Building an executable axiomatisation of the REA^2 ontology

Wim Laurier[1][0000−0002−9448−248X] and Satoshi Horiuchi[2]

[1] SMASH, Faculté ESPO, Université Saint-Louis - Bruxelles, Boulevard du jardin botanique 43, 1000 Bruxelles
wim.laurier@usaintlouis.be
https://www.researchgate.net/profile/Wim_Laurier

[2] Faculty of Commerce, CHUO University, 21225, 742-1 Higashinakano Hachioji-shi, Tokyo 192-0393 Japan
satoshi@tamacc.chuo-u.ac.jp
https://ir.c.chuo-u.ac.jp/researcher/profile/00010558.html?lang=en

Abstract. This project aims at producing an executable version of the Resource-Event-Agent ontology. This executable version is built with the MERODE methodology, which guarantees coherence between class and state diagrams. Where the MERODE CASE tool supports generating Java code from conceptual diagrams, our aim is to produce fully REA-compliant blockchain code with this methodology.
This project's main innovation is the use of existence dependency and event propagation to model increment and decrement as fundamental stand-alone concepts that simultaneously affect economic resources, event, agents and the nature of the stock-flows, participations and dualities between them.
The second – and most probably the most important – contribution of this paper is the formalisation of the REA axioms as executable finite state machines.
In the future, both of these innovations are believed to contribute to the reliability of a generic semantic blockchain technology for both the finance and logistics domain in both the traditional and the sharing economy. As such it is expected to promote both traceability and accountability in value networks and supply chains.

Keywords: REA ontology · MERODE · Blockchain · Model Driven Architecture

1 Introduction

1.1 Blockchain

Blockchain is – by some – believed to be the silver bullet to solve almost any issue in the world. The blockchain technology is particularly known for it's applications in finance (e.g. Bitcoin), although it's impact on the non-financial economy (e.g.

logistics) is also growing rapidly (e.g. Provenance[3]). Where blockchain has an impact in both finance and logistics, the full power of the technology has to the best of our knowledge not been realised in both domains simultaneously. What is needed is an approach that can link financial and logistical processes to each other, to promote transparency, fair trade, and fight money laundering, counterfeit and the funding of terrorism.

1.2 Introduction to REA & REA^2

The Resource-Event-Agent (REA) ontology [5] has the power to unite financial and non-financial value streams and has been shown to serve as a conceptual foundation for blockchain implementations. [2] REA has a conceptualisation for describing an organisation's metabolism (i.e. the value chain[6]) from the inside with it's dependent or trading-partner view and supply chains with its independent view [3], which is the perspective of a third party not taking part in an economic exchange. The REA^2 [4] ontology unites these two views demonstrating by means of a operationalised mapping that the three perspectives on an exchange (i.e. buyer, seller and third party) are – although semantically different – information equivalent.

1.3 Content of this paper

Section 2 introduces the methodology of this project, discussing the model mediated transformation of the ontology to blockchain code in subsection 2.1, and model coherence checking in subsection 2.2. The paper concludes with a summary of the contribution (i.e. subsection 3.1), an overview of the progress (i.e. subsection 3.2), and a discussion of the limitations (i.e. subsection 3.3) in section 3.

2 Methodology

2.1 Model-Driven Architecture

This project aims at building a truthful implementation of the REA ontology in blockchain technology. To this end a Model-Driven Architecture (MDA) [7] approach was used in which the REA ontology – and its formalisations [1] – serves as a computation independent model (CIM). This CIM is then implemented as a platform independent model (PIM), which is suitable for a particular implementation technology domain, while abstracting from the implementation details. This PIM is finally transformed in a platform specific model (PSM), which covers the implementation details in the technology of choice. In this project the PIM is a static conceptual model (i.e. existence dependency graph[4] (EDG)) complemented with a set of behavioural conceptual models (i.e. finite state machines

[3] `https://www.provenance.org`
[4] which is a formal subset of class diagrams

(FSM)). This projects considers these static and behavioural models to be a single PIM as the MERODE methodology explained in section 2.2 guarantees the consistency of the EDG and FSMs. Finally, the PIM is implemented as a platform specific model (PSM), which is suited for a chosen technological platform. The JMermaid[5] CASE[6] tool, which is used for building the MERODE compliant PIM in this project, is able to generate Java code, where Hyperledger Fabric[7], which has been chosen as the blockchain technology platform for this project, is used for building the PSM codes (i.e. Hyperledger Fabric code, composed of CTO, JavaScript, acl). In absence of a Hyperledger Fabric code generator in JMermaid, the Java code is used for validating the run-time behaviour of the model and Hyperledger Fabric code compliant with the Java code and is produced manually. Figure 1 summarises the project approach.

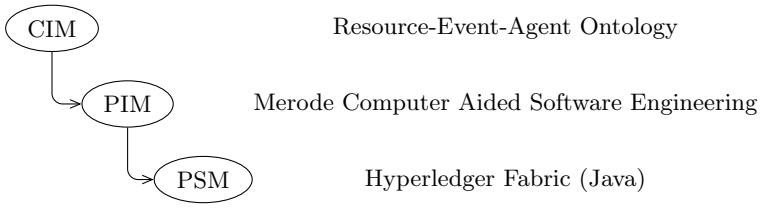

Fig. 1. Project approach CIM-PIM-PSM

2.2 MERODE

The MERODE methodology relies on the concept of existence dependency for its static conceptual model (i.e. the EDG).

"The existence-dependency relation is a partial ordering on objects and object types which is defined as follows: Let P and Q be object types. P is existence dependent on Q if and only if the life of each occurence p of type P is embedded in the life of one single and always the same occurence q of type Q. p is called the dependent object (P is the dependent object type) and is existence dependent on q, called the master object (Q is the master object type). "[8, p.83-84]

"A more informal way of defining existence dependency is as follows: If each object of a class P always is associated with minimum one, maximum one and always the same occurrence of class Q, then P is existence dependent on Q. In terms of life cycles, existence dependency means that the life of the existence-dependent object cannot start before the life of its master. Similarly, the life of an existence-dependent object ends at the latest at the same time that the life of its master ends."[8, p.83-84]

[5] `http://mermaid.econ.kuleuven.ac.be`
[6] Computer-Aided Software Engineering
[7] `https://www.hyperledger.org/projects/fabric`

For example, *"a hotel customer and a room object will have to be created before a reservation object by that customer for that room can be created, and reversely, due to referential integrity rules, the reservation object will have to be terminated before life of the customer or the room object can be ended."* [8, p.83]

Next to existence dependency, MERODE relies on event propagation to check the coherence of EDG and FSMs.

The event propagation rule implies that *if P is existence dependent on Q, then Q participates in each event in which P participates. In other words, each event marked for the dependent P must also be marked for the master Q. This can be explained as follows. When an existence-dependent object is involved in an event, its master objects are automatically involved in this event as well. For example, if a copy is created, the corresponding title is (implicitly) involved as well, as we now count one more copy for this title. Similarly, a state change of a loan, e.g. because of the return of the copy, automatically implies a state change of the related copy and member: the copy is back on shelf and the member has one copy less in loan. By marking each event type the dependent object type participates in also for the master object type, all implicit participations are made explicit, and as a result, all possible places for information gathering and constraint definition are identified. For example, the borrow method of the class MEMBER is the right place to update the number of copies a member has in loan and to check a rule such as a member can have at most five copies in loan at the same time. The borrow method of the class COPY is the right place to count the number of times a copy has been borrowed. At implementation time, methods that are empty because no relevant business rule or effect was identified can be removed to increase efficiency.* [8, p.115]

3 Conclusion

3.1 Contribution

This project aims at producing a platform independent model that formalises the REA ontology[8] and allows for the generation of executable code with the help of the MERODE methodology and JMermaid tool. This project's main contribution is the formalisation of the REA ontology using existence dependency and event propagation to expose increment, decrement and duality as the behavioural fundaments of the REA ontology. A second contribution of this paper is that it formalises the REA axioms as executable finite state machines.

3.2 Progress

This project was launched with a workshop at CHUO university (Japan) in March 2018 and continued with a workshop at universit Saint-Louis - Brussels (Belgium) in December 2018. The approach was developed at the workshop in Japan together with a first generation of PIMs (i.e. a set of coherent EDGs

[8] which served as a computation independent model

and FSMs) in JMermaid and a first generation of PSMs in Hyperledger Fabric. A second generation of PIMs was presented and discussed at the workshop in Brussels and a third generation of PIMs and a second generation of PSMs will be shown at the VMBO workshop in Stockholm.

3.3 Limitations

Currently, the JMermaid tool only allows for generating Java code. Consequently, the generated code was used to validate the EDG, OET and FSMs, testing the behaviour of the generated Java code. After this validation, blockchain code (i.e. Hyperledger Fabric) exhibiting the same behaviour was written manually. Full compliance with the REA literature still needs to be validated, and mappings with e3value and DEMO might be useful.

Acknowledgements

We would like to acknowledge Jun Azumi, and Satoshi Shimiza for their contribution at the workshop held at CHUO university (Japan), and Geert Poels, Monique Snoeck, and Graham Gal for their contribution at the workshop held at universit Saint-Louis Brussels (Belgium), and Hugues Dumont and Geert Poels for sponsoring the workshop in Belgium.

References

1. Gailly, F., Laurier, W. & Poels, G. (2008). Positioning and formalizing the REA enterprise ontology. Journal of Information Systems, 22(2), 219248. doi:10.2308/jis.2008.22.2.219.
2. Gal, G., & McCarthy, W. E. (2018). Implementation of REA Contracts as Blockchain Smart Contracts: An Exploratory Example.
3. ISO/IEC 15944-4:2015 Information technology – Business operational view – Part 4: Business transaction scenarios – Accounting and economic ontology
4. Laurier, W., Kiehn, J., & Polovina, S. (2018). REA 2: A unified formalisation of the Resource-Event-Agent ontology. Applied Ontology, vol. 13, no. 3, pp. 201-224, doi: 10.3233/AO-180198
5. McCarthy, W.E. (1982). The REA accounting model: A generalized framework for accounting systems in a shared data environment. Accounting Review, 57(3), 554578.
6. McCarthy, W.E. (2003). The REA modeling approach to teaching accounting information systems. Issues in Accounting Education, 18(4), 427441. doi:10.2308/iace.2003.18.4.427.
7. Osis, J., Asnina, E., & Grave, A. (2007, April). Formal Computation Independent Model of the Problem Domain within the MDA. In ISIM.
8. Snoeck, Monique (2014). Enterprise Information Systems Engineering, The MERODE Approach. The Enterprise Engineering Series, 280p. ISSN: 1867-8920, ISBN: 978-3-319-10144-6, doi: 10.1007/978-3-319-10145-3