A Domain Specific Modeling Language for REA

D. Mayrhofer[1], C. Sonnenberg[2], B. Hofreiter[2], C. Huemer[1]

[1]TU Vienna, [2]University of Liechtenstein
[1]{last}@big.tuwien.ac.at, [2]{first.last}@hochschule.li

1 Introduction

The Resource-Event-Agent (REA) ontology has its roots in the accounting discipline and was originally developed as a reference framework to conceptualize economic phenomena in an enterprise. In its proposal in 1982, McCarthy already had the vision to facilitate the design of data structures of accounting information systems by means of REA [1]. Since this time the REA model has been further extended and evolved into a domain ontology [2]. All REA concepts are based on well established concepts of the literature in economic theory - which is certainly one of the strengths of REA. However, REA has no dedicated representation format and, consequently, no graphical syntax. Thus, users may struggle when describing the REA models leading to the impression that REA is a rather heavy-weight approach. McCarthy feels (as he expressed during our joint work in standardization activities) that a dedicated graphical syntax - such as it exists for e3-value - may help in overcoming this problem and may lead to a much more significant adoption of REA. Accordingly, we have started the endeavor of developing a domain specific modeling language for REA. [1]

A domain-specific language (DSL) is a small, usually declarative language. It offers expressive power through appropriate notations and abstractions focused on – and usually restricted to – a particular problem domain [3]. Besides textual DSLs, we see an increasing interest in domain-specific modeling languages [4, 5]. According to [3] the benefits of a DSL approach are manifold: They allow solutions to be expressed at the level of abstraction of the problem domain. As a matter of fact, domain experts themselves can understand, validate and often modify DSL programs/models. The DSL programs/models are concise and self documenting to a large extent. They enhance productivity, reliability and maintainability. DSLs allow for validation and optimization at the domain level.

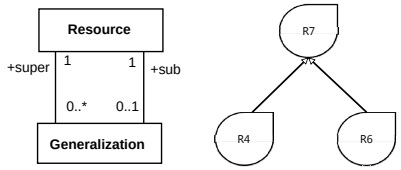
When developing our REA-DSL we followed methodological steps that have been suggested by Strembeck and Zdun for the systematic development of domain specific languages [6]. Amongst other variants, they describe the development process for extracting a DSL from an existing system, which is appropriate

[1] Note, a full version of this paper has been submitted to CAiSE 2011. William McCarthy encouraged us to submit an extended abstract as well to VMBO 2011 since he feels it provides the perfect audience to discuss the graphical syntax of the REA-DSL. We hope to present during VMBO not only the theoretical concepts, but also our modeling tool support. Furthermore, we want to discuss our thoughts on DSL features that go beyond the basic REA concepts.

for our needs, because we extract the DSL from the existing REA ontology. Accordingly, we started with (1) the identification of elements in the REA ontology. Next, we underwent a number of revision cycles of (2) deriving the abstract syntax of the REA model including the core language model and the language model constraints and (3) defining the DSL behavior, i.e. determining how the language elements of the DSL interact to produce the intended behavior. Once we had reached a stable state, we defined the DSL concrete syntax (4). Finally, we implemented a modeling tool support for the DSL (5), but we skipped the last step described in [6], i.e. integrating the DSL into a software platform, since REA stays at the platform independent level.

2 The REA DSL

We based the development of the REA DSL on The Object Management Group's(OMG) metamodeling architecture called Meta-Object Facility (MOF) [7]. MOF comes with a meta-meta model (M3 layer) that allows us to define the structure, i.e. the abstract syntax, of the REA DSL as a meta-model (M2 layer). The resulting REA DSL meta-model comprises three interlinked views: the *resource generalization hierarchy* (see Figure 1) , the *duality* (see Figure 2) and the *value chain* perspective (see Figure 3).

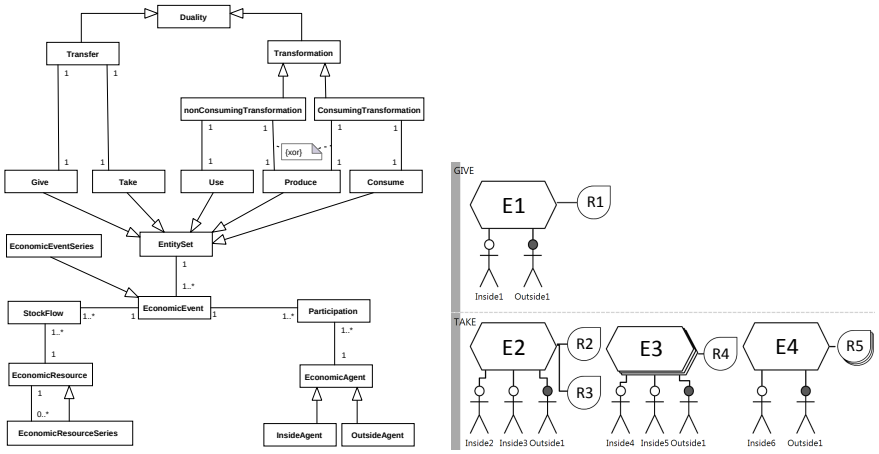


(a) Meta Model (b) Example

Fig. 1. Resource Generalization Hierarchy

We will gradually extend the REA-DSL. Currently, we are working on an integration of the REA policy infrastructure [8] covering commitments, agreements and, furthermore, the typification of the operational concepts [2]. Next, we plan to extend the REA-DSL by concepts to derive a database design for enterprise information systems, which has been one of the original goals of REA. For this purpose, we have already introduced the concepts of economic event series and economic resource series in the current DSL, because they will affect the multiplicities in the database design. By including the policy infrastructure and the REA-driven database design, it will become more obvious that REA models are of structural nature rather and do not concentrate on the behavioral aspects of process models.

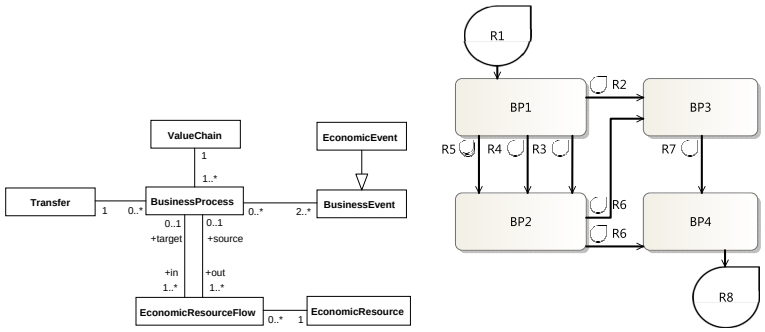
Another intended REA-DSL extension addresses the perspective from which the REA models are described. Currently, we focus on the perspective of a



(a) Meta Model (b) Example

Fig. 2. Duality

single enterprise which exchanges value with other enterprises. In the Open-edi reference model (ISO/IEC 15944-4) REA concepts are used to describe the exchanges of value among enterprises from a neutral observer's point of view. We plan to integrate the observer's perspective into our REA-DSL and to support semi-automatic mappings between the perspectives.



(a) Meta Model (b) Example

Fig. 3. Value Chain

3 REA DSL Tool Support

In the following we concentrate on the implementation of the graphical REA DSL tool based on *Microsoft's Visual Studio 2010 Visualization & Modeling*

SDK (V&M SDK). The *V&M SDK* enables the DSL designer to create the graphical definition of a DSL meta model. This meta model can automatically be integrated into a DSL designer, which allows to graphically define models on a conceptual level. Furthermore, the DSL designer can be integrated into a so called isolated shell, thereby creating a stand alone tool for our REA DSL. By the time of the workshop, we will allow a download of the REA DSL tool prototype as an isolated shell.

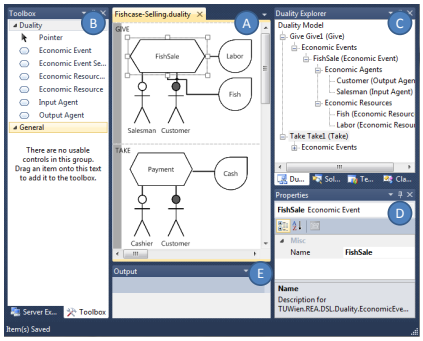


Fig. 4. REA-designer

The designer is separated into five major areas: The *modeling canvas* (A) the *toolbox* (B), the *solution explorer* (C), the *property window* (D) and the *validation window* (E). By dragging the modeling elements from the *toolbox* on the *modeling canvas*, a REA model can be assembled in a graphical representation. The *solution explorer* provides a tree based overview of the elements of the currently displayed model as well as a file and directory structure to hold different model instances. Properties of the selected model element can be changed in the *property window*. The *validation window* informs the user of any errors/warnings.

In the following we will explain the process of creating a REA model. Firstly, the user creates a *resource generalization hierarchy* model. The resources defined in this model will be made available to the *duality* and *value chain* model. Secondly, for each REA duality a *duality* model has to be created in a separated file. The user selects two appropriate swimlanes (e.g. *Give* and *Take*) and drops *event* or *event series* elements from the toolbox on the swimlane. For each event multiple agents can be connected. An agent can either be an *inside agent* or an *outside agent*. Additionally, the *resources* involved can be added to the *event*. A selection of the resources defined in the *resource generalization hierarchy* model ensures consistency between the models. When saving the model it gets validated and possible errors/warning will be shown in the *validation window*. After defining all dualities the user will create a *value chain* model. For each duality a process has to be created and resources flowing between the processes are added. Once more, by selecting the defined resources consistency is ensured between the models.

References

1. McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review **57**(3) (1982)
2. Geerts, G.L., McCarthy, W.E.: An ontological analysis of the economic primitives of the extended-rea enterprise information architecture. International Journal of Accounting Information Systems **3**(1) (2002) 1 – 16
3. van Deursen, A., Klint, P., Visser, J.: Domain-specific languages: an annotated bibliography. SIGPLAN Not. **35**(6) (2000) 26–36
4. Kelly, S., Tolvanen, J.P.: Domain-Specific Modeling. Wiley-IEEE Computer Society Press (2008)
5. Greenfield, J., Short, K., Cook, S., Kent, S.: Software Factories. John Wiley (2008)
6. Strembeck, M., Zdun, U.: An approach for the systematic development of domain-specific languages. Softw., Pract. Exper. **39**(15) (2009) 1253–1292
7. OMG: Meta Object Facility (MOF) Core Specification, Version 2.0 (January 2006)
8. Geerts, G.L., , McCarthy, W.E.: Policy-level specification in rea enterprise information systems. Journal of Information Systems **20**(2) (2006) 37–63