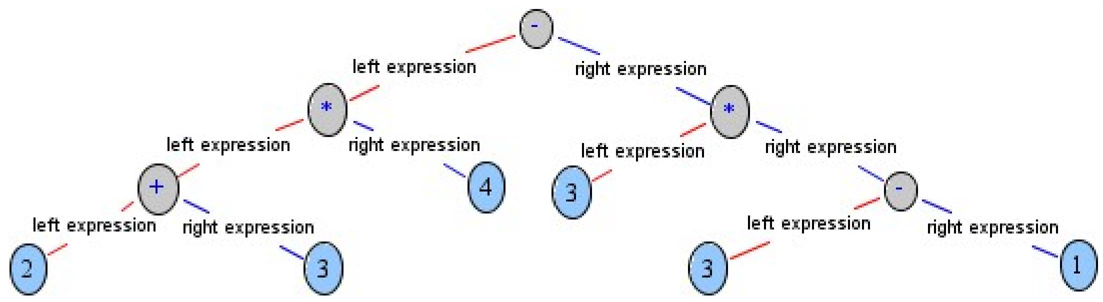
Example: A Java method for computing n!
```
/* PRECOND
 * n is a positive integer
 * POSTCOND
 * Return n!
 */
public int factorial (int n) {
   /* BASE CASE */
   if (n == 1) return 1;
   /* RECURSIVE PART */
   else return n * factorial (n - 1); // this command line shows a self-reference : the method calls itself
}
```

Appendix B2: Learning situations for recursion

Arithmetic expressions

A simple arithmetic expression composes of integer numbers and simple operators +, -, *, and / (integer division), for example, E = (2 + 3) * 4 – 3 * (3 – 1). We often use a binary tree to represent an arithmetic expression, as the following figure.

A recursive representation of the expression (2 + 3) * 4 – 3 * (3 – 1)



Recursive definition

The following definition defines arithmetic expressions in a recursive manner.

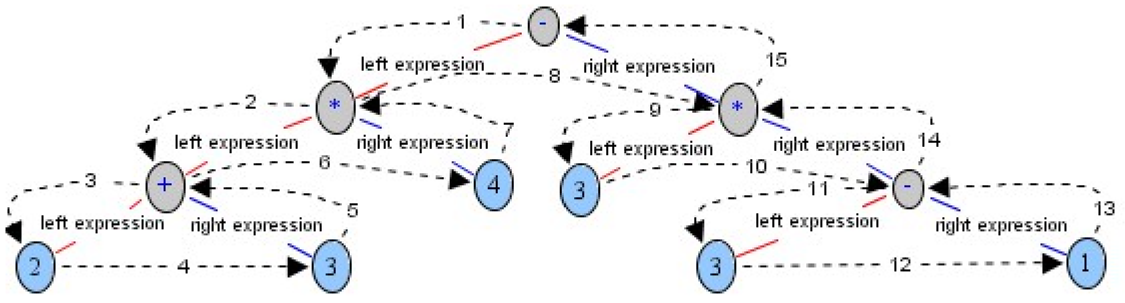
A recursive definition of arithmetic expressions

An *operator* is: + or - or * or /.
An *expression* is an: integer number
or an: *expression operator expression*.
For instance, tracing the recursive definition of the expression (2 + 3) * 4 - 3 * (3 - 1)
expression: *expression* *operator* *expression*
(2 + 3) * 4 - 3 * (3 - 1)
expression operator number number *operator expression*
2 + 3 * 4 3 * 3 - 1
number *operator* number number *operator* number
2 + 3 3 - 1

Recursive evaluation

The following figure shows how to evaluate an arithmetic expression in a recursive manner.

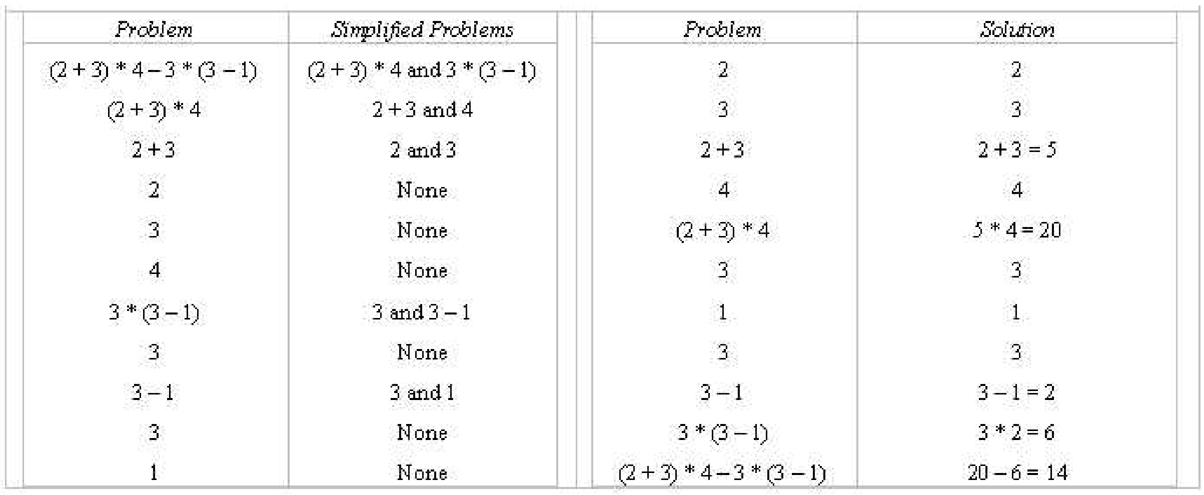
A tree-recursive evaluation of the expression (2 + 3) * 4 – 3 * (3 – 1)
(Follow the arrows from 1 to 15)



Recursive evaluation process

The following table points out how to decompose the evaluation of an arithmetic expression and how to combine sub-solutions to build the final solution.

Decomposing and combining sub-solutions of the evaluation of (2 + 3) * 4 – 3 * (3 – 1)



Problem	*Simplified Problems*	*Problem*	*Solution*
(2 + 3) * 4 – 3 * (3 – 1)	(2 + 3) * 4 and 3 * (3 – 1)	2	2
(2 + 3) * 4	2 + 3 and 4	3	3
2 + 3	2 and 3	2 + 3	2 + 3 = 5
2	None	4	4
3	None	(2 + 3) * 4	5 * 4 = 20
4	None	3	3
3 * (3 – 1)	3 and 3 – 1	1	1
3	None	3	3
3 – 1	3 and 1	3 – 1	3 – 1 = 2
3	None	3 * (3 – 1)	3 * 2 = 6
1	None	(2 + 3) * 4 – 3 * (3 – 1)	20 – 6 = 14

Java implementation

The following implementation shows how to use the Java programming language to represent and evaluate arithmetic expressions.

A Java implementation of arithmetic expressions

```
/* SPECIFICATION
 * This class is a data type for arithmetic expressions
 */
public class Expression{
  /* BASE CASE */
   /* identifier is the root of the binary tree representing the expression
      it could be + or - or * or / or a string representing an integer */
   private String identifier;
  /* RECURSIVE PART */
   /* left and right are the left sub-expression and the right sub-expression
      left and right are null if identifier is an integer */
   private Expression left, right;

   /* Construct an integer as an expression, s represents an integer */
   public Expression(String s){
      identifier = s;
      left = null;
      right = null;
   }
   /* Construct a compound expression, s represents an operator */
   public Expression(String s, Expression l, Expression r){
      identifier = s;
      left = l;
      right = r;
   }
   /* PRECOND
    * This is a correct arithmetic expression
    * POSTCOND
    * Return the integer value of this expression
    */
   public int evaluate(){
      /* RECURSIVE PART */
    if (identifier == "+" ) return left.evaluate() + right.evaluate();
      else if (identifier == "-" ) return left.evaluate() - right.evaluate();
      else if (identifier == "*" ) return left.evaluate() * right.evaluate();
      else if (identifier == "/" ) return left.evaluate() / right.evaluate();
      /* BASE CASE */
      /* Convert identifier into an integer and return it */
      else return Integer.parseInt(identifier);
   }
}
```

Java test class

The following implementation shows an example for testing arithmetic expressions.

A Java implementation testing arithmetic expressions

```
/* SPECIFICATION
 * This class tests the evaluation of an arithmetic expression
 */
public class TestExpression{
  public static void main (String[] args){
    Expression two = new Expression ("2");
    Expression three = new Expression ("3");
    Expression ex1 = new Expression ("+", two, three);
    Expression four = new Expression ("4");
    Expression ex2 = new Expression ("*", ex1, four);
    Expression one = new Expression ("1");
    Expression ex3 = new Expression ("-", three, one);
    Expression ex4 = new Expression ("*", three, ex3);
    Expression myExpression = new Expression ("-", ex2, ex4);
    System.out.println("The value of my expression is: " + myExpression.evaluate());
  }
}
```

Simple text search

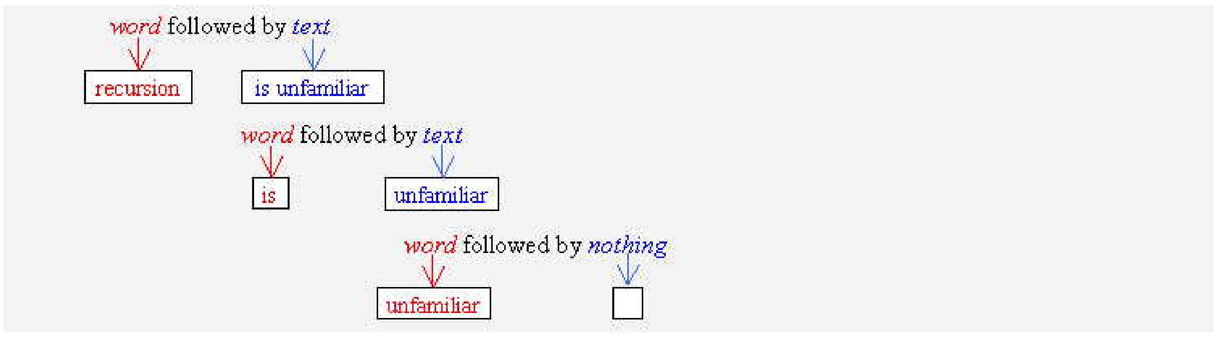
A text is a set of words such as a phrase or a paragraph or a document. The length of a text, meaning that the number of words in the text, could range between a few words such as a phrase to hundreds of thousand of words such as a document. An example of a short text is “*recursion is unfamiliar*” and an example of a longer text is “*re-cursion is a very difficult concept for students to learn because recursion is unfamiliar and complex*”. The problem is to identify whether a short text appears in a long text.

Recursive definition

A *word* is a group of lowercase letters (words are case insensitive).
A *text* is a: *word*
or a: *word* followed by a *text* (we omit punctuation marks in a text).
Note: We define the tail of a text to be the text without the first word.

Definition tracing

Tracing the recursive definition of the text "*recursion is unfamiliar*"



Recursive algorithm

A recursive algorithm for searching text T1 in text T2

By applying the DCG strategy, we divide the search problem into one sub-problem, as follows:

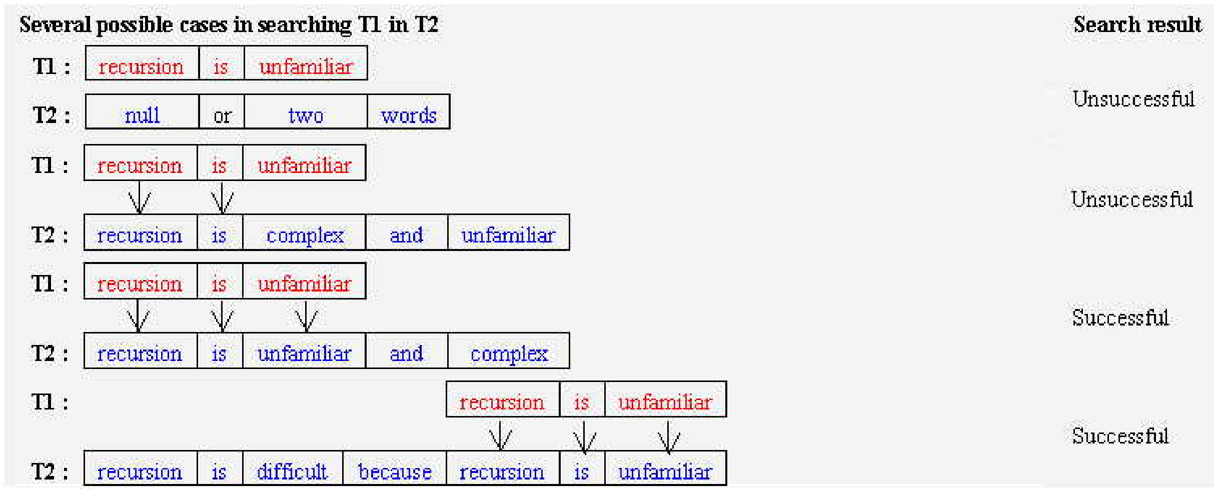
```
procedure search (T1, T2 : simple texts)
/* BASE CASE */
if T1 = null then search (T1, T2) : = true /* It is a convention */
else if begin (T1, T2) then search (T1, T2) : = true /* T1 appears at the beginning of T2 */
else if (tail of T2) = null then search (T1, T2) : = false /* We reach the end of T2 and T1 is not found */
/* RECURSIVE PART */
else search (T1, T2) : = search (T1, tail of T2) /* Only one sub-problem */
```

We divide the problem of checking whether T1 appears at the beginning of T2 into one sub-problem, as follows:

```
procedure begin (T1, T2 : simple texts)
/* BASE CASE */
if first word of T1 is different from first word of T2 then begin (T1, T2) : = false
else if (tail of T1) = null then begin (T1, T2) : = true /* We reach the end of T1 before we reach the end of T2 */
else if (tail of T2) = null then begin (T1, T2) : = false /* We reach the end of T2 before we reach the end of T1 */
/* RECURSIVE PART */
else begin (T1, T2) : = begin (tail of T1, tail of T2) /* Only one sub-problem */
```

You should note that there is no “glue” stage in this algorithm because we divide the original problem into only one sub-problem; this is a simple form of the DCG strategy.

Searching examples



Several possible cases in searching T1 in T2
Search result
T1 : recursion is unfamiliar
T2 : null or two words
Unsuccessful
T1 : recursion is unfamiliar
T2 : recursion is complex and unfamiliar
Unsuccessful
T1 : recursion is unfamiliar
T2 : recursion is unfamiliar and complex
Successful
T1 : recursion is unfamiliar
T2 : recursion is difficult because recursion is unfamiliar
Successful

Java implementation

A Java implementation of simple texts

```
/* SPECIFICATION
 * This class is a data type for simple texts
 */
public class SimpleText{
   /* BASE CASE */
   /* The first word of the text */
   private String word;
   /* RECURSIVE PART */
   /* The tail of the text */
   private SimpleText tail;

   /* Construct a word as a text, s is a word */
   public SimpleText(String s){
      word = s;
      tail = null;
   }
   /* Construct a compound text, s is a word and t is a text */
   public SimpleText(String s, SimpleText t){
      word = s;
      tail = t;
   }
   /* Get the word of the text */
   public String getWord(){
      return word;
   }
   /* Get the tail of the text */
   public SimpleText getTail(){
      return tail;
   }

   /* PRECOND
    * No comment
    * POSTCOND
   * Return true if t is null or t appears in this text, otherwise return false
   */
  public boolean contain(SimpleText t){
    /* BASE CASES */
    if (t == null) return true; // By convention
    else if (containAtTheBeginning(t)) return true; // t appears at the beginning
    else if (tail == null) return false; // Reach the end of this text but not found
    /* RECURSIVE PART */
    else return tail.contain(t); // Check whether t appears in the tail of the text
  }
  /* PRECOND
   * t is not null
   * POSTCOND
   * Return true if t appears at the beginning of this text, otherwise return false
   */
  private boolean containAtTheBeginning(SimpleText t){
    /* BASE CASES */
    if (word != t.getWord()) return false; // The first words are different
    else if (t.getTail() == null) return true; // Reach the end of t and found
    else if (tail == null) return false; // Reach the end of this text and not found
    /* RECURSIVE PART */
    /* Check if the tail of t appears at the beginning of the tail of this text */
    else return tail.containAtTheBeginning(t.getTail());
  }
}
```

Java test class

The following implementation shows an example for testing simple text search.

A Java implementation testing simple text search

```
/* SPECIFICATION
 * This class tests whether a text t1 is contained in another text t2
 */
public class TestSimpleText{
  public static void main (String[] args) {
    SimpleText t1 = new SimpleText("recursion", new SimpleText("is", new SimpleText("unfamiliar"));
    SimpleText t2 = new SimpleText("recursion", new SimpleText("is", new SimpleText("difficult", new SimpleText
    ("because", new SimpleText("recursion", new SimpleText("is", new SimpleText("unfamiliar")))));
    System.out.println("Search result: " + t2.contain(t1));
  }
}
```

Fibonacci numbers

The Fibonacci numbers, F_0, F_1, F_2, … are defined by the following equations.

A recursive formula for computing Fibonacci numbers

- $F_0 = 0$ and $F_1 = 1$.
- $F_n = F_{n-1} + F_{n-2}$ if $n >= 2$.

An iterative formula for computing Fibonacci numbers

- $F_0 = 0$ and $F_1 = 1$.
- $F_2 = F_0 + F_1 = 0 + 1 = 1$, $F_3 = F_1 + F_2 = 1 + 1 = 2$; $F_4 = F_2 + F_3 = 1 + 2 = 3$, …

First Java implementation

```
/* A recursive implementation using a recursive algorithm */
/* PRECOND
 * n is a nonnegative integer
 * POSTCOND
 * Return F(n)
 */
public int fibonacciRecursive (int n) {
   /* BASE CASES */
   if (n == 0) return 0;
   else if (n == 1) return 1;
   /* RECURSIVE PART */
   else return fibonacciRecursive(n - 1) + fibonacciRecursive(n - 2);
}
```

Second Java implementation

```
/* A recursive implementation using an iterative algorithm */
/* PRECOND
 * n is a nonnegative integer
 * POSTCOND
 * Return F(n)
 */
public int fibonacciIterative1(int n){
   /* BASE CASES */
   if (n == 0) return 0;
   else if (n == 1) return 1;
   /* ITERATIVE PART */
   // 0, 1 are F(0) and F(1).
   // 2 is the initial index of the iterative process: We iterate from 2 to n
   else return iterateFibonacci(0, 1, 2, n);
}
// A supplementary method helping the implementation of the previous method
private int iterateFibonacci(int number1, int number2, int index, int n){
   // Apply F(n) = F(n-1) + F(n-2) for calculating F(index)
   int number = number1 + number2;
   // We have reached the end of the iterative process, F(n) = F(index)
   if (index == n) return number;
   // A recursive call to calculate F(index + 1) in the iterative process
   else return iterateFibonacci(number2, number, index + 1, n);
}
```

Third Java implementation

```
/* An iterative implementation using an iterative algorithm */
/* PRECOND
 * n is a nonnegative integer
 * POSTCOND
 * Return F(n)
 */
public int fibonacciIterative2(int n){
   /* BASE CASES */
   if (n == 0) return 0;
   else if (n == 1) return 1;
   /* ITERATIVE PART */
```

```
  else{
    int number1 = 0; // F(0)
    int number2 = 1; // F(1)
    int number = 0; // number is used for calculating F(n)
    for (int i = 2; i <= n; i++){
      number = number1 + number2; // Calculate F(i)
      // Reassign intermediate numbers to calculate F(i+1)
      number1 = number2;
      number2 = number;
    }
    return number;
  }
}
```

Partition

A partition of a positive integer *m* is a way to write *m* as a sum of positive integers. Let P_m equal the number of different partitions of *m*, where the order of terms in the sum does not matter. Given *m*, find P_m.
We list all partitions of the number five in the following example.
This example is complex and compelling for recursion. This example is also good to explain how to go from a problem specification to a recursive solution. People must think the problem recursively to be able to solve it.

All partitions of the number five

(1) 5 = 5
(2) 5 = 4 + 1
(3) 5 = 3 + 2
(4) 5 = 3 + 1 + 1
(5) 5 = 2 + 2 + 1
(6) 5 = 2 + 1 + 1 + 1
(7) 5 = 1 + 1 + 1 + 1 + 1

Recursive formula

A recursive formula for computing P_m

Let $P_{m,n}$ be the number of different ways to express the positive integer *m* as the sum of positive integers not exceeding the positive integer *n*. The following recursive definitions are correct and can be used to compute P_m:

- $P_m = P_{m,m}$
- $P_{1,n} = 1$ and $P_{m,1} = 1$
- $P_{m,m} = 1 + P_{m,m-1}$ if $m > 1$
- $P_{m,n} = P_{m,m}$ if $m < n$
- $P_{m,n} = P_{m,n-1} + P_{m-n,n}$ if $m > n > 1$

Recursive computation process

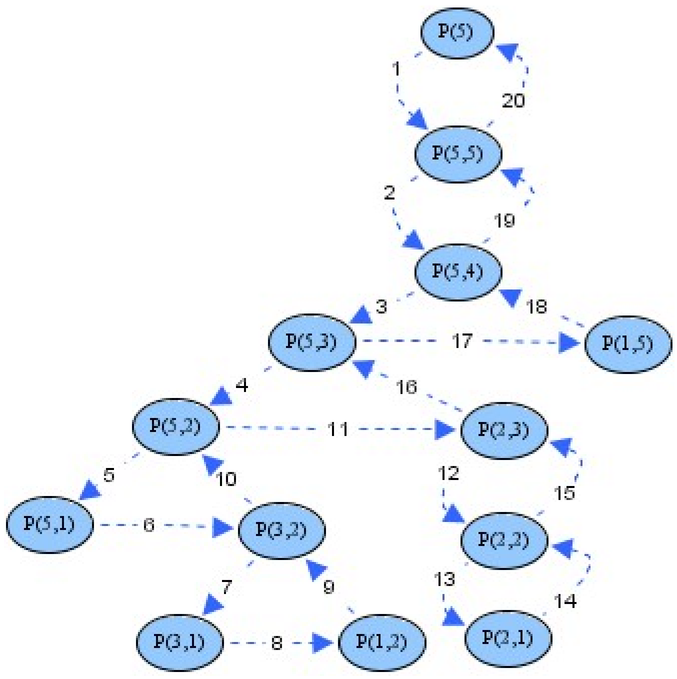
The following table points out how to decompose the computation of P_5 and how to combine sub-solutions to build the final solution.

Decomposing and combining sub-solutions of P_5

Problem	*Simplified Problem*	*Problem*	*Solution*
P_5	$P_{5,5}$	$P_{5,1}$	1
$P_{5,5}$	$1 + P_{5,4}$	$P_{3,1}$	1
$P_{5,4}$	$P_{5,3} + P_{1,5}$	$P_{1,2}$	1
$P_{5,3}$	$P_{5,2} + P_{2,3}$	$P_{3,2}$	$P_{3,1} + P_{1,2} = 1 + 1 = 2$
$P_{5,2}$	$P_{5,1} + P_{3,2}$	$P_{5,2}$	$P_{5,1} + P_{3,2} = 1 + 2 = 3$
$P_{5,1}$	None	$P_{2,1}$	1
$P_{3,2}$	$P_{3,1} + P_{1,2}$	$P_{2,2}$	$1 + P_{2,1} = 1 + 1 = 2$
$P_{3,1}$	None	$P_{2,3}$	$P_{2,3} = P_{2,2} = 2$
$P_{1,2}$	None	$P_{5,3}$	$P_{5,2} + P_{2,3} = 3 + 2 = 5$
$P_{2,3}$	$P_{2,2}$	$P_{1,5}$	1
$P_{2,2}$	$1 + P_{2,1}$	$P_{5,4}$	$P_{5,3} + P_{1,5} = 5 + 1 = 6$
$P_{2,1}$	None	$P_{5,5}$	$1 + P_{5,4} = 6 + 1 = 7$
$P_{1,5}$	None	P_5	$P_5 = P_{5,5} = 7$

Tree-recursive computing process

A tree-recursive computing process of P_5
(Follow the arrows from 1 to 20)



$$\begin{tikzcd}
 & & & & P(5) \arrow[d, dashed, bend right, "1"'] & & \\
 & & & & P(5,5) \arrow[u, dashed, bend right, "20"'] \arrow[d, dashed, bend right, "2"'] & & \\
 & & & & P(5,4) \arrow[u, dashed, bend right, "19"'] \arrow[dl, dashed, "3"'] & & \\
 & & & P(5,3) \arrow[rrr, dashed, "17"] \arrow[dl, dashed, "4"'] & & & P(1,5) \arrow[ull, dashed, "18"'] \\
 & & P(5,2) \arrow[rrrr, dashed, "11"] \arrow[dl, dashed, "5"'] & & & & P(2,3) \arrow[ulll, dashed, "16"'] \arrow[d, dashed, bend right, "12"'] \\
 & P(5,1) \arrow[rr, dashed, "6"] & & P(3,2) \arrow[ul, dashed, "10"'] \arrow[dl, dashed, "7"'] & & & P(2,2) \arrow[u, dashed, bend right, "15"'] \arrow[d, dashed, bend right, "13"'] \\
 & & P(3,1) \arrow[rr, dashed, "8"] & & P(1,2) \arrow[ul, dashed, "9"'] & & P(2,1) \arrow[u, dashed, bend right, "14"']
\end{tikzcd}$$

From problem to solution

A dialogue between tutor and student for finding a recursive solution of the partition problem

Tutor: You have examined all of the ways to write five as a sum of positive integers. Think recursively, what can you find out from this example?

Student: I cannot find out the relationship between P_5 and P_4, P_3, P_2, and P_1. Although, I see that there is only one way to write 5 = 5 and that there are six ways to write five as a sum of positive integers not exceeding four.

Tutor: Well, you have found that the number of ways to express five as the sum of positive integers not exceeding five is equal to the number of ways to write five as the sum of positive integers not exceeding four plus one, haven't you? Don't you see the sub-problem?

Student: Oh yes I do, however, how to divide this sub-problem now?

Tutor: You have divided the original problem by distinguishing two cases: one with the presence of the number five, and one without it. Why don't you apply this strategy now?

Student: Right, the number of ways to express five as the sum of positive integers not exceeding four is equal to the number of ways to write five as the sum of positive integers not exceeding three plus one, because there is only one way to express 5 = 4 + 1.

Tutor: Really? It is a particular case. Think to write 10 as the sum of positive integers not exceeding four.

Student: I understand. I would say the number of ways to express 10 as the sum of positive integers not exceeding four is equal to the number of ways to write 10 as the sum of positive integers not exceeding three plus the number of ways to write six (10 – 4) as the sum of positive integers not exceeding 4.

Tutor: Can you complete the definitions to compute P_m now?

Student: Yes, I should use $P_{m,n}$ to denote the number of different ways to express m as the sum of positive integers not exceeding the positive integer n, then give base cases as well as recursive definitions for that function.

Recursive algorithm

A recursive procedure for computing $P_{m,n}$

```
procedure partition (m, n : positive integer)
/* BASE CASE */
if m = 1 or n = 1 then partition (m, n) : = 1
/* RECURSIVE PART */
else if m < n then partition (m, n) : = partition (m, m)
else if m = n then partition (m, n) : = 1 + partition (m, m – 1)
else partition (m, n) : = partition (m, n – 1) + partition (m – n, n)
```

Appendix B3: Basic concepts related to linked lists

(Some part of learning contents for linked lists were reused from Friesen, 2003)

Linked-list data structure (from Friesen, 2003)

The linked-list data structure involves four concepts: the self-referential class, node, link field, and link:

Self-referential class: a class with at least one field whose reference type is the class name:

```
class Employee
{
   private int empno;
   private String name;
   private double salary;

   public Employee next;

   // Other members
}
```

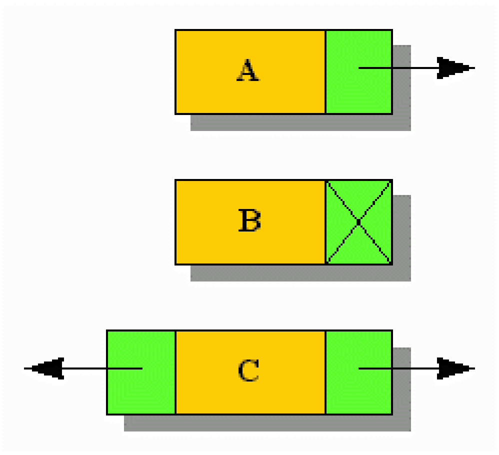
`Employee` is a self-referential class because its `next` field has type `Employee`.

Node: an object you create from a self-referential class.

Link field: a field whose reference type is the class name. In the code fragment above, `next` is a link field. In contrast, `empno`, `name`, and `salary` are nonlink fields.

Link: the reference in a link field. In the code fragment above, `next`'s reference to an `Employee` node is a link.
The four concepts above lead to the following definition: a *linked list* is a sequence of nodes that interconnect via the links in their link fields. Computer scientists use a special notation to illustrate linked lists. A variant of that notation appears in the following figure.

A special notation for illustrating linked lists

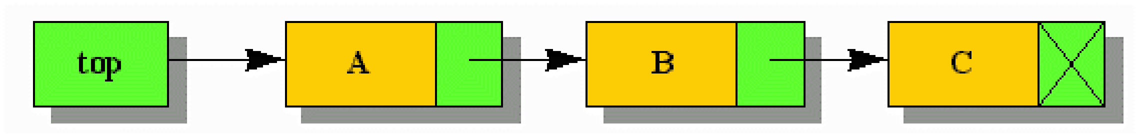


The figure presents three nodes: A, B, and C. Each node divides into a contents area (in orange) and one or more link areas (in green). The contents area represents all nonlink fields, and each link area represents a link field. A's single link area and C's link areas incorporate an arrow to signify a reference to some other node of the same type (or a subtype). B's single link area incorporates an X to signify a null reference. In other words, B doesn't connect to any other node.

Singly linked list (from Friesen, 2003)

A *singly linked list* is a linked list of nodes, where each node has a single link field. A reference variable holds a reference to the first node, each node (except the last) links to the next node, and the last node's link field contains null to signify the list's end. Although the reference variable is commonly named `top`, you can choose any name you want. The next figure presents a three-node singly linked list, where `top` references the A node, A connects to B, B connects to C, and C is the final node.

A three-node singly linked list contains connected nodes A, B, and C



Insertion

One common singly linked list algorithm is node-insertion. That algorithm is somewhat involved because it must deal with four cases: when the node must be inserted before the first node; when the node must be inserted after the last node; when the node must be inserted between two nodes; and when the singly linked list does not exist. Before studying each case, consider the following pseudo code:

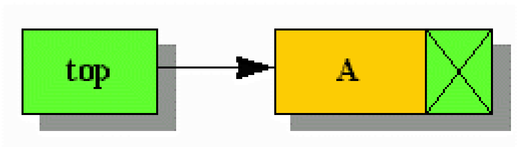
```
DECLARE CLASS Node
  DECLARE STRING name
  DECLARE Node next
END DECLARE

DECLARE Node top = NULL
```

The pseudo code above declares a `Node` self-referential class with a `name` non link field and a `next` link field. The pseudo code also declares a `top` reference variable (of type `Node`) that holds a reference to the first `Node` in a singly linked list. Because the list does not yet exist, `top`'s initial value is `NULL`. Each of the following four cases assumes the `Node` and `top` declarations:

The singly linked list does not exist: This is the simplest case. Create a `Node`, assign its reference to `top`, initialize its non link field, and assign `NULL` to its link field. The following pseudo code performs those tasks:

```
top = NEW Node
top.name = "A"
top.next = NULL
```

The following figure shows the initial singly linked list that emerges from the pseudo code above.

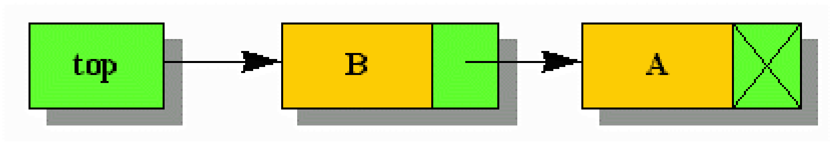


The initial singly linked list contains the solitary Node A

The node must be inserted before the first node:. Create a `Node`, initialize its non link field, assign `top`'s reference to the `next` link field, and assign the newly created `Node`'s reference to `top`. The following pseudo code (which assumes that the previous pseudo code has executed) performs those tasks:

```
DECLARE Node temp

temp = NEW Node
temp.name = "B"
temp.next = top
top = temp
```

The resulting two-`Node` list appears in the next figure.

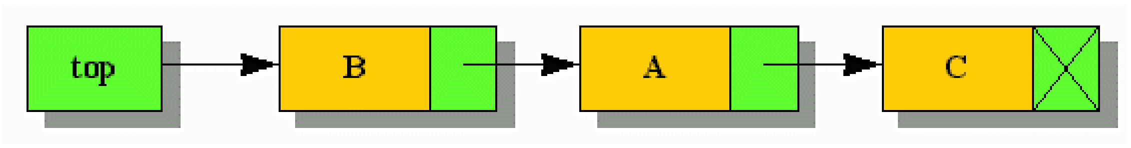


The expanded two-Node singly linked list places Node B ahead of Node A

The node must be inserted after the last node. Create a Node, initialize its non link field, assign NULL to the link field, traverse the singly linked list to find the last Node, and assign the newly created Node's reference to the last Node's next link field. The following pseudo code performs those tasks:

```
temp = NEW Node
temp.name = "C"
temp.next = NULL

DECLARE Node temp2

temp2 = top

// We assume top (and temp2) are not NULL
// because of the previous pseudo code

WHILE temp2.next IS NOT NULL
  temp2 = temp2.next
END WHILE

// temp2 now references the last node

temp2.next = temp
```

The next figure reveals the list following the insertion of Node C after Node A.

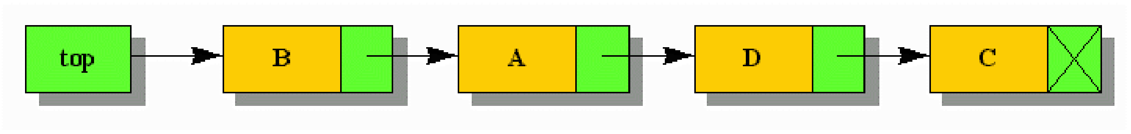


Node C comes last in the expanded three-node singly linked list

The node must be inserted between two nodes: This is the most complex case. Create a Node, initialize its non link field, traverse the list to find the Node that appears before the newly created Node to be inserted, assign the link in that previous Node's next link field to the newly created Node's next link field, and assign the newly created Node's reference to the previous Node's next link field. The following pseudo code performs those tasks:

```
temp = NEW Node
temp.name = "D"

temp2 = top

// We assume that the newly created Node is inserted after Node
// A and that Node A exists. In the real world, there is no
// guarantee that any Node exists, so we would need to check
// for temp2 containing NULL in both the WHILE loop's header
// and after the WHILE loop completes.

WHILE temp2.name IS NOT "A"
  temp2 = temp2.next
END WHILE

// temp2 now references Node A.

temp.next = temp2.next
temp2.next = temp
```

The following figure presents the list following the insertion of Node D between Nodes A and C.



The ever-growing singly linked list places Node D between Nodes A and C.

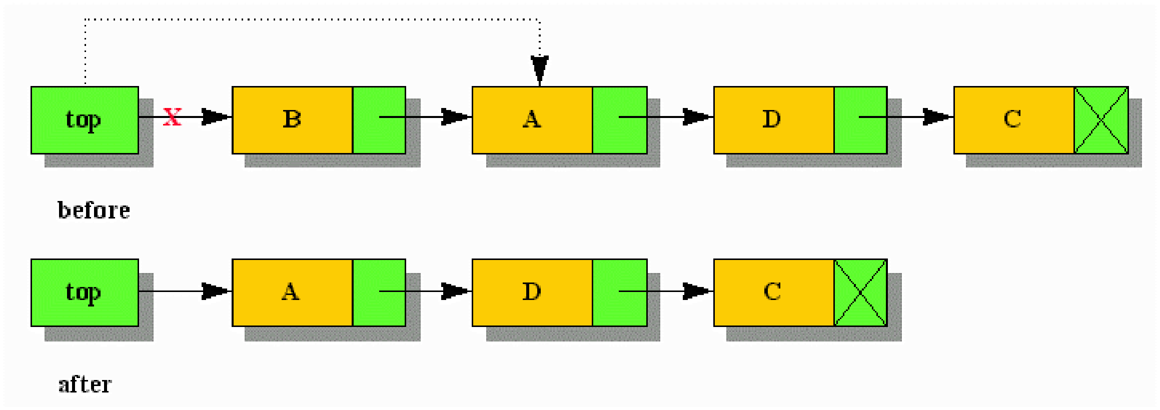
Deletion

A second common singly linked list algorithm is node-deletion. Unlike node-insertion, there are only two cases to consider: delete the first node and delete any node but the first node. Each case assumes a singly linked list with at least one node exists:

Delete the first node: Assign the link in the `top`-referenced `Node`'s `next` field to `top`:

top = top.next; // Reference the second Node (or NULL if there is only one Node)

The next figure presents before and after views of a list where the first `Node` is deleted. In that figure, `Node B` disappears and `Node A` becomes the first `Node`.

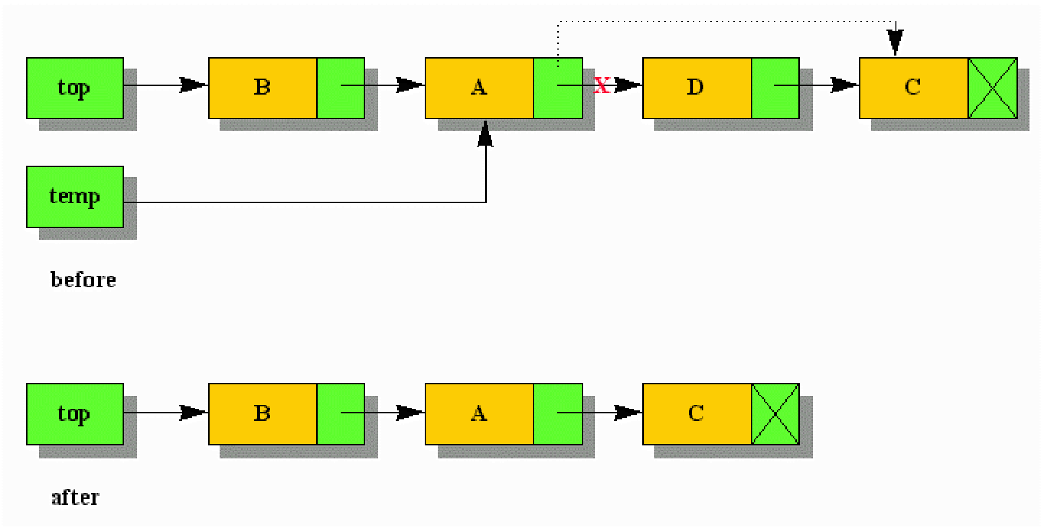


Before and after views of a singly linked list where the first Node is deleted. The red X and dotted lines signify top's change of reference from Node B to Node A.

Delete any node but the first node: Locate the `Node` that precedes the `Node` to be deleted and assign the link in the `Node`-to-be-deleted's `next` link field to the preceding `Node`'s `next` link field. The following pseudo code (which assumes Figure 6's linked list and extends that figure's associated pseudo code) deletes `Node D`:

```
temp = top
WHILE temp.name IS NOT "A"
   temp = temp.next
END WHILE
// We assume that temp references Node A
temp.next = temp.next.next
// Node D no longer exists
```

The following figure presents before and after views of a list where an intermediate `Node` is deleted. In that figure, `Node D` disappears.



Before and after views of a singly linked list where an intermediate Node is deleted. The red X and dotted lines signify Node A's change of link from Node D to Node C.

Appendix B4: Learning situations for linked lists

Phone book

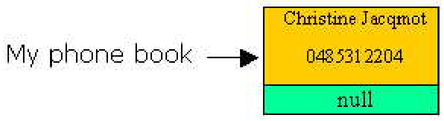
The phone book is a place wherein you store the name and the cell phone of your friends. The basic operations for the phone book include inserting a new entry, deleting an existing entry, and searching for an existing entry. The following table shows an example of a phone book.

Name	Cell phone
Nicolas Devos	0486234334
Mariane Frenay	0475223344
Christine Jacqmot	0485312204
Jean Cara	0475243189
Marc De Vylder	0474222999

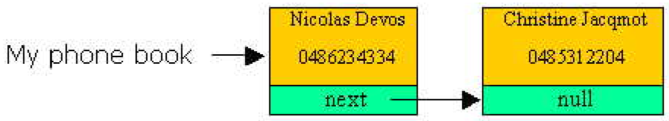
Insertion examples

My phone book is empty, now:

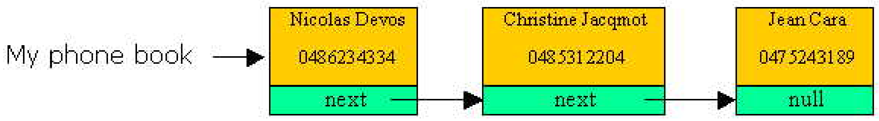
- add « Christine Jacqmot » to my phone book



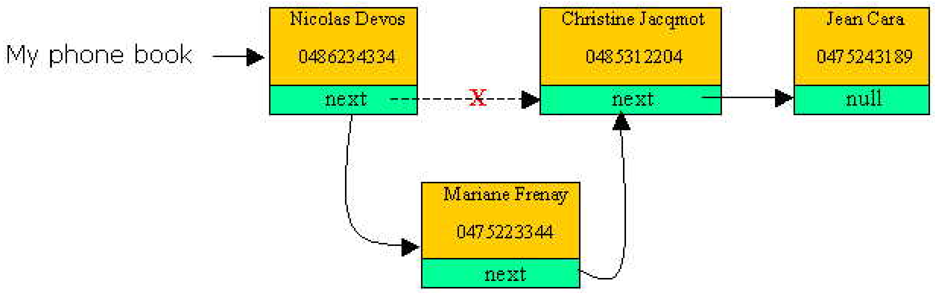
- add « Nicolas Devos » to the beginning of my phone book



- add « Jean Cara » to the end of my phone book



- add « Mariane Frenay » next to « Nicolas Devos »

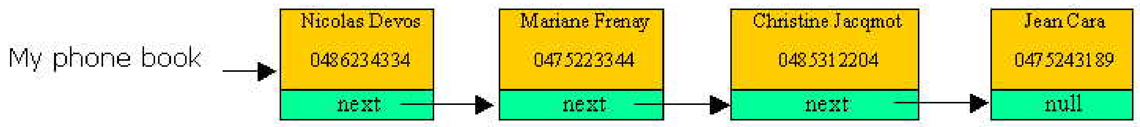


Insertion Java code

```
 * SPECIFICATION
 * This class defines insertion part of phone book
 */
public class PhoneBook{
    private FriendNode list;
    // Constructor
    public PhoneBook(){
        list = null;
    }
    // Methods
   /* PRECOND
    * fr is not null
    * POSTCOND
    * Create a new FriendNode object and add it to the beginning of the linked list
    */
    public void addFirst(Friend fr){
        FriendNode node = new FriendNode(fr);
        node.next = list;
        list = node;
    }
   /* PRECOND
    * fr is not null
    * POSTCOND
    * Create a new FriendNode object and add it to the end of the linked list
    */
    public void addLast(Friend fr){
        FriendNode node = new FriendNode(fr);
        if (list == null)
        list = node;
        else{
            FriendNode current = list;
            while (current.next != null)
            current = current.next;
            current.next = node;
        }
    }
   /* PRECOND
    * fr is not null
    * POSTCOND
    * Create a new FriendNode object and add it to the next of the node that contains a friend object whose "name"
    * field is name. If the node is not found, add to the end of the linked list
    */
    public void addMiddle(Friend fr, String name){
        FriendNode node = new FriendNode(fr);
        if (list == null)
        list = node;
        else{
            FriendNode current = list;
            while ((name.equals(current.friend.getName()) == false) && (current.next != null))
            current = current.next;
            node.next = current.next;
            current.next = node;
        }
    }
    /* SPECIFICATION
     * An inner class that represents a node in the phone book
     */
    private class FriendNode{
        public Friend friend;
        public FriendNode next;
        public FriendNode(Friend friend){
            this.friend = friend;
            next = null;
        }
    }
}
```

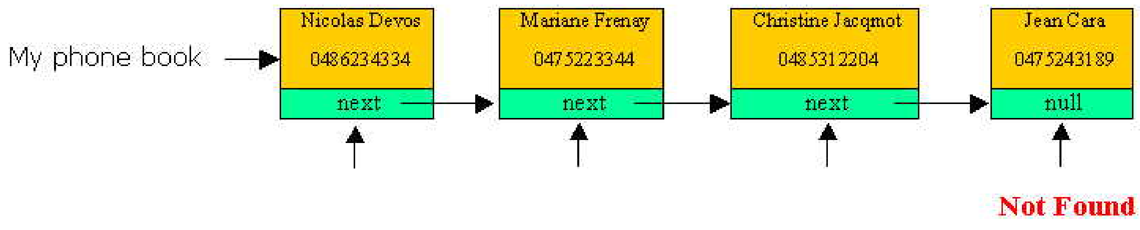
Search examples

My phone book has four entries, as follows:

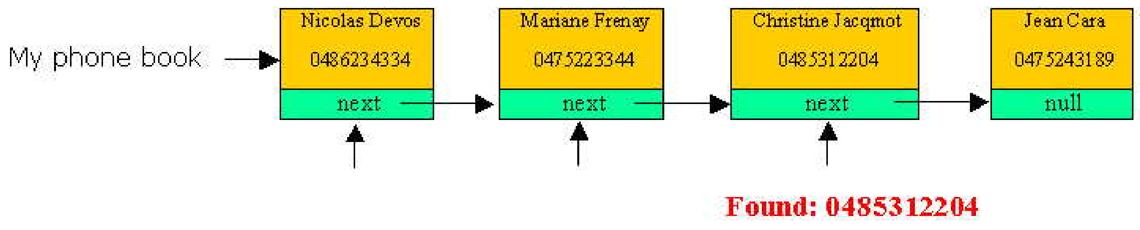


Now:

- search for the cellphone number of « Philippe Jonnaert »



- search for the cellphone number of « Christine Jacqmot »



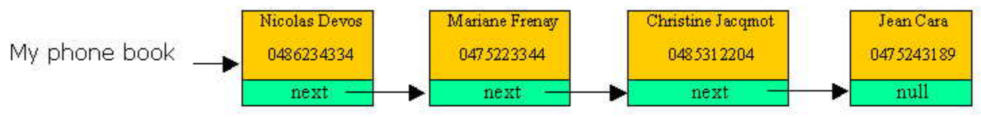
Search Java code

```
/* SPECIFICATION
 * This class defines search part of phone book
 */
public class PhoneBook{
    private FriendNode list;
    // Constructor
    public PhoneBook(){
        list = null;
    }
```

```
  // Methods
 /* PRECOND
  * POSTCOND
  * Return the cellphone number of the friend whose name is name
 */
  public String findCellPhone(String name){
    FriendNode current = list;
    while ((current != null) && (name.equals(current.friend.getName()) == false))
      current = current.next;
    if (current != null)
      return current.friend.getCellPhone();
    else
      return "Not Found";
  }
  /* SPECIFICATION
   * An inner class that represents a node in the phone book
   */
  private class FriendNode{
    public Friend friend;
    public FriendNode next;
    public FriendNode(Friend friend){
      this.friend = friend;
      next = null;
    }
  }
}
```

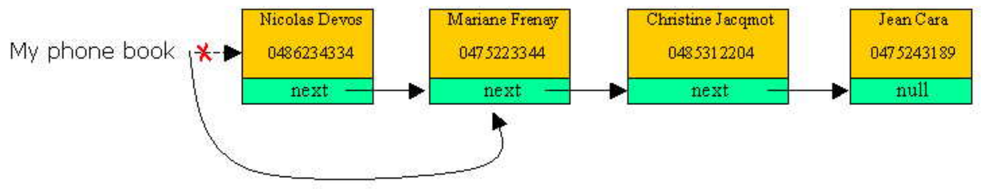
Deletion examples

My phone book has four entries, as follows:

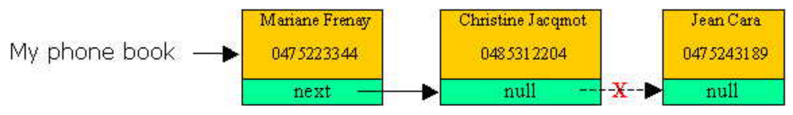


Now:

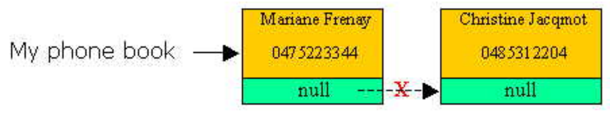
- delete the first entry



- delete the last entry



- delete the entry next to « Mariane Frenay »



Deletion Java code

```
/* SPECIFICATION
 * This class defines deletion part of phone book
 */
public class PhoneBook{
    private FriendNode list;
    // Constructor
    public PhoneBook(){
        list = null;
    }
    // Methods
   /* PRECOND
    * POSTCOND
    * Delete the first node of the linked list
    */
    public void deleteFirst(){
        if (list != null)
        list = list.next;
    }
   /* PRECOND
    * POSTCOND
    * Delete the last node of the linked list
    */
    public void deleteLast(){
    if (list != null) {
        if (list.next == null)
        list = null;
        else{
            FriendNode current = list;
            while (current.next.next != null)
                current = current.next;
            current.next = null;
        }
    }
    }
   /* PRECOND
    * POSTCOND
    * Delete the node following the one that contains a friend object whose "name" field is name
    */
    public void deleteMiddle(String name){
        if (list != null){
            if (name.equals(list.friend.getName()))
                list = list.next;
            else{
                FriendNode current = list;
                while ((current.next != null) && (name.equals(current.next.friend.getName()) == false))
                    current = current.next;
                if (current.next != null)
                current.next = current.next.next;
            }
        }
    }
    /* SPECIFICATION
     * An inner class that represents a node in the phone book
     */
    private class FriendNode{
        public Friend friend;
        public FriendNode next;
        public FriendNode(Friend friend){
            this.friend = friend;
            next = null;
        }
    }
}
```

Phone book Java code

```
/* SPECIFICATION
 * This class defines insertion part of phone book
 */
public class PhoneBook{
    private FriendNode list;
    // Constructor
    public PhoneBook(){
        list = null;
    }
    // Methods
   /* PRECOND
    * fr is not null
    * POSTCOND
    * Create a new FriendNode object and add it to the beginning of the linked list
    */
    public void addFirst(Friend fr){
        FriendNode node = new FriendNode(fr);
        node.next = list;
        list = node;
    }
   /* PRECOND
    * fr is not null
    * POSTCOND
    * Create a new FriendNode object and add it to the end of the linked list
    */
    public void addLast(Friend fr){
        FriendNode node = new FriendNode(fr);
        if (list == null)
        list = node;
        else{
            FriendNode current = list;
            while (current.next != null)
            current = current.next;
            current.next = node;
        }
    }
   /* PRECOND
    * fr is not null
    * POSTCOND
    * Create a new FriendNode object and add it to the next of the node that contains a friend object whose "name"
    * field is name. If the node is not found, add to the end of the linked list
    */
    public void addMiddle(Friend fr, String name){
        FriendNode node = new FriendNode(fr);
        if (list == null)
        list = node;
        else{
            FriendNode current = list;
            while ((name.equals(current.friend.getName()) == false) && (current.next != null))
            current = current.next;
            node.next = current.next;
            current.next = node;
        }
    }
   /* PRECOND
    * POSTCOND
    * Delete the first node of the linked list
    */
    public void deleteFirst(){
        if (list != null)
        list = list.next;
    }
   /* PRECOND
    * POSTCOND
    * Delete the last node of the linked list
    */
    public void deleteLast(){
    if (list != null) {
        if (list.next == null)
        list = null;
        else{
            FriendNode current = list;
            while (current.next.next != null)
                current = current.next;
            current.next = null;
        }
    }
    }
   /* PRECOND
    * POSTCOND
    * Delete the node following the one that contains a friend object whose "name" field is name
    */
    public void deleteMiddle(String name){
        if (list != null){
            if (name.equals(list.friend.getName()))
                list = list.next;
            else{
                FriendNode current = list;
                while ((current.next != null) && (name.equals(current.next.friend.getName()) == false))
                    current = current.next;
                if (current.next != null)
                current.next = current.next.next;
            }
        }
    }
   /* PRECOND
    * POSTCOND
    * Return the cellphone number of the friend whose name is name
    */
    public String findCellPhone(String name){
        FriendNode current = list;
        while ((current != null) && (name.equals(current.friend.getName()) == false))
           current = current.next;
        if (current != null)
           return current.friend.getCellPhone();
        else
           return "Not Found";
    }
   /* PRECOND
    * POSTCOND
    * Return this list of friends as a string
    */
    public String toString(){
        String result = "";
        FriendNode current = list;
        while (current != null){
           result += current.friend + "\n";
           current = current.next;
        }
        return result;
    }
    /* SPECIFICATION
     * An inner class that represents a node in the phone book
     */
    private class FriendNode{
        public Friend friend;
        public FriendNode next;
        public FriendNode(Friend friend){
           this.friend = friend;
           next = null;
        }
    }
}
```

Java test class

```
/* SPECIFICATION
   Driver to test the phone book class */
public class TestPhoneBook{
   public static void main(String[] args){
      PhoneBook myPhoneBook = new PhoneBook();
      myPhoneBook.addFirst(new Friend("Christine Jacqmot", "0485312204");
      myPhoneBook.addFirst(new Friend("Nicolas Devos", "0486234334");
      myPhoneBook.addLast(new Friend("Jean Cara", "0475243189");
      myPhoneBook.addMiddle(new Friend("Mariane Frenay", "0475223344", "Nicolas Devos";
      System.out.println(myPhoneBook);
      System.out.println(myPhoneBook.findCellPhone("Phillipe Jonnaert");
      System.out.println(myPhoneBook.findCellPhone("Christine Jacqmot");
      myPhoneBook.deleteFirst();
      System.out.println(myPhoneBook);
      myPhoneBook.deleteLast();
      System.out.println(myPhoneBook);
      myPhoneBook.deleteMiddle("Mariane Frenay";
      System.out.println(myPhoneBook);
   }
}
```

Appendix B5: Discussion questions

General questions (From Wright, 1995)

What mechanism would explain your hypothesis?

How would you explain that?

Are there other explanations?

Why?

What is the mechanism?

What do you mean?

How do you know that's true?

What does this mean?

What is the evidence?

Have you thought of everything that needs to be considered?

Does everyone in group agree?

Does anyone have a different opinion?

What was your source of information?

What is your hypothesis?

That is a good question. Can anyone answer?

Where might you get the information you need?

How might you get the information you need?

Can you be more specific?

Can you give an example?

How did we do as a group?

Domain-specific questions

Why recursion should be used in this problem?

Why recursion should not be used in this problem?

Can you see recursion in this problem, what is it?

Do you think this recursive method eventually ends?

Can you show me your iterative solution?

What is your own recursive solution? Can you explain it?

How did you go from the problem specification to your recursive solution?

What is wrong with this recursive solution?

Appendix B6: Teaching recursion in the literature

In this appendix, I tried to analyze several authors' teaching of recursion in classroom or in books or in computer-based learning systems. As in any teaching analysis, the following six criteria were considered:

1. *Learning objectives*: What the authors expected from students.
2. *Motivation*: Why the authors taught recursion.
3. *Problems*: Difficulties for teaching recursion.
4. *Pedagogical approach*: How the authors taught recursion.
5. *Learning outcomes*: How students learned recursion.
6. *Comments*: My critiques of the authors' teaching of recursion.

Lewis and Loftus (2003) published the book "Java software solutions", which cover recursion in a separate chapter with a certain number of examples.

- *Learning objectives*. The main objective is to help students use recursion to solve programming problems.
- *Motivation*. Recursion is a powerful programming technique, which provides elegant solutions to certain problems.
- *Problems*. No comment.
- *Pedagogical approach*. Explanation of the basic concepts underlying recursion (e.g. recursive thinking). Exploration of recursion with various programming examples (e.g., traversing a maze, the tower of Hanoi, fractals).
- *Learning outcomes*. No comment.
- *Comments*. The authors should explain how to go from a problem specification to a recursive solution.

Roberts (1986) published the book "Thinking recursively" exclusively devoted to the concept of recursion.

- *Learning objectives*. The authors want to help students think recursively and apply recursive thinking to solve complex programming problems.
- *Motivation*. Recursive algorithms are quite important in computing science; recursion is useful to solve complex problems; recursive solutions are concise and easily understood.
- *Problems*. Recursion is unfamiliar, obscure, difficult, and mystical.
- *Pedagogical approach*. Teaching the principle of recursive thinking: Solve a large problem by reducing it to one or more sub-problems that are identical in structure to the original problem and simpler to solve. Examining recursion from different perspectives: (a) the use of recursion outside the context of programming, (b) the use of recursion in mathematics, (c) the use of recursion to solve complex problems (e.g., sorting, permutations, fractals), (d) the use of recursion in defining data structures, (e) how recursion works in the computer.
- *Learning outcomes*. No comment.
- *Comments*. No comment.

Henderson and Romero (1989) used a ML programming environment (Standard ML) as a tool for teaching recursion in an introductory course on computing science.

- *Learning objectives*. The main objective is to help learners use recursively defined data structures and define recursive functions in ML.
- *Motivation*. Recursion is a central concept in computing science; recursive algorithms often provide elegant solutions to complex problems.

- *Problems*. Recursion is counter-intuitive and very difficult for students to learn.
- *Pedagogical approach*. Discovery learning: Doing programming exercises with Standard ML. Selecting Standard ML because of its recursive and simple-syntax characteristics. Teaching recursion at the beginning of the course.
- *Learning outcomes*. Most learners among 200 first-year students, with no prior programming experience, after three weeks, were able to define fairly powerful recursive functions in ML.
- *Comments*. To follow the course, students should have prior knowledge of recursive mathematical functions and definitions.

Turbak and colleagues (1999) emphasized teaching recursion before loops in their CS1 with the Java programming language.

- *Learning objectives*. The authors want to teach students how to think recursively in problem solving and how to use recursion in programming.
- *Motivation*. Recursion is a central concept in computing science.
- *Problems*. Recursion is difficult to teach because of interference arose from students' knowledge of iterations.
- *Pedagogical approach*. Teaching the DCG (Divide, Conquer, and Glue) strategy explicitly. Teaching recursion before loops. Showing examples that emphasize the nature of recursion rather than traditional examples (i.e., Factorials, Fibonacci numbers).
- *Learning outcomes*. Most students finished the course with a firm understanding of both recursion and loops. It seems that students left out the course with better problem-solving skills than in the previous incarnation of CS1.
- *Comments*. No comment.

Bhuiyan and associates (1994) supported the learning of recursive problem solving with the PETAL learning environment.

- *Learning objectives*. The principal objective is to help learners solve programming problems recursively.
- *Motivation*. No comment.
- *Problems*. Recursion is a difficult concept to teach and learn, especially for novice programmers.
- *Pedagogical approach*. Providing programming tools explicitly to assist learners in the use of three mental methods to solve problems recursively: the syntactic method, the analytic method, and the analysis/synthesis method (see chapter 6 for more details of these methods).
- *Learning outcomes*. Knowledge about recursive problem solving of five students in the PETAL group (students used PETAL as they learned recursion) improved both quantitatively and qualitatively over time. The five PETAL group learners also fared much better than the four learners in the traditional group (students used a standard LISP environment as they learned recursion).
- *Comments*. No comment.

Anderson and colleagues (1988) provided the GRAPES learning environment to help students learn recursion.

- *Learning objectives*. The main goal is to help learners write recursive functions in LISP.
- *Motivation*. No comment.
- *Problems*. Recursion is unfamiliar and complex; students could not determine what has to be done to the result produced by a recursive call in order to get a result for the current function call.
- *Pedagogical approach*. Learning by analogy: the learner solves a new problem by looking at worked-out examples. And learning by knowledge compilation: After each problem-solving session, the learner is asked to produce problem-solving operators (IF-THEN rules) so that he or she can apply them to new problems). Both kinds of learning were supported by GRAPES, which models the recursive programming behavior of an expert and visualizes students' problem-solving processes.
- *Learning outcomes*. Observing the behavior of a student during and after solving three recursive functions in LISP, the authors found that she improved from one function to the next, and that she eventually became quite effective at writing a wide variety of recursive functions.
- *Comments*. No comment.

APPENDIX C: Materials and evidences of the evaluation of COFALE

This appendix is a reference for the materials used to evaluate COFALE and several evidences collected during the survey process. The evaluation of COFALE was presented in chapter 9. In this appendix, I list the pretest, the posttest, the homework, and the interview questions. In the posttest, I also present the scale for grading students' tests (all implementations presented in the posttest were successfully tested) and students' solutions to a posttest question. Note that the original materials were written in French because the experiment was performed with French-native students. Several students had a little problem with understanding what I wanted to say in the posttest and the homework; however, they understood after listening to my explanation.

Because of limited space, one should contact me for having a complete copy of all the data collected in the survey of COFALE.

Pretest

(Recursion course, duration about 15 minutes)

Question 1

What would be the result of the following program? Justify your answer.

```
public class MyString{

  public static void main(String[] args){
    System.out.println (M1("hello"));
  }

  public static String M1(String list){
    if (list.equals("")) return "END";
    else return list.substring(0,1) + list.substring(0,1) + M1(list.substring(1));
  }
}
    // list.substring(0,1) returns the first character of list
    // list.substring(1) returns a substring of list, the substring begins with the character
    // at the index 1 and extends to the end of list
```

Possible solution:

Call `M1("hello")`:	return "hh" + `M1("ello")`.
Call `M1("ello")`:	return "ee" + `M1("llo")`.
Call `M1("llo")`:	return "ll" + `M1("lo")`.
Call `M1("lo")`:	return "ll" + `M1("o")`.
Call `M1("o")`:	return "oo" + `M1("")`.
Call `M1("")`:	return "END".

Print to the Java console: "hheellllooEND".

Question 2

What would be the result of the following program? Justify your answer.

```
public class MyInteger{

  public static void main(String[] args){
    System.out.println (M1(8));
  }

  public static int M1(int n){
    if (n == 1) return 1;
    else return 2 * M1(n/2) + 3;
  }
}
```

Possible solution:

Call `M1(8)`:	return 2 * `M1(4)` + 3.
Call `M1(4)`:	return 2 * `M1(2)` + 3.
Call `M1(2)`:	return 2 * `M1(1)` + 3.
Call `M1(1)`:	return 1.

So `M1(8)` = 2 * (2 * (2 * 1 + 3) + 3) + 3 = 29.

Print to the Java console: 29.

Question 3

Write the attributes of recursive methods below.

Possible solution:

- A recursive method must have one or more base cases, which permit the recursion to eventually end.
- A recursive method has a recursive part in which the method calls itself.

Question 4

Present in several lines your definition of the concept of recursion.

Possible solution:

Recursion is the process of defining something in terms of itself.

Question 5

In your opinion, what does "solving problems recursively" mean? Write your answer in several lines.

Possible solution:

"Solving problems recursively" means that we divide a large problem into one or more subproblems that are identical in structure to the original problem and somewhat simpler to solve.

Posttest

(Recursion course, duration about 60 minutes)

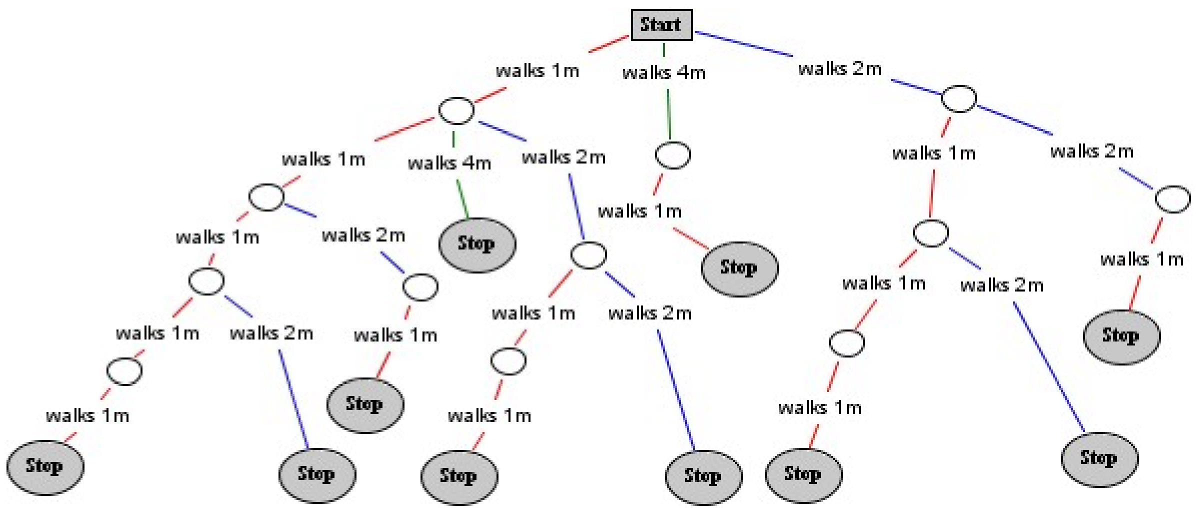
Situation

A robot can take steps of 1 meter, 2 meters, or 4 meters. Figure 1 shows all of the ways the robot can walk 5 meters. Let F_n denote the number of ways the robot can walk *n* meters where *n* is a positive integer. Table 1 presents some values of F_n.

Table 1. Some values of F_n

N	F_n
1	1
2	2
3	3
4	6
5	10
6	18
7	31

Figure 1. A ternary tree representing all of the ways the robot can walk 5 meters



Test 1 (4 points)

Answer the following questions. Justify each answer.

How many ways can the robot walk 8 meters if its first step is 1 meter?

The answer is: $F_7 = 31$. (1 point)

How many ways can the robot walk 8 meters if its first step is 2 meters?

The answer is: $F_6 = 18$. (1 point)

How many ways can the robot walk 8 meters if its first step is 4 meters?

The answer is: $F_4 = 6$. (1 point)

How many ways can the robot walk 8 meters?

The answer is: $F_7 + F_6 + F_4 = 55$. (1 point)

Test 2 (4 points)

Complete the following recursive method.

```
/* PRECOND
 * distance is a positive integer
 * POSTCOND
 * Return the number of ways the robot can walk distance meters
 */
public int numberOfWays(int distance) {
```

Two points for the correct base cases, 1 point for the incomplete base cases (e.g. absence of one or more base cases), and 0 point for the incorrect base cases.
Two points for the correct recursive part, 1 point for the incomplete recursive part (e.g. make correctly three recursive calls but not sum up the three return values), and 0 point for the incorrect recursive part (e.g. absence of one or more recursive calls).

Possible solution:

```
// BASE CASES                           (2 points)
 if (distance == 1) return 1;
 else if (distance == 2) return 2;
 else if (distance == 3) return 3;
 else if (distance == 4) return 6;
// RECURSIVE PART                       (2 points)
 else
 return numberOfWays(distance-1)+numberOfWays(distance-2)+numberOfWays(distance-4);





}
```

Test 3 (4 points)

Let ListOfSteps denote a simply linked list composed of positive integers, which represents a way the robot walks *n* meters where *n* is a positive integer:

```
/* SPECIFICATION
 * Definition of the class ListOfSteps: a simply linked list composed of positive
 * integers
 */
public class ListOfSteps {
  // Instance variables
  private int step; // value (meters)
  private ListOfSteps next; // link to the next element

  // Constructor
  public ListOfSteps(int step, ListOfSteps next) {
    this.step = step;
    this.next = next;
  }

  // Methods
/* PRECOND
 * POSTCOND
 * Return the value
 */
  public int getStep() {
    return step;
  }

/* PRECOND
 * POSTCOND
 * Return the link to the next element
 */
  public ListOfSteps getNext() {
    return next;
  }
}
```

A ListOfSteps is zigzag if there are no two equal consecutive steps, for example (1, 2, 2) is not zigzag, but (2, 1, 2) is. Complete the following recursive method.

```
/* PRECOND
 * POSTCOND
 * Return "true" if list is zigzag, otherwise return "false"
 */
public boolean isZigzag(ListOfSteps list) {
```

Three points for the correct base cases, 2 points if not check whether the first and second elements of the list are equal, 1 point if not check whether the list has only one element, and 0 point for the incorrect base cases.
One point for the correct recursive part and 0 point for the incorrect recursive part.

Possible solution:

```
// BASE CASES
 if (list == null) return true;                                   (1 point)
 else if (list.getNext() == null) return true;                    (1 point)
 else if (list.getStep() == list.getNext().getStep()) return false;  (1 point)
// RECURSIVE PART
 else return isZigzag(list.getNext());                            (1 point)
}
```

Test 4 (5 points)

Let TernaryTree denote a ternary tree that represents all the ways the robot can walk *n* meters where *n* is a positive integer:

```
/* SPECIFICATION
 * Definition of the class TernaryTree: a ternary tree
 */
public class TernaryTree {
  // Instance variables
  private TernaryTree stepOfOne;
  private TernaryTree stepOfTwo;
  private TernaryTree stepOfFour;

  // Constructors
  public TernaryTree() {
    stepOfOne = null;
    stepOfTwo = null;
    stepOfFour = null;
  }

  public TernaryTree(TernaryTree one, TernaryTree two, TernaryTree four) {
    stepOfOne = one;
    stepOfTwo = two;
    stepOfFour = four;
  }

  // Methods

}
```

Complete the following method.

```
/* PRECOND
 * distance is a positive integer
 * POSTCOND
 * Return a TernaryTree that represents all the ways the robot can walk distance
 * meters
 */
public TernaryTree walk(int distance) {
```

One point for the correct base case and 0 point for the incorrect base case.
Four points for the correct recursive part, minus 1 point for each absence of the construction of `one` or `two` or `four` or for the absence of the return statement.

<u>Possible solution</u>:

```
// BASE CASE
 if (distance == 0) return new TernaryTree ();                (1 point)
// RECURSIVE PART
 TernaryTree one, two, four;
 one = walk (distance - 1);                                   (1 point)
 if (distance >= 2) two = walk (distance - 2);                (1 point)
 else two = null;
 if (distance >= 4) four = walk (distance - 4);               (1 point)
 else four = null;
 return new TernaryTree (one, two, four);                     (1 point)



}
```

Test 5 (4 points)

Question 1: Write the attributes of recursive methods below.

Possible solution:

- *A recursive method must have one or more base cases, which permit the recursion to eventually end.* (1 point)
- *A recursive method has a recursive part in which the method calls itself.* (1 point)

Question 2: Present in several lines your definition of the concept of recursion.

Possible solution:

Recursion is the process of defining something in terms of itself. (1 point)

Question 3: In your opinion, what does "solving problems recursively" mean? Write your answer in several lines.

Possible solution:

"Solving problems recursively" means that we divide a large problem into one or more sub-problems that are identical in structure to the original problem and somewhat simpler to solve. (1 point)

Students' solutions to Test 5

Learners	Question 1	Question 2	Question 3
C1	They decompose the resolution of a complex problem into different small problems identical easier to solve. Although they require a higher degree of abstraction, they are generally easier to understand. They avoid using a list of complicated methods, and variables that would solve the same problem with the use of more memory and time.	Recursion is a technique to split a complex problem into many small problems identical but easier to solve.	Solve a problem small piece to small piece, each one is identical to another.
C2	It is a method that calls itself. It consists of a base case, which governs the number of calls that will be made.	It is back-tracking. We make many times a call to something known in order to advance to something unknown.	It is to divide a complex problem into a number of problems simple and of the same type. To solve the global problem, it is sufficient to solve the small problems, one to another.

C3	One or more base cases. Make a call to the method from the method itself.	Recursion allows us to solve without too many lines of code a problem that possesses one or more base cases and that must make a way.	Keep a base case and by taking into account one or more base cases. Make all of operations that arrive at those cases.
C4	A recursive method possesses a base case, which indicates where the method must terminate and a recursive part, which is the body of the method and in which the method calls itself with other parameters until the base case is reached.	Recursion is a concept that aims at executing a same method a number of times with other parameters until reaching the indicated limit.	Begin with a given problem, divide this problem into similar sub-problems, divide these problems … until reaching a list of problems simple and easy to solve (these are the base cases).
T1	They are methods that use themselves, until reaching something that is named a base case. The base case(s) are the cases where we arrive at the end of an iteration, where the treated problem is much too simple (so call base case).	Recursion is another way to treat problems of unknown size than iteration. The principle is a little different and recursion is more advantageous in certain cases. It is sufficient to find out all of the base cases (difficult) and treat them (easy). We also reduce the difficulty of the problem by transforming it into small problems solved fast.	We need to find base cases, treat them, it means give the solution of these small problems, of these particular cases, and ask the method to reduce, and reduce, and reduce the problem until we arrive at base cases, which return a response, and the method can then come up again to the (small) previous problem, and so on.
T2	They call themselves. They need a condition of exit. Fast to travel through a tree or to make fractals, but less interest for the rest.	Function including in its code one or more calls to itself. This permit to test all of the possibilities of a situation, for example.	It is to solve the problems by making call to a recursive function.
T3	They call themselves. They possess one or more cases of exit. They are relatively short and elegant.	Recursion is a concept that allows us to simplify the resolution of a complex problem by dividing it into small problems identical and easy to solve.	DCG: Divide the problem into small problems easy to solve and identical. Find the solution of these problems. Assemble the solutions to form the final solution.
T4	It employs itself. It possesses a condition of exit.	Recursion allows us to treat problems that are difficult to solve in an iterative manner. Recursion allows us to use the information previous, or, following in a same method.	Divide the solution of a problem into many small step to facilitate the resolution.
T5	Consist of a base case and a recursive part (position where they call themselves).	We find a base solution and we divide a problem into a list of base problems.	Divide the problems into pieces easy, then we solve the base case and we divide the problem into pieces equal to base case. Then we assemble the solutions to attain the one of the global problem.

Test 6 (4 points)

Let TernaryTree denote a ternary tree that represents all the ways the robot can walk *n* meters where *n* is a positive integer. Complete a method **traverse**:

```
/* SPECIFICATION
 * Definition of the class TernaryTree: a ternary tree
 */
public class TernaryTree {
  // Instance variables
  private TernaryTree stepOfOne;
  private TernaryTree stepOfTwo;
  private TernaryTree stepOfFour;

  // Constructors
  public TernaryTree() {
    stepOfOne = null;
    stepOfTwo = null;
    stepOfFour = null;
  }

  public TernaryTree(TernaryTree one, TernaryTree two, TernaryTree four) {
    stepOfOne = one;
    stepOfTwo = two;
    stepOfFour = four;
  }

 // Methods
 /* PRECOND
  * POSTCOND
  * Print to the Java console line by line all the ways of the robot represented by
  * this ternary tree, for example:
  * 11111
  * 1112
  * ...
  * 41
  */
  public void print(){
    traverse("");
  }
  // This recursive method helps the print process of the method "print"
  private void traverse(String path) {
```

One point for the correct base case and 0 point for the incorrect base case.
Three points for the correct recursive part, minus 1 point for each absence of the three recursive calls.

<u>Possible solution</u>:

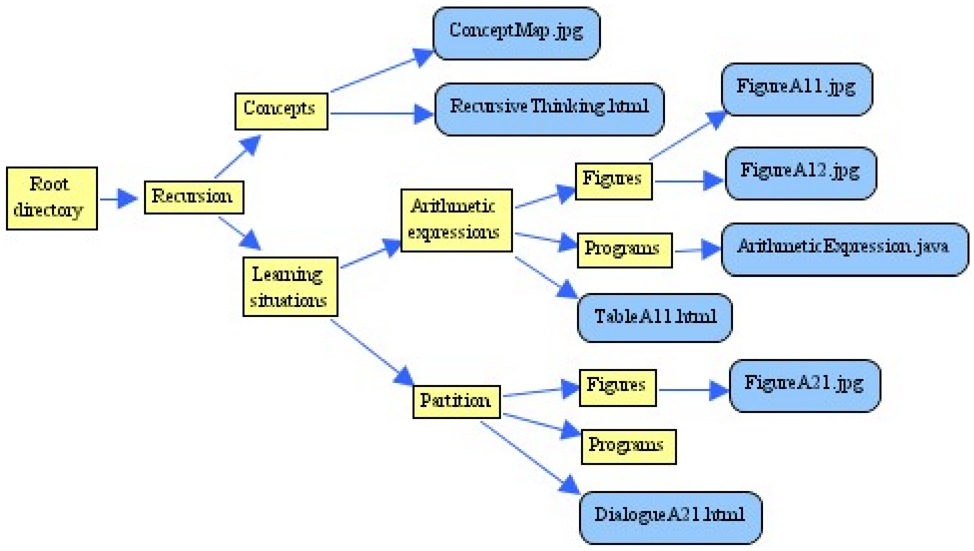
```
// BASE CASE
  if ((stepOfOne == null) && (stepOfTwo == null) && (stepOfFour == null))
    System.out.println (path);                                          (1 point)
// RECURSIVE PART
  else {
    if (stepOfOne != null) stepOfOne.traverse (path + "1");             (1 point)
    if (stepOfTwo != null) stepOfTwo.traverse (path + "2");             (1 point)
    if (stepOfFour != null) stepOfFour.traverse (path + "4");           (1 point)
  }

  }
}
```

Homework: A case study for recursion

(Recursion course, duration about 2 hours)

File management: A key concept supported by virtually all operating systems such as Windows is the **file system**. To provide a place to keep files, operating systems have the concept of a **directory** as a way of grouping files together. A directory can contain nothing, or one or more **directory entries**. Directory entries may be either files or other directories. The file system may be organized as a hierarchy (Figure 1). We can specify every file within the directory hierarchy by giving its **path name** from the top of the directory hierarchy, the **root directory**. For the file management, Java supports the class File. Table 1 describes a part of this class (for more details, see: http://java.sun.com/j2se/1.4.2/docs/api/).

Figure 1. A directory hierarchy



The round rectangles are **files** and the other ones are **directories**.

Table 1. A part of the specification of the class File

Constructor Summary

`File (String pathname)`

Creates a new `File` instance by converting the given pathname string into an abstract pathname.

Method Summary

boolean	exists()	Tests whether the file or directory denoted by this abstract pathname exists.
boolean	isDirectory()	Tests whether the file denoted by this abstract pathname is a directory.
boolean	isFile()	Tests whether the file denoted by this abstract pathname is a normal file.
long	length()	Returns the size, in bytes, of the file denoted by this abstract pathname.
String	getName()	Returns the name of the file or directory denoted by this abstract pathname.
String	getPath()	Converts this abstract pathname into a pathname string.
File[]	listFiles()	Returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname.

We ask you to:

- complete a class FileManager
- write a class MyTest for testing the methods of FileManager
- justify why recursion is very useful for solving this problem.

```
import java.io.*;
import java.io.File;

/* SPECIFICATION
 * Definition of the class FileManager that provides a certain number of methods
 * for the management of files
 */
public class FileManager{

  // Methods

  /* PRECOND
   * POSTCOND
   * Print to the Java console line by line the path name of all the files and sub
   * directories in the directory "dirName" (recursively)
   */
  public static void dirFunction(String dirName){
    // To be completed
  }

  /* PRECOND
   * POSTCOND
   * Return the total size in bytes of all the files in the directory "dirName"
   * (recursively)
   */
  public static long sizeFunction(String dirName){
    // To be completed
  }
}
```

Interview questions

(Recursion course, duration about 15 minutes)

Introduction

During the past several years, we have been encountering with students in FSA a certain number of difficulties about mastering the concept of recursion. This is why we seek for better understating the type of difficulties that students have and how we can help them with the pedagogical devices we set up to overcome these difficulties. To help us in this work, I would like to ask you some questions. They are open enough and do not hesitate to tell us any other element that seems to be important to you. It goes without saying there are no good or bad answers, because it is your specific opinion that interests us. We also asked a certain number of students to answer these questions, so as to have the most complete possible view of what students think.

Question 1

First of all, could you tell me how you would define the concept of recursion today?

Question 2

Undoubtedly, do you remember how you considered it before the course, according to you, what has been changed on the way in which you understand this concept today?

Question 3

Did you have particular difficulties when you were brought to work on this concept? Can you tell me your difficulties?

Question 4

Now, if we look at the way you used to build your own recursive solution to the problems presented in this posttest, can you explain how you constructed it?

Question 5

Finally, between the course that I gave you one week ago and this posttest, you worked alone with the help of a chapter (or of COFALE). Can you tell me how you used this chapter (or this tool)?

Question 6

Did you have particular difficulties? Can you explain your difficulties to me?

Question 7

Do you have suggestions to improve this tool?

Question 8

Were two weeks sufficient for this course?

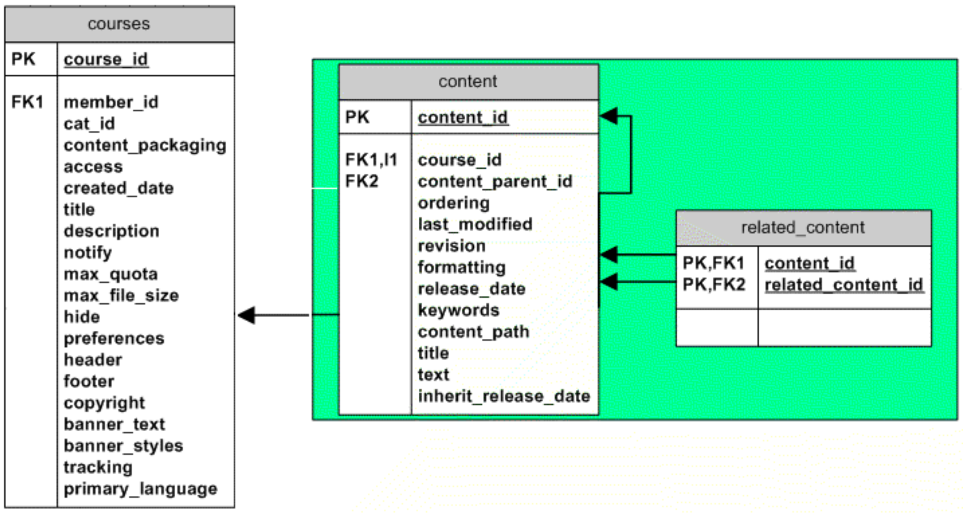
APPENDIX D: Development of COFALE

Implementation of ATutor

Here are two examples of how ATutor organizes information. Look at the ATutor developer documentation for more details.

Example 1. Figure D.1 shows how courses and content objects, information blocks, and learning objects are represented in ATutor, taking into account the standard suggested by IMS/SCORM (Advanced Distributed Learning, 2004).

Figure D.1. Representation of courses and content objects in ATutor (adapted from the ATutor developer documentation, PK = primary key, FK = foreign key, I = index)



The attributes of a course consist of:

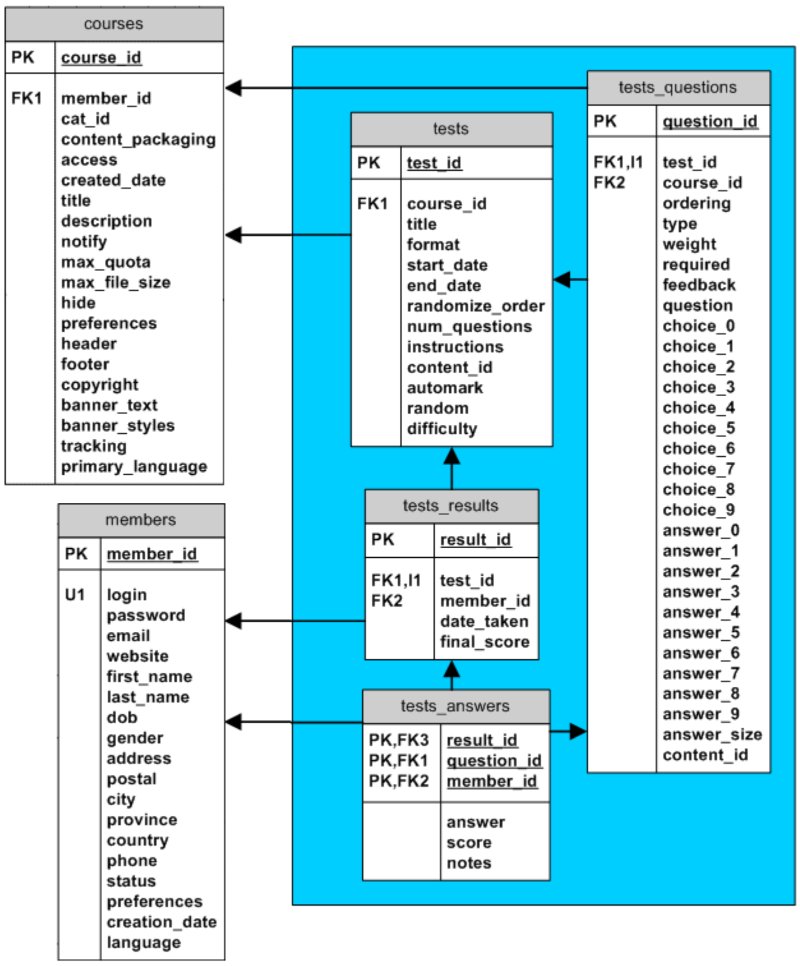
- `member_id`: The identity of the person who creates the course.
- `cat_id`: The identity of the category to which the course belongs.
- `title`: The title of the course.
- `description`: The short description of the course.
- `primary_language`: The first language used in the course. This is a new attribute added in versions greater than 1.4.
- And so on.

The attributes of a content object include:

- `course_id`: The identity of the course to which the content object belongs.
- `content_parent_id`: The identity of the parent content object of the current one (this attribute is equal to zero if the current content object has no parent). For example, the parent content object of "Java test class" is "Arithmetic expressions" (see Figures 6.1 and 7.3).
- `ordering`: The order of the content object within the parent content object (1 = the first child, 2 = the second child, etc.). For instance the order of "Java test class" in "Arithmetic expressions" is equal to five (see Figures 6.1 and 7.3).
- `title`: The title of the content object.
- `text`: The text or the HTML code describing the content object.

Table `related_content` describes the relationships among content objects (see also Figure 7.3). The data in this table are used to construct the menu "Related Topics" (see also Figure 6.1).

Figure D.2. Representation of courses, members, and tests in ATutor (adapted from the ATutor developer documentation, PK = primary key, FK = foreign key, U = unique, I = index)



Example 2. Figure D.2 shows how courses, users, and tests are represented in ATutor.

Table `members` represents the data of all kinds of users: learners, teachers, course designers, and administrators. Here are its attributes:

- `login`: The login of the user to the ATutor system. It must be unique.
- `password`: The password the user uses to log into ATutor.
- `status`: A non-negative integer indicating whether the user is a learner or a teacher or an administrator (ATutor considers the course designer as the teacher).
- And the user's personal information such as name, age, gender, address.

Each course may have zero or one or more tests. The attributes of a test include:

- `course_id`: The identity of the course to which the test belongs.
- `title`: The title of the test.
- `format`: The format (text or HTML) of the test.

`num_questions`: The number of questions of the test.

Each test may have one or more questions. The attributes of a question are:

- `test_id`: The identity of the test to which the question belongs.
- `type`: The type of the question (multiple-choice or true-false or open-ended).

- `weight`: The maximal score of the question.
- `feedback`: The feedback provided for the learner when he or she demands.
- `question`: The question in the form of text or HTML.
- `choice_0, …, choice_9`: The attributes for multiple-choice questions.
- `answer_0, …, answer_9`: The attributes for multiple-choice questions.
- And so forth.

Each result of a learner for a test is represented in table `tests_results`. Here are its attributes:

- `member_id`: The identity of the learner.
- `test_id`: The identity of the test.
- `date_taken`: The date the learner takes the test.
- `final_score`: The test final score marked by the teacher for the learner.

Each answer of a learner for a question of a test is represented in table `tests_answers`. Here are its attributes:

- `member_id`: The identity of the learner.
- `result_id`: The identity of the test result of the learner.
- `question_id`: The identity of the question.
- `answer`: The answer of the learner.
- `score`: The question score marked by the teacher for the learner.
- `notes`: The comments proposed by the teacher for the learner.

Implementation of COFALE

I illustrate here how to implement adaptability in ATutor. Sometimes, I do not explain why I used certain techniques because I assume that the reader has good programming knowledge when he or she wants to read this section.

Implementation of the learner model manager

Creating and editing components of learner models. I created an authoring tool (see section 7.3.1) allowing the course designer to create and edit components of learner models, as follows.

For the database. I created table `mental_models` to represent components of learner models. The structure of this table is shortly described next.

mental_models	
PK	mental_model_id
	model_name
	model_description
FK1	member_id
FK2	course_id
	model_default

`course_id`: The identity of the course the designer is editing.
`member_id`: The identity of the course designer.
`model_name`: The title of the component.
`model_description`: The description of the component.
`model_default`: A true/false value indicating whether or not the component is set as default for a new learner.

For the source code. Here is the process:

1. I logged into the recursion course designed in COFALE as a course designer.
2. I selected the menu "Tools" to know the PHP file needed to modify to add the desired authoring tool: `tools/index.php` seen on the address bar of the browser (Figure D.3).

3. I examined this file to understand how instructor tools had been implemented.
4. I modified this file to display the desired instructor tool (see Figure 7.1: Learner Model Manager). This tool is linked to a file `teachers/learner_model.php` created by me. This file leads the course designer to a set of tools (Figure 7.11) for managing components of learner models.
5. I created a file `teachers/edit_learner_model.php` to enable the course designer to add new components or edit existing components. For instance, when the course designer clicks on "Add New Component" (Figure 7.11) linked to this file, he or she is led to a tool (Figure 7.12) for adding a new component to the database (table `mental_models`).

Updating the learner model for a particular student. In section 7.3.1, I mentioned three evaluations for a particular student's learner model: self-evaluation, the teacher's evaluation, and the system's evaluation. Next, I explain the implementation for the last two evaluations.

For the database. Each student may possess one or more components of learner models. So, I created table `learner_mental_model` (notice that the student's favorite kind of evaluation is stored in table `course_enrollment`, see more information of this table in the ATutor developer documentation). The system can automatically detect several components of learner models, on the basis of students' test results (see section 7.3.1). Thus, I constructed table `mental_model_test` to represent means for diagnosing certain components of learner models. The brief description of the two tables is following:

learner_mental_model	
PK	learner_mental_model_id
FK1	member_id
FK2	mental_model_id
	evaluator
FK3	course_id

`course_id`: The identity of the course the designer is editing.
`member_id`: The identity of the learner.
`mental_model_id`: The identity of the component.
`evaluator`: The kind of evaluation.

mental_model_test	
PK	mental_model_test_id
FK1	mental_model_id
	means
	execution
FK2	member_id
FK3	course_id

`course_id`: The identity of the course the designer is editing.
`member_id`: The identity of the course designer.
`mental_model_id`: The identity of the component.
`means`: The means to detect the component (nothing or test).
`execution`: The expression the system can use to detect the component (if the means is "test").

For the source code. Here is the process:

1. I created a file `teachers/edit_learners_own_models.php`. When the teacher clicks on "Edit Learners' Own Models" (Figure 7.11) linked to this file, he or she is led to a set of tools for updating students' learner model to the database (table `learner_mental_model`).
2. In the file `teachers/edit_learner_model.php` mentioned earlier, I wrote a segment of code providing the course designer with a tool (see Figure 7.14) for introducing the expressions to detect certain components of learner models. These expressions are stored in table `mental_model_test`.
3. To let the system detect and update the learner model of a particular student, I first searched ATutor's PHP file system for a segment of code where students' test results are updated (I found it in `tools/tests/view_results.php`). Then, I modified this segment of code. The detecting algorithm can be informally explained, as follows:

Input:

- A set of tests, for instance, T1, T2, T3, T4, …
- A set of test results of a student, for example, T1 (passed), T2 (passed), T3 (fail), T4 (passed), …
- A component of learner models, for instance the loop model on recursion.
- A logic expression to detect the component, for example "T1 AND NOT T2 AND NOT T3 AND NOT T4" (the student possesses the component if he or she passes test T1 but not tests T2, T3, and T4).

Output:

- Return a true/false value indicating whether or not the student possesses the component.

Algorithm:

1. Replace the variables in the logic expression by their values: 1 if the student passes the test, 0 otherwise. For example: "T1 AND NOT T2 AND NOT T3 AND NOT T4" => "1 AND NOT 1 AND NOT 0 AND NOT 1".
2. Make a SQL select command with the new expression. For instance: "**SELECT** 1 AND NOT 1 AND NOT 0 AND NOT 1 **AS** possessed".
3. Use the MySQL server to run this command, the value of variable `possessed` indicates the result: 1 means the student possesses the component, 0 means the student does not possess the component.

Note: Because this algorithm relies on the one implemented in the MySQL server, I do not show here its proof.

Implementation of adaptability

Adaptive presentation of learning contents. See sections 6.3.2 and 7.3.2 to understand how the course designer makes this feature available for the student.

For the database. To represent associations between components of learner models and content objects, I created table `mental_model_content`, as follows:

mental_model_content	
PK	mental_model_content_id
FK1	mental_model_id
FK2	content_id
FK3	member_id
FK4	course_id

`course_id`: The identity of the course the designer is editing.
`member_id`: The identity of the course designer.
`mental_model_id`: The identity of the component.
`content_id`: The identity of the content object defined by the course designer to be appropriate to students possessing the component.

For the source code.

1. In the file `teachers/edit_learner_model.php` mentioned previously, I wrote a code segment providing the course designer with a tool (see Figure 7.15) so that he or she can define appropriate content objects for the component of learner models being editing. These associations are stored in table `mental_model_content`.
2. I searched ATutor's PHP file system for the segment of code that manages the set of content objects presented for the student: The function `initContent` in the file `include/classes/ContentManager.class.php` was found.
3. I modified this function in such a way that the learning content is adapted to the current learner model of the student: Table `mental_model_content` was used to retrieve the appropriate content objects for the student, according to his or her current learner model.

Adaptive use of pedagogical devices. To know how the course designer makes this characteristic available for the student, the reader should refer back to sections 6.3.2 and 7.3.2.

For the database. To represent associations between learning activities and learner models and content objects, here are the two tables `learning_activities` and `learning_activity_content` I created. For the former, I filled a set of learning activities in advance (see also Figure 6.3).

learning_activities	
PK	learning_activity_id
	activity_name
	activity_description
	activity_action
FK1	member_id
FK2	course_id

`course_id`: The identity of the course the designer is editing.
`member_id`: The identity of the course designer.
`activity_name`: The title of the learning activity.
`activity_description`: The description of the learning activity.
`activity_action`: Null or the hyperlink leading the student to the tool(s) for performing the learning activity.

learning_activity_content	
PK	learning_activity_content_id
FK1	learning_activity_id
FK2	content_id
FK3	mental_model_id
FK4	member_id
FK5	course_id

`course_id`: The identity of the course the designer is editing.
`member_id`: The identity of the course designer.
`learning_activity_id`: The identity of the learning activity.
`mental_model_id`: The identity of the component.
`content_id`: The identity of the content object at the end of which the activity is presented for the student possessing the component.

For the source code.

1. The way to add a learning activity manager (Figure 7.1: Learning Activities and Learning Content) to ATutor is more or less similar to the one to add a learner model manager presented earlier. The file I created is `teachers/learning_activity.php`.
2. The way to present the course designer with a tool (Figure 7.10) for defining associations between learning activities and learner models and learning contents is also identical to the one to provide the course designer with a tool (Figure 7.15) for defining associations between learning contents and learner models. The file I constructed is `teachers/edit_learning_activity.php`.
3. I searched for the segment of code that manages the presentation of a particular content object: It was found in `index.php`. Then, I inserted into this file a segment of code that retrieves the appropriate learning activities from the two tables `learning_activities` and `learning_activity_content` and presents them for the student at the end of the content object being considered.

Adaptive communication support. This kind of adaptation support was showed in sections 6.3.2 and 7.3.2.

For the database. To represent help relations among components of learner models, I created the following table:

mental_model_help_relation	
PK	mental_model_help_relation__id
FK1	model_left_id
FK2	model_right_id
FK3	member_id
FK4	course_id

`course_id`: The identity of the course the designer is editing.
`member_id`: The identity of the course designer.
Students possessing the component identified by `model_left_id` can help students possessing the component identified by `model_right_id`.

For the source code.

1. In the set of PHP files proposed for the learner model manager, I created `teachers/edit_model_constraints.php` producing the tool (Figure 7.16) for the course designer to define help relations among components of learner models.
2. In the set of PHP files related to "Peers' Learning Hyperspace" (see Example 2 in the previous section), I constructed `search_peers.php` that produces a list of appropriate peers (Figure 6.12), arranged from the highest "level of appropriateness" to the lowest one, for the student when he or she makes a demand. The algorithm calculating the "level of appropriateness" can be informally explained, as follows:

Input:

- Let C be the set of components of learner models: C_1, C_2, … C_N. For instance, N = 6 in the course on recursion.
- Let T be the two-dimension table $T_{N, N}$ defining help relations among components of learner models: $T_{i, j} = 1$ if students possessing C_i can help students possessing C_j, otherwise $T_{i, j} = 0$ ($1 \le i, j \le N$, $i \neq j$). For instance the table presented in Figure 7.16.
- Let CS be the set of components of learner models possessed by the student: C_{s1}, C_{s2}, … C_{sa} (CS is a subset of C, $1 \le s1 < s2 < … < sa \le N$). For example, Bob with the loop model on recursion and the "novice" model on the use of COFALE.
- Let CP be the set of components of learner models possessed by the peer: C_{p1}, C_{p2}, … C_{pb} (CP is also a subset of C, $1 \le p1 < p2 < … < pb \le N$). For example, Alice with the analysis-synthesis model on recursion and the "expert" model on the use of COFALE.

Output:

- Return a non-negative integer indicating the "level of appropriateness" about the fact that the peer can help the student. This number is defined to be the sum of $T_{pi, sj}$ ($1 \le i \le b$ and $1 \le j \le a$). I assume that the higher this number is, the higher the probability of the fact the peer can help the student is.

Algorithm:

1. Set *level_appropriateness* = 0.
2. For each i in (p1, p2, … pb) and for each j in (s1, s2, … sa), sum up *level_appropriateness* with $T_{i, j}$.
3. Return *level_appropriateness*. For instance, return the value 2 in the case of Bob and Alice.

Note: Because this algorithm is quite simple, I do not show here its proof.

Glossary

The document "Making sense of learning specifications & standards" created and maintained by Masie Center (2003) provided many resources in which a great number of terms related to the field of learning technology. Santrock (2001) also provided many definitions of terms related to learning and instruction.

Accommodation. In Piaget's theory, the process in which individuals adjust existing cognitive structures to account for new information.

Adaptability. (Also adaptation) The ability of a learning system to provide a learning experience that is continuously tailored to the needs of the individual learner.

Adaptive assessment. A technique of providing a specific learner with appropriate assessment problems and methods at any given time.

Adaptive communication support. A technique of identifying appropriate peers who could help a specific learner.

Adaptive learning system. A learning system that can adapt the learning materials to different kinds of students.

Adaptive presentation of learning contents. A technique of providing a specific learner with appropriate learning contents at any given time.

Adaptive problem-solving support. A technique of providing appropriate feedback during the problem-solving process of a specific learner.

Adaptive use of pedagogical devices. A technique of providing a specific learner with appropriate learning activities at any given time.

Assessment. (A component of constructivist learning environments) Problems, methods, and tools for determining whether learners have achieved the learning objectives.

Asset. The learning content in its most basic form such as electronic media, text, images, and sound.

Assimilation. In Piaget's theory, the process in which individuals incorporate new knowledge into existing cognitive structures.

Authentic assessment. Evaluating a student's knowledge or skill in a context that approximates the real world life as closely as possible (Santrock, 2001, p. 513).

Cognitive constructivist approach. (Also cognitive constructivism) An educational approach that emphasizes that individuals construct knowledge by transforming, organizing, and reorganizing previous knowledge and information.

Cognitive flexibility. The ability to spontaneously restructure one's knowledge, in many ways, in adaptive response to radically changing situational demands (Spiro & Jehng, 1990, p. 165).

Cognitive structure. (Also schema) A concept or framework that exists in an individual's mind to organize and interpret information (Santrock, 2001, p. 49).

Computer-based instruction. (Also computer-assisted instruction) Instruction that is provided by a computer.

Constructivism. A learning theory that emphasizes that individuals learn best when they actively construct their own knowledge and understanding. Constructivism has two main paradigms: a cognitive constructivist approach and social constructivist approaches.

Constructivist learning environment. A place where learners may use a variety of information resources, pedagogical and assessment devices, and interact with the tutor and peers through communication means in their guided pursuit of learning objectives.

Declarative knowledge. The conscious recollection of information, such as specific facts or events that can be verbally communicated (Santrock, 2001, p. 282).

Distance education. Teaching and learning in which learning normally occurs in a different place from teaching.

E-Learning. Learning or training that is prepared, delivered, or managed using a variety of learning technologies, and that can be deployed either locally or globally. Term covering a wide set of applications and processes, such as Web-based learning, computer-based learning, and digital collaboration. It includes the delivery of content via Internet, intranet, extranet, virtual private network, audiotape, videotape, satellite broadcast, virtual classroom, interactive television, CD-ROM, DVD, PDA, and other delivery platforms (Masie Center, 2003).

Human interactions. (A component of constructivist learning environments) Means and techniques for engaging tutors and learners in exchanges.

ICT-based learning environment. Learning or training that is prepared, delivered, or managed using ICT (see also e-Learning).

Information and communication technology. (ICT) The study of the technology used to handle information and aid communication.

Information block. A set of sharable content objects organized to present concepts, learning situations, and so on.

Instructional design activity. One or more operations the teacher should perform in order to create or evaluate certain learning conditions for students.

Instructional design process. A set of instructional design activities.

Interactive phase. (In instructional design) The process in which the teacher specifies what should be done during the learning session.

Knowledge. (In a Piagetian point of view) Cognitive structures an individual constructs about the new information on the basis of his or her own experiences and the interaction with the environment surrounding him or her.

Learning. (In a Piagetian point of view), The process in which individuals construct and transform cognitive structures.

Learning content management system. (LCMS) A multi-user software application that enables content authors to manage the life-cycle of learning content by allowing them to create, register, store, assemble, re-use, and publish digital learning content for delivery via Web, print, CD, etc., within a central object repository (Masie Center, 2003).

Learning contents. (A component of constructivist learning environments) Sources of information provided for learners for exploring their learning objectives.

Learning materials. A general term denoted for learning contents, pedagogical devices, assessment, and human interactions.

Learning object. A set of information blocks or sharable content objects organized to meet a particular learning objective.

Learning objective. (Also instructional objective) A statement of what students should know or be able to do as a result of instruction (Santrock, 2001, p. 499).

Mental model. (Also mental approach, mental method, and mental representation) A conceptual structure of declarative knowledge or procedural knowledge or both of them a person holds of a concept or a device or a system.

Mindful reflection and epistemic flexibility. The ability of students to be aware of their own role in the knowledge construction process.

Multiple modes of learning. (One of two main learning conditions for cognitive flexibility) Multiple representations of contents, multiple ways and methods for exploring contents.

Multiple perspectives on learning. (One of two main learning conditions for cognitive flexibility) Expression, confrontation, and treatment of multiple points of view.

Operational approach. An approach for designing learning environments that is based on operational criteria used as guidelines and means of validation.

Operational criterion. (for cognitive flexibility) A test that allows a straightforward decision about whether or not a learning situation reflects the pedagogical principles that are underlying cognitive flexibility.

Pedagogical devices. (A component of constructivist learning environments) Methods and tools provided for learners for exploring learning contents.

Performance assessment. Assessments that require students to perform a task such as write an essay, conduct an experiment, carry out a project, and solve a real-world problem (Santrock, 2001, p. 502).

Portfolio. A systematic and organized collection of a student's learning activities such as the student's work.

Post-active phase. (In instructional design) The process in which the teacher and the course designer specify what should be done after the learning session.

Pre-active phase. (In instructional design) The process in which the course designer specifies what should be done before the learning session.

Procedural knowledge. Cognitive structures in the form of skills and cognitive operations about how to do something (Santrock, 2001, p. 282).

Reasoning, critical thinking, and problem solving. The ability of the learner to write persuasive essays, engage in informal reasoning, explain how data relate to theory in scientific investigations, and formulate and solve moderately complex problems that require mathematical reasoning.

Retention, understanding, and use. The ability of the student to actively apply the new knowledge in various situations, particularly in interactions with other people, in order to reinforce his or her retention and understanding of the new knowledge.

Scaffolding. A technique of changing the level of support (learning contents, pedagogical devices, assessment, communication, problem-solving) over the course of a learning session of a particular learner.

Self-regulation. The ability of learners to identify and pursue their own learning goals.

Sharable content object. (Also content object) The lowest level of granuality of learning content that can be tracked by a learning content management system.

Social constructivist approach. (Also social constructivism) Educational approach that emphasizes that individuals construct knowledge through social interactions with others.

Standard deviation. A statistic that indicates how tightly all the various examples are clustered around the mean in a set of data.

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}.$$

Virtual learning environment. (Also virtual classroom) An online learning environment that provides facilitated, interactive instruction and peer-to-peer learner interaction during real-time events (Masie Center, 2003).

Bibliography

Adaptive Technology Resource Center (2004). *ATutor learning content management system*. Retrieved June 2, 2004, from: http://www.atutor.ca/

Advanced Distributed Learning (2004). *SCORM content aggregation model*. Retrieved November 8, 2004 from: http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=993

Ala-Mutka (2002). *Computer-assisted software engineering courses*. Paper presented at The 5th IASTED International Multi-Conference on Computers and Advanced Technology in Education. Cancun, Mexico.

Alliance of Remote Instructional Authoring and Distribution Networks for Europe (ARIADNE) (2005). Retrieved April 28, 2005, from: http://www.ariadne-eu.org/index.html

AMZI Inc. (1997). *Adventure in Prolog*. Retrieved January 27, 2004 from: http://oopweb.com/Prolog/Documents/AdventureInProlog/VolumeFrames.html

Anderson, J.R., Pirolli, P., & Farrell, R. (1988). Learning to program recursive functions. In M. Chi, R. Glaser, & M. Farr (Eds.), *The nature of expertise* (pp. 153–184). Hillsdale, NJ: Erlbaum.

Bayman, P., & Mayer, R.E. (1983). A diagnosis of beginning programmers' misconceptions of basic programming statements. *Communication of the ACM, 26(9)*, 677–679.

Bhuiyan, S., Greer, J., & McCalla, G. (1994). Supporting the learning of recursive problem solving. *Interactive Learning Environments, 4(2)*, 115–139.

Bonk, C. J., & Cunningham, D. J. (1998). Searching for learner-centered, constructivist, and sociocultural components of collaborative educational learning tools. In C. J. Bonk, & K. S. Kim (Eds.), *Electronic collaborators: Learner-centered technologies for literacy, apprenticeship, and discourse* (pp. 25–50). New Jersey: Erlbaum.

Bourgeois, E., & Nizet, J. (1999). *Apprentissage et formation des adultes*. Paris, FR: Presses Universitaires de France.

Bourgeois, E., & Frenay, M. (2002). *Les modèles théoriques de l'apprentissage: Note de synthèse*. Louvain-la-Neuve, BE: Cours EDFO 2106 Psychologie de l'apprentissage.

Bruner, J.S. (1973). *Going beyond the information given*. New York: Norton.

Bruner, J.S. (1986). *Actual minds, possible worlds*. Cambridge, MA: Harvard University Press.

Bruner, J.S. (1996). *The culture of education*. Cambridge, MA: Harvard University Press.

Brusilovsky, P. (1999). Adaptive and intelligent technologies for Web-based education. In C. Rollinger, & C. Peylo, *Special Issue on Intelligent Systems and Teleteaching* (pp. 19–25), Künstliche Intelligenz, 4.

Brusilovsky, P. & Peylo, C. (2003). Adaptive and intelligent Web-based educational systems. *International Journal of Artificial Intelligence in Education, 13*, 156–169.

Cho, K. & Schunn, C. D. (2004). *The SWoRD is mightier than the pen: Scaffolded writing and rewriting in the discipline*. Paper presented at the 4th IEEE International Conference on Advanced Learning Technologies, Joensuu, Finland.

Cho K. & Schunn, C.D. (2003). *Validity and reliability or peer assessments with a missing data estimati*on technique. Paper presented at the ED-Media 2003 World Conference on Educational Multimedia, Hypermedia & Telecommunications, Honolulu, Hawaii.

Cognition and Technology Group at Vanderbilt (1991a, May). Technology and the design of generative learning environments. *Educational Technology, 31*, 34–40.

Cognition and Technology Group at Vanderbilt (1991b, September). Some thoughts about constructivism and instructional design. *Educational Technology, 31*, 16–18.

Culler, J. (1990, April). *Fostering post-structuralist thinking*. Paper presented at the Annual Meeting of the American Educational Research Association, Boston.

Cumming, A. (1998). *Introduction to ML language*. Retrieved January 27, 2004 from: http://www.dcs.napier.ac.uk/course-notes/sml/manual.html

Cunningham, D.J. (1987). Outline of an education semiotic. *American Journal of Semiotics, 5*, 201–216.

Cunningham, D.J. (1992). Beyond educational psychology: Steps toward an educational semiotic. *Educational Psychology Review, 4*, 165–194.

Cunningham, D.J., Duffy, T.M., & Knuth, R.A. (1993). Textbook of the future. In C. McKnight (Ed.), *Hypertext: A psychological perspective*. London: Ellis Horwood Publishing.

De Bra, P. & Calvi, L. (1998). AHA! an open adaptive hypermedia architecture. *The New Review of Hypermedia and Multimedia, 4*, 115–139.

De Jong, T., van Joolingen, W., & van der Meij, J. (2004). *SimQuest discovery learning*. Retrieved November 8, 2004 from: http://www.simquest.nl

De Praetere, T., et al. (2004). *Claroline: Open source e-Learning*. Retrieved June 2, 2004, from: http://www.claroline.net/index.php

Deville, Y., Barette O., & Van Hentenryck, P. (1999). Constraint satisfaction over connected row-convex constraints. *Journal of Artificial Intelligence, 109*, 243–271.

Doise, W., & Mugny, G. (1997). *Psychologie sociale et développement cognitif*. Paris, FR: Armand Colin.

Dougiamas, M. (2004). *Moodle: Course management system*. Retrieved November 8, 2004 from: http://moodle.org

Driscoll, M.P. (2000). *Psychology of learning for instruction*. Massachusetts: Allyn and Bacon.

Du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research, 2(1)*, 57–73.

Duffy, T., & Jonassen, D., editors (1992). *Constructivism and the technology of instruction*. Mahwah, NJ: Lawrence Erlbaum Associates.

Duffy, T.M., & Cunningham, D.J. (1996). Constructivism: Implications for the design and delivery of instruction. In D.H. Jonassen (Ed.), *Handbook of research for educational communications and technology* (pp. 170–198). New York: Macmillan.

Duval, E., Hodgins, W., Sutton, S., & Weibel S.L. (2002). *Metadata principles and practicalities*. Retrieved November 8, 2004 from: http://www.masie.com/standards/s3supplement/edu_dlib_04weibel_042002.pdf

Eck, D.J. (2004). *Introduction to programming using Java*. Retrieved January 27, 2004 from: http://oopweb.com/Java/Documents/IntroToProgrammingUsingJava/VolumeFrames.html

Edelson, D.C., Pea, R.D., & Gomez, L. (1996). Constructivism in the collaboratory. In B.G. Wilson (Ed.), *Constructivist learning environments: Case studies in instructional design* (pp. 151–164). Englewood Cliffs, NJ: Educational Technology Publications.

Feltovich, P.J., Spiro, R.J., Coulson, R.L., & Feltovich, J. (1996). Collaboration within and among minds: Mastering complexity, individually, and in groups. In T. Koschmann (Ed.), *CSCL: Theory and practice of an emerging paradigm* (pp. 25–44). Mahweh, NJ: Erlbaum.

Frenay, M. (1994). *Apprentissage et transfert dans un contexte universitaire*. Louvain-la-Neuve, BE: Université catholique de Louvain, Faculté de psychologie et des sciences de l'éducation. Thèse de doctorat.

Frenay, M., & Bédard, D. (2004). Des dispositifs de formation s'inscrivant dans la perspective d'un apprentissage et d'un enseignement contextualisés pour favoriser la construction de connaissances et leur transfert. In A. Presseau & M. Frenay (Dir.), *Le transfert des apprentissages : comprendre pour mieux intervenir* (pp. 241–268). Québec, CA: Les presses de l'Université Laval.

Friesen, J. (2003). Data structures and algorithms, part II. In *Java World*. Retrieved

March 23, 2005 from: http://www.javaworld.com/javaworld/jw-06-2003/jw-0613-java101.html

GNU General Public License (1991). Retrieved June 2, 2004 from: http://www.gnu.org/copyleft/gpl.html

Google (2005). *The Google search engine*. Retrieved May 11, 2005 from: http://www.google.com/

Greenberg, L. (2002). *LMS and LCMS: What's difference?*. Retrieved April 19, 2005 from: http://www.learningcircuits.org/2002/dec2002/greenberg.htm

Greer, J., McCalla, G., Collins, J., Kumar, V., Meagher, P., & Vassileva, J. (1998). Supporting peer help and collaboration in distributed workplace environments. *International Journal of Artificial Intelligence in Education, 9*, 159–177.

Götschi, T., Sanders, I., & Galpin, V. (2003, February). *Mental models of recursion*. Paper presented at the ACM 34th SIGCSE Technical Symposium on Computer Science Education. Reno, Nevada.

Hannafin, M.J. (1992). Emerging technologies, ISD, and learning environments: Critical perspectives. *Educational Technology Research and Development, 40*, 49–64.

Henderson, P.B., & Romero, F.J. (1989). *Teaching recursion as a problem-solving tool using Standard ML*. Paper presented at the ACM 20th SIGCSE Technical Symposium on Computer Science Education. Louisville, Kentucky.

Henze, N. & Nejdl, W. (2001). Adaptation in open corpus hypermedia. *International Journal of Artificial Intelligence in Education, 12*, 325–350.

Honebein, P.C., Chen, P., & Brescia, W. (1992). Hypermedia and sociology: A simulation for developing research skills. *Liberal Arts Computing, 1*, 9–15.

Honebein, P.C., Duffy, T.M., & Fishman, B.J. (1993). Constructivism and the design of authentic learning environments: Context and authentic activities for learning. In T.M. Duffy, J. Lowyck, & D. Jonassen (Eds.), *Designing environments for constructive learning* (pp. 87–108). Hillsdale, NJ: Erlbaum.

Huitt, W. (2001). Motivation to learn: An overview. *Educational Psychology Interactive*. Valdosta, GA: Valdosta State University. Retrieved July 2, 2005 from: http://chiron.valdosta.edu/whuitt/col/motivation/motivate.html

IEEE Learning Technology Standards Committee (2005). Retrieved April 28, 2005 from: http://ltsc.ieee.org

IMS Global Learning Consortium (2005). Retrieved April 28, 2005 from: http://www.imsglobal.org/

Institute for Human and Machine Cognition (2005). *CmapTools*. Retrieved March 15, 2005 from: http://cmap.ihmc.us/

Java World (2004). *Java Q&A*. Retrieved January 27, 2004 from:

http://www.javaworld.com/columns/jw-qna-index.shtml

jEdit (2005). *Programmer's text editor*. Retrieved March 30, 2005 from: http://www.jedit.org

Johnson, M.J. (1987). *The body in the mind: The bodily basis of meaning*. Chicago: University of Chicago Press.

Jonassen, D.H. (1999). Designing constructivist learning environments. In C.M. Reigeluth, *Instructional design theories and models: Their current state of the art* (pp. 215–239). Mahwah, NJ: Lawrence Erlbaum Associates.

Jonassen, D.H., & Rohrer-Murphy, L. (1999). Activity theory as a framework for designing constructivist learning environments. *Educational Technology, Research & Development, 47(1)*, 61–79.

Jonassen, D. H., Peck, K. L., & Wilson, B. G. (1999). *Learning with technology: A constructivist perspective*. Upper Saddle River, NJ: Merrill.

Jonnaert, P. (1995). Entrer dans l'apprentissage scolaire. In G. Forges, (éd.), *Enfants issue de l'immigration et apprentissage du français langue seconde* (pp. 15–53). Paris, FR: Didier-Érudition.

Jonnaert, P., & Vander Borght, C. (2003). *Créer des conditions d'apprentissage: Un cadre de référence socioconstructiviste pour une formation didactique des enseignants.* Bruxelles, BE: De Boeck-Université, Perspectives en Éducation et Formation.

Kaplan, D.A. (2000). *The Silicon boys and their valley of dreams*. New York, NY: HarperCollins.

Kearsley, G. (2003). *Theory into practice*. Retrieved September 24, 2004 from: http://tip.psychology.org.

Kinshuk, Looi, C.K., Sutinen, E., Sampson, D., Aedo, I., Uden, L., & Kähkönen, E. (2004). *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies*. IEEE Computer Society.

Kirsh, D. (2000). *A few thoughts on cognitive overload*. Retrieved September, 12, 2005 from: http://icl-server.ucsd.edu/~kirsh/Articles/Overload/published.html

Kjell, B. (2003). *Introduction to computer science using Java*. Retrieved November 8, 2004 from: http://chortle.ccsu.edu/CS151/cs151java.html

Klahr, D., & Dunbar, K. (1988). Dual space search during scientific reasoning. *Cognitive Science, 12*, 1–48.

Knuth, R.A., & Cunningham, D.J. (1993). Tools for constructivism. In T. Duffy, J. Lowyck, & D. Jonnasen (Eds.), *Designing environments for constructive learning* (pp. 163–188). Berlin, DE: Springer-Verlag.

Koschmann, T. (1996). *CSCL: Theory and practice of an emerging paradigm*. Mahweh, NJ: Erlbaum.

Kuhn, Th. (1983). *La structure des révolutions scientifiques*. Paris, FR: Flammarion.

Laanpere, M., Põldoja, H., & Kikkas K. (2004). The second thoughts about pedagogical neutrality of LMS. Paper presented at the 4th IEEE International Conference on Advanced Learning Technologies, Joensuu, Finland.

Lajoie, S. P. & Lesgold, A. M. (1992). Dynamic assessment of proficiency for solving procedural knowledge tasks. *Educational Psychologist 27(3)*, 365–384.

Lakoff, G. (1987). *Women, fire and dangerous things: What categories reveal about the mind*. Chicago: University of Chicago Press.

Language Development and Hypermedia Group (1992). Bubble dialogue: A new tool for instruction and assessment. *Educational Technology, Research & Development, 40(2)*, 59–67.

Legendre, R. (1988). *Dictionnaire actuel de l'éducation*. Paris, FR: Larousse.

Leslie, S. (2003). *Open source course management systems*. Retrieved June 2, 2004, from: http://www.edtechpost.ca/gems/open_source_cms3.htm

Lewis, J. & Loftus, W. (2003). *Java software solutions*. Boston: Addison Wesley Longman.

Malmi, L. & Korhonen, A. (2004). *Automatic feedback and resubmissions as learning aid*. Paper presented at the 4th IEEE International Conference on Advanced Learning Technologies, Joensuu, Finland.

Marshall, H.H. (1996). Implications of differentiating and understanding constructivist approaches. *Educational Psychologist, 31*, 235–240.

MASIE Center e-Learning CONSORTIUM (2003). *Making sense of learning standards and specifications*. Retrieved November 8, 2004 from: http://www.masie.com/standards/s3_2nd_edition.pdf

Milgrom, E., Jacqmot, Ch., Blaise, O., Cohen, A., D'Hautcourt, F., Lammé, A., & Uyttebrouck, E. (1997). Evaluation of web-based tools for building distance education systems. *Journal of interactive instruction development*, 3–11.

Mitrovic, A. (2005). *Constraint-based tutors: A success story*. Invited talk presented at the 12th International Conference on Artificial Intelligence in Education, Amsterdam, The Netherlands (soon appear).

Morrison, D., & Collins, A. (1996). Epistemic fluency and constructivist learning environments. In B.G. Wilson (Ed.), *Constructivist learning environments: Case studies in instructional design* (pp. 107–120). Englewood Cliffs, NJ: Educational Technology Publications.

Murray, T. (1999). Authoring intelligent tutoring systems: Analysis of the state of the art". *International Journal of Artificial Intelligence in Education, 10*, 98–129.

OOPWeb (2004). *Java Programming Tutorials*. Retrieved January 9, 2004 from: http://www.oopweb.com/Java/Files/Java.html

Pea, R.D. (1993). The collaborative visualization project. *Communications of the ACM, 36*, 60–63.

Pedagogy Group at FSA/UCL (2005). *Candis 2000: Apprendre par les problèmes*. Retrieved March 10, 2005 from: http://www.fsa.ucl.ac.be/candis/publications/index.html

Perkins, D.N. (1991a, May). Technology meets constructivism: Do they make a marriage? *Educational Technology, 31*, 18–23.

Perkins, D.N. (1991b, September). What constructivism demands of the learner. *Educational Technology, 31*, 19–21.

Perkins, D.N. (1996). Minds in the 'hood. In B.G. Wilson (Ed.), *Constructivist learning environments: Case studies in instructional design* (pp. v–viii). Englewood Cliffs, NJ: Educational Technology Publications.

Piaget, J. (1975). *L'équilibration des structures cognitives*. Paris, FR: PUF.

Reeves, T.C., & Okey, J.R. (1996). Alternative assessment for constructivist learning environments. In B.G. Wilson (Ed.), *Constructivist learning environments: Case studies in instructional design* (pp. 191–202). Englewood Cliffs, NJ: Educational Technology Publications.

Reimann, P. (1991). Detecting functional relations in a computerized discovery environment. *Learning and Instruction, 1*, 45–65.

Robbins, S.R. (2002). *The evolution of the learning content management system*. Retrieved April 19, 2005 from: http://www.learningcircuits.org/2002/apr2002/robbins.html

Roberts, E.S. (1986). *Thinking recursively*. John Wiley & Sons, NY.

Rogoff, B. (1998). Cognition as a collaborative process. In W. Damon, D. Kuhn, & R.S. Siegler (Eds.), *Handbook of child psychology* (5th ed., Vol. 2). New York: Wiley.

Rubin, E. (1915). *Visuell wahrgenommene figuren*. Gyldenalske Boghandel, Copenhagen.

Santrock, J.W. (2001). *Educational psychology*. New York: McGraw-Hill.

Sasse, M.A. (1991). How to t(r)ap users' mental models. In M.J. Tauber & D. Ackermann, *Mental models and human-computer interaction* (Vol. 2). Amsterdam, NL: Elsevier Science Publishers.

Schach, S.R. (1999). *Classical and object-oriented software engineering*. New York: McGraw-Hill.

Shepard, L.A. (1991). Psychometricians' beliefs about learning. *Educational Researcher*, 2–16.

SimCity. (2004). *A city simulator*. Retrieved January 6, 2004 from: http://www.mcli.dist.maricopa.edu/proj/sw/games/simcity.html

Soderman, A.K., Gregory, K.M., & O'Neill, L.T. (1999). *Scaffolding emerging literacy*. Boston: Allyn & Bacon.

Spiro, R.J., Coulson, R.L., Feltovich, P.J., & Anderson, D.K. (1988). Cognitive flexibility theory: Advanced knowledge acquisition in ill-structured domains. In *Tenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Spiro, R.J., & Jehng, J.C. (1990). Cognitive flexibility and hypertext: Theory and technology for the nonlinear and multidimensional traversal of complex subject matter. In D. Nix, & R.J. Spiro, *Cognition, education and multimedia* (pp. 163–205). Hillsdale, NJ: Erlbaum.

Spiro, R.J., Feltovich, P.J., Jacobson, M.J., & Coulson, R.L. (1991, May). Cognitive flexibility, constructivism, and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. *Educational Technology, 31*, 24–33.

Stoyanov, S., & Kirschner, P. (2004). Expert concept mapping method for defining the characteristics of adaptive e-Learning: ALFANET project case. *Educational Technology, Research & Development, 52(2)*, 41–56.

Sun Microsystems Corp. (2004). *The online Java tutorial*. Retrieved January 9, 2004 from: http://java.sun.com/docs/books/tutorial/index.html

Suomela, J. (2005). *Jari's astronomy site*. Retrieved May 12, 2005 from: http://www.kolumbus.fi/jimenez/quotes/index.htm

Sweller, J. (2005). *Cognitive load theory*. Retrieved September, 12, 2005 from: http://tip.psychology.org/sweller.html

Turbak, F., Royden, C., Stephan, J., & Herbst J. (1999). Teaching recursion before loops in CS1. *Journal of Computing in Small Colleges, 14(4)*, 86–101.

Van Hentenryck, P., Saraswat V., & Deville, Y. (1998, October). The design, implementation, and evaluation of the constraint language cc(FD) ". *Journal of Logic Programming*, Special Issue on Constraint Logic Programming, 26 pages, Vol. 37 (1–3).

Van Joolingen, W., & De Jong, T. (1997). An extended dual search space model of scientific discovery learning. *Instructional Science, 25*, 307–346.

Van Joolingen, W. (1999). Cognitive tools for discovery learning. *International Journal of Artificial Intelligence in Education, 10*, 385–397.

Vietshare.com (2004). *Câu chuyện cát đá*. Retrieved October 25, 2004 from: http://www.vietshare.com.

Von Foerster, H. (1988). La construction d'une réalité. In P. Watzlawick, (dir.), *L'inven-*

tion de la réalité. Contributions au constructivisme (pp. 45–69). Paris, FR: Seuil.

Vygotsky, L.S., (1962). *Thought and language*. Cambridge, MA: MIT Press.

Weber, G. (1996). Episodic learner modeling. *Cognitive Science, 20 (2)*, 195–236.

Weber, G., & Specht, M. (1997). *User modeling and adaptive navigation support in WWW-based tutoring systems*. Paper presented at The 6th International Conference on User Modeling. Vienna, New York.

Weber, G. & Brusilovsky, P. (2001). ELM-ART: an adaptive versatile system for Web-based instruction. *International Journal of Artificial Intelligence in Education, 12*, 351–384.

Wiley, D.A. (2002). *The instructional use of learning objects*. Retrieved February 17, 2004 from: http://www.reusability.org/read/

Wilson, B., Teslow, J., & Osman-Jouchoux, R. (1995). The impact of constructivism (and postmodernism) on ID fundamentals. In B. B. Seels (Ed.), *Instructional design fundamentals: A review and reconsideration* (pp. 137–157). Englewood Cliffs NJ: Educational Technology Publications.

Wilson, B.G. (1996). *Constructivist learning environments: Case studies in instructional design*. Englewood Cliffs, NJ: Educational Technology Publications.

Wilson, B.G. (1997). Reflections on constructivism and instructional design. In C.R. Dills & A.A. Romiszowski (Eds.), *Instructional development paradigms*. Englewood Cliffs NJ: Educational Technology Publications.

Wittrock, M.C. (1985a). Teaching learners generative strategies for enhancing reading comprehension. *Theory into Practice, 24*, 123–126.

Wittrock, M.C. (1985b). The generative learning model and its implications for science education. *Studies in Science Education, 12*, 59–87.

WorldPeace J. (1997). *The contemporary Tao of peace and harmony*. Retrieved May 11, 2005 from: http://www.johnworldpeace.com/taoteching.html

Wright, W.A. (1995). *Teaching improvement practices. Successful strategies for higher education*. Bolton: Anker Publishing Company.

Yahoo (2005). *The Yahoo search engine*. Retrieved from May 11, 2005 from: http://search.yahoo.com/