Building a Rationale Diagram for Evaluating User Story Sets

Yves Wautelet*, Samedi Heng †, Manuel Kolp†, Isabelle Mirbel ‡, Stephan Poelmans*
* KU Leuven, Belgium. Email: {yves.wautelet, stephan.poelmans}@kuleuven.be
† Université catholique de Louvain, Belgium. Email:{samedi.heng, manuel.kolp}@uclouvain.be
‡ University of Nice Sophia Antipolis, France. Email: isabelle.mirbel@unice.fr

***Abstract*—Requirements representation in agile methods is often done on the basis of User Stories (US) which are short sentences relating a WHO, WHAT and (possibly) WHY dimension. They are by nature very operational and simple to understand thus very efficient. Previous research allowed to build a unified model for US templates associating semantics to a set of keywords based on templates collected over the web and scientific literature. Since the semantic associated to these keywords is mostly issued of the i* framework we overview in this paper how to build a custom rationale diagram on the basis of a US set tagged using that unified template. The rationale diagram is strictly speaking not an i* strategic rationale diagram but uses parts of its constructs and visual notation to build various trees of relating US elements in a single project. Indeed, the benefits of editing such a rationale diagram is to identify depending US, identifying EPIC ones and group them around common Themes. The paper shows the feasibility of building the rationale diagram, then points to the use of these consistent sets of US for iteration content planning. To ensure the US set and the rationale diagram constitute a consistent and not concurrent whole, an integrated *Computer-Aided Software Engineering* (*CASE*) tool supports the approach.**

***Index Terms*—User Story, User Story Template, Rationale Diagram, Agile Requirements Modeling, eXtreme Programming, SCRUM**

I. Introduction

Following [1], *User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.* [2] acknowledged that no unification is provided in *User Story* (*US*) templates. Indeed, the general pattern (which is the one tackled in this paper with no further extensions) relates a WHO, a WHAT and possibly a WHY dimension[1], but different keywords are used in these dimensions in practice (e.g. Mike Cohn's *As a <type of user>, I want <some goal> so that <some reason>* [1]). Moreover, in the literature, no semantics are ever associated to these keywords (see [2]). This is why, [2] conducted research to find the majority of templates used in practice, sort them and associate semantics to each keyword. These semantics were derived from several sources and frameworks (by order of importance [3], [4], [5], [6]); some of these are derived from Goal-Oriented Requirements Engineering (GORE, see [7]). After performing a redundancy analysis, a first selection of keywords with associated semantics was performed. Then, applying them on large test sets led to a sub-selection of keywords forming a unified model (see Section III-C). In the end, most of the semantics adopted for the remaining keywords were selected from the i* framework (i-star [3], [8], [9], [10]); this is due to the research design that favored adopting a consistent framework. Note that it is not the i* framework that is fully rebuilt for tagging US since concepts like the resource, the agent, etc. are not included in the unified model. The capability, another concept not existent in i*, has been included in the unified model (see [2] for details). Authors demonstrated the applicability of this unified model of US templates onto a test set made of both US examples and cases.

One may indeed question about the utility of such a model for agile practitioners. In the end, why should US be "tagged" to a certain template or keyword and not simply expressed following the WHO/WHAT and WHY structure without more refinement. The main advantage of tagging is that, if done respecting the semantics associated to the concepts, it gives information about the nature but also the granularity of the US element. Such information could possibly be used later on for analysis or structuring of the problem as for example pointed out by [11]. Structuring of US is often done with the *User Story Mapping* technique (see [12]); the latter uses *Story Maps* – which are hard to maintain and read – so that other techniques for visual representation could be welcome.

In this perspective, we suggest to explore the visual representation of US starting from a set of US tagged following the model of [2]. Since the latter unified model is largely inspired by i* semantics, this paper overviews how one can build a diagram in the form of a tree using the constructs of the i* Strategic Rationale (SR) diagram with the elements contained in sets of US. The goal is thus *not* to use the SR as such, but to build a graphical notation largely inspired by the SR convenient for the representation of the US elements and studying their refinements, compositions and decompositions in order to group them consistently. In the requirements engineering process built out of our contribution, we point to keep up with agile principles and to build the set of US first then to generate a rationale diagram[2] on their basis.

[1]Examples are provided in Table I.

[2]In this paper, we refer to the *rationale diagram* as the diagram that we build in order to visually represent US elements issued of a US set and their links. It is strictly speaking not a SR diagram but uses close notation and constructs (this is build-up and motivated in the paper). For a complex case this rationale diagram is made of several decomposition trees.

We indeed do not believe that starting from i* modeling in agile development could, as such, be adopted as an alternative to USM because the approach is very abstract and often starts with as-is requirements representation so is not really in line with what agile modelers are expecting for requirements representation. Nevertheless, an i*-like diagram furnishing a consistent visual representation of an existing US set thus providing a graphical representation of the system-to-be only provides added value and is in line with agile expectations.

Our proposal is illustrated through a running example about carpooling. Carpooling deals with the sharing of car journeys so that more than one person travels into the car. In this context, it takes increasing importance to save gas, reduce traffic, save driving time and control pollution. *ClubCar* is a multi-channel application available as an Android application, SMS service and *Interactive Voice Response* (IVR) system. Users of ClubCar are riders and/or drivers that can register by SMS, voice or through an Android app. Roughly speaking the software allows drivers to propose rides and submit their details with *dates*, *times*, *sources* and *destinations* while riders can search for available rides [13]. This example is part of a bigger case study and US have been selected to be representative. Nevertheless, the full application of the methodology on a real-life case with a study of all encountered issues is left for future work.

The paper is structured as follows. Related work is discussed in Section II while Section III exposes a meta-model of elements aiming at grouping US (macro-level) as well as decomposing US (micro-level). The meta-model to build US templates issued from previous research [2] is also described. Section IV explicitly maps the elements of the US template meta-model with the elements of the SR model to use its reasoning techniques with a project's US. Section V abstracts the different cases we can face within the edition of US using the reasoning approach of the SR model, how Epic US can be identified in these cases and how US can be grouped around Themes. Section VI discusses its inclusion in the agile software process while Section VII discusses the automation of the approach and its support through a *Computer-Aided Software Engineering* (*CASE*) tool. Section VIII discusses the validity, the threats to validity, the scalability of the approach and future works. Finally, Section IX concludes the paper.

II. Related Work

Our work aims to organize US on the basis of a proper granularity analysis. Among other sources, the need for granularity levels in US-based modeling has been identified in [11]. The problem of poor scalability of agile methods because of poor granularity identification in requirements representation has been identified in [14], [15].

The process of transformation from a set of US to an SR diagram can be compared to a more formal approach to *User Story Mapping* (*USM*, see [16]). USM is the industry adopted technique that relates the most to our approach. Within a project, USM is intended to produce a *Story Map* (*SM*) which is a map of a project's US according to (i) the level of abstraction, (ii) the sequence (horizontal dimension), and (iii) the priority (vertical dimension).

Basically, USM defines three layers: the *backbone* which represents an entire user activity (or process), the *walking skeleton* which represents a user task and the *slice US* which represents a small, implementable and concrete US [16]. We informally evaluate a possible alignment with elements present in our approach (see Section III-C for their definition). The USM *user activity* – which is an abstract objective – could then be compared to a US containing a *Goal* in its WHAT dimension while the USM *user task* – which makes the former objective more concrete – could be compared to a US containing a *Task* in its WHAT dimension and, finally, the USM *implementable US* – which is the most concrete and atomic one – could be compared to a US containing a *Capability* in its WHAT dimension. We could thus say that, by nature, granularity of elements is not what distinguishes our approach from USM. This is rather interesting since the model of [2] that we base our approach upon has been built from existing sets of US templates and examples and, empirically, also distinguished 3 required granularity levels. Nevertheless, our approach diverges from USM in the use of a graphical notation inducing the use of formal links between US elements while USM uses story cards with a color-coding technique. This allows only limited expressiveness and flexibility in US manipulation. Indeed, one finer-grained US can then only relate to one coarser-grained US where we could define multiple links.

When compared to USM, a bit more effort is required with the technique we propose in this paper since we split a US in 2 or 3 dimensions and we use the graphical representation of i*. Nevertheless, this leads to:

- a graphical representation of requirements. SM remain limited to post-its on a board or even on the ground;
- a structuring of requirements where we can:
  - systematically eliminate redundant US elements. A SM is aimed to achieve a comparable process but with a refinement on the basis of the WHAT and WHY dimensions; our process offers finer possibilities.
  - study the dependencies of US to other US (thus multiple possible dimensions) notably useful for the identification of Theme US. SM hierarchy is limited since a US can only be under the column (scope) of one Epic US;

We thus argue that with comparable modeling effort we make use of a more precise model to build the system-to-be.

[17] envisages a transformation approach from sets of US tagged with the model of [2] to a Use Case Diagram. Roughly speaking the authors point to the transformation of goal elements as well as some task elements to use-cases in a use-case diagram. The approach delivers a coarse-grained representation of the system-to-be but fails to bring a decomposition approach necessary to study US inter-dependencies as we build-up in this paper.

III. Research Design

This section exposes the "building blocks" used within this research. Our main goal is to be able to group US on the basis of their interdependencies. A few concepts have been proposed in agile literature[3] from which we have built up a meta-model. At the macro-level, detailed in Section III-A, relationships with US concepts are highlighted. US of various granularity levels require to be composed/decomposed into other US or be grouped with other US around common themes. Features are also required for the proper fulfillment of US. We also propose to decompose US through their dimensions in so called *Descriptive Concepts* to allow their analysis. Section III-B depicts this micro-level. Finally, Section III-C depicts the meta-model to generate US templates.

A. Macro-level: Ways to Organize User Stories

We first distinguish a US macro-level dealing with the grouping of US into depending (i.e. relating) sets. We thus envisage here the US as potential building blocks serving for project composition/decomposition. US are indeed subject to a process of sorting and dropping early on into the development project [18]; an initial identification of the US hierarchy using precise semantics could help (i) addressing priorities among coherent and non-redundant sets of US and (ii) manage changing requirements by understanding the impact of changes.

Figure 1 presents our meta-model of the US concepts in its project environment. At this macro-level, the aim is to identify what elements could be used (i) to group US, (ii) to abstract them or (iii) to see what technical elements should be provided by the system and that they are concerned with.

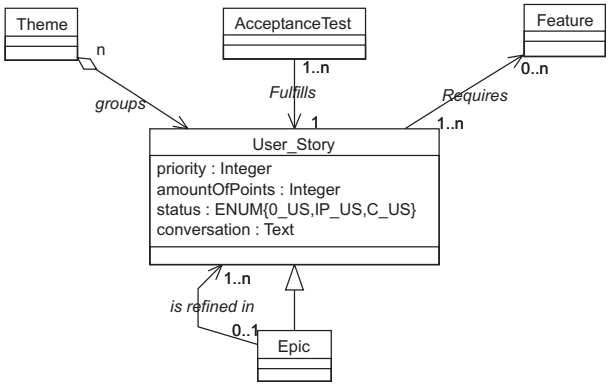


Fig. 1. US as Macro-Level Structures: Meta-Model

The *User_Story* class represents the US characteristics as a whole. Chronologically, *US* are written by the customer or product owner at the earliest stages of the project and put in the product backlog with an (implementation) *priority* and an *amountOfPoints* which refers to the number of *User Story Points* (*USP*)[4] [19], [20]. These elements have thus been added as attributes to the *User_Story* class. Other attributes required for process management are included within the *User_Story* class. Indeed, US are written onto *User Story Cards* (*USC*). To support their implementation, we enrich the *User_Story* class with a *status* attribute which contains the status of the US on the USC. The value of the *status* in the USC can be threefold: *Open User Story* (*O_US*), *In Progress User Story* (*IP_US*) and *Completed User Story* (*C_US*) [21]. In addition, the *conversation* attribute contains the detailed discussion about the *US*.

The *AcceptanceTest* [19] class encapsulates the set of predefined tests for a US. This is used to validate whether the US satisfies stakeholders' requirement(s). This can be a normal/abnormal scenario and is defined by the tester.

Some US need to be refined into other ones since they are too abstract (coarse-grained) to be estimated, implemented and tested at once. These are called *Epic US* [19]. The latter are indeed US with a high-level of abstraction meaning that they must be refined/decomposed into smaller US to describe the requirement more precisely. These US are represented by the class *Epic* inheriting from the class *User_Story*.

A *Theme* is a collection of related US [19]. We model it using the *Theme* class as a grouping of a set of lower level US.

Finally, *Features* inherently relate to technical elements not expressed into US but that must be provided by the system. Indeed, *a feature is a delimitable characteristic of a system that provides value for stakeholders* [6]. This definition can be refined by *... a unit of functionality of a software system that satisfies a requirement, represents a design decision, and provides a potential configuration option* [22]. As highlighted in [23], the feature is unique when compared to the other semantics because it refers to part of the system that satisfies a functional or non-functional requirement and thus shapes part of the structure of the system-to-be. The *Feature* class represents the concept.

B. Micro-level: Decomposing a User Story in Descriptive Concepts

Within Figure 2, the meta-model of the previous section is enriched with the constituting elements of the US; we refer here to this US view as the micro-level.

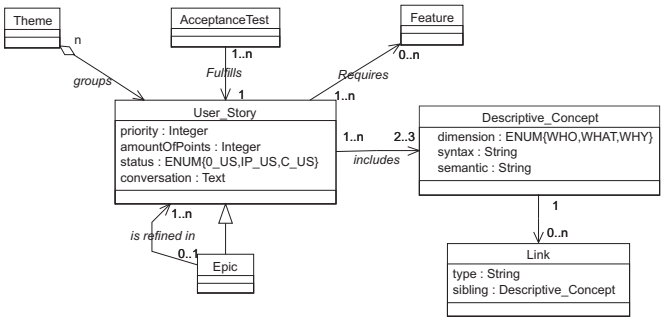


Fig. 2. US as Macro and Micro-Level Structures: Meta-Model

[3]We have focused on the available sources describing the eXtreme Programming (XP) and SCRUM methods.

[4]The amount of USP represents the estimated effort required to implement the US.

Rather than using the US as a whole within the requirements analysis process, we suggest, in our research design, to decompose the US on the basis of their *WHO*, *WHAT* and, when available, *WHY* dimensions. For the sake of uniformity, these elements are all characterized as *Descriptive_Concepts* (*D_C*). When decomposed into a set of *D_C*, the dependency between *D_C* is intended to be further studied (see Section III-C). Each element of a US template relating to one of the 3 dimensions is then an instance of the *D_C* class. For the template *As a <role>, I need a <task> so that <goal>*, we have 3 instances of the *D_C* class: one for *role*, one for *task* and one for *goal*. The *dimension* attribute thus compulsorily takes one of the values *WHO* (for *role*), *WHAT* (for *task*) or *WHY* (for *goal*) and the *syntax* attribute takes the syntax of the concept name (e.g. role, task, goal, ...). The *semantic* attribute relates to the definition of the *D_C*. The list of all the possible *D_C* is given in the form of a meta-model allowing to define US templates in Section III-C.

Finally, since different *D_C* can be linked together, we introduce the *Link* class that represents the possible different types of links between two *D_C*. The possible instances of the *Link* class will be studied in Section IV-C.

C. Unified-Model of User Stories' Descriptive_Concepts

As evoked, [2] suggests to build a unified model for designing US templates. The interested reader will refer to the latter reference for the research details and process and we use this model as reference. Figure 3 represents the meta-model of US templates. A US template can be designed taking an element from the WHO, WHAT and possibly WHY dimensions. The link between the classes conceptually represents the link from one dimension to the other. Concretely, the unidirectional association from the *Role* to one of the *Capability*, *Task* or *Goal* classes implies that the target class instantiates an element of the WHAT dimension (always tagged as *wants/wants to/needs/can/would like* in the model). Then, the unidirectional association from one of these classes instantiating the WHAT dimension to one of the classes instantiating the WHY dimension (always tagged as *so that* into the model) implies that the target class eventually (since 0 is the minimal cardinality) instantiates an element of the WHY dimension. A US template supported by our model is for instance: *As a <Role>, I would like <Task> so that <Hard-goal>*.

Each concept is associated with a particular syntax (identical to the name of the class in Figure 3) and a semantic. The syntax and semantics of the model are summarized here. As a result of the research conducted in [2], the couples syntax/semantic are the following:

- *A role is an abstract characterization of the behavior of a social actor within some specialized context or domain of endeavor*;
- *A task specifies a particular way of attaining a goal*;
- *A capability represents the ability of an actor to define, choose, and execute a plan for the fulfillment of a goal, given certain world conditions and in the presence of a specific event*;
- *A hard-goal is a condition or state of affairs in the world that the stakeholders would like to achieve*;
- *A soft-goal is a condition or state of affairs in the world that the actor would like to achieve. But unlike a hard-goal, there are no clear-cut criteria for whether the condition is achieved, and it is up to the developer to judge whether a particular state of affairs in fact achieves sufficiently the stated soft-goal*.

Only the *Role* is used as possible element in the WHO dimension, this is justified in [2].

These are thus the semantics that we consider valid for our requirements modeling framework, other agile frameworks may have a different interpretation (so other semantics/definitions associated to these concepts, e.g. the Scaled Agile Framework 4.0 very recently introduced the capability concept as a very abstract element, see [24], [25]). A few more explanations are nevertheless required to be able to distinguish a *Hard-goal* from a *Task* and a *Capability* element.

The *Hard-goal* is the most abstract element; there is no defined way to attain it and several ways could be followed in practice. It is indeed part of the problem domain. The *Task* that represents an operational way to attain a *Hard-goal*. It is thus part of the solution domain. An example of a *Hard-goal* could be to *Be transported from Brussels to Paris*; it can be the *Hard-goal* of a traveler but there are several ways to attain this *Hard-goal* (by train, by car, etc.).

The *Task* and the *Capability* represent more concrete and operational elements but these two need to be distinguished. The *Capability* could in fact be modeled as a *Task* but the *Capability* has more properties than the former since it is expressed as a direct intention from a role. In order to avoid ambiguities in interpretation, we point to the use of the *Capability* element only for *an atomic Task* (i.e., a task that is not refined into other elements but is located at the lowest level of hierarchy). A *Task* could then be *Move from Brussels to Paris by car* and a *Capability* would be *Sit in the car*.

Finally, since the reader may raise the question *why is the Capability concept required if it represents a Task that is atomic, thus why not represent it as a Task?*. First of all, the study of [2] started on an empirical basis and lead to the identification of 3 levels of refinement, therefore these were required "from the field". The approach used by User Story Mapping (see Section II) also consists in three levels of refinement.

The set possible US templates that can be derived from the meta-model in Figure 3 and thus resulting of the research in [2] is:

- *As a <Role>, I want/want to/need/can/would like <Task>*;
- *As a <Role>, I want/want to/need/can/would like <Task> so that to <Task>*;
- *As a <Role>, I want/want to/need/can/would like <Task> so that to <Hard-goal>*;
- *As a <Role>, I want/want to/need/can/would like <Task> so that to <Soft-goal>*;

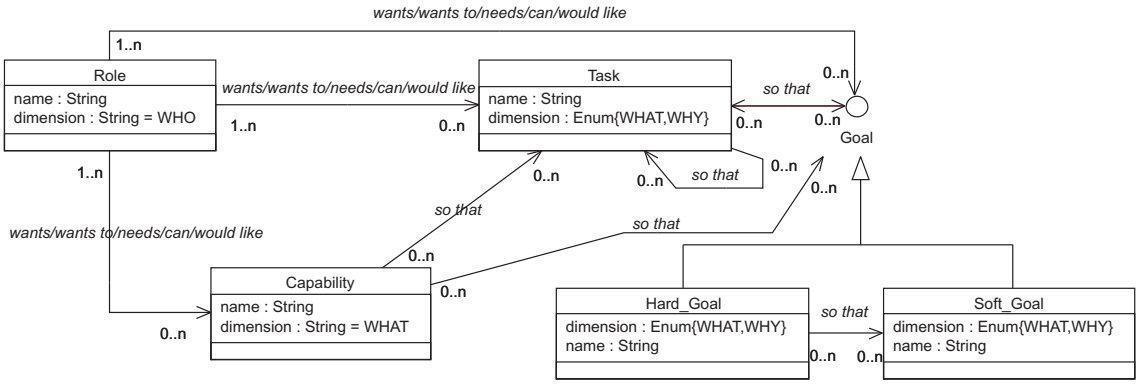


Fig. 3. Unified Model for User Story Descriptive Concepts

- *As a <Role>, I want/want to/need/can/would like <Capability>*;
- *As a <Role>, I want/want to/need/can/would like <Capability> so that to <Task>*;
- *As a <Role>, I want/want to/need/can/would like <Capability> so that to <Hard-goal>*;
- *As a <Role>, I want/want to/need/can/would like <Capability> so that to <Soft-goal>*;
- *As a <Role>, I want/want to/need/can/would like <Hard-goal>*;
- *As a <Role>, I want/want to/need/can/would like <Hard-goal> so that to <Hard-goal>*;
- *As a <Role>, I want/want to/need/can/would like <Hard-goal> so that to <Soft-goal>*.
- *As a <Role>, I want/want to/need/can/would like <Soft-goal>*.

IV. A Graphical Notation for US Dependency Analysis: Micro-Level Approach

The purpose of this section is to explicitly map the concepts of the unified model of US templates with the concepts issued of the Strategic Rationale (*SR*) Model; this would allow to derive a relevant graphical notation to be used for reasoning around a project US.

A. The WHO Dimension: Graphical Notation

Within the WHO dimension, we only find, in the unified model, the *Role* concept.

The *Role* concept has semantics issued from the i* framework and is thus "natively" supported by the SR model with a defined icon (see Figure 4). Similarly, the boundary of the actor is defined as a circle associated to the role as within the SR model. This graphical notation is thus also adopted here within the graphical representation of WHO-dimension US elements.

B. The WHAT and WHY Dimensions: Graphical Notation

Within the WHAT and WHY dimensions we find, in the unified model, the *Task*, *Capability* and *Goal* concepts. The latter must be a *Hard-goal* or *Soft-goal* so that we in total have 4 concepts that need to be represented in these two dimensions.

All of these concepts – except the *Capability* – have semantics issued from the i* SR model and are thus supported by a defined icon (see Figure 4). These graphical notations are thus also adopted here for these 3 concepts within the graphical representation of WHAT and WHY US elements.

The *Capability* concept with its associated semantics is not a requirements modeling concept but rather an agent-oriented design one not "natively" supported by the SR model. The relevancy of its inclusion in the unified model has been discussed in [2] and, as evoked earlier, we keep it for the modeling of atomic *Tasks* (of course performed by a determined *Role*). As shown in Figure 4, we introduce a genuine icon for this concept. We also add a constraint on this element: *it cannot be used as an entire mean in a means-end decomposition (see next section) because it is atomic (i.e. low level and concrete)*.

C. Linking Descriptive_Concepts of the Unified User Story Model

Now that we have set-up the icons for the different elements, we need to study the possible types of links between elements of the WHAT and/or of the WHY dimension. Three types of links between elements are specifically defined for the SR model; these are:

- Means-end links which *indicate a relationship between an end, and a means for attaining it. The "means" is expressed in the form of a task, since the notion of task embodies how to do something, with the "end" is expressed as a goal. In the graphical notation, the arrowhead points from the means to the end* [3];
- Decomposition links are more specifically associated to tasks, indeed *a task element is linked to its component nodes by decomposition links* [3]. Moreover, *A task can be decomposed into four types of elements: a subgoal, a subtask, a resource, and/or a softgoal - corresponding to the four types of elements. The task can be decomposed into one to many of these elements. These elements can also be part of dependency links in Strategic Dependency model(s) when the reasoning goes beyond an actor's boundary* [3];
- Contribution links for contributions to Soft-goals, indeed *any of these Contribution Links can be used to link any of*

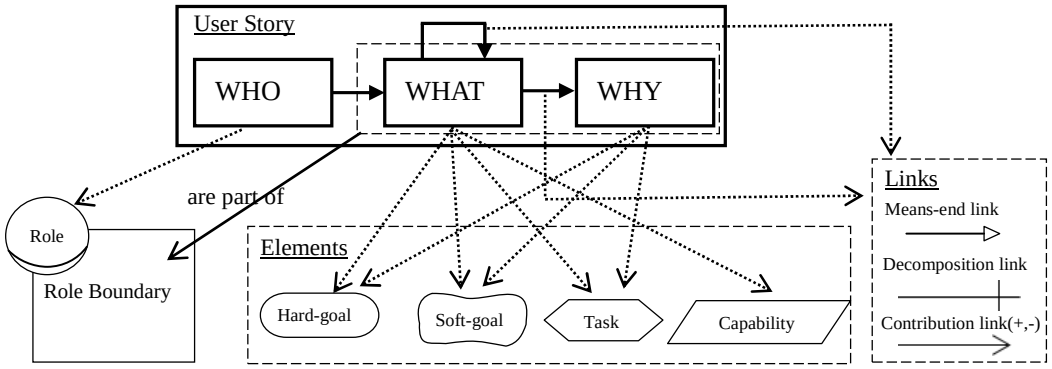


Fig. 4. Icons Used Within the Representation of the US Elements Using the Strategic Rationale Reasoning

the elements to a Soft-goal to model the way any of these Elements contributes to the satisfaction or fulfillment of the Soft-goal [3].

When a WHY dimension is present into a US, we can deduce that there is a link between the elements of the WHAT and the WHY dimensions even if this link is not necessarily a direct one. Intuitively we could think that there is compulsorily a means-end link (the WHAT element is a mean to attain the WHY element) but this cannot be stated as a rule (thus nor automated). Indeed, empirically a lot of cases can be found where the element in the WHY dimension is very coarse-grained and the element in the WHAT dimension atomic. Then, the WHAT element is just a step in the realization scenario rather than an entire mean to achieve the element in the WHY dimension. This means that if the WHAT element is a *Capability* or a *Task* located on a low level basis, that is, a step or partial set of steps in the realization of an element expressed in a very coarse-grained manner in the WHY dimension (like a *Hard-goal* but could also be a *Task* or a *Soft-goal*), then there can be, in the diagram representing the set of US, elements between the elements of the WHAT and WHY dimensions of this single US.

The modeler has to create the links between the *D_C* in function of the requirements/domain analysis. The study and linking of elements lead to a tree hierarchy in a SR diagram fashion. That way an analysis of the alternatives (means-end) and of the possible redundancy (in the decompositions) could also be performed.

Note that a decomposition within our model can cross the boundaries of a single role. This is represented in the form of a dependency within a classical i* Strategic Dependency model. We here focus on decomposition only, so that we do not include dependency associations that would increase the complexity of the diagram (see examples in Table II and Figure 6).

V. Towards a Rationale Analysis for US Hierarchy and Grouping: From Micro to Macro Level

This section is aimed to study how the i* SR models' constructs can be used to build a custom rationale diagram (so we do not pretend that it is an i* SR diagram but it is however largely inspired by it) aligned with a set of US in order to highlight *Epic* US and group US belonging to the scope of an Epic one (so sharing a same Theme). Indeed, after the graphical mapping of elements made within the previous section, the remaining relevant question is to determine *how to characterize an Epic US and determine US relating to the same Theme on the basis of a rationale analysis?*

Intuitively, we envisage the *Epic* US as the ones containing elements at the highest level of the hierarchy of a decomposition model. *Capabilities* can thus not be considered for possible inclusion in the top-level elements category because they should be expressed as role decisions on a very low (atomic) level. Similarly, *Soft-goals* are by nature non-functional so that they are also not included in the category. Relevant top-level elements can thus be *Hard-goals* or *Tasks*. *Hard-goals* are abstract and need to be operationalized so that we choose to focus on elements with a concrete realization scenario and they will not be considered for being *Epic* US. Only the *Task* concept is then remaining and we define a **top-level *Task* as a *Task* element that is not issued of the refinement of another *Task* element but that itself needs to be refined in more elements**. Decomposition complexity and possible finer grained hierarchization of elements will be further discussed in Section VI.

Different possible cases for *Epic* US and *Theme* US identification are discussed in the rest of this section.

A. A Top Level Hard-goal (End), One Mean

1) Description: If, for a top-level *Hard-goal*, there is only one means-end decomposition (which represents a possible way of satisfying the *Hard-goal* through a *Task*), then the

US containing the *Task*[5] at the source of the means-end decomposition as *D_C* is an *Epic* US. The rest of the (lower level) elements in the scope of this means-end decomposition belong to US of the same *Theme*.

2) Example: Table I presents a set of US issued of the ClubCar application development. The scenario described in this section is represented in Figure 5 both in canonical form and instantiated to the US of Table I. The US including the *Task* "*Propose a ride from A to B with the price, location and time of departure, and number of seats available*" is thus an *Epic* US because it is the top-level *Task* issued of the means-end decomposition of the *Hard-goal* "*Propose a ride to go from A to B*". Moreover, the US containing all the elements refining the top-level *Task* are part of the same *Theme*[6]. Also note that the *Capability* "*Select appropriate service*" was not initially present into the US set but has been added to ensure the consistency of the rationale analysis. Also note that the presence of the *Soft-goal* "*Rider satisfied of my driver service*" has no impact on the rest of the *Epic* and *Theme* identification process but its identification and representation can constitute a guidance for designers later on in the software engineering process (the use of softgoals in the software architecture and design is however outside the scope of this paper).

B. A Top Level Hard-goal (End), Several Means

1) Description: If, for a top-level *Hard-goal*, there are two or more means-end decompositions (which represent possible ways of satisfying the *Hard-goal* through *Tasks*), then the US containing the *Tasks* involved in the means-end decomposition as *D_C* are considered as *Epic* US. The rest of the (lower level) elements in the scope of each particular means-end decomposition belong to US of the same *Themes*.

2) Example: Table II presents a set of US issued of the ClubCar application development. The scenario described in this section is represented in Figure 6 both in canonical form and instantiated to the US of Table II. The US including the *Task* "*Pay by SMS in domestic country*" as well as the US including the *Task* "*Pay by credit card*" are thus *Epic* US because these are top-level *Tasks* issued of means-end decompositions of the *Hard-goal* "*Pay for the car pooling service in function of the country he is traveling in*". Moreover, the US containing all the elements refining these two top-level *Tasks* are part of the same *Themes* (so we have two *Themes* represented in Figure 6).

C. A Top Level Task, a Direct Decomposition

1) Description: For a top-level *Task* not linked with a *Hard-goal* through a means-end decomposition, the US containing this *Task* is considered as an *Epic* US. The rest of the (lower level) elements in the scope of this *Task* decomposition belong to US of the same *Theme*.

2) Example: This scenario is similar to scenario V-A but without an upper *Hard-goal* of which the *Task* represents the means-end analysis. Because of this similarity, it is not illustrated here.

VI. Impact on the Agile Software Process

This section studies the possible impact of integrating US analysis with the SR diagram into a US-based agile method.

A. Impact of Changing Requirements

The transformation approach from a set of US to the rationale diagram using the constructs of an SR diagram as presented in the paper is applied to a static set of US. Discovery and ability to deal with changing requirements is nevertheless one of the core willingness of agile methods. When requirements change US are adapted so that the rationale diagram is impacted. The impact of a change in a US on other requirements (US) can be studied on other US. In other words, the impact of a change of a US or several US can be studied on the rationale diagram by overviewing the links of changing US elements with other US elements. This cannot (or hardly) be achieved on the basis of the list of US only or with USM and could thus provide added value at this level. Consistency between the set of US and the rationale diagram is ensured by the use of a CASE-tool (see section VII).

B. Impact Iterative Planning

One of the main potential use of the rationale diagram built out the set of US is as input to the planning game[7]. Indeed, as evoked, building trees within the rationale diagram allows to distinguish Epic US and group US under a common Theme. Iterative and incremental development precisely requires such information for consistent iteration content planning.

Not all top-level *Tasks* found in an SR diagram should necessarily be the exclusive focus of one iteration. Indeed, an *Epic* US grouping a set of common *Theme* US could require to be treated in multiple iterations (or the opposite). This depends on the couple time/effort that can be/or is willing to be deployed on a single iteration:

- the type of iterative method to be used may vary. An agile method like *eXtreme Programming* (*XP*) [26] or Scrum [28] tends to iterate more times than a method like the Rational Unified Process (RUP[8]) [29], [30], [31] which includes 8 or 9 iterations at maximum in the whole project. Consequently, the time spent on an iteration may vary in function of the used methodology or life-cycle template;

[5]Note that the *Task* could potentially be present in different US in the WHY dimension. We refer here to the US in the WHAT one. It may be that we need to create the Epic US (thus with the *Task* in the WHAT dimension) because it can be that it is not expressed as such in the set of US of the project.

[6]In our approach, a US containing a *Capability* element (necessarily in the WHAT dimension, see Section III-C) can belong to different *Theme* groupings because these steps are relevant for the execution scenarios of different *Epic* US.

[7]The planning game is the process (in agile methods like XP) of selecting the requirements on which the development team will focus during an iteration and aligning these requirements with the available development capacity (see for example [26], [27]).

[8]Note that the RUP is not US driven but use-case driven. It is also strictly speaking not considered as an agile method. Its life cycle template could nevertheless be driven by US and used in an agile fashion.

TABLE I
US SAMPLE ISSUED OF THE CLUBCAR APPLICATION DEVELOPMENT

Dimension	*Element*	*D_C Type*
WHO	*As a DRIVER*	Role
WHAT	*I want to register to the service*	Task
WHY	*so that I can propose ride to go from A to B*	Hard-goal
WHO	*As a DRIVER*	Role
WHAT	*I want to propose a ride from A to B with the price location and time of departure, and number of seats available*	Task
WHO	*As a DRIVER*	Role
WHAT	*I want to log in to the platform*	Capability
WHY	*so that I can register to the service*	Task
WHO	*As a DRIVER*	Role
WHAT	*I want to select the ride characteristics*	Capability
WHO	*As a DRIVER*	Role
WHAT	*I want to confirm the proposal*	Capability
WHO	*As a DRIVER*	Role
WHAT	*I want the RIDER to be satisfied of my service*	Soft-goal

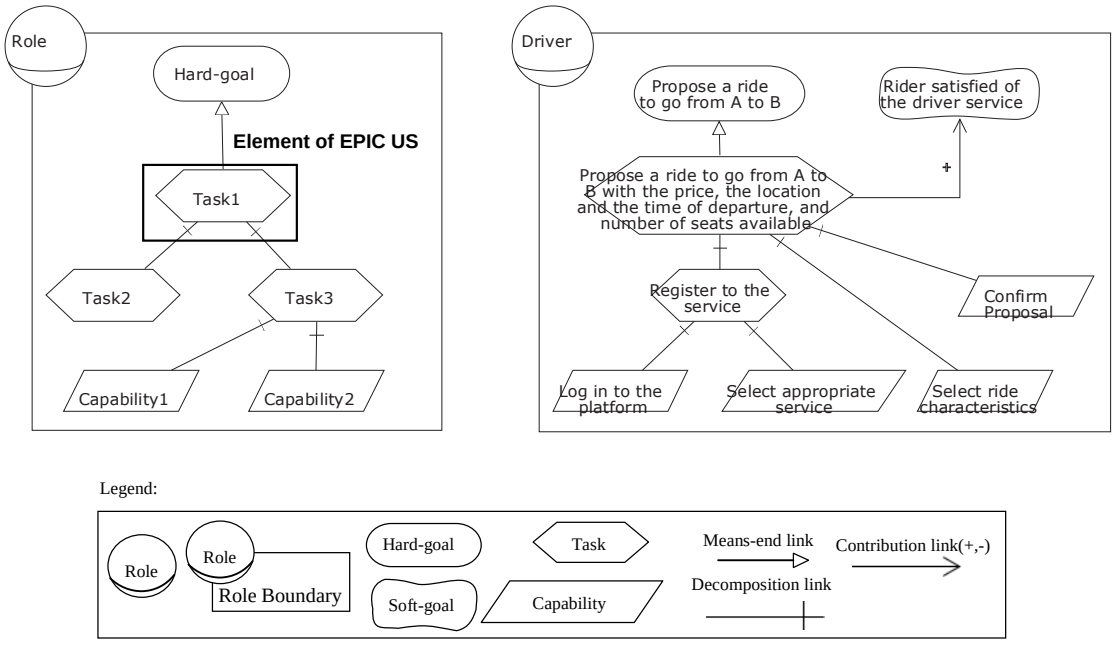


Fig. 5. Top-Level Hard-goal, One Means-End Decomposition.

- the amount of available resources to work on a single iteration may vary;
- the preferences of the software development team may vary leading to be willing to do more or less iterations in various amounts of time.

Calibration in the planning of *Epic* US in the fashion developed in the paper is thus required in function of the evoked parameters.

C. Generic Iterative Planning Template

With respect to the elements seen so far and the discussion that preceded in this section, we illustrate a possible decomposition of the i*-like rationale diagram for iterative planning. Figure 7 shows such a generic diagram in its canonical form. We refer to scope elements as elements that can be used as a basis for iterative planning. As shown in the picture, the scope element can be the entire EPIC US meaning that the entire theme should be prototyped/developed for the iteration or just a decomposed US meaning that we consider just part of the theme US elements for prototyping/development into that iteration and that the entire theme elements could be validated over multiple iterations. An approach to determine the right scope element could also be to evaluate the number of US points required for the development of the US containing the potential scope element.

Figure 7 presents a generic template that represents a possible approach for iterative content planning of the i*-like rationale diagram developed with the transformation method overviewed in this paper. Prioritization of Epic US for iterative planning can be done on discussion with stakeholders or with a structured approach like in [32], this however remains outside

TABLE II
US SAMPLE ISSUED OF THE CLUBCAR APPLICATION DEVELOPMENT

Dimension	*Element*	*D_C Type*
WHO	*As a RIDER*	Role
WHAT	*I want to pay for the car pooling service in function of the country I'm traveling in*	Hard-goal
WHO	*As a RIDER*	Role
WHAT	*I want to pay by credit card*	Task
WHO	*As a RIDER*	Role
WHAT	*I want to pay by sms in my domestic country*	Task
WHO	*As a RIDER*	Role
WHAT	*I want to be able to select the payment desiderata*	Capability
WHY	*so that I can pay by credit card*	Task
WHO	*As a RIDER*	Role
WHAT	*I want to log in to the platform*	Capability
WHO	*As a RIDER*	Role
WHAT	*I need to defined the amount and the driver*	Capability
WHY	*so that I can pay by SMS*	Task
WHO	*As the ClubCar Application*	Role
WHAT	*I want to send a confirmation SMS*	Capability
WHY	*so that when the payment by SMS has been performed*	Task

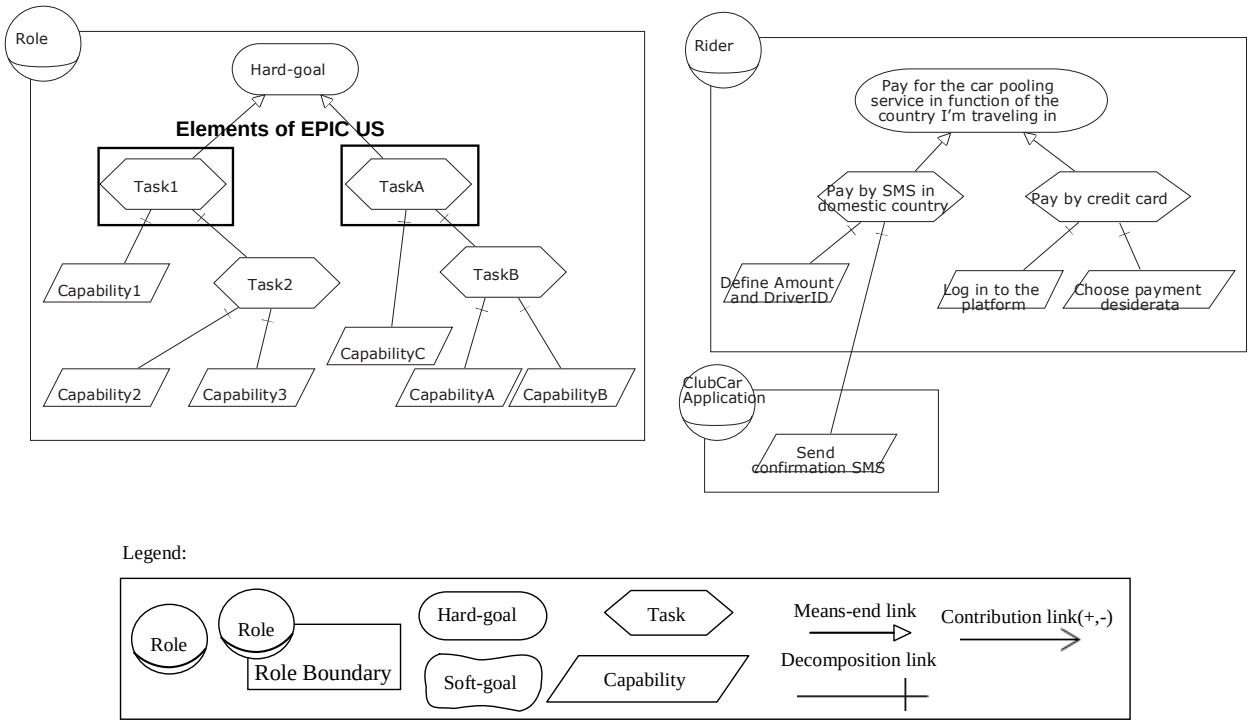


Fig. 6. Top-Level Hard-goal, Several Means-End Decompositions.

the scope of this paper.

VII. A CASE-TOOL FOR AUTOMATING THE APPROACH AND ROUND-TRIPPING BETWEEN VIEWS

In order to support the edition of US sets on US cards as well as the rationale diagram, we have build an add-on to the cloud version of the Descartes Architect CASE-Tool [33] that, for the present purpose, allows three views:

- the *User Story View* (*USV*) to edit US through virtual US cards. Each US element in a dimension must be tagged with a concept of the unified model;
- the *Rationale View* (*RV*) to edit rationale diagrams following the specification made in this paper. The graphical elements can be automatically generated from the US defined in the USV; the modeler is then in charge of further editing of the links between elements. When changes are made to graphical elements in the RV, the elements are automatically updated in the USV and vice-versa. These do indeed form the same logical element represented in different views;
- the *Structural View* (*SV*) to edit a structural agent diagram. This agent architecture is outside the scope of this

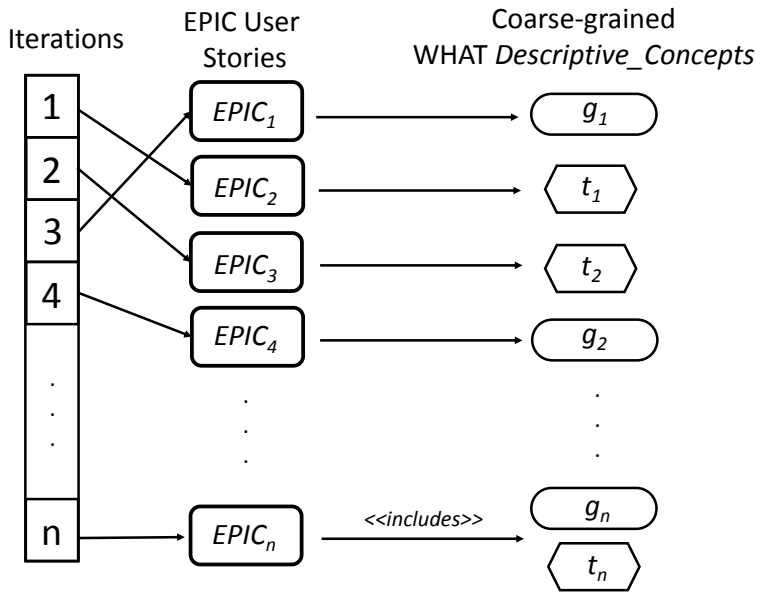


Fig. 7. Portfolio Optimization Problem

paper but is also integrated with US elements in the way defined in [34];
- next to this, we can also edit classical UML diagrams.

Once again, as a prerequisite, the set of US needs to be tagged to start the transformation and round-trip between the views. The editing process is continuous and intensive over the requirements analysis stage[9] (and to a certain extend over the entire project life cycle). This process is of course fully supported by the tool and leads to automatic updates of complementary views; consistency is ensured by separating the conceptual element in the CASE-tool memory from its representation in a view.

VIII. Validity, Threats to Validity, Scalability of the Approach and Future Work

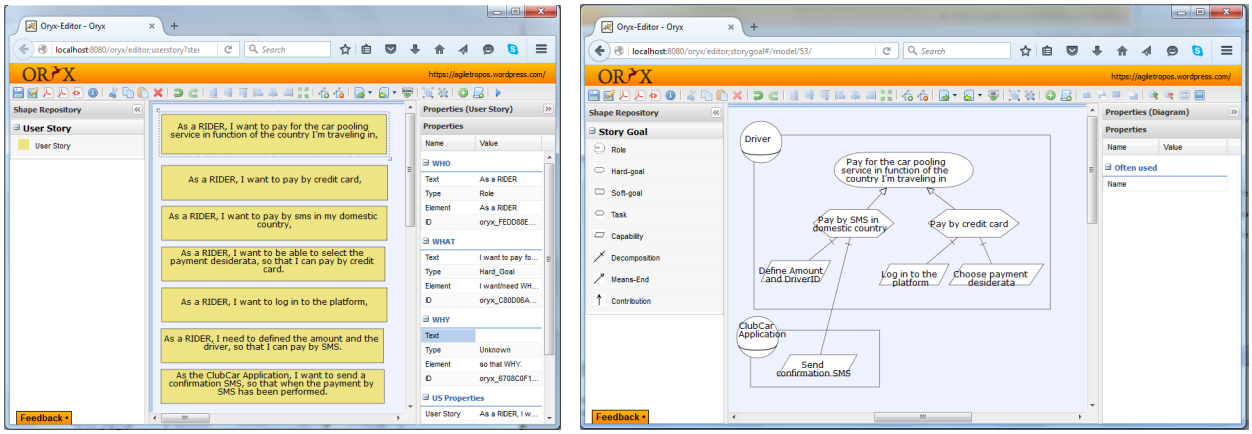
As already evoked, the prerequisite to the use of our approach is to tag the US when setting them up. Nevertheless, in terms of time, the investment is very limited; at maximum a few minutes per US, encoding them in the CASE-tool included. More time would then be necessary to create and edit the links between US elements in the RV. This is, however, similar to classical modeling. Moreover, we dispose of a clustering algorithm based on US syntax that allows to make clusters of relating US as a first step in the analysis process. This allows to start modeling on the basis of clusters of US that are rather similar which saves considerable time when compared to building rationale trees on the basis of a random organization of US. This allows a more consistent first basis for US elements structuring and grouping.

[9]In practice, during requirements analysis some US elements are "retagged" in an ad-hoc manner in later modeling stages. Indeed, the composition of the rationale model mostly leads to reconsider the nature of some elements (of which the granularity and structure was hard to determine when only seen in an isolated manner in the first stages) like in any modeling method.

A few threats to validity could also be evoked and should be clarified in later validation of the work:

- *Accuracy in US tagging could impact the relevancy of the RV*. A study on the perception of US elements' granularity using the unified model has been performed in [35]. The study distinguished different groups of users from students to software development professionals. The results were better with experienced modelers, but identifying granularity did not lead to major issues in any group with the condition that the set of US was taken as a consistent whole. This indeed allows to evaluate the relative links and hierarchy of US elements leading to adequate granularity identification and thus US elements tagging. As "stand alone" elements, granularity identification makes no sense and is nearly random;
- *The accuracy of the rationale diagram with respect to the system-to-be*. In order to asses the validity of the rationale diagram in the RV view, we will proceed to the following experience. At first, subjects (issued of 3 groups: researchers, students and business analysts) will be informed about a case and asked to carefully read and tag a set of US. At second, these same subjects will be asked to rank their perceived relevancy of 3 rationale diagrams:
  - Rationale diagram 1 generated and built from their own tagging of US set (so from the first part of the experiment);
  - Rationale diagram 2 generated and built by the tagging of the US set by two main researcher of this paper;
  - Rationale diagram 3 purely randomly generated and built out of the US set.

  The perceived relevancy/validity of the rationale diagrams



a. User Story View b. Rationale View

Fig. 8. The supporting CASE-Tool.

will then be evaluated by the subjects. Other metrics for the evaluation of rationale diagram will also be envisaged.

Future work also includes the application of the full validation of the technique on more real-life cases. We will notably proceed to a statistical analysis of the stakeholders perception of the relevancy and contribution of the rationale diagram for a project they have worked on. Also, they will be interviewed about the value of the use of the rationale diagram complimentary to the US sets in the agile requirements process.

The technique of transforming a set of US into an rationale diagram is virtually applicable to all sizes of projects. The complexity of the reasoning trees making part of the rationale diagram may then vary from project to project and the larger the number of US, the larger the modeling effort required. The tricky question of scalability can thus legitimately be posed. As evoked previously, our technique could be compared to USM; the latter is applied to projects of any size. Splitting US through their 3 dimensions will induce more modeling effort but using the CASE-Tool will save effort by adding flexibility in model management comparing to build a USM on a board or on the ground as it is the case for larger projects. The tradeoff of using our "transformation" technique with respect to USM must thus be balanced between a stronger US elements links' identification plus management using a CASE-Tool against a less formal approach where US are written on post-its and presented on a board or on the ground but are organized faster.

Above this, the question of agility may also be raised, i.e. *Are we not hampering agility?* The willingness is not to revolutionize an entire practice but to add a more formal representation of requirements to be able to better manage requirements across the whole development life cycle so with a positive impact on scalability. We argue that the larger the set of US, the most benefiting should be the rationale analysis because it proposes a structuring and consistent grouping of requirements possibly used as input for iteration content planning. That said, we do not believe that the building of the RV as an impact on the agility of the current process. Indeed, we suggest our approach as a side one (but constantly kept consistent with the US set if our CASE-Tool is used). We thus want to provide structure and complementary information that can be used in the fashion that the software development team wishes (not constraining).

In the illustrative example, the sets of US allowed to build a nearly perfect decomposition which is seldom the case in real-life case studies so that the domain analysis not only requires to build the links between the elements but also to possibly add some elements so to introduce new US leading (as already evoked) to a round-trip between the set of US and the rationale diagram rather than an unidirectional transformation process. We consider this as an advantage because it adds consistency to the entire requirements set.

Finally, further studies could be achieved in order to evaluate the value of representing roles' interaction in an advanced manner. We can for now only visualize that different roles are possibly involved in the fulfillment of a coarse-grained element and what their specific fine-grained contribution is. We could study the added value of the use of dependencies in the sense defined by the i* strategic dependency diagram.

IX. Conclusion

US are often expressed in a very operational manner leading, in huge projects, to an explosion of US' number, consistency issues and consequently an inherent difficulty to define their hierarchy and consistently plan iterations' content. Building a visual representation where US elements could be sorted, the links between them could be studied and consistent sets of US elements could be build could thus represent a useful guidance for the agile modeler and project manager.

Enhancing on a unified-model for US templates proposed in previous research, we have suggested in this paper to build a rationale model largely inspired by the i* SR diagram to dispose of such a graphical notation. Epic US contain top-level *Task* elements and interrelated elements are grouped around sets of *Theme* US. As an alternative to USM, this construction

allows a more accurate US decomposition analysis leading to an elimination of redundant elements as well as a more accurate study of changing requirements' impact. We also point to the use of consistent sets of US elements for iterative content planning.

To fully support the edition of models, an integrated CASE-Tool has been built. Thanks to its use a change in one view immediately updates the other views in order to keep the requirements analysis consistent.

Future validation on more projects will be done in the future, we will also carefully collect user feedback on the relevancy and validity of the approach.

REFERENCES

[1] M. Cohn, *Succeeding with Agile: Software Development Using Scrum*, 1st ed. Addison-Wesley Professional, 2009.

[2] Y. Wautelet, S. Heng, M. Kolp, and I. Mirbel, "Unifying and extending user story models," in *CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, ser. LNCS, M. Jarke, J. Mylopoulos, C. Quix, C. Rolland, Y. Manolopoulos, H. Mouratidis, and J. Horkoff, Eds., vol. 8484. Springer, 2014, pp. 211–225.

[3] E. Yu, P. Giorgini, N. Maiden, and J. Mylopoulos, *Social Modeling for Requirements Engineering*. MIT Press, 2011.

[4] A. Van Lamsweerde, *Requirements engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[5] OMG, "Business process model and notation (bpmn). version 2.0.1," Object Management Group, Tech. Rep., 2013.

[6] M. Glinz, "A glossary of requirements engineering terminology, version 1.4," 2012.

[7] A. van Lamsweerde, "Goal-oriented requirements enginering: A roundtrip from research to practice," in *12th IEEE International Conference on Requirements Engineering (RE 2004), 6-10 September 2004, Kyoto, Japan*. IEEE Computer Society, 2004, pp. 4–7. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/RE.2004.25

[8] E. Yu, *Modeling Strategic Relationships for Process Reengineering*. Cambridge, USA: MIT Press, 2011, ch. 1–2, pp. 1–153.

[9] E. S. K. Yu, "Social modeling and i*," in *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*, ser. Lecture Notes in Computer Science, A. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. K. Yu, Eds., vol. 5600. Springer, 2009, pp. 99–121.

[10] ——, "Towards modeling and reasoning support for early-phase requirements engineering," in *3rd IEEE International Symposium on Requirements Engineering (RE'97), January 5-8, 1997, Annapolis, MD, USA*. IEEE Computer Society, 1997, pp. 226–235.

[11] O. Liskin, R. Pham, S. Kiesling, and K. Schneider, "Why we need a granularity concept for user stories," in *Agile Processes in Software Engineering and Extreme Programming - 15th International Conference, XP 2014, Rome, Italy, May 26-30, 2014. Proceedings*, ser. LNBIP, G. Cantone and M. Marchesi, Eds., vol. 179. Springer, 2014, pp. 110–125.

[12] J. Patton and P. Economy, *User Story Mapping: Discover the Whole Story, Build the Right Product*, 1st ed. O'Reilly Media, Inc., 2014.

[13] M. P. K. Shergill and C. Scharff, "Developing multi-channel mobile solutions for a global audience: The case of a smarter energy solution," *SARNOFF'12, New Jersey*, 2012.

[14] J. R. Nawrocki, M. Ochodek, J. Jurkiewicz, S. Kopczynska, and B. Alchimowicz, "Agile requirements engineering: A research perspective." in *SOFSEM*. Springer, 2014, pp. 40–51.

[15] C. Larman and B. Vodde, *Practices for scaling lean & agile development: large, multisite, and offshore product development with large-scale Scrum*. Pearson Education, 2010.

[16] J. Patton and P. Economy, *User Story Mapping: Discover the Whole Story, Build the Right Product*. O'Reilly Media, Incorporated, 2014.

[17] Y. Wautelet, S. Heng, D. Hintea, M. Kolp, and S. Poelmans, "Uml transformation of user story sets," *Submitted to the 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016)*, 2016.

[18] K. Logue and K. McDaid, "Handling uncertainty in agile requirement prioritization and scheduling using statistical simulation," in *Agile, 2008. AGILE '08. Conference*, 2008, pp. 73 –82.

[19] M. Cohn, *User Stories Applied: For Agile Software Development*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2004.

[20] D. Leffingwell, *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional, 2010.

[21] J. Patton, "Finding the forest in the trees," in *OOPSLA Companion*, R. E. Johnson and R. P. Gabriel, Eds. ACM, 2005, pp. 266–274.

[22] S. Apel and C. Kästner, "An overview of feature-oriented software development," *Journal of Object Technology*, vol. 8, no. 5, pp. 49–84, 2009.

[23] J. Bosch, *Design and use of software architectures: adopting and evolving a product-line approach*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000.

[24] D. Leffingwell. (2016) Welcome to safe 4.0. [Online]. Available: http://scaledagileframework.com/welcome-to-safe-40/

[25] ——. (2015) Safe 4.0.: Capabilities and features. [Online]. Available: http://scaledagileframework.com/features-and-capabilities/

[26] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change (2Nd Edition)*. Addison-Wesley Professional, 2004.

[27] K. Auer and R. Miller, *Extreme programming applied*. Addison-Wesley Boston, 2002.

[28] K. S. Rubin, *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.

[29] IBM, *The Rational Unified Process, Version 7.0.1*, 2007.

[30] R. D. Gibbs, *Project Management with the IBM®Rational Unified Process®: Lessons From The Trenches*. IBM Press, 2006.

[31] P. Kruchten, "The rational unified process : An introduction," *Longman (Wokingham), Addison-Wesley, December*, 2003.

[32] Y. Wautelet, M. Kolp, and S. Poelmans, "Requirements-driven iterative project planning," in *Software and Data Technologies - 6th International Conference, ICSOFT 2011, Seville, Spain, July 18-21, 2011. Revised Selected Papers*, ser. Communications in Computer and Information Science, M. J. Escalona, J. Cordeiro, and B. Shishkov, Eds., vol. 303. Springer, 2011, pp. 121–135.

[33] (2016) The descartes architect case-tool. [Online]. Available: http://www.isys.ucl.ac.be/descartes/

[34] Y. Wautelet, S. Heng, M. Kolp, and C. Scharff, "Towards an agent-driven software architecture aligned with user stories," in *ICAART 2016 - Proceedings of the International Conference on Agents and Artificial Intelligence*, 2016.

[35] M. Velghe, "Requirements engineering in agile methods: Contributions on user story models," Master's thesis, KU Leuven, Belgium, 2015. [Online]. Available: http://www.isys.ucl.ac.be/descartes/ThesisMattijs.pdf