Defining Business Rules for REA based on Fragments and Declarations

Bernhard Wally, Alexandra Mazak, Dieter Mayrhofer, and Christian Huemer

Vienna University of Technology, Institute of Software Technology and Interactive Systems, Business Informatics Group, Favoritenstrasse 9–11, 1040 Vienna, Austria,
{wally, mazak, mayrhofer, huemer}@big.tuwien.ac.at,
http://www.big.tuwien.ac.at/

Abstract. Sophisticated business rule engines provide a way to enable users with a non-software-engineering background to take action during the runtime of e.g. an enterprise information system: rules for customers can be defined, based on the turnover realized through them. The REA accounting model defines a concept for the modeling and excecution of business models, yet it does not go into detail how to implement and integrate business rules. We propose the integration of a proven business rule engine in contrast to the definition of a REA specific rule language or model by defining a set of *anchor points* into which specific business rules can be hooked that are fired when these anchor points are reached.

Keywords: Business Rules, Business Ontology, Software Engineering

1 Introduction

Users of enterprise information systems often have a non-technical, non-sofware-engineering background [3], and they might change frequently. The former issue requires such systems need to provide ways to be configured by making use of either intuitive graphical languages or written languages that are close to natural languages. The latter requires rules to be defined, changed and removed at runtime, without the need for complex software engineering procedures, such as recompilations and redeployments. Common tasks performed by users of enterprise information systems include the classification of customers, employees or suppliers into different classes in order to grant them e.g. different benefits, based on classificators like yearly turnover, individual performance, etc. Business rule languages, algorithms and engines have been built in order to equip software applications with such functionality [6].

Business rules are closely related to decision models, i.e. business rules might provide decision logics to argue whether a certain business process should follow one or the other path. The recently adopted DMN standard [5] provides a graphical syntax and exchange format for such "decision models". In our work we do not currently use or support DMN, but it might represent a feasible way to define which factors influence certain business rules, and as such should be injected as data into the business rule engine.

Previous works have introduced methods for specifying business rules for REA [1,2], but an important aspect has been left out so far: how are these rules integrated, i.e. how does the system know when to call which rule? This facet represents an important issue for systems that are too large to keep all system entities in memory and run the rule engine continously. We are therefore proposing a set of "anchor points" that are bound to specific lify cycle changes of REA entities, which can be used to deposit business rules by hooking them into these anchor points. When that anchor point is entered, the corresponding business rules are fired.

2 Anchor Points for Business Rules

In [7] a business domain agnostic software architecture for REA is presented that provides a clear, extendible class contract for all REA entities. With that architecture it is possible the declare the *structural contract* of a specific business domain, but there is no mechanism specified to influence the *behavior* of the system. To tackle this issue, we are using that architecture and define anchor points that are bound to certain life cycle state changes of REA entities that can be used to dynamically hook business rules therein. These rules provide means for affecting the pre-defined application flow by interrupting entities' lifelines, and/or entering additional application paths.

The various entities of REA define common and specific anchor points for hooking in business rules with regards to the life cycle of each entity. Each rule can veto the current life cycle or property change of entities, in other words it can hinder the creation, modification or destruction of an entity. Rules can also interrupt the life cycle of other entities than the one where the rule was hooked into—in that case, the destruction of the corresponding entity is triggered (the rules that are hooked into the corresponding destruction anchor points are fired as they would under a "normal" trigger).

The following is a discussion over the various anchor points we have identified so far, starting with common and continuing with specific anchor points. The anchor points are presented in tables, as to keep them compact and easily comparable. The columns of the tables can vary depending on the type of anchor point, but the meaning is always the following: (i) *name* is a short descriptor of the anchor point, in order to quickly reference it, (ii) *synopsis* is the description of the temporal location within the life cycle of an entity or the overall system, (iii) *persistence status* informs whether the entity under inspection is already (or still) registered in the main system and persisted in the storage backend, (iv) *context metadata* explains which special meta data, if defined, is available for the rule[1], and (v) *examples* lists possible application scenarios for that specific anchor point.

Due to the clear class contract defined by fragments and declarations, rule authors can refer to attributes in a very intuitive way. Content assist modules can

[1] Each business rule can access the main system through a global variable, and thus query for any registered entity in order to perform advanced tasks.

use this class contract together with its meta information in order to provide the rule author with context sensitive information and code suggestion/completion methods.

2.1 System Anchor Points

We suggest the following system anchor points, i.e. anchor points that are not bound to a specific entity, but that are fired based on system events, or based on a global timer component. These are the most generic anchor points that do not relate to a specific entity; as such, the persistence status can be stripped from the anchor point definition, as only entities that are registered and persisted can be accessed from the rule context through the global variable mentioned in footnote 1.

Tbl. 1: System anchor point definitions.

Name	Synopsis	Context Metadata	Examples
`Timed Invocation`	○ Given a start date, an optional end date and an optional interval ○ Called by the system when it is about time	○ Reference to the main system	○ Regular maintenance work ○ Grouping of customers based on the generated turnover, every night ○ Yearly performance evaluation of employees ○ Periodic creation of reports
`Manual Invocation`	○ Invoked by hand	○ Reference to the main system	○ Creation of reports ○ Calculation of key performance indicators

2.2 Anchor Points for REA Entities

Anchor points that are bound to REA entities can be hooked into from three different contexts of the generic software architecture:

- **Declaration Context:** Entities can be equipped with business rules directly in the declaration context, i.e. when they are initially declared. Rules defined here apply to all instances of this declared entity, regardless of the context they are in. Only in this context, the anchor points `Before Creation` and `After Creation` are available, because in either of the other contexts, the entity must have been created already.
- **Constellation Context:** An entity in a specific position in a certain business case constellation is provided with business rules, e.g. the hardware resource in the packaging event of the sales value chain. Any instance that at runtime finds itself in the given context for that specific situation executes the business rules hooked into the corresponding anchor points.
- **Instance Context:** Single instances can be supplied business rules. It is clear from this definition, that only the specific instance is affected and that the rules are respected regardless of the context the entity is in.

REA entity anchor points can also be hooked into from various other (even entity specific) contexts, such as e.g. resources from within the *linkage pattern*

[4]: anchor points of resources can be equipped with rules that are only valid inside the linkage pattern context, e.g. when a resource is linked to another resource rules can be fired.

Common Entity Anchor Points We suggest the anchor points summarized in Tbl. 2 to be defined for any entity of the REA accounting model.

Tbl. 2: Anchor point definitions valid for all REA entities.

Name	Synopsis	Persistence Status	Context Metadata	Examples
`Before Creation`	Just before the entity is created or a typification is realized	○ Not persisted ○ Not registered in main system	○ The creator of the entity ○ The not-yet persisted entity bean	○ Modifying properties before introduction of the entity to the system
`After Creation`	Just after the entity has been created or a typification has been realized	○ Persisted ○ Registered in main system	○ Nothing special	○ Notification of other entities ○ Adding to specific groups
`Before Destruction`	Just before the entity is destroyed	○ Persisted ○ Registered in main system	○ Cause of destruction ○ The destructor of the entity	○ Notification of other entities ○ Transferring data to another entities
`After Destruction`	Just after the entity has been destroyed	○ Not persisted "alive" ○ Not registered in main system	○ The unpersisted entity bean	○ Notify other entities about the successful destruction
`Before Property Change`	Just before a property is changed	○ Persisted ○ Registered in main system	○ The old and the envisioned new value of the property	○ Data consistency checking
`After Property Change`	Just after a property has been changed	○ Persisted ○ Registered in main system	○ The old and the already persisted new value of the property	○ Notify other entities about the changed value
`Timed Entity Invocation`	○ Given a start date, an optional end date and an optional interval ○ Called by the system when it is about time	○ Persistance status not deterministic; must be queried at runtime	○ Reference to the entity	○ Notifications after some time or at a specific point in time
`Manual Entity Invocation`	○ Invoked by hand	○ Persisted ○ Registered in main system	○ Reference to the entity	○ Calculating entity specific indicators ○ Entity specific maintenance

Entity Specific Anchor Points We suggest the anchor points summarized in Tbl. 3 to be defined for specific entities of the REA accounting model in addition to the ones shown in Tbl. 2. As a rule of thumb, the anchor points of a specific entity correspond to the relations that this entity can establish.

3 Conclusion

We have presented a concept for a framework for the integration of business rules into the REA accounting model. Other than the specification and modeling of the rules themselves, we have considered the contextual location of these rules with respect to REA concepts. As such we frame the notion of *anchor points*, which are state changes in the lifeline of entities or timed events that are triggered from a scheduling service. Anchor points are contextualized in a way that allows defining rules that are valid only in specific situations. We consider business rules

Tbl. 3: Anchor point definitions for specific REA entities. This list is not to be regarded final, but should give an impression on possible anchor points.

Name	Synopsis	Context Metadata	Examples
Resources			
`On Resource Reservation`	Just after a resource has been reserved by a commitment	○ Nothing special	○ Order the resource from the supplier ○ Trigger a transfer of the resource from a central warehouse to a branch office
`On Stockflow`	Just after a resource has been assigned to an event	○ Nothing special	○ Check whether the resource should be reordered
`Before Linkage`	Just before a resource is linked to another one	○ Nothing special	○ Consistency checks (is the resource eligable for linkage?) ○ Check if the resource is already reserved
`After Linkage`	Just after a resource has been linked to another one	○ Nothing special	○ Changing the availability of the resource ○ Putting it into a specific group
Events			
`Interruption`	The life cycle of an event has forcibly come to an end	○ The cause of the interruption	○ Notification of entities about the premature life cycle ending ○ Performing clean up work
`Before Ending`	Just before the event is finished	○ The envisioned end date	○ Check for consistency with event goals
`After Ending`	Just after the event has ended	○ Nothing special	○ Notification of other entities about the successful destruction
Agents			
`On Agent Reservation`	Just after an agent has been assigned to an event	○ Nothing special	○ Notification of the agent about the reservation
`On Participation`	Just after an agent has been reserved by a commitment	○ Nothing special	○ Notification of a supervisor about the participation
Dualities			
`On Event`	Just after an event of this duality has ended	○ Nothing special	○ Run calculations on the balance/imbalance of the duality and create/update claims
`On Completed`	Just after the last expected event of this duality has ended	○ Nothing special	○ Notification of other entities about the new state of the duality
Commitments			
`On Fulfillment`	Just after the event that is responsible for the partial or complete fulfillment of this commitment has ended	○ The fulfillment grade of the commitment	○ Notification of other entities about the fulfillment state of the commitment
`On Realization`	Just after the commitment has been realized by a term which has been met	○ The fulfillment grade of the commitment	○ Notification of other entities about the fulfillment state of the commitment

to be defined in a rule engine specific syntax and executed in that rule engine on behalf of the REA runtime system. The required information for the rule engine to work properly is to be provided by our runtime system.

In a next step we will implement a prototype to adhere to the concepts presented in this paper in order to test the feasibility of this approach: (i) in terms of software engineering, (ii) in terms of usability, and (iii) in terms of performance (especially with regards to the conetxtualization of anchor points).

4 Acknowledgements

This work was supported as part of the BRIDGE program of the Austrian Research Promotion Agency (FFG) under grant number 841287—a joint research effort of Vienna University of Technology and eventus Marketingservice GmbH.

References

1. Andersen, J., Elsborg, E., Henglein, F., Simonsen, J.G., Stefansen, C.: Compositional specification of commercial contracts. International Journal on Software Tools for Technology Transfer 8(6), 485–516 (October 2006)

2. Gailly, F., Geerts, G.L.: Formal definition of business rules using REA business modeling language. In: 7th International Workshop on Value Modeling and Business Ontology (VMBO 2013), Proceedings (2013)

3. Gottesdiener, E., Consulting, E.: Business RULES show power, promise. Application Development Trends 4(3), 36–42 (1997)

4. Hrubỳ, P., Kiehn, J., Scheller, C.V.: Model-Driven Design using Business Patterns. Springer (2006)

5. Object Management Group, Inc.: Decision model and notation (DMN) specification (August 2013), `http://www.omg.org/spec/DMN/1.0/Alpha/`

6. Rosenberg, F., Dustdar, S.: Business rules integration in BPEL—a service-oriented approach. In: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC05) (2005)

7. Wally, B., Mazak, A., Mayrhofer, D., Huemer, C.: A generic REA software architecture based on fragments and declarations. In: 8th International Workshop on Value Modeling and Business Ontology (VMBO 2014) (March 2014)