Ontology-Driven Audit using the REA-ontology

Graham Gal[1] [0000-0001--6526-9367] , Monique Snoeck[2][0000-0002-3822-3214] and Wim Laurier[3] [0000-0002-9448-248X]

[1] Isenberg School of Management, University of Massachusettes Amherst, MA 01003 USA
[2] Research Centre for Information Systems Engineering (LIRIS), KU Leuven, Leuven, Belgium
[3] NODES, Université Saint-Louis–Bruxelles, Brussels, Belgium

gfgal@isenberg.umass.edu
monique.snoeck@kuleuven.be
wim.laurier@usaintlouis.be

Abstract. While blockchains are not yet ubiquitous in business practice, they are expected to serve as a platform to handle an increasing number of business transactions in a not too distant future. Smart contracts can be used to code and to enforce agreements between business parties. A significant difference between traditional and smart contracts is that once the actual events of the smart contract become part of a block in the blockchain, they are almost impossible to undo. Therefore, it is important that critical validity aspects of these smart contracts are explicitly represented. As smart contracts are software products too, it is therefore also critical that the coding of these critical validity aspects guarantees a faithfull implemenation of the validity checks. This project combines two approaches (i.e. ontology engineering and model-driven engineering) applying them to the design and the implementation of smart contracts, in order to facilitate their audit through a clear separation of concerns. More precisely, this paper discusses the example of the REA ontology to provide the ontological commitment of the critical validity aspects of a contract, while MDE provides a tool to unambiguously translate the REA ontology's contracting terms into a well-designed Smart Contract. This paper argues how the resulting Smart Contract can support auditors' assertions regarding exchanges between business partners, and support the audit process.

Keywords: Software Audit, Model-Driven Engineering, Ontology, Smart contracts.

1 Introduction

In the audit of a company's financial statements the independent auditor conducts a substantive audit of account balances and a test of controls over the processes that create these balances [1]. Auditing Statement (AS) 5 requires two types of control reviews [2]. First is the design of the controls (para. 42 – 43) and second is their operational effectiveness (para. 44-45). For certain types of processes, such as computer mediated processes, the testing of the design becomes crucial as deploying flawed software can result in ubiquitous errors accross all of a business's operations. Various software

testing regimes discuss the generation and use of test cases [3–5] . As software increases in size and complexity model-based testing has been suggested [6]. For applications which create transactions to be mined into blockchains this approach has some promise; particularly for the audit of these blockchain applications as once a transaction has been mined it becomes an immutable part of the chain. This presents a problem for companies and auditors as incorrect transactions cannot be "backed out" and corrected. The use of smart contracts to create these immutable transactions becomes even more problematic as inconsistences or errors within a smart contract could result in many incorrect transactions. What is needed is an approach that allows for an ex-ante audit of smart-contracts' validity at run-time, as ex-post audits seem to lose their rationale in a world where even substantial errors are almost impossible to reverse.

This paper proposes that a particular model, the resource-event-agent (REA) ontology [7], be used to design smart contracts, and thus support the auditor's requirement to test and confirm the design of controls. Additionally, it argues that model-driven engineering (MDE) could allow auditors to assess the design quality (i.e. ontological commitment) of smart contracts that operationalize the REA and other ontologies.

Section 2 offers an introduction to ontology-aware MDE, with an introduction to the MDE development chain, an introduction on ontology-engineering a platform-indepenent model, and the need for generating ontology-aware smart contracts. Section 3 describes a concrete realisation as a combination of the MERODE MDE methodology and the REA-ontology, while discussing their combined potential. Section 4 concludes this paper, with the support this combination can offer auditors at three different levels.

2 Model-driven Engineering

2.1 The MDE Development Chain

In model-driven engineering (MDE), software is created from models rather than hand-coded. The development chains start with a computation-independent model (CIM), which is *"a domain model developed by domain experts that does not show the details of the structure of the system"*[8]. A platform-independent model (PIM) is derived from this CIM, showing the structure of the system independent of the peculiarities of a specific computer platform. The PIM is the result of the analysis phase, which follows the requirements elicitation phase. Although requirements engineering [9] is a discipline in its own right, requirements are considered as given and hence outside the scope of this paper. Here, we assume that the PIM results from the assembly of domain and task ontologies into an application ontology, in which the application ontology components are determined by the requirements, as first described by Guarino [10]. Where informations systems are generally tailored to a specific audience or application and thus require discovering specific requirements, we believe that in public chains requirements are of lesser importance to smart contracts as the requirements for smart contracts should be as generic as possible and supported by the largest community possible given that they are generally accessible. In private chains, we assume that a traditional MDE

development chain could be applied, in which the analysis phase can still be supported by application ontologies.

In the further MDE development chain, the plaform-specific model (PSM) is derived from the PIM with a set of validated model-to-model transformation rules taking into account platform specificities. Subsequently, the PSM is transformed in code with a set of validated model-to-code transformation rules. Direct PIM-to-code transformations are possible as well.

2.2 Ontology Engineering a Platform Independent Model

Smart contracts (as with any other software artefact) need to reliably represent a relevant part of reality. Hence the domain model on which they are based needs to be a truthful representation of reality, preferably supported by an as large as possible community of domain experts. As it is our aim to use domain models as input for a formal transformation process they also need to be formal. The construction of such shared and formal domain models lays in the realm of ontology engineering, since ontologies are *"formal and explicit conceptualizations of a shared conceptualization"* [11, 12]. Like requirements engineering, ontology engineering is considered outside the scope of this paper as we consider ontologies a given.

As ontology engineering is a mostly human [13] and hence costly endeavour, and because they are also validated throught their use in practice, their reuse should be promoted through interoperability. Guarino [14] requires the formalizations of domain and task ontologies to be specializations of top-level ontologies to operationalize this interoperability and distinguish between valid and invalid combinations of domain and task ontology concepts in an application ontology. Top-level ontologies (e.g. UFO, SUMO) describe generic concepts (e.g. space and time). Domain and task ontologies (e.g. REA) capture domain knowledge about a specific domain (e.g. food) or task (e.g. sales) by specializing top-level ontology constructs. Application ontologies then combine and specialize domain and task ontology constructs to form an application-specific ontology (e.g. catering, which is food sales), while respecting the logic of the domain, task and top-level ontologies they specialize.

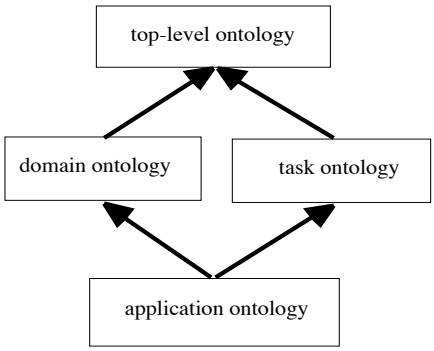

Fig. 1. Guarino's [14] ontology Aufbau principle.

As requirements engineering is an integral part of the ontology engineering process [15, 16], ontology engineers deliver and maintain ontologies that formalize expert domain knowledge. Hence, an auditor can rely on the expertise and reputation of the ontology engineer to cover the requirements engineering and analysis phase of smart-contract (and software) design that result in a CIM (i.e. the ontology) supported by a community of experts. Additionally, the auditor can rely on the respect of the formal combinatory rules coined by Guarino [14] to evaluate the face validity of the constructed application ontology, as it is virtually impossible to design ontologies for every possible application.

2.3 Generating ontology-aware smart contracts

In order to guarantee that the semantics of the ontology are reliably represented by the smart contract (or any other software artefact), the ontology needs to be refined into a PIM, that is then transformed into code. The transformation rules need to be transparent and reproducible. As a programer's ways of transforming a PIM in code are not always transparent and human creativity might hamper reproducibility, it is from the perspective of the auditor better to entrust machines with this transformation, limiting the application of human creativity to the rigorous design and maintenance of a set of transformation rules that is validated and improved through repeated use.

As with the ontology, the auditor can rely on the expertise and reputation (e.g. track-record, certificates, brand) of the software engineer responsible for the design and maintenance of the code-generators and the transformation rules to evaluate the quality of the product. As the ontology engineering and PIM-to-code engineering are orthogonal (i.e. the PIM-to-code transformation is a linguistic instantiation, where the application ontology development is an ontological instantiation according to [8]), both engineers should be able to trace undesirable smart contract behaviour back to either the ontology or the PIM-to-code transformation. For example, when smart contracts that are generated using the same ontology but different code-generators all exhibit undesirable behavior, the origin of the defect must lay in the ontology (unless in the unlikely case that all code generators exhibit the exact same error), while if different contracts generated by means of the same PIM-to-code transformation exhibit undesirable behaviour, the origin of the defect must reside within the PIM-to-code transformation.

3 Generating REA-based smart contracts with MERODE

3.1 MERODE

MERODE [17] is an enterprise information systems engineering method focusing on domain modelling. It uses existence dependency graphs (i.e. a sub-language of UML class diagrams) and finite state machines to model business objects and their behaviour, complemented with an object-event table (i.e. a version of a CRUD matrix [18]) to model object interaction. The method has been formalized by means of process algebra [19, 20], which ensures the consistency of the existence dependency graphs and the finite state machines [21] before transforming them to code. MERODE-models are

stored as xml-files, and can be transformed to working Java-applications in just three clicks. The transformations have been designed so as to support the transformation of high-level models by means of built-in defaults and PIM-to-PSM transformation rules [22], thus freeing the business analysts from the need of providing lots of details prior to code generation. Code-generation has proved a powerful instrument to validate models through fast prototyping [23]. While the default code generator can be used to validate an ontology (i.e. as *"a formal and explicit specification of a shared conceptualization"* [11, 12]) as has been demonstrated by the authors of [24], the generation of Smart Contracts using Blockchain technology requires and extention of the MERODE method. In [25], the authors develop B-MERODE as an artefact-centric approach to smart contracts. B-MERODE adds an extension to address permissions and to distinguish between the model aspects that result in code on and off the blockchain [26].

3.2 The REA ontology

The REA Ontology [7] is a domain ontology which specifies the objects and relations for the accounting and business domain. Fig. 2 provides a graphical representation of the core concepts of the ontology. The ontology is organized around three conceptual layers. The top-layer, which is colorcoded yellow in REA models, incorporates a description of what could or should be. The classes in this layer are "types". These are similar to Plato's forms [27]. The middle or contracting layer, which is colorcoded red in REA models, includes classes for contracts and their bundled commitments. For example, a contract to build a house includes commitements to pour the foundation, frame the house, put on the roof, etc. while the reciprocal commitments include a payment schedule. Finally, the bottom layer, which is colorcoded green in REA models, includes classes for what economic events have occurred, who was involved in them, and what resources were affected.

This section looks at how smart contracts can be modeled as bundles of Economic Commitments and how their correct execution can be verified by the independent auditor examining the semantic associations of the REA ontology (e.g. specify and typify). The typify association between the Economic Event Type and Economic Event classes in conjunction with the specify association between the Economic Commitment and Economic Event classes allows auditors to check wether the Economic Event that fulfils the Economic Commitment matches the type specified in the contract. For example, the contract to build a particular type of house (two bedroom) would include a commitment that specifies the Economic Resource Types (lumber, concrete, roofing shingles, etc.) that are to be included in the finished house. The typify association between the Economic Recourse Type and Economic Resources classes would allow for the verification that the acutual components of the house are of the type specified in the contract. The association between an Economic Commitment and the Economic Event Types would specify the steps that should be used to build the house while the Typify association between the Economic Event Types and Economic Events allows for the determination that the steps specified in the contract's commitment are actually followed. In addition to these specify associations,the stockflow association between the Economic Resources and Economic Events can allow auditors to check the materials

to build the house were used in the indicated stepts (Economic Events). Finally, the typify association between the Economic Agent Type and Economic Agent classes in conjunction with the specify association between the Economic Commitment and Economic Agent classes allows auditors to check wether the Economic Agent associated through a inside- or outsideParticipate association with the Economic Event that fulfils the Economic Commitment match the type specified in the contract. For example, before fulfilling the commitment to make the Economic Event Type of "final payment", the Economic Event Type – "final inspection" – must be completed by the Economic Agent Type of a "certified engineer." The buyer of the house could simply accept that the person actually performing the inspection Economic Event met these criteria, or the could verify this by asking for the inspector's credentials. In the development of a smart contract this could be accomplished with an oracle that examines the builder's human resources information system for example [28, 29]. This ability of a smart contract to use oracles as software connectors could allow the contract to verify that individuals fulfilling certain commitments are in fact of the type specified in the contract.

FIGURE 3-16
The REA Metamodel (M2) for All Temporal Layers

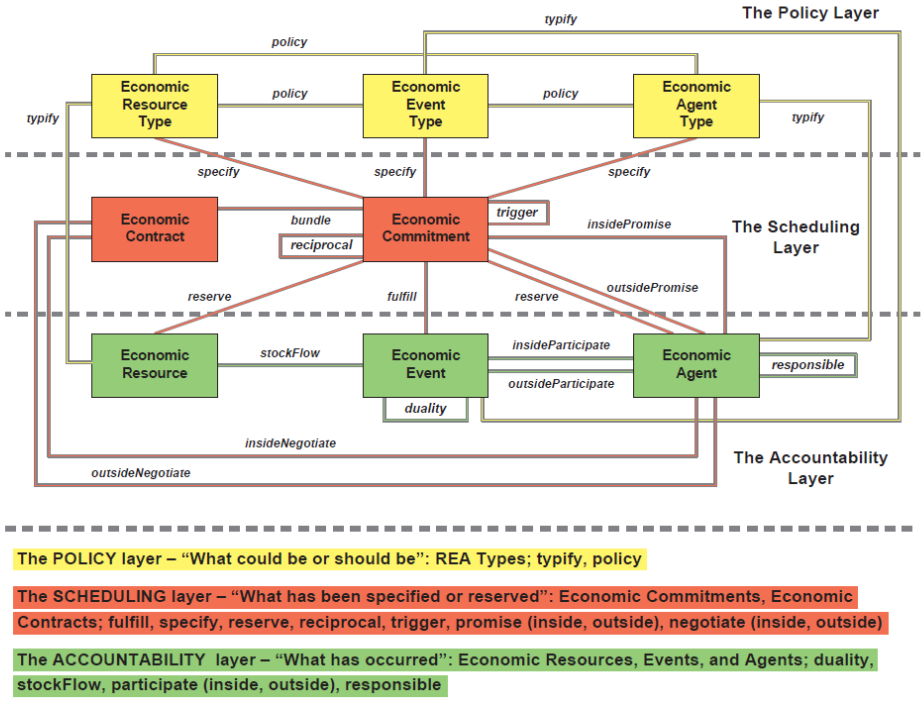

Fig. 2. - *REA Metamodel* [7)]

3.3 REA-aware smart contracts for economic networks

The authors of [24] published a proof-of-concept for the ontology-aware model-driven engineering approach advocated by this paper. They model the REA^2-ontology that is

equivalent to the green layer in Fig. 2 and an REA^2 interpretation of the REA axioms. They subsequently validate their model with a Java-application generated using MERODE's Merlin modeling tool and the associated code generator. Where the green layer in Fig. 2 is valid for a single trading-partner at a time, REA^2 is simultaneously valid for all stake-holder perspectives in a business transaction (i.e. buyer, seller and third-party (e.g. governing body)) [30].

The authors of [31] argue that REA-aware blockchains and smart contracts could improve resilience of and interoperability in economic networks, and especially in decentralized economic networks such as the peer-to-peer economy, through (1) an explicit representation of an agreed upon business vocabulary (i.e. the shared conceptualization in [11, 12]'s definition of ontology) that is an inherent characteristic of ontologies and indespensible for trade and (2) the use of the REA^2-ontology, of which the unified semantics are claimed to promote interoperability, throught the elimination of stovepipe architectures for the information systems, blockchains and smart contracts that operate economic networks. The economic networks considered in [31] are value networks - defined as any set of roles and interactions by [32] - of collaboration spaces in which people participate in both tangible and intangible exchanges of economic resources (i.e. goods, services and rights) to achieve economic or social benefit. The authors of [31] also list potential application of the ontology-aware model-driven engineering approach for blockchain development for the sharing economy as an example of external-facing value networks, collaboratives of independent workers as an example of internal-facing value networks, wood and vaccine traceability. Based on these examples, we can conclude that the ontology-aware model-driven engineering approach for blockchain development and particularly the REA^2-aware model-driven engineering approach for smart contract and blockchain development has considerable potential for audit in many domains of the traditional and the new economy.

4 Conclusion

In conclusion, this paper argues that the application of ontology-aware model-driven engineering practices has the potential to enable the ex-ante (i.e. that is before the are published) audit of smart contracts. Additionally, it argues that the REA ontology augments this potential with the ex-post (i.e. that is after the transation data have been recorded) audit of transaction data both on and off-blockchain resulting from the execution of these smart contracts. Finally, it argues that the design of REA-aware model-driven design has the potential to enable run-time monitoring of smart-contract execution (e.g. through the use of oracles).

First, ontology-aware model-driven engineering is expected to allow auditors to achieve a clear separation of concerns through the definition of clear deliverables (i.e. ontologies and code-generators) and a clear scoping of the tasks that lead to them. The auditor can rely on ontology engineers for the design of well-formed formal top-level, domain, task and application ontologies that truthfully represent a relevant part of reality through the application of ontology engineering methods. Subsequently, the auditor can rely on software engineers and their methods to design trustworthy code-generators

that transform these formal ontologies in smart contracts (and other types of software code) that truthfully reflect a relevant part of reality.

Second, the work of McCarthy et al. [7] as discussed above shows the potential of the REA-ontology for ex-post auditing the transaction data both on- and off-blockchain resulting from the execution of these smart contracts.

Third, the implementation of the REA axioms in [24] demonstrates the potential of the REA axioms and REA^2-aware MDE for monitoring the execution of smart-contracts at runtime.

References

1. Rittenberg, L.E., Bradley J. Schwieger: Auditing Concepts for a Changing Environment. Harcourt College Publishers, Orlando, Florida (2001).
2. Public Company Accounting Oversight Board: Auditing standard no. 5 – An audit of internal control over financial reporting that is integrated with an audit of financial statements. Exch. Organ. Behav. Teach. J. (2007).
3. Emam, S., Miller, J.: Test Case Prioritization using Extended Digraphs. ACM Trans. Softw. Eng. Methodol. 25, 1–41 (2015).
4. Demillo, R.A.: Software Testing, (2003).
5. Fraser, G.: Gamification of Software Testing. In: IEEE/ACM 12th International Workshop on Automation of Software Testing. pp. 2–7 (2017).
6. Hemmati, H., Arcuri, A., Briand, L.: Achieving scalable model-based testing through test case diversity. ACM Trans. Softw. Eng. Methodol. (2013). https://doi.org/10.1145/2430536.2430540.
7. McCarthy, W.E., Geerts, G.L., Gal, G.: The REA Ontology. (2021).
8. Gašević, D., Djurić, D., Devedžić, V.: Model driven engineering and ontology development. (2009). https://doi.org/10.1007/978-3-642-00282-3.
9. Dick, J., Hull, E., Jackson, K.: Requirements engineering. (2017). https://doi.org/10.1007/978-3-319-61073-3.
10. Guarino, N.: Understanding, building and using ontologies. Int. J. Hum. Comput. Stud. 46, 293–310 (1997). https://doi.org/10.1006/ijhc.1996.0091.
11. Guarino, N., Oberle, D., Staab, S.: What Is an Ontology? Handbook on Ontologies. In: Handbook on Ontologies SE - International Handbooks on Information Systems (2009).
12. Studer, R., Benjamins, V.R., Fensel, D.: Knowledge Engineering: Principles and methods. Data Knowl. Eng. 25, (1998). https://doi.org/10.1016/S0169-023X(97)00056-6.
13. Iqbal, R., Murad, M.A.A., Mustapha, A., Sharef, N.M.: An analysis of ontology engineering methodologies: A literature review. Res. J. Appl. Sci. Eng. Technol. 6, 2993–3000 (2013). https://doi.org/10.19026/rjaset.6.3684.
14. Guarino, N.: Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 139–170 (1997).

https://doi.org/10.1007/3-540-63438-x_8.
15. Al-Arfaj, A., Al-Salman, A.: Ontology Construction from Text: Challenges and Trends. Int. J. Artif. Intell. Expert Syst. 6, 15–26 (2015).
16. Cimiano, P., Völker, J., Studer, R.: Ontologies on demand? A description of the state-of-the-art, applications, challenges and trends for ontology learning from text. Information-wiss. und Prax. 57, 315–320 (2006).
17. Snoeck, M.: Enterprise Information Systems Engineering: The MERODE Approach. Springer (2014).
18. Martin, J.: Information engineering. Prentice Hall, Englewood Cliffs, N.J. (1989).
19. Snoeck, M., Dedene, G.: Existence dependency: The key to semantic integrity between structural and behavioral aspects of object types. IEEE Trans. Softw. Eng. 24, 233–251 (1998).
20. Dedene, G., Snoeck, M.: Formal deadlock elimination in an object oriented conceptual schema. Data Knowl. Eng. 15, 1–30 (1995).
21. Snoeck, M., Michiels, C., Dedene, G.: Consistency by Construction: The Case of MERODE. In: Jeusfeld, M. and Pastor, Ó. (eds.) Conceptual Modeling for Novel Application Domains SE - 11. pp. 105–117. Springer Berlin Heidelberg (2003).
22. Monsieur, G., Snoeck, M.: PIM to PSM transformations for an event driven architecture in an educational tool. Milestones, Model. Mappings Model. Archit. 55–63 (2006).
23. Sedrakyan, G., Snoeck, M., Poelmans, S.: Assessing the effectiveness of feedback enabled simulation in teaching conceptual modeling. Comput. Educ. 78, 367–382 (2014). https://doi.org/10.1016/j.compedu.2014.06.014.
24. Laurier, W., Horiuchi, S., Snoeck, M.: An executable axiomatization of the REA2 ontology. J. Inf. Syst. ISYS-19-026. (2021). https://doi.org/https://doi.org/10.2308/ISYS-19-026.
25. Amaral de Sousa, V., Burnay, C., Snoeck, M.: B-MERODE: A Model-Driven Engineering and Artifact-Centric Approach to Generate Blockchain-Based Information Systems. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-49435-3_8.
26. Scheynen, N.: Construction of web services using the MERODE approach, (2016).
27. Tarnas, R.: The passion of the western mind: Understanding the ideas that have shaped our world view. In: The Passion of the Western Mind (1991).
28. Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A.B., Chen, S.: The Blockchain as a Software Connector. In: 13th Working IEEE/IFIP Conference on Software Architecture (WICSA). pp. 182–191 (2016).
29. Chen, Y., Liu, J.: Distributed Community Detection Over Blockchain Networks Based on Structural Entropy. In: ACM International Symposium on Blockchain and Secure Critical Infrastructure. pp. 3–12 (2019).
30. Laurier, W., Kiehn, J., Polovina, S.: REA2: A unified formalisation of the resource-event-agent ontology. Appl. Ontol. 13, 201–224 (2018). https://doi.org/10.3233/AO-180198.

31. Laurier, W., Collet, R., Desguin, S., Fauconnier, B.: Ontology-aware Model-driven Architecture A Resource-Event-Agent implementation for the Blockchain. 日本情報経営学会誌. 41, 1–12 (2021).
32. Allee, V.: Value network analysis and value conversion of tangible and intangible assets. J. Intellect. Cap. 9, 5–24 (2008). https://doi.org/10.1108/14691930810845777.