Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 112 (2017) 2354–2362

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France

A Domain-specific Rule Generation Using Model-Driven Architecture in Controlled Variability Model

Neel Mani*, Markus Helfert [a], Claus Pahl[b]

[a] *ADAPT Centre for Digital Content Technology, Dublin City University, School of Computing, Dublin, Ireland*
[b] *Free University of Bozen-Bolzano, Faculty of Computer Science, Bolzano, Italy*

Abstract

The business environment changes rapidly and needs to adapt to the enterprise business systems must be considered for new types of requirements to accept changes in the business strategies and processes. This raises new challenges that the traditional development approaches cannot always provide a complete solution in an efficient way. However, most of the current proposals for automatic generation are not devised to cope with rapid integration of the changes in the business requirement of end user (stakeholder's and customer's) resource. Domain-specific Rules constitute a key element for domain specific enterprise application, allowing configuration of changes, and management of the domain constraint within a domain. In this paper, we propose an approach to the development of an automatic generation of the domain-specific rules by using variability feature model and ontology definition of domain model concepts coming from Software product line engineering and Model Driven Architecture. We provide a process approach to generate a domain-specific rule based on the end user requirement.

Peer-review under responsibility of KES International

Keywords: Rule Generation; Domain-specific rules; Business Process Model; Variability Model; Model Driven Architecture;

1. Introduction

Nowadays, Enterprises struggle with rapid changes due to the dynamic and competitive nature of the

* *E-mail address:* neel.mani2@mail.dcu.ie

1877-0509
Peer-review under responsibility of KES International
10.1016/j.procs.2017.08.206

environment. The businesses need to adapt and implement changes promptly. The changes occur almost at every front of the business process. For example, changes in demand of customers, changes in business strategies (internal and external stakeholder) or changes in various laws. Process model's languages[1] provide expressive and various verification techniques to ensure the reliable processes[2]. However, the languages restrict domain experts to make changes such as defining explicitly the process execution plan as pre-defined task control flow, data flow, and work/process allocation schema, etc. The changes reflect at the modeling stage or design phase, which makes the process model rigid[3]. These may pose challenges in customization, adaptability and maintenance of the system. These languages limit flexibility of enterprises[4,5] and their suitability (or sustainability) for the dynamic environment. As the nature of organisations is volatile (polices of business) and processes are often excessively rigid, Domain specific solution will make process model more focussed for a particular domain in statically and dynamically adaptable in terms of the process execution plan, reducing the dependency on programming, and to software developers.

Advent of model approach in Business Process Model (BPM) systems, domain or business experts usally work on high level of model designing, and maintaining the complex behaviours of their enterprise application. The software professional or programmers are using work on very low level of code who can modify code. Since, Domain experts are often working, designing, learning and thinking in critical way to solve the complex process of enterprise in form of decision-making rules, and policy, their intent is rather simple to understandable. The enterprises are looking for a new standard configurable domain solution for defining the rule, expressing and facilitating their integration for process model constraint. Domain expert and business expert(stakeholder) often work on very high level of abstraction (design model and use modeling language and designing tools) and think in conditional rules, mapping programmer and software expert intent into low level of programming code.

The several steps required for implementation of model transformation and configuration system. These are the steps of high-level design to low level execution. Enterprises, usually, have high level of legacy model and design as a domain model or process model. Automatic code generation is a well-known process of getting the low level of executable code from a given abstract model. Rule generation is the process by which higher level model is translated or transformed into the lower level program. It is a process of conversion of one form to another; it may be platform specific or platform independent or generic[6]. In Model-Driven Architecture(MDA)[7], techniques are expressed by models as the primary development artifact and use them as a basis for obtaining a configurable domain-specific rule for process model customisation in different ways[8], but it does not talk about variability of models (domain model and process model).

We are using digital content process domain for a case study. The web application's machine translation translates source language to target language. The system provides a web based platform where domain expert can edit the generated domain-specific rule in the natural language configurable editor. In this paper, we focus on two main elements: (1) How domain-specific rules(DSR)[9] can be generated from domain models automatically? (2) what are processes of approach to get a rule from domain model and what are the core component models with variability model. We also discuss the benefit of domain models, and configurable rule in process model customisation.

We discuss the related work in Section 2. In Section 3, we proposed approach and core component of this research. In Section 4, we describe the integration of process approach for DSR generation. Then, we saw how the rule generation approach could be implemented using domain model, process model and variability model the solution implementation in Section 5. We finish with an evaluate of the solution in Section 6 and some conclusion with future work in Section 7.

2. Related Work

The use of rule and ontology modeling formalism as the MDD source model, along with defining a metamodel to put it into the framework of MDA has been our proposal. Abdullah et al. (2007) [10], who proposes the possibility of depicting the profile elements to a Jess platform-specific representation, also embarks upon the idea of using a UML profile for the framing of knowledge.

Hecht, Piveta, Pimenta and Price[11] uses high-level programming as well as code generation approach in expediting the process of alteration between software design and its implementation in executable code. In order to

lessen the programming code writing effort in a manual set up in addition to checking errors inherent to it like spelling mistake, Code generation technique is used.

Code generation constitutes the means of obtaining programming constructs from design-level constructs. Automatic code generation deals with the conversion of software design into tools fit for execution with very little or no manual intervention. On the other hand, Hecht, Piveta, Pimenta and Price[11] opine that Aspect Oriented code generation[12] is the blend of automatic code generation techniques with aspect oriented programing concepts to derive both their advantages. To implement this separation of crosscutting behavior has to be ensured at both designs and coding levels. An approach by Groher & Schulze[10] uses UML extension medium to segregate and encase the design of crosscutting behavior whereas Dijkstra[12] constituted a mechanism that deals with the code part of crosscutting behavior.

Models fitting web application development, as defined by web engineering uses methodologies such as[13], WebML[14] and WebDSL[15] which provide conceptual tools[16]. These emphasize on content, navigation and presentation models[17] outlines a proposal wherein Web application development rests upon MVC and JavaServer Faces but doesn't follow rule modeling. Whereas we have constituted an approach which considers it and hence is inventive. Instead of considering other modeling concerns, we used fixed and preset define functionality operationa links to define functionality operations.

In the context of Rule language, Rule Markup Initiative (RuleML)[18], the REWERSE Rule Markup Language (R2ML)[19], and the Semantic Web Rule Language (SWRL)[20] are the important measures in standardization and exchanging of rules. Likewise, the World Wide Web Consortium (W3C) advocates rule-based interchange for the Web with the Rule Interchange Format (RIF)[21].

Dioufet al. (2007)[22] describes means to amalgamate OWL ontologies and UML models for generation of business rule automatically. They recommend the usage of ontologies to attach for semantics to UML models and applying the MDA approach in the extraction of deducible rules from models. Their business rules are generated in the Semantics of Business Vocabulary and Business Rules (SBVR) syntax[23] but they propound a common framework for applying MDA and OWL ontologies for the creation of rulesets in a target rule engine, and till now only the first abstraction version of business rules have been generated. It is still a subject of research, and no precise development methodology has been proposed for creating rule based systems embedded in Semantic Web Application. Although, the proposed the combination of UML and ontologies semantic for extracting the set of rules in target rule engine, they only generated the first level of the abstraction of the rules.

We have proposed process approach for generating the rule from high-level of domain model. The domain-specific rule language (DSRL)[24] is transformed from domain model based on the requirement of the domain user. Our approach provides the steps and component of rule generation from high-level of model.

3. Proposed Approach

3.1. Model Driven Architecture

For constructing Our approach and its fundamentals comes from MDA concepts which are related to transformation of model in domain-specific applications. The developing of the prototyping of the application, the application must have Platform Specific Models (PSM). The transformation of PIM obtains these models and adds to relative technical information to platforms. These models provide the platform for facilitating the rule generation. The MDA approach is commonly used for advanced and complexed model generators. The architecture of the DSRL generator follows the MDA four-level model organization presented by Bézivin[25] as illustrated in Figure 1. At the top level, the M3 is the Syntax Definition Formalism (SDF) metametamodel which is the grammar of the SDF. This level is also known as Computational Independent Model(CIM) or metametamodel which is defined and thus conforms to itself[26]. The BNF notation takes as a self-representation metasyntax. This notation facilities to define multiple well-formed grammar. A given grammar allows defining the infinity of syntactically correct DSR configuration.

At the M2 level, we define the DSRL metamodel, i.e., the grammar of DSRL with ECA defined in SDF and this level is called Platform Independent Model (PIM). The metamodel conforms to the metametamodel at level M3. At the M1 level, we define DSRL models of configuration applications. This is known as Platform Specific Model

(PSM), consisting of entity and definitions. The model conforms to the metamodel at level M2. The bottom level is called M0, we define the configuration of BPM customization consisting with DSR and XML rules, which represent the models at the M1 level.

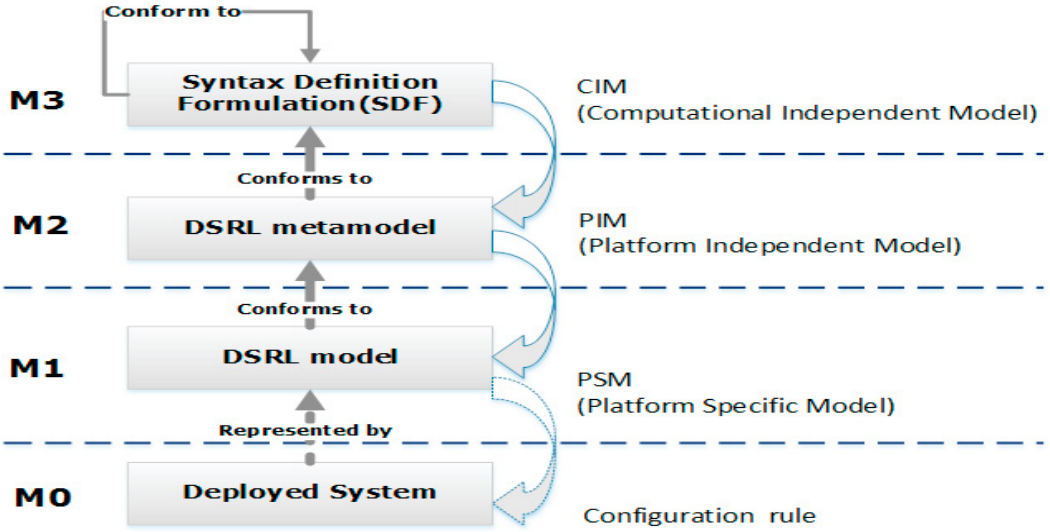

Fig. 1. MDA organisation view of models approach and artifacts of DSRL generator

3.2. Software Product Line Engineering

The key contribution of this paper is the use of a feature model to bridge between an assumed domain model (here in ontology form) and business process. The feature model streamlines customisation for domain template (domain model and process model) and configures constraints of domain. It acts as a bridge between domain model and process model for activation and deactivation of their component.

A software product line framework or lifecycle has two phases and two focuses (see Fig. 2): (i) Problem Space will address the problem of the application is terms of in individual and group of the application to define the belongness of the family; (ii) Solution Space will address the software components for solving that problem; (iii) Domain engineering phase is formal representation of a common platform where a number of arbitrary products are developed and implemented. This includes a variability model and the core assets of the product family. Feature models are standard variability modeling techniques in SPLE that represents variability in a hierarchical way to simplify or differentiate the feature of products that belong to a software family. A feature is a logical unit that is specified by a set of functional (what the system should do) and non-functional requirements (how the system works, should behave and quality attributes) [27, 28], and; (iv) Application engineering phase provide a platform for the end users or customers that capture their requirement for the target application. This is responsible for concreate or final product from variability model trough a generation of rule, configuration of the rule and deploying the final product.

3.3. Domain-specific Language

The domain experts, business expert and subject matter experts (SMEs) are the process developers of enterprise business design. They require a high-level language platform or environment to develop and configure their application that is domain-specific language (DSLs)[29]. Domain expert and software developer can solve a domain development tasks issue. Domain experts are required to use their domain knowledge, experience, expertise and intellect to solve the challenges. Software developers provide a solution through code and it will be in the form of some executable in a programing language to obtain a program/application/solution that can be performed to run on the systems to solve development task issue. In both the cases, cognitive challenges and significant amount of logical, mental and thinking activities are involved.

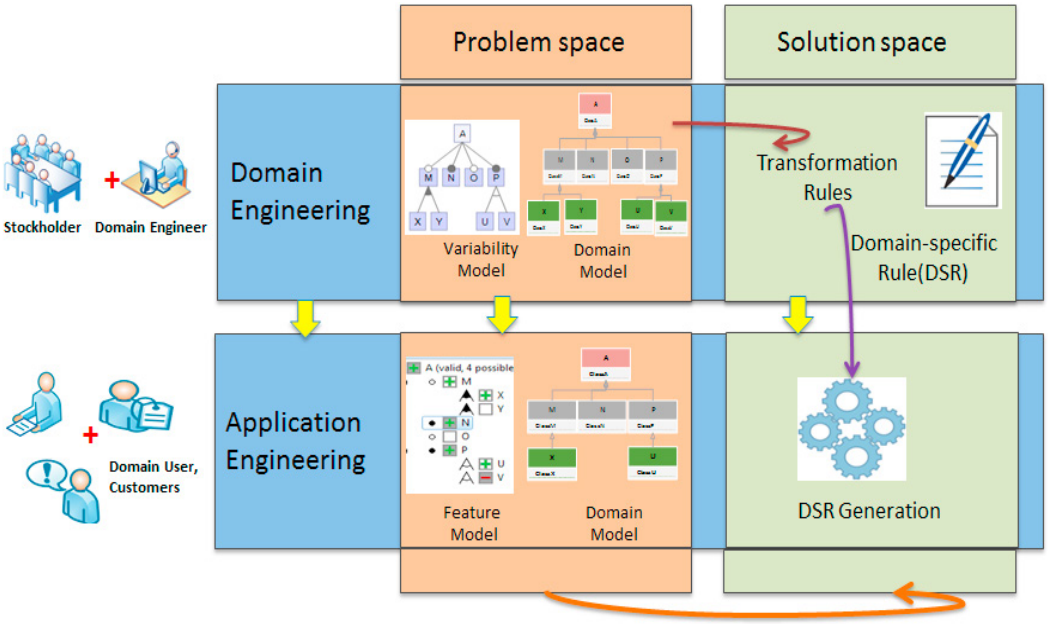

Fig.2. Framework for SPLE: domain and application engineering, problem and solution space

Moreover, the considerable amount of cognitive activity on the part of developer cannot be reduced just because significant research has been done in machine learning and automatic programming. Although a considerable amount of research in machine learning and automatic programming has been carried out, still reducing the cognitive activity of the domain expert is stupendously difficult, if not impossible. The programmer has to invest a lot of mental and logical effort, even though, the semantic gap between the domain solution procedure (DSP) and programming language is still present. The domain experts are thinking at a very high level of abstraction while designing the DSP. This results in a huge semantic gap when the developers use the low level of programming language instead of high level. If we have a specific language that have the right level of abstraction at which domain user can design and think, the tasks of creating the program will be much easier[30]. A domain-specific language technology can make it possible.

DSLs are always depended on GPL (General Purpose Language) languages, support of GPL is required either at the time of compilation or while designing and developing a new compiler/ interpreter for a particular domain. The DSRL is a combination of rules and BPMN process; it is simpler and easier as compared to other modular language systems. Moreover, DSR generation process is based on Template Model which carry domain model and process model(BPMN) and it is a responsible for the primary flow and functional process of the whole process. ECA (Event Condition Action) Language is the focus on the operation part of the DSR system (fulfil the condition and actions perform based on process model event). In DSRL, there is no need GPL language, though we use a little bit of GPL language for process but it is very less in comparison of DSL.

4. Approach for Domain-specific Rule Automatic Generation

The principle argument here is that customisation of process models can be carried out at run time which is possible by the software product line engineering (SPLE). An SPLE facilitates mass customization and satisfies the different stakeholder requirement[31]. The SPLE can be implemented in two steps: domain engineering and application engineering. The domain engineering is responsible for reusable platform and defines the commonality and the variability of the product line. In this research, we are using for domain template model which is the combination of domain model and process model. The application engineering is responsible for facilitating a platform for end user applications interface which through connect the domain engineering. Therefore, we consider the SPLE as an enabler for mass customisation. Rule generation is another challenge. MDA concept can be used to generate rules, provide definition and composition, and validation of domain constraint challenges.

The MDA concept can provide a multiple conceptual platform that allows a domain user to create models of the application, concept of the business logic and generate rule for a target model or platform by means of transformations. By using the MDA concept domain expert would be able to solve and focus on domain engineering and domain related challenges that are specific to the application domain. The generated DSR is completely independent from any platform, the application developer also get benefited from writing platform-specific codes[32]. In this way, MDA provides the abstraction level of the new software development and existing customisation(reuse).

We express rule generation of DSR (Domain-specific Rule) in terms of functional and operational. The representation of the rules is done with based XML-based ECA language called DSRL. In the figure 3, a case of rule generation illustrates the DSRL processor is loaded (or as loaded in?) the UML class diagram and a semantic model is expressed as a relationship between attributes (Length:int) and operations (fileType():bool).

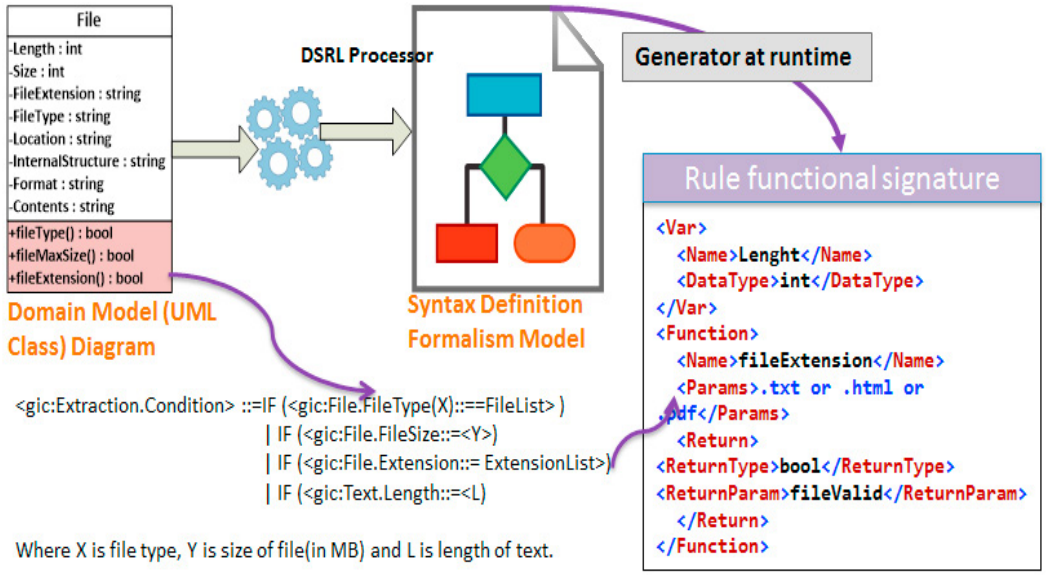

Fig. 3. DSR generation and signature

Both are a major component of the rule process builder for the appropriate rule conversation processor. During the rule conversion, the DSRL processor would produce rule that will save in a rule repository or process by the BPMN system, perhaps some kind of rules may be verified and evaluated by user end or human based computing.

A DSR is a XML rule which is independent from system and its platform. The Semantic model provides a semantic meaning in term of structure and syntax. It can be varied to do away with rule constraints like inconsistency, redundancy and incorrectness resulted from multiple views and abstraction. Semantic consistency is a major issue when compatibility of specific behaviour. A typical example is compatibility between sequence diagram and state chart diagram when class is implemented [33].

5. Implementation

Our paper is being focused on the conceptual aspects; the prototype system has been implemented. The web application prototype provides a platform for a domain user or expert that selects the feature based on their requirement and desires to build a rule generation and configurable processes for Digital content problem and constraints. In this architecture, a software product line used as a platform and MDA to perform participate in rule generation. Overall application is used by non-technical domain expert or business expert (understand the particular domain) to specify the future application. The flow of application is started from domain engineering where domain expert creates template models (domain, variability, and process model) and connect all models through a waving model. The feature selection as input is processed through the domain user or customers, based on selected feature

from template models, components are activated and de-activated. The final domain model is transformed in rule by MDA. During MDA process, entire rule specification such as: grammar definition, abstract and concreate rule formation based on syntax, apply the DSR grammar and rule language semantic and validation etc. After rule generation, next step is configuration of process model constraint; it could be automatic or semi-automatic. The automatic configuration is used as the constraint value in the form of parametric at the time when user selects the feature. In the semi-automatic, experts can configure the constraint value after rule generation.

After this step of the rule generation and formalism the next the stage was to implement our approach on DSRs generation automatically. From the Fig.1 and 3, a first prototype of our approach in a Microsoft environment. Our model is a web based application model, our domain mode is used as a digital content domain. For our domain model definition Metamodel arrives just in time, while we were using the ontology directly in form of UML with MOF model. We use a Microsoft .Net framework for implementation of user feature model selection to DSRs generation. The process step is the follow as: use Object Model for transforming our domain model to data model. This data model is a simple own define mode (follow the grammar) l which contain different object models for abstract syntax, concreate syntax, grammar and semantic models. In Fig. 3, we illustrated a signature of DSR with one example of file uploading, where there are two parametric functions, one can calculate the file length and other is to validate the file type or extension (like .txt,.pdf,.html, etc.) After this, using the modeling language as UML in .Net, we enrich the object model with three different class to care varies object model based on UML fundamental properties: attribute, function and data structure. The implementation this state, rules are generated from domain model with semantic through the Domain Vocabulary[34] and Domain-specific Rules like format.

6. Evaluation

In this section, we describe how to evaluate the generated DSR. A model-based design and automated rule generation are new concept and there is no standard system which can be used for evaluating this proposed solution. We validate the type of generation output in terms correctness, completeness, output effectiveness and efficiency. Our primary goal is to have a proof of full functional and operational correctness, and completeness of the rule with respect to its feature requirement selected by the domain user. Among the types of investigations (strategies), we plan to carry out two different evaluation strategies to evaluate this DSR, generated rule evaluation will consist of following:

6.1. Fixed Validation of generated rule

In this section, we describe how to evaluate the generated DSR. A model-based design and automated rule generation are new concept and there is no standard system which can be used for evaluating this proposed solution. We validate the type of generation output in terms of correctness, completeness, output effectiveness and efficiency. Our primary goal is to have a proof of full functional and operational correctness, and completeness of the rule with respect to its feature requirement selected by the domain user. Among the types of investigations (strategies), we plan to carry out two different evaluation strategies to evaluate this DSR generated rule evaluation will consist of following:

6.2. Fixed Validation of generated rule

Firstly, this approach investigates the generated rule in terms of whether the end user selected features are converted into rule or not? We validate this issue with under and over generation of rule. Secondly, as the grammar of the rule is based on operational and functional of the models via validation, we can argue that the selection of the feature model can be verified easily.

- Under generation – We define under generation as missing instance (for example events, actions etc.) at the time of generation or after generation.
- Over generation- This would be identified as some extra information in terms of syntax and semantic (functional and operational information).

Thus, using this approach validate we can validate the generated rule with find some extra or less information generated during the model transformation. This is quickly along with the reuse of existing opaque behavior(s), creating a flexible and easy to reconfigure complex systems environment.

7. Conclusion

In this paper, we have proposed a process approach of a domain-specific rule generation through variability management. We have presented a novel approach to generate the rule from domain model and process model based on the end user requirement by applying variability modeling as a systematic approach. We provide a process support and its approach to allow the end user (non-technical domain experts). They can select suitable features that satisfy their business constraint within domain aspects. We have added adaptively to domain model. We provide a conceptual view of domain-specific rule generation and manage the domain model, and variability model using MDA. It helps in managing frequent changes of the business process along with variability schema of a set of structured variation mechanisms for the specification. The domain user can generate the DSRs and configure domain constraints in a dynamic environment.

We plan to extend this approach in combination with our existing work on business process model customization based on user requirement (feature model, domain model and process models), so that a complete development life cycle for the customization and configuration of business process models are supported. We also see the need for other domain, how to apply in different domain as a generic approach.

Acknowledgement

This research is supported by Science Foundation Ireland (SFI) as a part of the ADAPT Centre at Dublin City University (Grant No: 12/CE/I2267).

References

1. Van der Aalst, W.M. and A.H. Ter Hofstede, YAWL: yet another workflow language. Information systems, 2005. 30(4): p. 245-275.
2. Boukhebouze, M., et al., A rule-based approach to model and verify flexible business processes. International Journal of Business Process Integration and Management, 2011. 5(4): p. 287-307.
3. Rangiha, M.E. and B. Karakostas. Goal-driven social business process management. in Science and Information Conference (SAI), 2013. 2013. IEEE.
4. van Eijndhoven, T., M.-E. Iacob, and M.L. Ponisio. Achieving business process flexibility with business rules. in Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE. 2008. IEEE.
5. Ayora, C., et al. Towards run-time flexibility for process families: open issues and research challenges. in Business Process Management Workshops. 2013. Springer.
6. Bergmayr, A. and M. Wimmer. Generating Metamodels from Grammars by Chaining Translational and By-Example Techniques. in MDEBE@ MoDELS. 2013.
7. Poole, J.D. Model-driven architecture: Vision, standards and emerging technologies. in Workshop on Metamodeling and Adaptive Object Models, ECOOP. 2001.
8. Gonçalves, R.C.A., Parallel programming by transformation. 2015.
9. Mani, N., M. Helfert, and C. Pahl, Business Process Model Customisation using Domain-driven Controlled Variability Management and Rule Generation. International Journal on Advances in Software, 2016. 9(Numbers 3 & 4, 2016): p. 179 - 190.
10. Groher, I. and S. Schulze. Generating aspect code from UML models. in The 4th AOSD Modeling With UML Workshop. 2003.
11. Hecht, M.V., et al., Aspect-oriented code generation. Simpsio Brasileiro de Engenharia de Software, 2005.
12. Dijkstra, E.W., A discipline of programming. Vol. 1.
13. Koch, N., et al., UML-based web engineering, in Web Engineering: Modelling and Implementing Web Applications. 2008, Springer. p. 157-191.
14. Ceri, S., et al., Morgan Kaufmann series in data management systems: Designing data-intensive Web applications. 2003: Morgan Kaufmann.
15. Groenewegen, D.M., et al. WebDSL: a domain-specific language for dynamic web applications. in Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications. 2008. ACM.
16. Ceri, S., P. Fraternali, and M. Matera, Conceptual modeling of data-intensive Web applications. IEEE Internet Computing, 2002. 6(4): p. 20-30.

17. Moreno, N., et al. Addressing new concerns in model-driven web engineering approaches. in International Conference on Web Information Systems Engineering. 2008. Springer.
18. Boley, H., S. Tabet, and G. Wagner. Design rationale of RuleML: A markup language for semantic web rules. in Proceedings of the First International Conference on Semantic Web Working. 2001. CEUR-WS. org.
19. Wagner, G., A. Giurca, and S. Lukichev, A usable interchange format for rich syntax rules integrating OCL, RuleML and SWRL. Proc. of WSh. Reasoning on the Web, 2006.
20. Horrocks, I., et al., SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission, 2004. 21: p. 79.
21. Kifer, M. Rule interchange format: The framework. in International Conference on Web Reasoning and Rule Systems. 2008. Springer.
22. Diouf, M., S. Maabout, and K. Musumbu. Merging model driven architecture and Semantic Web for business rules generation. in International Conference on Web Reasoning and Rule Systems. 2007. Springer.
23. (OMG), O.M.G., Semantics of Business Vocabulary and Business Rules (SBVR). (Version 1.0.).
24. Mani, N. and C. Pahl. Controlled variability management for business process model constraints. in ICSEA 2015, The Tenth International Conference on Software Engineering Advances. 2015. IARIA XPS Press.
25. Bézivin, J., On the unification power of models. Software & Systems Modeling, 2005. 4(2): p. 171-188.
26. Visser, E., Syntax definition for language prototyping. 1997: Eelco Visser.
27. Acher, M., Managing, multiple feature models: foundations, languages and applications. 2011, Nice.
28. Soltani, S., et al. Automated planning for feature model configuration based on functional and non-functional requirements. in Proceedings of the 16th International Software Product Line Conference-Volume 1. 2012. ACM.
29. Fowler, M., Domain-specific languages. 2010: Pearson Education.
30. Gupta, G., Language-based software engineering. Science of Computer Programming, 2015. 97: p. 37-40.
31. Pohl, K., et al., Software Product Line Engineering: Foundations, Principles and Techniques. 2005: Springer-Verlag New York, Inc.
32. Lewis, G.A., B.C. Meyers, and K. Wallnau, Workshop on Model-Driven Architecture and Program Generation. 2006, DTIC Document.
33. Engels, G., R. Heckel, and J.M. Küster, Rule-based specification of behavioral consistency based on the UML meta-model, in ≪ UML≫ 2001—The Unified Modeling Language. Modeling Languages, Concepts, and Tools. 2001, Springer. p. 272-286.
34. Gonçalves, R.C., D. Batory, and J.L. Sobral, ReFlO: An interactive tool for pipe-and-filter domain specification and program generation. Software & Systems Modeling, 2016. 15(2): p. 377-395.