A Model-driven Perspective on the Rule-based Specification of Services

Maria-Eugenia Iacob
University of Twente
m.e.iacob@utwente.nl

Henk Jonkers
BiZZdesign
h.jonkers@bizzdesign.nl

Abstract

The focus in this position paper is on business rules as a means to raise the level of abstraction (and automation) at which business logic is incorporated in model driven application design in the context of service oriented architectures. More specifically, next to providing a classification framework for business rules and investigating the existing standards and languages for the formal specification of business rules, we propose a model-driven framework for the rule-based design of services. We provide an example to illustrate this framework and to demonstrate the role business rules can play in the context of MDD of SOAs. Furthermore, we also explore, in terms of existing tool support, the extent to which, the model-driven design process can be complemented and combined with business rules written in nearly natural language, which can become, at the platform specific level, an executable way to specify business knowledge and decisions.

Keywords: model-driven architecture, business rules, service oriented architecture, service orchestration, business rule standards

1. Introduction

Service-oriented architecture (SOA) and the idea of "Software as a Service" are two current trends that begin to lead to a fundamental change in the way in which ICT applications are developed and used. The central idea is that instead of building or buying monolithic systems, in which the business logic is hard-coded, applications should be assembled in a flexible way, using well-defined software services that may be distributed over the internet.

However, this new way of building applications also requires a new way of approaching the development of reusable services and the composition of these services into end-user applications.

In this context, the model-driven development (MDD) paradigm [16] is of great relevance. However, although the number of practical applications of MDD is growing, the current state-of-the-art is that platform-specific code that is generated from platform-independent models is still incomplete: in most cases, code for specific business logic still has to be added manually. One of the reasons for this is that there is no suitable way to represent this business logic at the higher modelling layers. In this paper we argue that business rules (BR), aimed at these higher layers, are very well suited to fulfil this role. Thus, the general question we address in this position paper is how business rules can be incorporated in a MDD process as a means to raise the level of abstraction (and automation) at which business logic is integrated in application design in the context of SOA.

The paper is organised as follows: In Section 2 we briefly discuss the problems we address in this paper. In Section 3 we provide a classification framework for business rules and we investigate the existing standards and languages for the formal specification of business rules. In Section 4 we present our vision for combining business rules with model-driven design. The main goal is to analyse in terms of method, specification languages and tools the extent to which business rules can be combined with design models in all Model Driven Architecture (MDA) layers and become eventually, at the platform specific level, an executable way of specifying business knowledge and decisions. Finally, in Section 5 we draw some conclusions and point out future work.

2. Problem statement

In this section we will address the three main paradigms central in this research and research issues arising from their integration.

2.1. Model-driven development

In most traditional software application development practices, the ultimate product of the design process is "the realization", deployed on available realization platforms. In several model-driven approaches, however, intermediate models are reusable and are also considered final products of the design process. These models are carefully defined such that they abstract from details in platform technologies, and are therefore called computation-independent (CIMs) and platform-independent models (PIMs), in line with OMG's MDA [16][28]. MDA has emerged as a new approach for the *design and realisation of software* and has eventually evolved in a collection of standards that raise the level of abstraction at which software solutions are specified. Thus, MDA fosters a design process and tools that support the specification of software in languages such as UML rather than in languages such as Java.

The central idea is that design models at different levels of abstraction are derived from each other through model transformations. More specifically, different platform-specific models (PSMs) can be derived (semi-) automatically from the same platform-independent model, making use of information contained by a platform model. More recently, MDA has extended its focus to more business-oriented concepts and languages, reflecting the growing awareness that it is important to take into account business considerations in software development decisions. For this purpose, MDA has been extended with a CIM layer. Nevertheless, we believe that business-oriented concepts and languages may also have correspondents at the PIM level (such as, business process models describing the logical structure of the processes) and at the PSM level (describing the realisation and/or orchestration of processes in terms of, e.g., BPEL4WS, WSFL, XLANG, WSCI, and BPML specifications [1]). Also notice that UML, originally developed as a standard for software design, is probably not the most suitable language to express business-oriented models at the PIM level; specific business process modelling languages, such as BPMN [4], EPCs (implemented in ARIS) [26] or Amber (implemented in BiZZdesigner) [7] are better equipped for this purpose.

2.2. SOA and business rules

The central idea of SOA is that a service denotes the functionality that is relevant to the user of the service, without burdening the user with irrelevant details on how the service is implemented. SOA therefore holds the potential of allowing the development on-the-fly of flexible applications that can adapt rapidly to rapidly changing business needs by combining and reusing existing services. However, the technological state-of-the-art with respect to SOA (i.e., Web service technology [23]) so far only partly realizes the SOA potential. Design approaches incorporating the business view and with clear architectural guidelines are to a large extent still a subject of research.

One way to incorporate the business view in SOA is to express this view formally in terms of business rules and integrate it in the design and composition of services. Using business rules to achieve this has the advantage of allowing the *decoupling* of the business logic (expressed as business rules) from business operations, such as business processes and their supporting applications. Furthermore, the effects of rapid changes of the business logic (e.g., new laws and regulation, change of the internal business policy or a new business strategy etc.) can be thus isolated, affecting the business operations only to a limited and controllable extent (since business rules can be stored and maintained separately from process models). In this way, it becomes possible for organisations to explicitly manage and maintain business rules, which are no longer hidden and hard-coded in processes and applications [10], and to achieve higher business process and software agility. Also, such an approach would enhance the reuse of business rules.

In this paper we argue that business rules are very well positioned to be combined with or incorporated in the model-driven design of SOAs. The idea of combining business rules with SOA (in particular in relation to web service technology) has been already around for a while (e.g., [8], [21]). Currently, several commercial software platforms (e.g., the Oracle SOA suite, BEA Aqualogic, Web Methods etc.) support the use of business rules for *controlling services and the orchestration of services*. Thus, combining business rules with SOA is to some extent technically already possible. However the BR specification languages used by these tools are in most cases proprietary and have significant limitations. Furthermore, it should be noted that combining SOA and BRs is only possible at the platform-specific level, which does not yet fulfil the promise of SOA being an architectural style in which software design is driven by and fully aligned with the business needs. Fulfilling this promise would assume that the (partial) specification of both applications and business rules is possible independent of specific implementation platforms in an intuitively understandable manner, accessible to the primary user/creator of these specifications: the non-technical business person. The idea of developing means to specify business rules in nearly natural language is therefore essential. To raise the level of abstraction at which business rules are specified, the availability of a model-driven approach for business rules is a prerequisite.

2.3. MDD, SOA and business rules

Recently, the idea of applying the principles of model-driven design not only to software but also to business rules has captured the attention of standardisation bodies such as the OMG and W3C. Work is currently done to finalise standards for BR specification languages in all MDA layers of models (e.g., SBVR[18], RIF [25], PRR [19]). Furthermore, results have been reported with respect to the definition of model transformations between BR specification languages positioned in the different MDA abstraction layers (e.g., [22]). However, although the two model-driven approaches (for business rules and for software design) follow the same principle, they seem to evolve in parallel and somewhat independently from each other. In this paper we argue that they must be combined and that they will eventually converge into a model-driven approach for SOA in which business rules constitute the expression of business logic and through which the decoupling of the business logic from applications can be effectively achieved. Thus, the goal of this position paper is threefold: firstly, we aim to provide an overview regarding the theoretical and technological state-of-the-art in the areas of BR, SOA and MDD; secondly, we propose a framework for the integration of the three aforementioned approaches; and, thirdly, we outline some open research directions that emerge as a consequence of this integration.

3. Business rules

Business rules have received a lot of attention lately since they have been recognised as being the ideal vehicle for capturing and encapsulating business logic. One of the problems most commonly mentioned by practitioners is that any change in the business logic leads to complicated and costly software maintenance issues because the business logic is currently hard-coded in applications. This is why the idea that BRs would facilitate the separation between the business and the application logic, and thus, enhance their maintainability and agility, makes BRs very attractive for the software architecture community.

3.1. Definition, characteristics and classification

According to the Business Rules Group[1] a *business rule* can be defined as:
"A statement that defines or constrains some aspect of the business. It is intended to assert business structure, or to control or influence the behaviour of the business."

Business rules have (as indicated in [5]) a number of distinctive features that motivated us to put them at the foundation of this research:

- Business rules are "by and for business people, not IT people"
- Business rules must be specified in a *declarative* way in (almost) natural-languages accessible for the business audience.
- Business rules are *decoupled* from processes, procedures and applications.
- Business rules are *atomic*.

The above definition and characteristics of business rules cover a wide range of business rule types. We identify two main categories:

1. Rules that influence the operational process:
   - *Derivation rules* (deduction rules and computation rules) that are used to establish information that is used in a process.
   - *Action rules* that establish when certain activities should place. Two variants of action rules can be distinguished: condition-action rules (production rules) and event-condition-action (ECA) rules.
2. Constraints, which impose certain limitations to the structure, behaviour or information of an organisation or system:
   - *Structural constraints* (deontic assertions).
   - *State constraints*.
   - *Process constraints*.

Constraints can be either static or dynamic. Static constraints may lead to additional requirements for a design, while dynamic constraints can only be checked at 'runtime'.

Figure 1 positions the different types of business rules with respect to the architecture of an enterprise by indicating which of the architecture domains the respective rule types may partly capture.

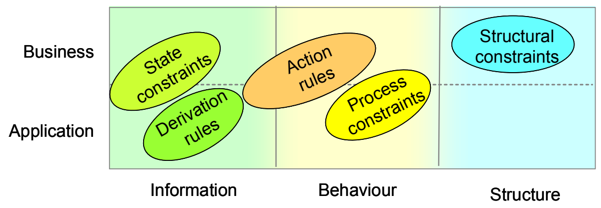


Figure 1. Classification of business rule types

To this purpose we have used a simplified version of the ArchiMate architecture framework [14].

[1] see http://www.businessrulesgroup.org/defnbrg.shtml

3.2. Business rules specification standards and languages

Several business rules specification standards, ranging from the higher business level to the execution level, are under development. There are two main standardisation streams [9]. The first one, driven primarily by the semantic web community and academia (and delivered via the W3C), is related to ontology standards and entails standards such as RuleML [26] and Semantic Web Rule Language (SWRL) [11] - the combination of OWL & RuleML. The second one, primarily driven by the business rules community and dominated by software suppliers and consultants, is working on standards, such as SBVR and PRR that are delivered via the OMG.

Since our research is mostly concerned with the relation between business rule specification and model-driven development, we discuss the second group of standards in somewhat more detail. However, for an exhaustive discussion of business rule and business-rule related standards we refer to [9].

Aimed at business users, the Object Management Group (OMG) proposes the standard for Semantics of Business Vocabulary and Business Rules (SBVR)[17]. This includes constructs to express business rules of different types in semi-natural language. Specifically for production rules, the Production Rule Representation (PRR) is under development [19]. For business constraints at the operational level, OMG's Object Constraint Language (OCL) is a possible candidate [20], although its notation might be too complex to meet the requirement that it is readable for all stakeholders. It is still very much an open question how these standards and languages can be derived from each other (although the OMG claims that a mapping from SBVR to PRR and OCL is possible, see [22]).

Most business rule management systems, however, do not currently support these standards and use their own proprietary rule languages. Figure 2 roughly positions a number of standards and languages in the layers in the Model Driven Architecture (MDA) framework.

4. Model-driven rule-based specification of services

As we said, business rules provide an easy-to-understand, yet executable way to specify business knowledge and decisions. However, it is unclear how business rules and different types of design models can be used in an integrated way. In this section we will explain how the integration of business rules in the model-driven design of service-oriented architectures can be achieved. Furthermore, in Section 4.1 and Section 4.2 we point out several ways in which business rules can intervene in service design and we address the integration of business rule specification languages with modelling languages. The model-driven and rule-based approach we are proposing is explained in Section 4.3 and illustrated by means of an example in Section 4.4. In Section 4.5 we investigate the tools that may support our vision and we identify some gaps and integration issues.

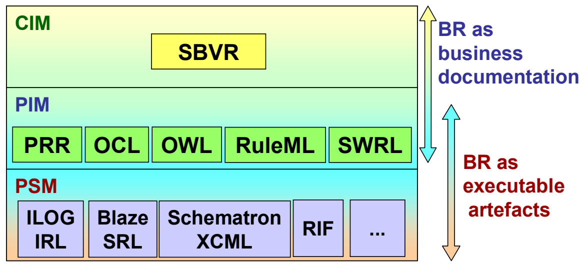


Figure 2. Business rules standards and languages in the different MDA layers

4.1. Types of integration

One obvious way to combine business rules and SOA (also embraced by several of the SOA platform vendors) is to use them for *controlling services* and for *the orchestration of services*. Orchestration languages (e.g., BPEL) contain conditional control structures that determine how the orchestration further unfolds (e.g., by invoking alternative services), based on information that is available at runtime. The condition associated with such a control structure in the orchestration definition can be captured in a business rule [6]. However, most orchestration languages have significant limitations with respect to their support of business rules. In order to resolve this shortcoming there are two options to be considered: (a) the integration of existing orchestration languages with business rules languages as well as of their supporting tools (e.g., BEA's Aqualogic with iLOG's JRules) and (b) the extension of orchestration languages with rule specification constructs. The first option can be achieved through a so called "service design and execution environment" in which tools supporting the design, analysis, visualisation and execution of both services and business rules are integrated. An example of such a service-oriented approach for the integration of a BPEL orchestration engine with a rule engine, using an enterprise service bus is proposed in [21]. A possible architecture of such an environment is depicted in Figure 3 and builds upon the solution we proposed in [2].

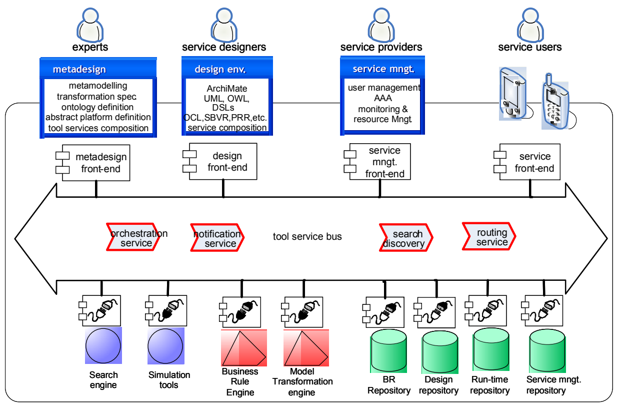


Figure 3. Architecture of a service design and execution environment

The second option should be resolved on a higher abstraction level through enhancement of existing specification languages. An example of this, in the case of BPEL, is the work done by BPMI.org for the development of the Business Process Extension Layers (BPXL) standard that could eventually complement BPEL by supporting transactions, business rules, task management, and human interactions.

Besides controlling the orchestration of services, one other way to use business rules in the context of SOA is to *provide and invoke them in the form of independent web services*. Thus, rule engines may expose the effect of business rules resulting in decision or derivation (web) services as depicted in Figure 4.

4.2. Relating non-functional aspects, business rules and service design

Current software development approaches have a strong focus on functional properties. Non-functional aspects, e.g., security, cost and QoS, are often added as an 'afterthought'. However, it becomes more and more accepted that they should be integral part of the development process, from global architectural descriptions to detailed specifications [12]. Once a design has been produced, performance problems, for example, can seldom be fixed by adding functions and generally the solution lies in redesign. In the context of SOA, where applications are generally a composition of distributed services involving multiple parties, issues such as service levels and security are particularly important, since quantitative aspects drive the actual SOA design: if there are several services that meet the functional requirements, the non-functional aspects usually determine the choice between the alternatives.

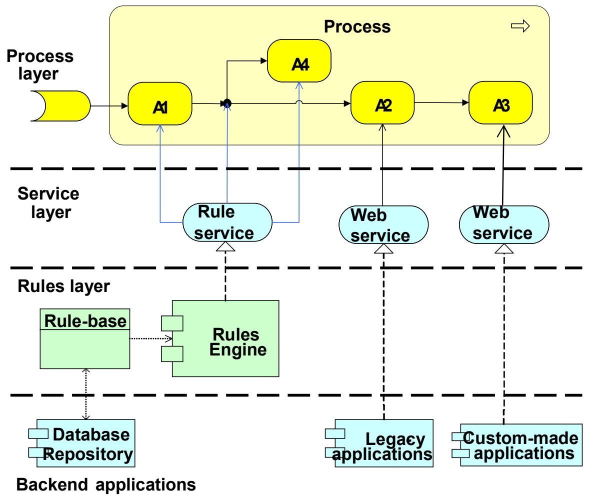


Figure 4. Business rules exposed as a service

At the technical level, a number of standards exist to specify quality attributes of services in Service Level Agreements, e.g., WSLA. However, there is still no consensus on the question whether services offering the same functionality, but at different quality levels, should be considered the same: e.g., does a service that sends a message with a delay of one second fundamentally differ from a service that sends the same message with a delay of two days? From the business point of view this type of differences are essential. At business level, "service levels" are derived from and constrained by business rules (expressed in natural or nearly natural language) that define the boundary conditions in which the business is supposed to function. It is natural to expect that business rules may play a similar role at the application level and that service level agreements may be derived from and constrained by formally expressed executable business rules. Thus, the following related issues are open to research: (1) to relate business rules to SLA specification and to investigate the extent to which this relation can be automated and (2) to raise the level of abstraction at which this relation is established at the PIM or even CIM level, which extends the idea addressed in [12] that analysis of non-functional properties is applicable at all levels, from high-level architectural descriptions to detailed designs in the context of the Model-Driven Architecture (MDA) paradigm for software development.

4.3. Model-driven rule-based design

The combination of MDA with SOA design is an area that has been extensively researched in the Freeband A-MUSE project (http://a-muse.freeband.nl), which has proposed and validated a design methodology in this sense [3]. As sketched already in the two previous paragraphs, new interesting areas of research emerge from the combination of the two aforementioned paradigms and business rules, which could reuse and extent the Freeband A-MUSE results. As we have shown, business rules may not just play a role in designing services and designing/controlling the orchestration of services, but they could also play a role in specifying and controlling the non-functional properties of the resulting composite service (e.g., performance). Furthermore, we argue that this should be possible throughout the whole stack of MDA models, from high level computation-independent models to platform-specific models. In MDA, model transformations play a central role. Transformations are used to maintain relationships between models at different abstraction levels in the MDA model stack. Typically, one of the languages from OMG's Query-View-Transformation (QVT) standard [17] is used as the language to specify these transformations. The left-hand side of Figure 6 (which is a "service-oriented" version of MDA) illustrates this.

As in these top-down transformations information is added (i.e., the lower-level models are refinements of the higher-level models), it is still unclear to what extent these transformations can be performed fully automatically.

In Figure 6 a distinction has been made between the *design space*, with models expressed in design languages such as UML, business process modelling languages or architectural description languages, and the *business rule space*, with rules expressed in special-purpose specification languages (see Section 3.2). The integration of design models and rule specifications can also be seen as a special type of (horizontal) model transformations – model merging [14].

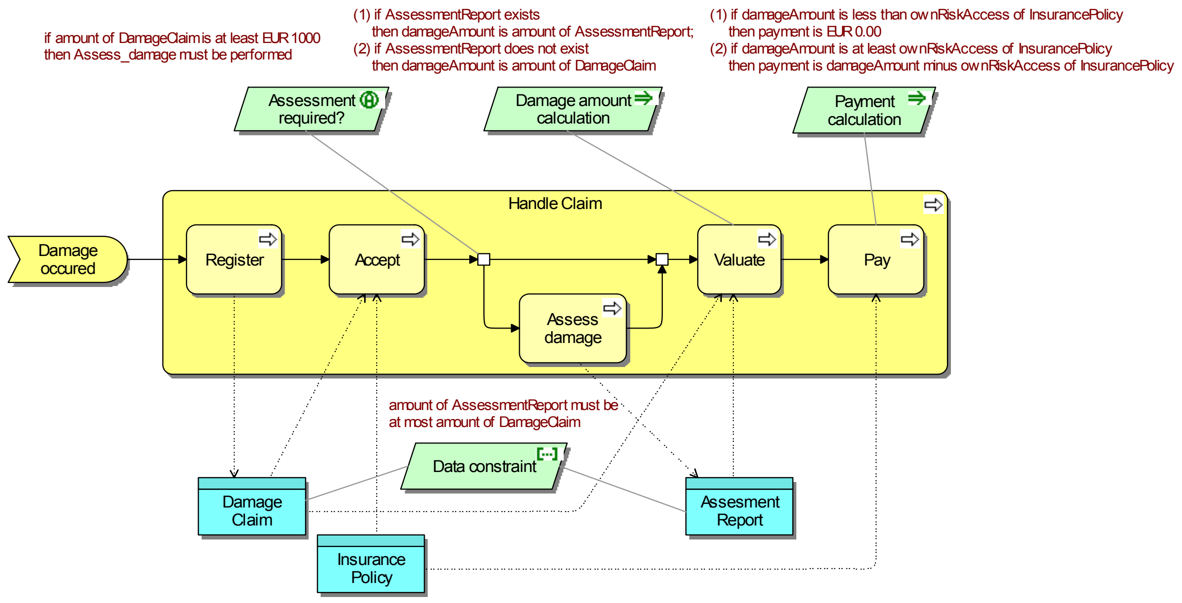

if amount of DamageClaim is at least EUR 1000
then Assess_damage must be performed
(1) if AssessmentReport exists
then damageAmount is amount of AssessmentReport;
(2) if AssessmentReport does not exist
then damageAmount is amount of DamageClaim
(1) if damageAmount is less than ownRiskAccess of InsurancePolicy
then payment is EUR 0.00
(2) if damageAmount is at least ownRiskAccess of InsurancePolicy
then payment is damageAmount minus ownRiskAccess of InsurancePolicy
Assessment required?
Damage amount calculation
Payment calculation
Handle Claim
Damage occured
Register
Accept
Assess damage
Valuate
Pay
amount of AssessmentReport must be at most amount of DamageClaim
Data constraint
Damage Claim
Insurance Policy
Assessment Report

Figure 5. Service architecture enhanced with BRs

As the Figure 6 suggests, there is a strong symmetry between the design space and the rule space: for any design model, there may be a corresponding rule set specification. Furthermore, a rule specified at a higher abstraction level may be refined (i.e., transformed) into a rule (set) specification at a lower abstraction level.

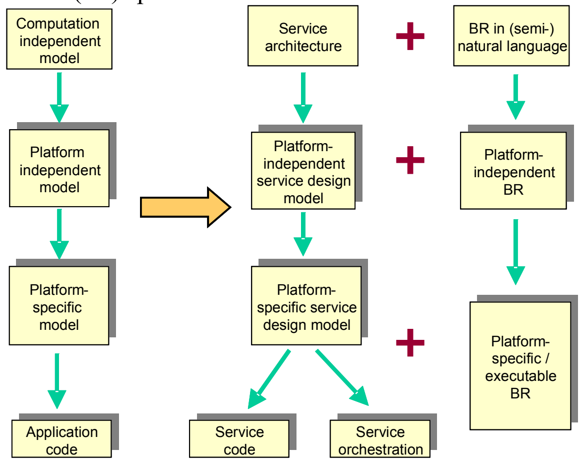

Computation independent model
Service architecture
BR in (semi-) natural language
Platform independent model
Platform-independent service design model
Platform-independent BR
Platform-specific model
Platform-specific service design model
Platform-specific / executable BR
Application code
Service code
Service orchestration

Figure 6. A model-driven view on the integration of service design enhanced and business rules

In summary, the following types of model transformations are relevant (see Figure 6):

- Vertical model-to-model and model-to-code transformations in the design space as identified in the MDA.
- Horizontal model merging transformations between design models and rule specifications,

either at the architectural, platform-independent or platform-specific level.

- Vertical transformations in the rule space going from (and refining) rules expressed in near-natural language to executable rule specifications.

At the PIM level the service architecture has been refined and transformed into:

- A behaviour model expressed in the BiZZdesign modelling language Amber [7] which has been annotated with rules specifications (see Figure 7)

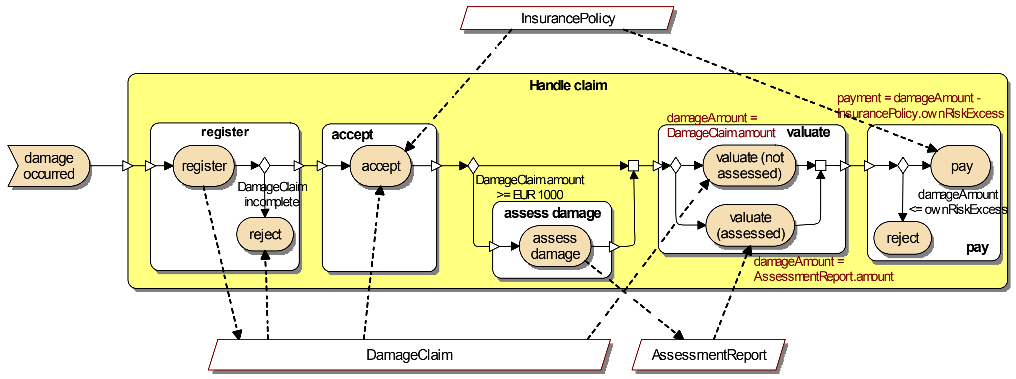


Figure 7. Rule-annotated behaviour

4.4. Example

In order to illustrate our vision we consider the example of a car damage claim handling service within an insurance company. At the CIM level a service architecture model is proposed (see Figure 5), using the ArchiMate design language [14] that has been supplemented with the business rule concept and specializations hereof (for a summary of the notation see Figure 8).

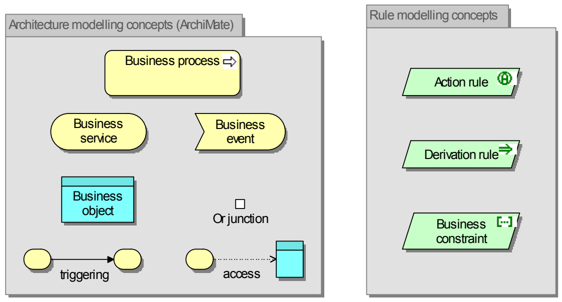


Figure 8 Selection of ArchiMate notation extended with BR concepts

At this level, business rules are specified in a near-natural language. Note that the chosen example contains both rules for controlling the orchestration of services (action rules, e.g., Assessment required?) and rules that can be implemented and used as independent rule services (derivation rules, e.g., Damage amount calculation, and constraints, e.g., Data constraint).

- A class diagram also annotated with a data constraint rule expressed as an OCL condition (see Figure 9).

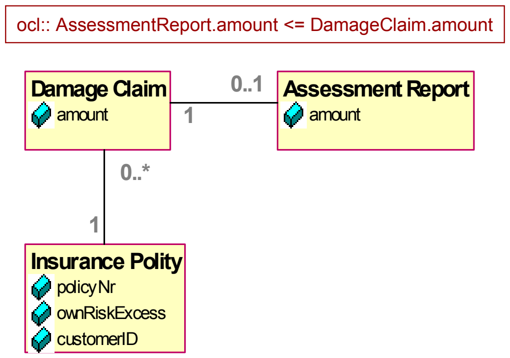


Figure 9. Rule-annotated class diagram

Finally, at the platform-specific level, the previous service design models can be transformed into the BPEL specification, a fragment of which is depicted in Figure 10 and realized using the Oracle SOA suite that integrates among others a BPEL engine with a BR authoring tool, engine and repository. Please note that the decisions points in the process have been externalised so-called decision services (e.g., “AssesmentDecisionServicePL”) wrapping business rules stored in dictionaries in a rule repository (e.g., for the abovementioned service see the “aboveLimit” rule depicted in Figure 11).

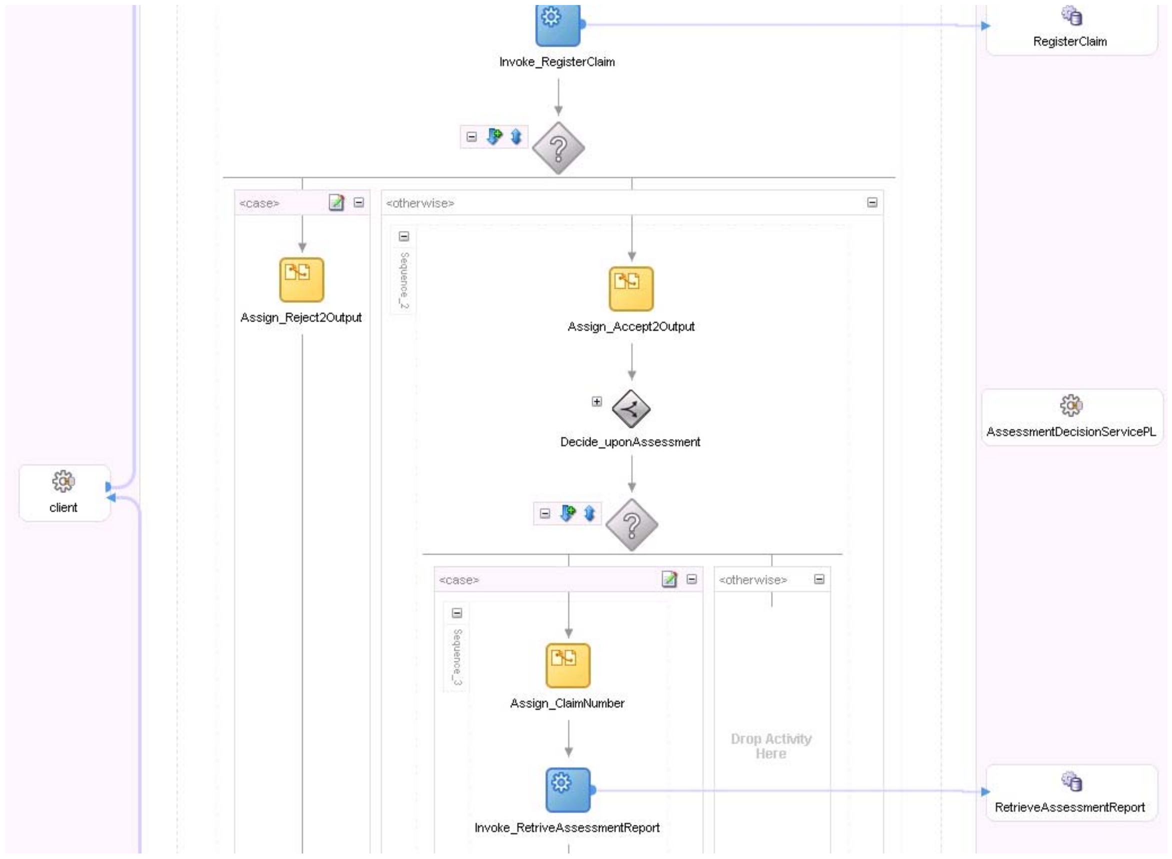


Figure 10. BPEL screenshot fragment

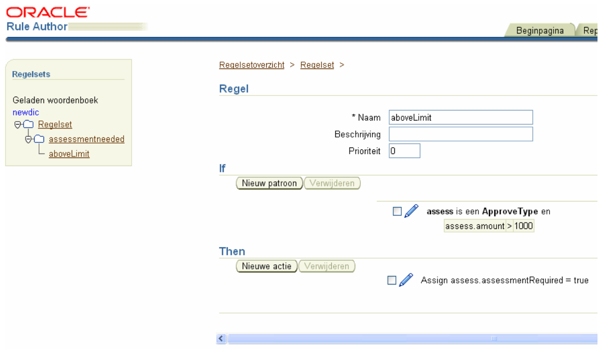
4.5. Gaps and integration issues in the tool support for model-driven service-oriented and rule-based design

It has been suggested in Section 4.1 that the vision presented in this paper could be achieved through a service design and execution environment in which tools supporting the model-driven design (i.e., modelling and model transformation specification) and execution of both services and business rules are integrated.

We have conducted a survey in order to explore the extent to which the existing technological state-of-the-art could support the realization of the presented approach. In the remainder of this section the main conclusions of this survey (also summarised in [13]) are briefly presented.

In the design space:

- For the design of CIM-level models several ***architecture modelling tools***, such as BiZZdesign Architect, Casewise, Metis etc. are available and could be used;
- Also the design of PIM-level models is fairly well covered by ***business process modelling*** (BPM) tools, such as BiZZdesigner, Aris, etc.


Rule Author

Beginpagina | Rep

Regelsets

Geladen woordenboek
newdic
Regelset
assessmentneeded
aboveLimit

Regelsetoverzicht > Regelset >

Regel

* Naam	aboveLimit
Beschrijving	
Prioriteit	0

If

Nieuw patroon | Verwijderen

assess is een **ApproveType** en
assess.amount > 1000

Then

Nieuwe actie | Verwijderen

Assign assess.assessmentRequired = true

Figure 11. Rule specified using the Rule Author

- At the PSM level several ***SOA development platforms*** that incorporate process/service orchestrations engines are also available on the market. Few examples hereof are Oracle SOA suite, Websphere, Cordys, etc.

- Finally, there are several so-called ***MDA tools*** covering both the PIM and PSM levels, which could be used to support the automated generation of code and the specification of model transformations: AndroMDA, OptimalJ, Arcstyler, IKV++ Medini, etc.

In the rule space most of the existing ***Business Rule Management Systems***, such as ILOG JRule, Corticon, MicrosoftBizTalk, Rule Burst, InRule, PegaRule, etc., can be positioned at the PSM level in our framework. An extensive survey of these tools led us to the following observations that are documented in [13]:

- most of these tools use proprietary BR specification languages;
- the BR specification standards positioned in CIM and PIM levels are not supported by tools, which might be explained by the fact that they have been just recently adopted or are under development;
- a lot of attention is paid recently to the integration of BRMS tools with SOA tools (e.g., Oracle Application Server or the combination Aqualogic - iLOG);
- there is almost no support for expressing rules in near-natural language;
- there is a strong focus on production rules.

When exploring the extent to which these tools could be integrated to build a comprehensive service design and execution environment we have identified the following gaps:

- although both BRMS and BPM tools are both capturing business knowledge, surprisingly enough very limited attention has been paid to their integration;
- integration between (a) MDA tools and SOA tools and (b) between BPM tools and SOA tools platforms is practically inexistent;
- and integration between modelling tools and BR specification tools/standards at the CIM and PIM level is extremely scarce.

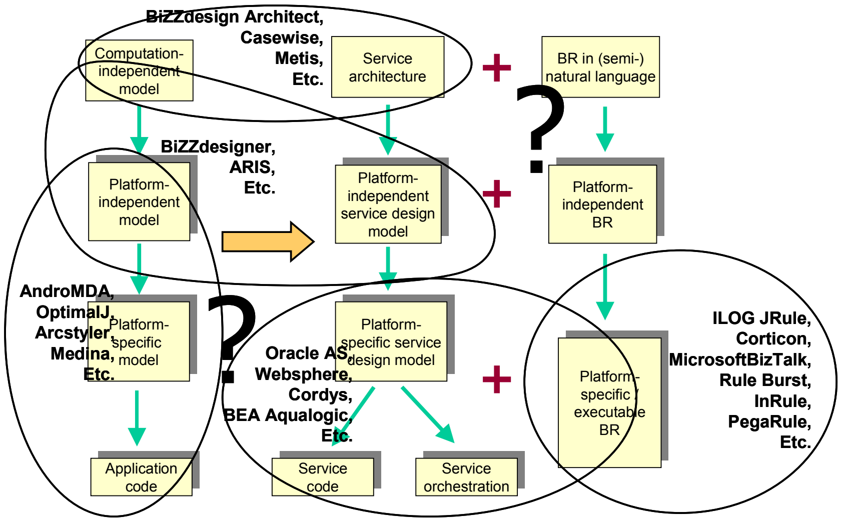

BiZZdesign Architect, Casewise, Metis, Etc.
Computation-independent model
Service architecture
+
BR in (semi-) natural language
?
BiZZdesigner, ARIS, Etc.
Platform-independent model
Platform-independent service design model
+
Platform-independent BR
AndroMDA, OptimalJ, Arcstyler, Medina, Etc.
Platform-specific model
?
Oracle AS, Websphere, Cordys, BEA Aqualogic, Etc.
Platform-specific service design model
+
Platform-specific executable BR
ILOG JRule, Corticon, MicrosoftBizTalk, Rule Burst, InRule, PegaRule, Etc.
Application code
Service code
Service orchestration

Figure 12. Tool coverage: gaps and integration issues

5. Conclusions

In this paper we have presented our vision on what is needed in order to be able to design service-oriented applications in a model-driven and rule-based way.

This research also pointed out a number of areas having significant research potential. In particular, we believe more work is needed in the area of integration between design languages and BR specification languages in all layers of the MDA stack. Furthermore, we argue that business rules could play a significant role in the specification and analysis of non-functional properties of services. Also, it has been suggested that current MDA, SOA, BRMS and other modelling tools can be used to partly support the model-driven approach presented in this paper. However, several integration issues and gaps have been identified that lead us to two important conclusions: software supporting the (model-driven) specification standards of business rules is still missing and integration between design and BR specifications has been only partly realised, and only at the platform-specific level. Finally, one important issue to be addressed is the extent to which automated model and BR transformations within and between the MDA layers are possible.

Acknowledgments

The work presented in this paper is part of the Freeband A-MUSE project (http://a-muse.freeband.nl), which is sponsored by the Dutch government under con-tract BSIK 03025.

References

[1] Aalst, W.M.P. van der , "Don't go with the flow: Web services composition standards exposed", *Trends & Controversies Jan/Feb 2003 issue of IEEE Intelligent Systems Web Services - Been there done that?*

[2] Almeida, J.P.A., Iacob, M.-E. Jonkers, H., Lankhorst, M., and Leeuwen, D. van, "An integrated model-driven service engineering environment", in *Proc. 2nd Interoperability for Enterprise Software and Applications Conferernce (I-ESA'06)*, Bordeaux, France, 22-24 March, 2006.

[3] Almeida, J.P.A., Iacob, M.-E., Jonkers, H. & Quartel, D., "Model-driven development of context-aware services", In: Frank Eliassen, Alberto Montresor (Eds.) *Distributed Applications and Interoperable Systems: 6th IFIP WG 6.1 International Conference*, DAIS 2006 Lecture Notes in Computer Science, Volume 4025, pp.213-227, 2006, ISSN: 0302-9743, Springer-Verlag.

[4] BPMN, http://www.bpmn.org/.

[5] Business Rule Group, *Business Rules Manifesto*, retrieved on April, 6-th, 2008 from http://www.businessrulesgroup.org/brmanifesto/BRManifesto.pdf.

[6] Charfi, M. Mezini, "Hybrid web service composition: business processes meet business rules", in *Proc. 2004 International Conference on Service-Oriented Computing (ICSOC'04)*, Nov. 2004, pp. 30-38.

[7] Eertink H., W. Janssen, P. Oude Luttighuis, W. Teeuw and C. Vissers, "A business process design language" , in *Proc. 1st World Congress on Formal Methods*, Toulouse, France, 1999.

[8] Geminiuc, K., "A Services-Oriented Approach to Business Rules Development", *SOA Best Practices: The BPEL Cookbook (Oracle white paper)*, retrieved on April, 6-th, 2008 from http://www.oracle.com/technology/pub/articles/bpel_cookbook/geminiuc.html.

[9] Hall, J., "Developments in Business Rules Standards", *Presentation at the 6th European Business Rule Conference*, 18-20 June, 2007, Dusseldorf, Germany, retrieved on April, 6-th, 2008 from http://www.ebrc2007.com/Uploads/Files/Hall_2c_20J_20presentatie.pdf.

[10] Hermans L., Lemahieu W., Vanthienen J., "Real agility and transparency requires a combination of BPM/SOA, EDA and BRA", In *Proceedings of the 6th European Business Rules Conference*, Düssseldorf (Germany), Jun. 18-19, 2007.

[11] Horrocks, I., Patel-Schneider, P.F.,Boley, H., Tabet, S., Grosof, B., Dean, M., *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, W3C Member Submission 21 May 2004, retrieved on April, 6-th, 2008 from http://www.w3.org/Submission/SWRL/.

[12] Iacob, M.-E., Jonkers, H., "Quantitative analysis of service-oriented architectures", *International Journal of Enterprise Information Systems*, vol. 3, no. 1, Jan.-Mar. 2007, pp. 42-60.

[13] Jonkers, H., Doest, H. ter, *Business rules voor BiZZdesign*, Technical report, Telematica Instituut, Enschede, 2007.

[14] Kolovos, D.S., Paige, R.F., and Polack, F.A. C., "Merging models with the Epsilon Merging Language (EML)", *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, Volume 4199, 2006, p. 215-229.

[15] Lankhorst, M. et al., *Enterprise Architecture at Work - Modelling, Communication, and Analysis*, Springer, 2005.

[16] Miller, J. and J. Mukerji (eds.), *MDA Guide Version 1.0.1*, Object Management Group, Document Nr: omg/2003-06-01, June 2003.

[17] Object Management Group, *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification*, Final Adopted Specification ptc/05-11-01, Nov. 2005, http://www.omg.org/docs/ptc/05-11-01.pdf (1-2-2008).

[18] Object Management Group, *Semantics of Business Vocabulary and Business Rules Specification*, OMG Adopted Specification, 2006.

[19] Object Management Group, *Production Rule Representation: Request for Proposal*, br/2003-09-03, Sept. 2003. http://www.omg.org/docs/br/03-09-03.pdf.

[20] Object Management Group, *UML 2.0 OCL Specification,* ptc/03-10-14, Oct. 2003. http://www.omg.org/docs/ptc/03-10-14.pdf.

[21] F. Rosenberg, S. Dustdar, "Business Rules Integration in BPEL – A Service-Oriented Approach", in Proc. 7th IEEE International Conference on E-Commerce Technology (CEC'05), Munich, Germany, July 2005.

[22] Linehan, M.H., *Semantics in Model-Driven Business Design*, IBM T.J. Watson Research Center, New York, 2006.

[23] Papazoglou, M. P., *Web services: principles and technology*. Harlow: Pearson Prentice Hall, 2008.

[24] Raj, A., Prabhakar, T. V., Hendryx, S., Transformation of SBVR business design to UML models, In *Proceedings of the 1st conference on India software engineering conference*, Pages 29-38, 2008.

[25] RIF Working group, http://www.w3.org/2005/rules/wiki/RIF_WorkingGroup

[26] RuleML, http://www.ruleml.org/.

[27] Scheer, A.-W., *Business Process Engineering: Reference Models for Industrial Enterprises*, Springer, Berlin, 2nd ed., 1994.

[28] Soley, R. and the OMG Staff Strategy Group, *Model Driven Architecture*, Object Management Group White Paper, Draft 3.2, Nov. 2000.