# Controlled Variability Management for Business Process Model Constraints

Neel Mani

ADAPT Centre for Digital Content Technology
Dublin City University, School of Computing
Dublin, Ireland

Claus Pahl

ADAPT Centre for Digital Content Technology
Dublin City University, School of Computing
Dublin, Ireland

***Abstract*—Business process models are abstract descriptions that are applicable in different situations. To allow a single process model to be reused, configuration and customisation features can help. Variability models, known from product line modelling and manufacturing, can control this customisation. While activities and objects have already been subject of similar investigations, we focus on the constraints that govern a process execution. We report here on the development a rule-based constraints language for a workflow and process model. The aim is a conceptual definition of a domain-specific rule variability language, integrated with the principles of a common business workflow or process notation. This modelling framework will be presented as a development approach for customised rules through a feature model. Our use case is content processing, represented by an abstract ontology-based domain model in the framework.**

***Keywords–Business Process Modelling, Process Constraints, Variability Model, Domain-specific Rule Language.***

## I. Introduction

Business process models are abstract descriptions that can be applied in different situations and environments. To allow a single process model to be reused, configuration and customisation features help. Variability models, known from product line engineering, can control this customisation. While activities and objects have already been subject of customisation research, we focus on the customisation of constraints that govern a process execution here. Specifically, the recent emergence of business processes as a services in the cloud (BPaaS) highlights the need to implement a reusable process resource together with a mechanism to adapt this to consumers.

We are primarily concerned with the utilisation of a conceptual domain model for business process management, specifically to define a domain-specific rule language for process constraints management. We present a conceptual approach in order to define a Domain Specification Rule Language (DSRL) for process constraints [1] based on a Variability Model (VM). To address the problem, we follow a feature-based approach to develop a domain-specific rule language, borrowed from product line engineering. It is beneficial to capture domain knowledge and define a solution for possibly too generic models through using a domain-specific language (DSL). A systematic DSL development approach provides the domain expert or analyst with a problem domain at a higher level of abstraction. DSLs are a favourable solution to directly represent, analyse, develop and implement domain concepts. DSLs are visual or textual languages targeted to specific problem domains, rather than general-purpose languages that aim at general software problems. With these languages or models, some behaviour inconsistencies of semantics properties can be checked by formal detection methods and tools.

The key contribution is a model development approach using of a feature model to bridge between an assumed domain model (here in ontology form) and the domain-specific rule extension of a business process to define and implement process constraints. The feature model streamlines the constraints customisation of business processes for specific applications. It acts as a bridge between domain model and rule language. The novelty lies in the use of software product line technology to customise processes.

We choose content processing here as a specific domain context to illustrate the application the proposed domain-specific technique (but also look at the transferability to other domains in the evaluation). We use a text-based content process involving text extraction, translation and post-editing as a sample business process. Note that we have also investigated the subject in the context of e-learning processes, which we will also address later on. We also briefly discuss a prototype implementation. However, note that a full integration of all model aspects is not aimed at as the focus here is on models. The objective is to outline principles of a systematic approach towards a domain-specific rule language for content processes.

The paper is organised as follows. We review process modelling and constraints in Section 2. In Section 3, we content processing from a feature-oriented DSL perspective. Section 4 introduces rule language background and ideas for a domain-based rule language. We then discuss formal process models into with the rule languages can be integrated.

## II. Business Process Models and Constraints

At the core is a process model that defines possible behaviour. This is made up of some frame of reference for the system and the corresponding to the attributes used to describe the possible behaviour of the process. The set of behaviours constitutes a process referred to as the extension of the process and individual behaviours in the extension are referred as instances. Constraints can be applied at states of the process to determine its continuing behaviour depending on the current situation. The rules combine a condition (constraint) on a resulting action. The target of our rule language (DSRL) is

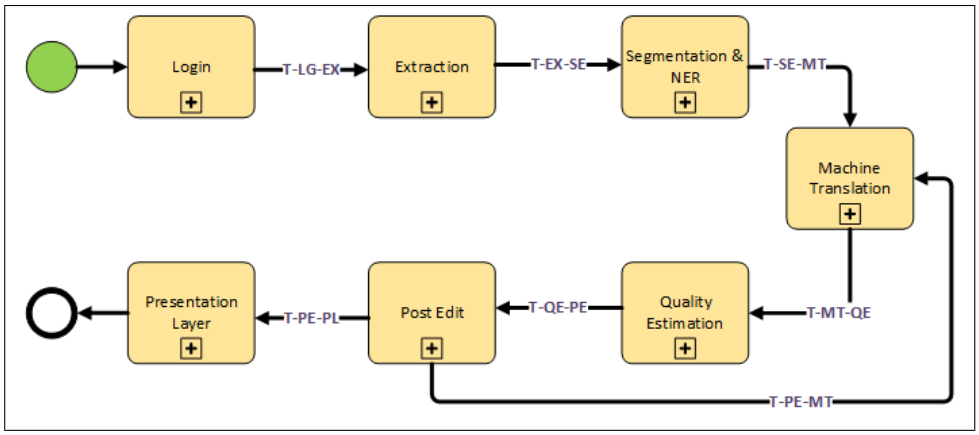


Figure 1. Workflow design of content process.

a standard business process notation (as in Fig. 1). Rules shall be applied at the processing states of the process.

Our application case study is intelligent content processing. Intelligent content is digital content that allows to users to create, curate and consume content in a way that satisfies dynamic and individual requirements relating to task design, context, language, and information discovery. The content is stored, exchanged and processed by a Web architecture and data will be exchanged, annotated with meta-data via web resources. Content is delivered from creators to consumers. Content follows a particular path which contains different stages such as extraction and segmentation, name entity recognition, machine translation, quality estimation and post-editing. Each stage in the process has its own challenges and complexities.

We assume the content processing workflow as in Figure 1 as a sample process for the rule-based instrumentation of processes. Constraints govern this process. For instance, the quality of a machine-based text translation decides whether further post-editing is required. Generally, these constraints are domain-specific, e.g., referring to domain objects, their properties and respective activities on them.

## III. Domain and Feature Model

Conceptual models (CM) are part of the analysis phase of system development helping to understand and communicate particular domains [1]. They help to capture the requirements of the problem domain and, in ontology engineering, a CM is the basis for a formalized ontology. We utilise a conceptual domain model (in ontology form) to derive a domain-specific process rule language. A domain specific language (DSL) is a programming or specification language that supports a particular application domain through appropriate notation, grammar and abstractions [2]. DSL development requires both domain knowledge and language development expertise. A prerequisite for designing DSLs is an analysis that provides structural knowledge of the application domain.

### A. Feature Model

The most important result of a domain analysis is a feature model. A feature model covers both the aspects of software family members, like commonalities and variabilities, and also reflects dependencies between variable features. A feature diagram is a graphical representation of dependences between a variable feature and its components. Mandatory features are present in a concept instance if their parent is present. Optional features may be present. Alternative features are a set of features from which one is present. Groups of features are a set of features from which a subset is present if their parent is present. Mutex and Requires are relationships that can only exist between features. Requires means that when we select a feature, the required featured must be selected too. Mutex means that once we choose a feature the other feature must be excluded (mutual exclusion).

A domain-specific feature model can cover languages, transformation, tooling, and process aspects of DSLs. For feature model specification, we propose the FODA (Feature Oriented Domain Analysis) [3] method. It represents all the configurations (called instances) of a system, focusing on the features that may differ in each of the configurations [4]. We apply this concept to constraints customisation for processes. The Feature Description Language (FDL) [5] is a language to define features of a particular domain. It supports an automated normalization of feature descriptions, expansion to disjunctive normal form, variability computation and constraint satisfaction. It shall be applied to the content processing use case here. The basis here is a domain ontology called GLOBIC (global intelligent content), which has been developed as part of our research centre. GLOBIC elements are prefixed by gic.

Feature diagrams are a FODA graphical notation. They can be used for structuring the features of processes in specific domains. Figure 2 shows a feature diagram for the GLOBIC content extraction path, i.e., extraction as an activity that operates on content in specified formats. This is the first step in a systematic development of a domain-specific rule language (DSRL) for GLOBIC content processing use case. The basic component gic:Content consists of a gic:Extraction element, a mandatory feature. A file is a mandatory component of gic:Extraction and it may either be used for Document or Multimedia elements or both. The closed triangle joining the lines for document and multimedia indicates a non-exclusive (more-of) choice between the elements. The gic:Text has two mandatory states Source and Target. Source contains ExtractedText and Target can be TranslationText. Furthermore, expanding the feature Sentence is also a mandatory component of ExtractedText. The four features Corpora, Phrase, Word

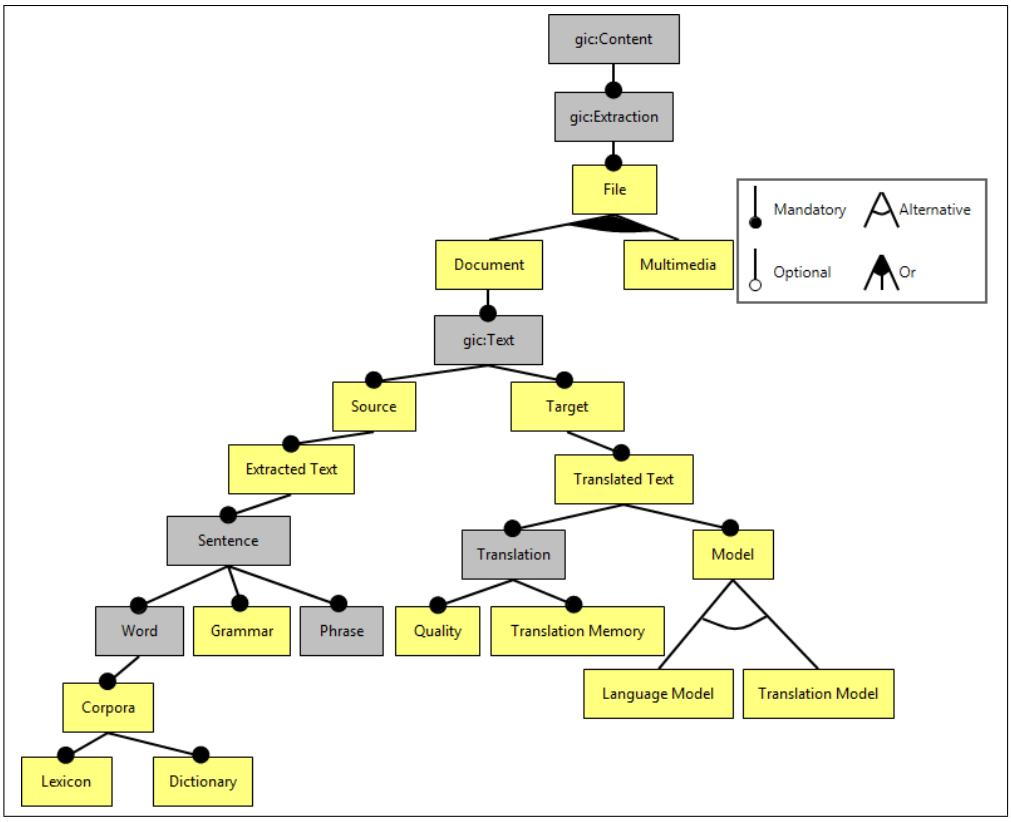


Figure 2. Workflow design of content process.

and Grammar are mandatory. On the other side of gic:Text, a TranslationText is a mandatory component of Target, also containing a mandatory component Translation. A Translation has three components: TranslationMemory and Model are mandatory features, Quality is an optional feature. A Model may be used as a TranslationModel or a LanguageModel or both models at same time. An instance of a feature model consists of an actual choice of atomic features matching the requirements imposed by the model. An instance corresponds to a text configuration of a gic:Text super class. The number of possible gic:Text feature combinations is 512 for the given model, structured and made accessible through the model.

The feature model might include for instance duplicate elements, inconsistencies or other anomalies. We can address this situation by applying consistency rules on feature diagrams. Each anomaly may indicate a different type of problem. The feature diagram algebra consists of four set of rules [4]:

- Normalization Rules - rules to simplify the feature expression by redundant feature elimination and normalize grammatical and syntactical anomalies.
- Expansion Rules - a normalized feature expression can be converted into a disjunctive normal form.
- Satisfaction Rules - the outermost operator of a disjunctive normal form is one-of. Its arguments are All expressions with atomic features as arguments, resulting in a list of all possible configurations.
- Variability Rules - feature diagrams describe system variability, which can be quantified (e.g. number of possible configurations).

The feature model is the key element. Thus, checking internal coherence and providing a normalised format is important for its accessibility for non-technical domain experts. In our setting, the domain model provides the semantic definition for the feature-driven variability modelling.

### B. *Domain Model*

Semantic models have been widely used in process management [6,7]. This ranges from normal class models to capture structural properties of a domain to full ontologies to represent and reason about knowledge regarding the application domain or also the technical process domain [8,9]. Domain-specific class diagrams are the next step from a feature model towards a DSL definition. A class is defined as a descriptor of a set of objects with common properties in terms of structure, behaviour, and relationships. A class diagram is based on a feature diagram model and helps to stabilise relationship and behaviour definitions. Note that there is an underlying domain ontology here, but we use the class aspects (subsumption hierarchy only).

In the content use case, class diagrams of gic:Content and its components based on common properties are shown in Figure 3. The class diagram focuses on gic:Text. The two major classes are Text (Document) and Movie files (Multimedia), consisting of different type of attributes like content : string, format : string, or frame rate : int. Figure 3 is the presentation of an extended part of the gic:Content model. For instance, gic:Text is classified into the two subclasses Source and Target. One file can map multiple translated texts or none. gic:Text is multi-language content (source and target content).

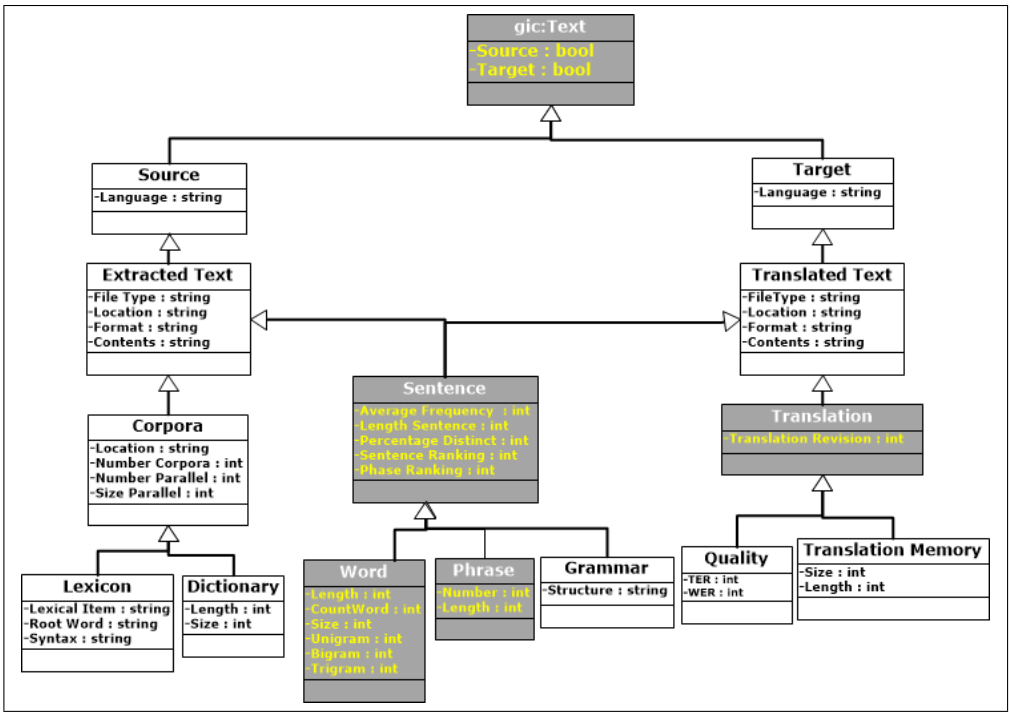


Figure 3. Domain model for global content.

## IV. Constraints Rule Language

Rule languages typically borrow their semantics from logic programming [12]. A rule is defined in the form of If-then clauses containing logical functions and operations. A rule language can enhance ontology languages, e.g., by allowing one to describe relations that cannot be described using for instance description logic (DL) underlying the definition of OWL. We adopt Event-Condition-action (ECA) rules to express rules on content processing activities. The rules take the constituent elements of the GLOBIC model into account:

- Content objects (e.g., text) that are processed.
- Content processing activities (e.g., extraction or translation) that process content objects.

An example shall illustrate ECA rules for extraction as the activity. Different case can be defined using feature models:

- We can customise rules for specific content types (text files or multimedia content).
- We can also vary according to processing activities (extraction or translation).

Three sample rule definitions are:

- On uploading file notification from user and if filetype is valid, then progress to Extraction
- On a specific key event and Text is inputted by user and if text is valid then progress Process to Segmentation
- On a specific key event and Web URL input by user and if URL is valid then progress to Extraction and Segmentation

We define the rule language as follows using GLOBIC concepts (examples here):

gicRule ::= [gic:Event] || [gic:Cond] || [gic:Action]
gic:Event::= {Upload} || {Translate} || {Extract}

While the rule syntax is simple, the important aspect is that that the syntactic elements refer to the domain model, giving it semantics and indicating variability points. Variability points are, as explained, defined in the feature model. The above three examples can be formalised using this notation. Important here is thus the guidance in defining rules that a domain expert gets through the domain model as a general reference framework and the feature model definition the variability points.

## V. Implementation

While this paper focuses on the conceptual aspects, a prototype has been implemented. Our implementation (cf. Fig. 4) provides a platform that enables building configurable processes for content management problems and constraints running in the Activiti workflow engine. In this architecture, a cloud service layer perform data processing and avail of different resources using the Content Service Bus (based on the Alfresco content management system) to perform activities.

Every activity has its own constraints. The flow of entire activities is performed in a sequential manner so that each activitys output becomes input to the next. The input data is processed through the content service bus (Alfresco) and the rule policy is applied to deal with constraints. The processed data is validated by the validation & verification service layer. After validation, processing progresses to the next stage of the Activiti process - e.g., if Segmentation & Extraction data is validated, then it will automatically move to the Name Entity Recognition stage, otherwise sent for reprocessing.

Reasons to architecturally separate a Service Layer include:

- To provide the capability of grouping, interlinking and coupling services within components.

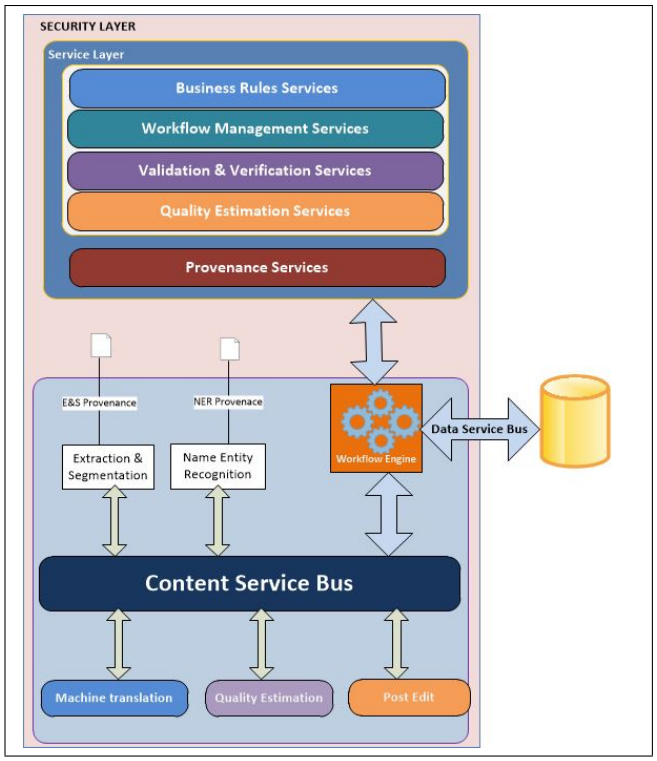


Figure 4. Prototype implementation architecture.

- To share data across multiple component of an application at any time.
- To execute long running operations without overloading the engine, as each component may have individual specific services.
- To use the common content management bus and provisioning infrastructure provided by the Service layer.
- To provide a common policy engine service for the entire platform to reduce the code complexity and improve maintainability.
- To monitor workflow, process executions and record task names, execution durations and parameters use through a provenance service.

The architecture of the system is based on services and standard browser thin clients. We follow the Toolkit script of JavaScript functions that can be used to deploy entire application on a Tomcat web server. The application can be hosted on a Tomcat web server and all services could potentially and be hosted on cloud-based server.

### A. Discussion

Explicit variability representation has benefits for the modelling stage. The feature and domain models control the variability, i.e., add dependability to the process design stage. It also allows formal reasoning about families of processes.

Furthermore, the general utility can be demonstrated. The domain and feature models here specifically support domain experts. We have worked with experts in the digital media and language technology space as part of our research centre. Their qualitative feedback, based on expert interviews as the mechanism, confirms the need to provide a mechanism to customise business processes in a domain-specific way. Using the feature model, rule templates can be filled using the different feature aspects guided by the domain model without in-depth modelling expertise. The majority of experts (more than 2/3) in the evaluation have confirmed simplification or significant simplification in process modelling.

In addition, we looked at another process domain to assess the transferability of the solution. In the learning domain, we examined learner interaction with content in a learning technology system [30,31,32]. Again, the need to provide domain expert support to define constraints and rules for these processes became evident. Here, educators act as process modellers and managers, specifically managing the educational content processing as an interactive process between learners, educators and content. Having been involved in the development of learning technology systems for years, tailoring these to specific courses and classes is required.

## VI. Related Work

Current open research concerns for process management includes customisation of governance and quality policies and the non-intrusive adaptation of processes to policies. Today, one-size-fits-all service process modelling and deployment techniques exist. However, their inherent structural inflexibility makes constraints difficult to manage, resulting in significant efforts and costs to adapt to individual domains needs.

We discuss related work in the field of constraints and policy definition and adaptive BPEL processes. While a notation such as BPMN is aimed at, there is more work on WS-BPEL in our context. Work can be distinguished into two categories.

- BPEL process extensions designed to realize platform-independence: Work in [23] and [25] allows BPEL specifications to be extended with fault policies, i.e., rules that deal with erroneous situations. SRRF [13] generates BPEL processes based on defined handling policies. We do not bind domain-specific policies into business processes directly, as this would not allow to support user/domain-specific adaptation adequately.
- Platform-dependent BPEL engines: Dynamo [3] is limited in that BPEL event handlers must be statically embedded into the process prior to deployment (recovery logic is fixed and can only be customised through the event handler). It does not support customisation and adaptation. PAWS [1] extends the ActiveBPEL engine to enact a flexible process that can change behaviour dynamically, according to constraints.

Furthermore, process-centricity is a concern. Recently, business-processes-as-a-service (BPaaS) is discussed. While not addressed here as a cloud technology specifically, this perspective needs to be further complemented by an architectural style for its implementation [20].

We have proposed a classification of several quality and governance constraints elsewhere [30]: authorisation, accountability, workflow governance and quality. This takes the BPMN constraints extensions [22,23] into account that suggest containment, authorisation and resource assignment as categories into account, but realises these in a less intrusive process adaptation solution.

The DSRL is a combination of rules and BPMN. Moreover, DSLR process based on BPMN and ECA rules is the main focus on the operational part of the DSRL system (i.e., to check conditions and perform actions based on an event of a BPMN process). There is no need for a general purpose language in a DSLR, though aspects are present in the process language. [33,34,35] discuss business process variability, though primarily from a structural customisation perspective. However, [33] also uses an ontology-based support infrastructure.

## VII. Conclusion

In presenting a variability and feature-oriented development approach for a domain-specific rule language for business process constraints, we have added adaptivity to process modelling. We can provide domain experts with a set of structured variation mechanisms for the specification, processing and management of process rules as well as managing frequency changes of business processes along the variability scheme at for notations like BPMN. The novelty of our variability approach is a focus on process constraints and their rule-based management, advancing on structural variability.

Cloud-based business processes-as-a-service (BPaaS) as an emerging trend signifies the need to adapt resources such as processes to different consumer needs (called customisation of multi-tenant resources in the cloud). Furthermore, self-service provisioning of resources also requires non-expert to manage this configuration.

We see the need for further research that focuses on how to adapt the DSRL across different domains and how to convert conceptual models into generic domain-specific rule language which are applicable to other domains. So far, this translation is semi-automatic, but shall be improved with a system that learns from existing rules and domain models, driven by the feature approach, and to result in an automated DSRL generation.

## Acknowledgment

This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/CE/I1142 as part of the Centre for Global Intelligent Content (www.cngl.ie) at DCU.

## References

[1] Ö. Tanrver and S. Bilgen, "A framework for reviewing domain specific conceptual models," CompStand & Interf, vol. 33, pp. 448-464, 2011.

[2] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," ACM computing surveys, vol. 37:316-344, 2005.

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

[4] A. Van Deursen and P. Klint, "Domain-specific language design requires feature descriptions," Jrnl of Comp and Inf technology, vol. 10, pp. 1-17, 2002.

[5] M. Acher, P. Collet, P. Lahire, and R. B. France, "A domain-specific language for managing feature models," in ACM Symp on Applied Computing, 2011, pp. 1333-1340.

[6] C. Pahl and Y. Zhu, "A semantical framework for the orchestration and choreography of web services," Electronic Notes in Theoretical Computer Science, vol. 151(2), pp. 3-18, 2006.

[7] C. Pahl, "Semantic model-driven architecting of service-based software systems," Information and Software Technology, vol. 49(8), pp. 838-850, 2007.

[8] C. Pahl, "An ontology for software component matching," International Journal on Software Tools for Technology Transfer, vol 9(2), pp. 169-178, 2007.

[9] M.X. Wang, K.Y. Bandara, and C. Pahl, "Integrated constraint violation handling for dynamic service composition," IEEE International Conference on Services Computing, 2009, pp. 168-175.

[10] Y.-J. Hu, C.-L. Yeh, and W. Laun, "Challenges for rule systems on the web," Rule Interchange and Applications, 2009, pp. 4-16.

[11] A. Paschke, H. Boley, Z. Zhao, K. Teymourian, and T. Athan, "Reaction RuleML 1.0" in Rules on the Web: Research and Applications, 2012, pp. 100-119.

[12] H. Boley, A. Paschke, and O. Shafiq, "RuleML 1.0: the overarching specification of web rules," Lecture Notes in Computer Science. 6403, 162-178, 2010.

[13] T. Soininen and I. Niemel, "Developing a declarative rule language for applications in product configuration," in practical aspects of declarative languages, ed: Springer, 1998, pp. 305-319.

[14] D. Curry, H. Debar, and B. Feinstein, "Intrusion detection message exchange format data model and extensible markup language (xml) document type definition," IDWG, 2002.

[15] K. Williams, M. Brundage, P. Dengler, J. Gabriel, A. Hoskinson, M. R. Kay, et al., Professional XML databases: Wrox Press, 2000.

[16] E. Wilde and D. Lowe, XPath, XLink, XPointer, and XML: A practical guide to Web hyperlinking and transclusion, 2002.

[17] L. Wood, V. Apparao, L. Cable, M. Champion, M. Davis, J. Kesselman, et al., "Document object model (DOM) specification," W3C recommendation, 1998.

[18] E. R. Harold, Processing XML with Java: Addison-Wesley, 2002.

[19] R. Mohan, M. A. Cohen, and J. Schiefer, "A state machine based approach for a process driven development of web-applications," in Advanced Information Systems Engineering, 2002, pp. 52-66.

[20] S. Van Langenhove, "Towards the correctness of software behavior in uml: A model checking approach based on slicing," Ghent Univ, 2006.

[21] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, and Y. Bontemps, "Generic semantics of feature diagrams," Computer Networks, vol. 51, pp. 456-479, 2/7/ 2007.

[22] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, "FORM: A feature-oriented reuse method with domain-specific reference architectures," Annals of Software Engineering, vol. 5, pp. 143-168, 1998.

[23] M. L. Griss, J. Favaro, and M. d'Alessandro, "Integrating feature modeling with the RSEB," in Intl Conf Software Reuse, 1998, pp. 76-85.

[24] K. Czarnecki and U. W. Eisenecker, "Generative programming," 2000.

[25] D. Beuche, "Modeling and building software product lines with pure variants," in Intl Software Product Line Conference-Volume 2, 2012, pp. 255-255.

[26] D. Benavides, S. Segura, P. Trinidad, A. R. Corts, "FAMA: Tooling a framework for the automated analysis of feature models," VaMoS, 2007.

[27] M. Antkiewicz and K. Czarnecki, "FeaturePlugin: feature modeling plug-in for Eclipse," Workshop on Eclipse Techn, 2004, pp. 67-72.

[28] A. Classen, Q. Boucher, and P. Heymans, "A text-based approach to feature modelling: Syntax and semantics of TVL," Science of Computer Programming, vol. 76, pp. 1130-1143, 2011.

[29] A. v. Deursen, P. Klint, and J. Visser, "Domain-specific languages: an annotated bibliography," SIGPLAN Not., vol. 35, pp. 26-36, 2000

[30] C. Pahl and N. Mani. Managing Quality Constraints in Technology-managed Learning Content Processes. In: EdMedia'2014 Conference on Educational Media and Technology. 2014

[31] S. Murray, J. Ryan, C. Pahl. A tool-mediated cognitive apprenticeship approach for a computer engineering course. 3rd IEEE Conference on Advanced Learning Technologies, 2003.

[32] X. Lei, C. Pahl, and D. Donnellan, "An evaluation technique for content interaction in web-based teaching and learning environments." The 3rd IEEE International Conference on Advanced Learning Technologies 2003, IEEE, 2003.

[33] M.X. Wang, K.Y. Bandara, and C. Pahl, "Process as a service distributed multi-tenant policy-based process runtime governance." IEEE International Conference on Services Computing (SCC 2010), IEEE, 2010.

[34] Y. Huang, Z. Feng, K. He, Y. Huang: Ontology-based configuration for service-based business process model. In: IEEE SCC, pp. 296303. 2013

[35] N. Assy, W. Gaaloul, B. Defude: Mining configurable process fragments for business process design. DESRIST. LNCS 8463, pp. 209224. 2014