A Framework for Generating Domain-specific Rule for Process Model Customisation

Neel Mani[1], Markus Helfert[1] and Claus Pahl[2]
[1]*ADAPT Centre for Digital Content Technology, Dublin City University, School of Computing, Dublin, Ireland*
[2]*Free University of Bozen-Bolzano, Free University of Bozen-Bolzano, Bolzano, Italy*

Keywords: Domain Model, Domain-Specific Rule, Rule Language, Rule Generation, Business Process Model, Variability Model, Model Driven Architecture, Model Transformation, Domain-Specific Language.

Abstract: The domain-specific model-driven development requires effective and flexible techniques for implementing domain-specific rule generators. In this paper, we present a framework for rule generation through model translation with feature model, a high-level of the domain model to translate into low-level of rule language based on the paradigm of software reuse in terms of customisation and configuration with domain-specific rule strategies benefit mode-to-text translations. This framework is domain-specific where non-technical domain user can customise and configure the business process models. These compositions support two dimensional of translation modularity by using software product line engineering. The domain engineering is achieved by designing the domain and process model as a requirement space, it is also called template model, connecting with feature model through weaving model. The feature model is a high-level input model to customise the template model to an implementation. The application engineering is achieved by supporting the rule definition and configuring the generated rules. We discuss the development approach of the framework in a domain-specific environment; we present a case study in a Digital Content Technology (DCT) domain.

1 INTRODUCTION

Nowadays the reuse of software is one of challenges to implement to customise the application based on the end-user requirements. The traditionally model-driven engineering use for extracting the knowledge from a high-level design model to low-level language. How to effectively fill the gap between the software product line engineering and the models to extract the knowledge from high-level models to low-level language. The primary requirement of the framework where end user can opt their requirement to adapt the desirable models. For achieving the desirable model, there are several steps to require the implementation of configuration-based systems where translate the high-level design to low level execution. Enterprises, usually have high level legacy models and they are domain and process based models. Automatic code generation (Edwards, Brun et al. 2012, Prout, Atlee et al. 2012, Ringert, Roth et al. 2015) is a well-known approach for getting the execution code of a system from a given abstract model. Rule is an extended version of code that requires compiling and building it in a configurable mode. Rule generation is an approach by which the higher-level design model as input and the lower level of execution code as output are shaped. It may be platform independent or platform specific (Bergmayr and Wimmer 2013) approach. In Model-Driven Architecture (MDA), the techniques are expressed by design models as the primary artefact of development and use them as a basis for obtaining a configurable domain-specific rule for business process model customisation in different ways (Gonçalves 2015). But it does not talk about the variability of models (domain model and process model).

The major motivations is to address the automated customise the models based on the end-user and then generate the domain-specific rule(DSR) (Mani, Helfert et al. 2016) from high-level complex domain models at run time. The domain models are characterised by the complexity of their structures as specified in the metamodel or the non-trivial nature of constraints imposed on them or a combination of these two factors. It is a problem by which the existing fully-automated model generators fail.

Mani N., Helfert M. and Pahl C.
A Framework for Generating Domain-specific Rule for Process Model Customisation.
DOI: 10.5220/0006512201630171
In *Proceedings of the International Conference on Computer-Human Interaction Research and Applications (CHIRA 2017)*, pages 163-171
ISBN: 978-989-758-267-7

Our contribution is a framework for customising and configuring process model in digital content technology domain in dynamic environment. The production capability is based on reusable model as core assets that configure the constraints of the domain problems by non-technical domain user. At runtime, we propose the model-based rule generation for process models that use the variability models. They capture the user requirement to process automatically the rule generation and configuration of the process model's composition. Mass customisation updates are possible through software Product Line Engineering (SPLE), which provides a platform to domain expert for designing and developing the domain template (domain model and process model).

The paper is organised as follows. We compare the related work Section 2. Section 3 is the motivation of research paper. In Section 4, we discuss the overall proposed approach, in which we discuss DSLs, feature model, domain model and the rule. Section 5 introduces and describe the design of the framework and its component and In Section 6, we evaluate the overall approach in terms of usability The Section 7 concludes the paper reflecting the future work.

2 RELATED WORKED

In this section, we review the present concept, technologies and techniques that motivate us to present the relevant problem under domain constraint study. It includes a short description of automatic code generation, language and applications based on Model-driven principles.

Our research work uses the SPLE life cycle for customising the feature based on the user requirement and MDA framework that uses in generating the domain-specific rule of the customised domain model.

The domain expert performing on the high level of abstraction, the machine works low level of language (Gupta 2015). The code generation technique accelerates the transformation process between designing the application and its implementtation in terms of executable code (Prout, Atlee et al. 2012, Gurunule and Nashipudimath 2015). The code generation is an approach for converting the high level of design constructs into low level of executable code constructs. The automatic code generation is converting software design into executable code without little bit intervention of programmer or developer (Gurunule and Nashipudimath 2015). The generation technique reduces the effort of the manual writing of programming code, avoiding the manual programming errors like syntactical (spelling mistake) and programmatically semantical errors.

The web application development is description of processes, techniques and selecting appropriate models for web engineering. The web engineering uses automatic web application methodologies such as UWE (Koch, Knapp et al. 2008), WebML (Ceri, Fraternali et al. 2000) and WebDSL (Groenewegen, Hemel et al. 2008) approaches. The design and development of Web applications and tools provide mainly conceptual models (Ceri, Fraternali et al. 2002), focusing on content, navigation and presentation models (Linaje, Preciado et al. 2007, Moreno, Meliá et al. 2008). Now , the model driven approach for dynamic web application, based on MVC and server is described by Distante et al. (Distante, Pedone et al. 2007). However, these methods do not consider as the user requirement of the variability model. To simplify our description, we have considered the user requirement and according to the need of the user, the user selects the feature and customises the enterprise application in the dynamic environment.

Currently, there is no such type of methodology or process of development for creating a rule-based system in a web application (semantic). Dioufet et al. (Diouf, Maabout et al. 2007, Musumbu, Diouf et al. 2010) propose a process which merges UML models and OWL ontologies for business rule generation. The solution for semantic web is ontologies, UML and applying the MDA approach for generating or extraction rules from high level of models. Although, the proposed combination of UML and semantic ontologies are for extracting the set of rules in target rule engine, but they only generate the first level of the abstraction of the rules.

We have proposed a classification of several quality and governance constraints elsewhere (Pahl and Mani 2014): authorisation, accountability, workflow governance and quality. The domain-specific rule language (DSRL) (Mani and Pahl 2015) is a combination of rules and BPMN. Our approach provides a framework architecture for generating the domain-specific rule and configuring the generated rule for process model customisation using the variability model.

3 MOTIVATIONS

The enterprise needs to change and adapt progressively the dynamic behaviour at runtime in response to changing conditions, updating new features in support of enterprise and business applications and in the surrounding business process models. Nowadays, the requirements for developing

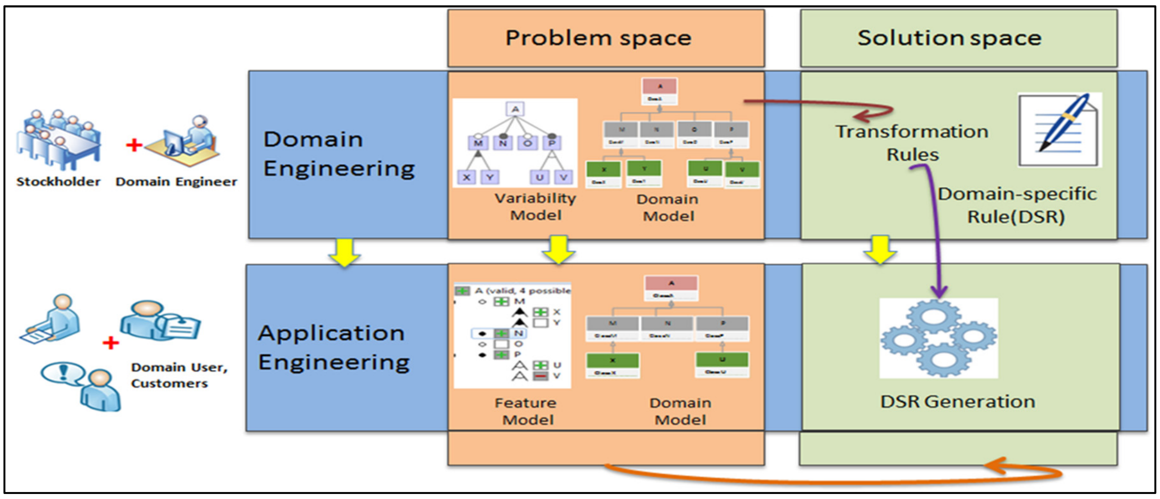


Figure 1: Framework for SPLE: domain and application engineering, problem and solution space (Mani, Helfert et al. 2017).

and customisation process are increasing. However, the process model languages (Standard , Reichert and Dadam 1998, White 2004, Van der Aalst and Ter Hofstede 2005, List and Korherr 2006) restrict domain user and designer to describe explicitly the process execution plan as pre-defined task control flow, data flow and work/process allocation schema. The changes reflect at modelling stage or design phase which make the process model rigid (Boukhebouze, Amghar et al. 2011, Gromoff, Kazantsev et al. 2012, Rangiha and Karakostas 2013). Thus, adaptability is emerging as crucial and an underlying capability in highly dynamic environment.

During business process customisation and adaptation in dynamic and domain-specific environments, the end users have only domain knowledge without technically customising and adapting the changes and configuration is done in process models at runtime. For example, process activities and sub process could be activated and deactivated (removed) (Mani, Helfert et al. 2016) based on user requirement at application engineering level.

4 SOFTWARE PRODUCT LINE ENGINEERING

The key contribution is the use of a feature model to bridge the gap between an assumed domain model (here in ontology form) and the domain-specific rule extension of a business process. The feature model streamlines the constraints customisation for business processes. It acts as a bridge between the domain model and the rule language.

We propose to abstractly support the different SPLE aspects from MDA. The MDA aims at capturing all the relevant aspects of the framework through appropriate models. The stakeholders' motives are more prominently captured by models than the implementation codes. Models capture the requirements or the intentions of the end users more effectively, help to avoiding accidental implementation details and also are more suited for analysis. Models, in MDA context, are much more than being supportive artifacts; rather, these are actually source artifacts which can be utilized for automated analysis and/or rule generation.

In our approach, the aim is the solve the need for expressive and easy-to-understand adaptation. Therefore, the domain model is implemented as a variability model which describes the variants in which the domain expert designs the domain template composition. Since the domain model abstracts the rule generation, the definition of a bridge between the elements in the variability model and the elements in the domain model could be used to support the dynamic rule generation in the underlying domain-specific environment.

To this end, we use a weaving model as an additional software asset input to project the changes in the features of the variability model, on abstract elements in a domain model. In other words, the weaving model works as a bridge between the elements in these models.

The goal of Software Product Lines Engineering (SPLE) is developing a set of software components and systems with similar characteristics and catering to the requirements of a domain through management of certain features (Kang, Cohen et al. 1990). SPLE effectively tunes down the development cost and

Table 1: Weaving model.

	Variablity Model	Feature Model	Process Model	Domain Model
1.	V1	F1	Task1	Class1
2.	V2	F2	Task2	Class 2
3.	V3	F3	Task3	Class 3

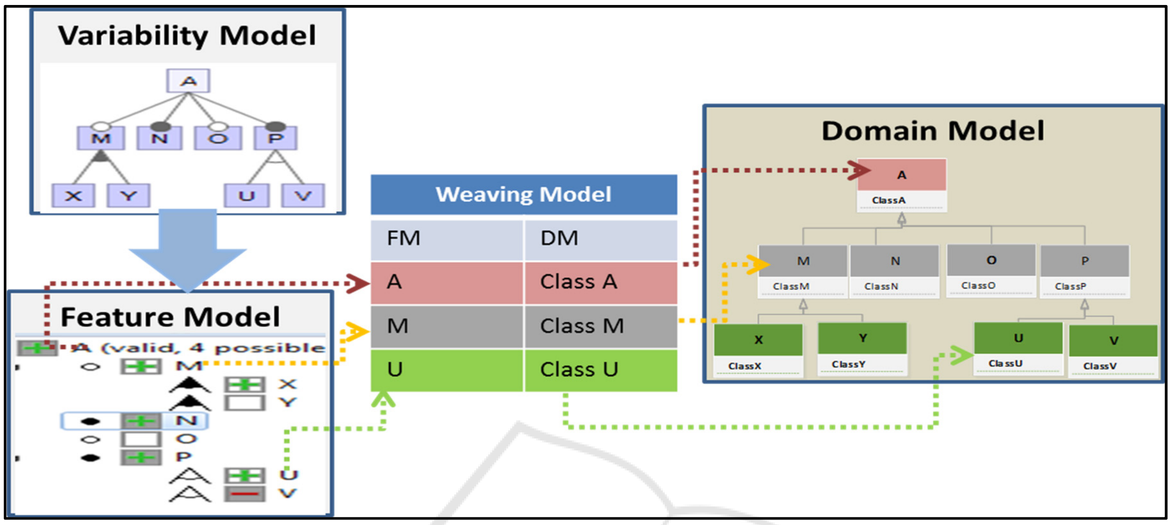


Figure 2: Models that support the SPL for autonomic rule generation.

market time of the software, enhances the overhead quality and engineering by reusing assets strategically within the domain. SPLE uses adopted techniques to manage reusability with commonality and variability model that effectively categorizes the common assets and their variabilities. The software product line framework has two phases (see Figure 2): (i) the Problem Space for describing the problem description, the type of applications, or an individual application in the category; (ii) Solution Space for providing the software components to solve that problem; iii) Domain engineering phase may be defined as a formally represented platform in which development and implementation products take place. In SPLE, the variability modeling technique known as "Feature Models" are used for the purpose of portraying variability in hierarchical manner differentiation or simplification of features in hierarchy of products belonging to a software family.

The DSRs are products, based on activate or deactivate features in the feature model to manage the requirement of domain user or stack holder during feature selection. Therefore DSRs depend upon the feature of the feature model. The rule generation in the variability model produces the adaptation space with 1) and all possible constraint configurations of the process model (in terms of active and inactive features in the feature model with parametric value) and 2), customising the process model in terms of the functional and operational use. In order to avoid the problem or an interruption (error, system halt, malfunctioning, wrong interpretation, etc.) during rule generation and or configuration in critical service application, we argue that the feature model and its possible configurations should be validated and verified at the runtime.

The SPLE and MDA are not only complementary, but their integration may lead to significant gains in various applications. On one side MDA provides for abstractly representing various aspects of a product line, while on the other SPLE provides for a well-defined application scope. This provides a sound basis for the development and selection of appropriate modeling languages. Further, the automated generation of system configurations is made possible by accurate models as a result of automated analysis and rule. MDA provides effective techniques for conveying the results of specifying variability as follows:

- Metamodeling: It refers to type of systems that express, for specific domains, having the constraints that are associated with a product line, with key abstract syntax characteristics and static semantic constraints

  Domain-specific languages (DSLs): In order to formalize the specifics of structure of the product line, its behavior and requirements with respect to domain, the DSLs provide notations governed by extendable metamodels.

- Model transformations and rule generators: It refers to ensuring the consistency of implementations of the product line along with the corresponding analysis. The analysis may be retrospect to functional and QoS requirements.

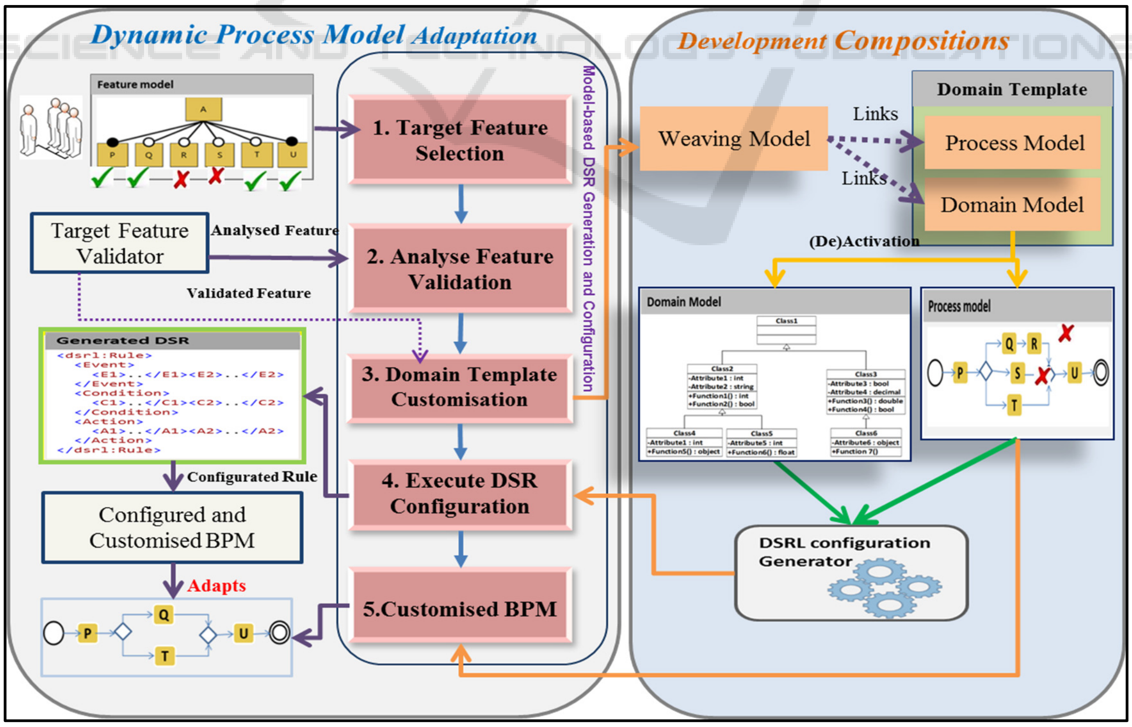
Key advantages of using MDA in conjunction with variability of SPLs are (1) rigorously capturing the commonalities and variabilities in a family of systems and (2) helping automated repetitive tasks that must be accomplished for each product instance.

Figure 2 shows how to combine modelling and model transformations to develop an SPLE-MDA for Digital Content Technology. First, the assets of the SPLE-MDA model elements are described with a family of Digital Content Technology. These model elements are conformed to the metamodel (of Domain Model), which is a DSL for Digital Content Technology. Second, the decision model is another model which specifies the aspects or characteristics (named features) of a particular Digital Content Technology. Third, a weaving model projects the features on the DSL for scoping the domain model. Finally, the output system is obtained through a model (scoped DSRL) to text (domain-specific rule) the translation.

5 DESIGNS OF FRAMEWORK

The dynamic process model adaptation is possible by customising and configuring the overall architecture of the framework at runtime through predefine domain template (domain model and process model) models. This approach works on principal of activetion and deactivation of models. However, the proposed framework follows strategy for the dynamic activation and deactivation of features in the domain template, based on the end-user requirement. After, the domain template is modeled and the weaving model has connected the feature model and the domain template (via a virtual relational table) at design time, the customisation and adaptation may take place at run time. The customisation of the process model and rule generation achieve at runtime. This approach works under the domain-specific environment.

An overview of our approach, enabling customisation and configuration of the component connections for the rule is shown in Figure 3. As illustrated, the approach covers both dynamic process model adaptation and development compositions. The aim is to design appropriate architecture modes of the feature selection based on user requirements and generate the configurable rule for process model customisation. At run time, the users configure the



```
<dsrl:Rule>
  <Event>
    <E1>..</E1><E2>..</E2>
  </Event>
  <Condition>
    <C1>..</C1><C2>..</C2>
  </Condition>
  <Action>
    <A1>..</A1><A2>..</A2>
  </Action>
</dsrl:Rule>
```

Figure 3: Detailed Framework of Rule Generation and Process Model Customisation.

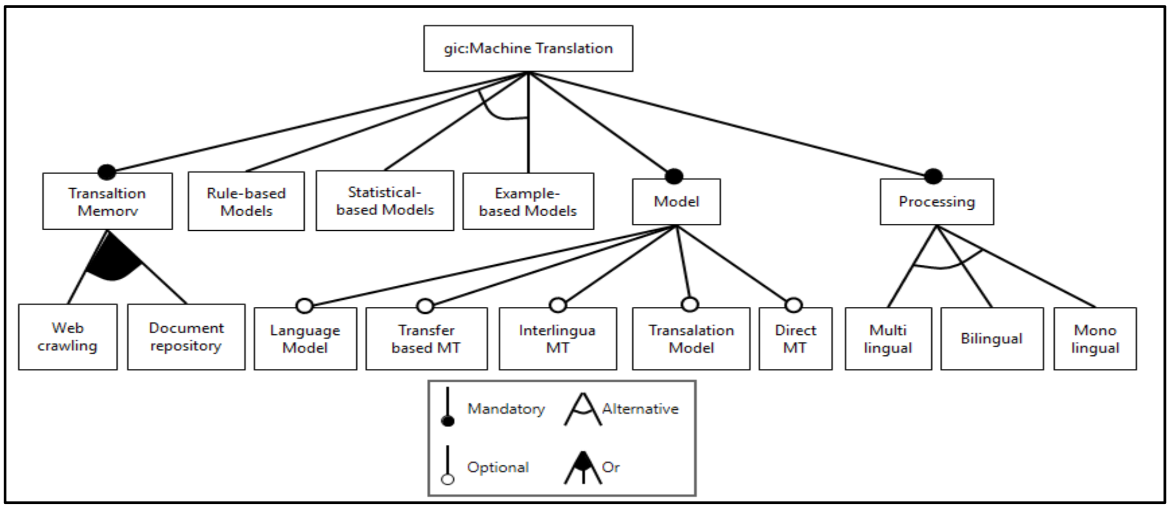


Figure 4: Feature model of Machine translation.

rule, thereby facilitating the framework to quality configuration in terms of effective, efficient and satisfactory usability. The feature model requirements are continually analysed and validated every feature by target feature validator with respect to the runtime feature section. It reflects in the template model environment as an activation and deactivation through weaving model. In the case of the feature selection violation, the target feature validator gets prompted and validated accordingly on basic criteria of the feature model. For example, the user can remove the mandatory feature from the system. In the following, we briefly discuss each phase and describe their actives.

The framework offers a dynamic solution for rule generation and process model customisation. The framework consists of two sections: Development Composition and Dynamic Process Adaptation. Development composition consist of different elements such Weaving Model, Process model, metamodel, DSRL configuration generator. We have developed two building blocks to provide utilisation of the domain model variability and community at run time: (1) the domain model customiser uses a weaving model (table 1). The weaving models are models which capture different kind of relationships between models (Del Fabro and Valduriez 2007). It is about finding a similar type of transformation patterns between model elements integrating transformations in the tabular form.

The weaving helps to execute the customisation process base on validate feature. The validate feature is a set of analysed and verified features, which is captured by the requirements of a domain user or stakeholder. (2) In the Rule generator, the models and metamodel that created in the domain model customiser are used in the MDA at different level M1 and M2 (Figure 3). The generated rule configuration has two different types: per-configured (at design time) and post-configurable (dynamic) metamodel

In the Dynamic Process Adaptation, the proposed structure carries out the following steps to support dynamics adaptation. First, the model-based configurator collects the information from various models. If the requirement of target feature is violated, the feature selection or customisation activities such as: rename, move, update and delete feature, affect the activation and deactivation process. The execute customisation models pass the captured requirement to weaving model for further activities. The customised and generated sets of rules are output from the Development Composition.

We propose an approach that offers a solution for the dynamic adaptation of rule generation and, BPM customisation and configuration composition. The variability model services are responsible for carrying mandatory or optional feature of the BPM and metamodel. The feature of a process can be activated or deactivated at any moment of time, i.e. design or run time. We introduce a mechanism where domain expert and user can perform their tasks in a simply way. A domain expert can design high-level of solution for domain, based on that solution, domain user can modify and customise model elements (activities) in any process over time.

The model elements activation and deactivation depend on the variability model and what are the requirement of end user. In this research, there are two different groups of users, involving one expert in domain with modelling knowledge and other have a functional domain knowledge, but they are non-technical. Therefore, we select software product line engineering (SPLE) platform where the user can perform the tasks. The SPLE is a standard model to develop software applications using platform and mass customisation (Böckle, van der Linden et al. 2005).

5.1 Feature Model

Figure 4 shows the feature model used in our case study. For instance, the Language Model, Transfer based MT, Interlingua MT, Direct MT and Translation Model features are variants that can be used during execution to accomplish the machine translation functionality in the Model point.

5.2 Weaving Model

A product line feature model represents variabilities and commonalities. The features in a feature model are simply symbols with their type. Mapping features to other models (feature model, domain model and process model,) expressed in Table 1. Next, we show how to perform the mapping by means of a weaving model (Geyer and Becker 2002). We use a static weaving model for managing the variability relationships among all models. The principle argument for using the static weaving model is with domain-specific environment, when the domain experts have significant domain knowledge. They design and develop the domain template at design (static) time. This weaving approach enables us for scoping and configuring the Domain Models from a set of given Features.

6 EVALUATIONS

We evaluate the DCT, analysing the time and efficiency of its configuration including satisfaction and operational compliance based on computing human interaction by the end user. The goal of the analysis is to get a comparative analysis of the time and efficiency of configured DCT and its sub-process systems. The emphasis is on analysing the relative benefit of the proposed framework and a manual or traditional or baseline approach regarding the efficiency, effectiveness, and satisfaction with function and operational compliance support. The feature selection and configuration scenarios involve to modifications resulting from improvement of the complex process activity that affected the function and operation of the process.

The current example is a part of a digital content processing process model as a sample process for the rule composition of business processes and domain constraints that conduct this process. The source text is translated into target language by the machine translation activity. The translated text quality decides whether further post-editing activity is required. Usually, these constraints are domain-specific, e.g., referring to domain objects, their properties and respective rules.

The capability of the solution is effective, efficient and satisfactory by the user when used under specified conditions. We evaluate our contribution in usability and forcing it on the following aspects which is illustrated in Figure 5.

There are different component in the ISO standard 9241, applying the specification of usability into both hardware and software designs. We discuss the usability criteria and its goals.
In this overall framework, the usability criteria define as effectiveness, efficiency and satisfaction. It specified that end-users achieve specified goals are in domain-specific environments.

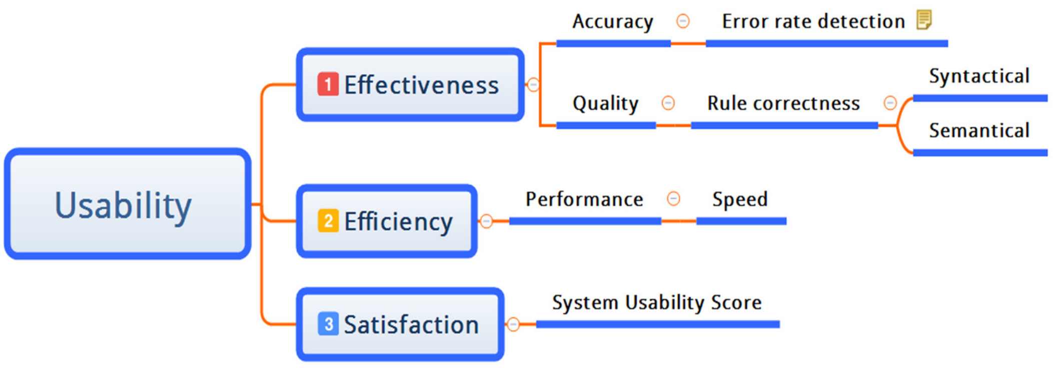


Figure 5: Usability evaluation criteria.

- **Efficiency.** The comparison between the time taken of configuring domain constraints in manual and semi-automatic process, based on that find which process is more efficient.
- **Effectiveness.** The generated rule configuration in terms of accuracy to prevent or protect errors to achieved the configuration of domain constraints goal.
- **Satisfaction.** The measure of end-user's comfort and acceptability of the overall framework.

7 CONCLUSION AND FUTURE WORK

In this paper, we have presented a model-based framework, generating the domain-specific rule for process model configuration and customisation in dynamic environment. A case study on digital content technology shows the applicability of this framework, a realisation development approach and a proof of concept prototype validating the feasibility of the proposed approach. The main benefit of the framework has: 1) Non-technical domain experts can customise and configure the business process without knowing any technical knowledge;2) The framework can configure the domain constraints in dynamic environment; and 3) A third is to evaluate the framework architecture mechanisms in terms of usability. It can be handled the completeness of configuration of rule in terms: effectiveness, efficiency and satisfaction base on computer human interface by end use. The completeness of rule configuration means performance, error free rule, syntactically and semantically correctness after configuring the rule.

We plan to extend this approach in combination with our existing work on business process model customisation based on user requirement (feature model, domain model and process models), so that a complete development life cycle for the customisation and configuration of business process models is supported. We also see the need for further research that focuses on how to define the DSRL in terms of abstract and concreate syntactical definition with grammar formation across different domains and how to convert conceptual models into generic domain-specific rule language which are applicable to other domains. So far this is a model to text translation, but shall be improved with a system that learns from existing rules and domain models, driven by the feature model approach with automatic constraints configuration, and to result in an automated DSRL generation.

ACKNOWLEDGEMENTS

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

Bergmayr, A. and M. Wimmer (2013). *Generating Metamodels from Grammars by Chaining Translational and By-Example Techniques.* MDEBE@ MoDELS.

Böckle, G., F. J. van der Linden and K. Pohl (2005). *Software product line engineering: foundations, principles and techniques*, Springer Science & Business Media.

Boukhebouze, M., Y. Amghar, A. c.-N. Benharkat and Z. Maamar (2011). *A rule-based approach to model and verify flexible business processes.* International Journal of Business Process Integration and Management 5(4): 287-307.

Ceri, S., P. Fraternali and A. Bongio (2000). *Web Modeling Language (WebML): a modeling language for designing Web sites.* Computer Networks 33(1): 137-157.

Ceri, S., P. Fraternali and M. Matera (2002). *Conceptual modeling of data-intensive Web applications.* IEEE Internet Computing 6(4): 20-30.

Del Fabro, M. D. and P. Valduriez (2007). *Semi-automatic model integration using matching transformations and weaving models.* Proceedings of the 2007 ACM symposium on Applied computing, ACM.

Diouf, M., S. Maabout and K. Musumbu (2007). M*erging model driven architecture and Semantic Web for business rules generation.* International Conference on Web Reasoning and Rule Systems, Springer.

Distante, D., P. Pedone, G. Rossi and G. Canfora (2007). *Model-driven development of web applications with UWA, MVC and JavaServer faces.* International Conference on Web Engineering, Springer.

Edwards, G., Y. Brun and N. Medvidovic (2012). *Automated analysis and code generation for domain-specific models.* Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on, IEEE.

Geyer, L. and M. Becker (2002). On the influence of variabilities on the application-engineering process of a product family. Software Product Lines, Springer: 1-14.

Gonçalves, R. C. A. (2015). *Parallel programming by transformation*.

Groenewegen, D. M., Z. Hemel, L. C. Kats and E. Visser (2008). *WebDSL: a domain-specific language for* *dynamic web applications*. Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications, ACM.

Gromoff, A., N. Kazantsev, K. Evina, M. Ponfilenok and D. Kozhevnikov (2012). *Modern era in business architecture Design*. Far East Journal of Psychology and Business 9(2): 15-34.

Gupta, G. (2015). *Language-based software engineering*. Science of Computer Programming 97: 37-40.

Gurunule, D. and M. Nashipudimath (2015). *A Review: Analysis of Aspect Orientation and Model Driven Engineering for Code Generation*. Procedia Computer Science 45(0): 852-861.

Kang, K. C., S. G. Cohen, J. A. Hess, W. E. Novak and A. S. Peterson (1990). *Feature-oriented domain analysis (FODA) feasibility study*, DTIC Document.

Koch, N., A. Knapp, G. Zhang and H. Baumeister (2008). *UML-based web engineering*. Web Engineering: Modelling and Implementing Web Applications, Springer: 157-191.

Linaje, M., J. C. Preciado and F. Sánchez-Figueroa (2007). *Engineering rich internet application user interfaces over legacy web models*. IEEE internet computing 11(6): 53-59.

List, B. and B. Korherr (2006). *An evaluation of conceptual business process modelling languages*. Proceedings of the 2006 ACM symposium on Applied computing, ACM.

Mani, N., M. Helfert and C. Pahl (2016). *Business Process Model Customisation using Domain-driven Controlled Variability Management and Rule Generation*. International Journal on Advances in Software 9(Numbers 3 & 4, 2016): 179 - 190.

Mani, N., M. Helfert and C. Pahl (2017). *A Domain-specific Rule Generation Using Model-Driven Architecture in Controlled Variability Model*. Procedia Computer Science 112: 2354-2362.

Mani, N. and C. Pahl (2015). *Controlled variability management for business process model constraints*. ICSEA 2015, The Tenth International Conference on Software Engineering Advances, IARIA XPS Press.

Moreno, N., S. Meliá, N. Koch and A. Vallecillo (2008). *Addressing new concerns in model-driven web engineering approaches*. International Conference on Web Information Systems Engineering, Springer.

Musumbu, K., M. Diouf and S. Maabout (2010). *Business rules generation methods by merging model driven architecture and web semantics*. 2010 IEEE International Conference on Software Engineering and Service Sciences, IEEE.

Pahl, C. and N. Mani (2014). *Managing quality constraints in technology-managed learning content processes*.

Prout, A., J. M. Atlee, N. A. Day and P. Shaker (2012). *Code generation for a family of executable modelling notations*. Software & Systems Modeling 11(2): 251-272.

Rangiha, M. E. and B. Karakostas (2013). *Goal-driven social business process management*. Science and Information Conference (SAI), 2013, IEEE.

Reichert, M. and P. Dadam (1998). *ADEPTflex—Supporting dynamic changes of workflows without losing control*. Journal of Intelligent Information Systems 10(2): 93-129.

Ringert, J. O., A. Roth, B. Rumpe and A. Wortmann (2015). *Code Generator Composition for Model-Driven Engineering of Robotics Component & Connector Systems*. arXiv preprint arXiv:1505.00904.

Standard, O. *Web services business process execution language version 2.0*.

Van der Aalst, W. M. and A. H. Ter Hofstede (2005). *YAWL: yet another workflow language*. Information systems 30(4): 245-275.

White, S. A. (2004). *Business process modeling notation. Specification*, BPMI. org.