Business Process Model Customisation using Domain-driven Controlled Variability Management and Rule Generation

Neel Mani, Markus Helfert

ADAPT Centre for Digital Content Technology
Dublin City University, School of Computing
Dublin, Ireland
Email: [nmani|mhelfert]@computing.dcu.ie

Claus Pahl

Free University of Bozen-Bolzano
Faculty of Computer Science
Bolzano, Italy
Email: claus.pahl@unibz.it

***Abstract*—Business process models are abstract descriptions and as such should be applicable in different situations. In order for a single process model to be reused, we need support for configuration and customisation. Often, process objects and activities are domain-specific. We use this observation and allow domain models to drive the customisation. Process variability models, known from product line modelling and manufacturing, can control this customisation by taking into account the domain models. While activities and objects have already been studied, we investigate here the constraints that govern a process execution. In order to integrate these constraints into a process model, we use a rule-based constraints language for a workflow and process model. A 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 and implemented by a customisation engine. The key contribution is a conceptual definition of a domain-specific rule variability language.**

***Keywords*–Business Process Modelling; Process Customisation; Process Constraints; Domain Model; Variability Model; Constraints Rule Language; Rule Generation.**

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 emergence of business processes as a services in the cloud context (BPaaS) highlights the need to implement a reusable process resource together with a mechanism to adapt this to consumers [1].

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 [2] 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 semantic properties can be checked by formal detection methods and tools.

Our contribution is a model development approach using of a feature model to bridge between a 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, bridging between domain model and rule language. The novelty lies in the use of software product line technology to customise processes.

We use digital content processing here as a domain context to illustrate the application of the proposed domain-specific technique (but we will 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. We also 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 discuss the State-of-the-Art and Related Work in Section II. Here, we review process modelling and constraints to position the paper. In Section III, we introduce content processing from a feature-oriented DSL perspective. Section IV introduces rule language background and ideas for a domain-based rule language. We then discuss formal process models into which the rule language can be integrated. Then, we describe the implementation in Section V and evaluate the solution in Section VI.

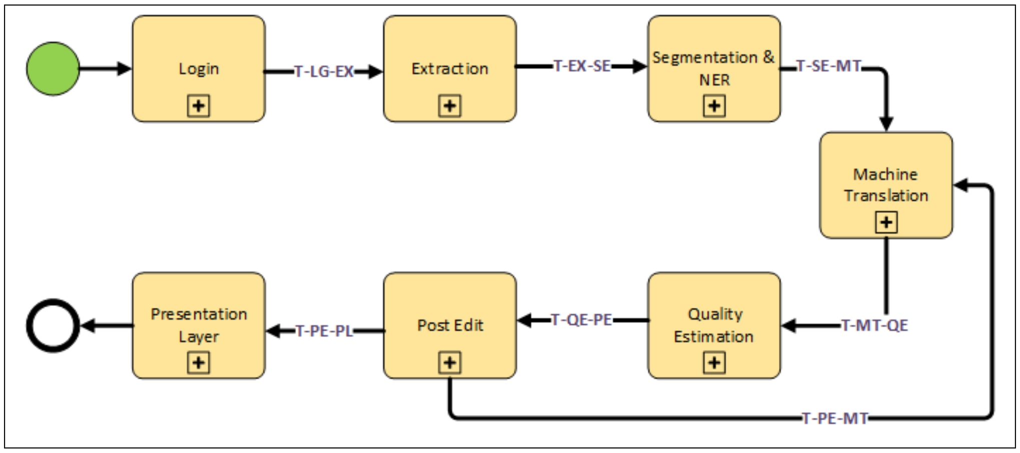

Figure 1. Sample content lifecycle process.

II. State-of-the-Art and Related Work

Current open research concerns for process management include 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.

A. SPL and Variability Modeling

Recently, many researchers have started applying software product line (SPL) concepts in service-oriented computing [3], [4], [5], [6]. We focus on approaches that used the SPL technique for process model configuration. For instance, [7] proposes a BPEL customization process, using a notion of a variability descriptor for modeling variability points in the process layer of service-oriented application.

There are many different approaches of process based variability in service compositions, which enable reuse and management of variability and also support Business Processes [8], [9]. Sun [9] proposes an extended version of COVAMOF; the proposed framework is based on a UML profile for variability modeling and management in web service based systems of software product families. PESOA [10] is variability mechanism represented in UML (activity diagram and state machines) and BPMN for a basic process model, which has non-functional characteristics, like maintenance of the correctness of a syntactical process. Mietzner et al. [7] propose variability descriptors that can be used to mark variability in the process layer and related artifacts of a SaaS application. The SaaS application template allows to customise processes.

B. Dynamic BPEL/BPMN Adaptation

There is related work in the field of constraints and policy definition and adaptive BPEL processes. While here 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 [11] and [12] 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 [40] 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 [2] 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 [14]. We propose a classification of several quality and governance constraints elsewhere [15]: authorisation, accountability, workflow governance and quality. This takes the BPMN constraints extensions [16], [11] 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. [17], [18], [19] discuss business process variability, though primarily from a structural customisation perspective. However, [17] also uses an ontology-based support infrastructure [20].

Several research works related to dynamic adaptation of service compositions have tended to implement variability constructs at the language level [21]. For example, VxBPEL [22] is an extension of the BPEL language allowing to capture variation points and configurations to be defined for a process in a service-centric system. SCENE [23] is also a language for composition design which, extends WS-BPEL by defining the main business logic and Event Condition Action (ECA) rules that define consequences to guide the execution of binding and rebinding self-configuration operations. Rules are used to associate a WS-BPEL workflow with the declaration of the policy to be used during (re)configuration.

C. Configuration Process Models and Templates

Recent years have resulted in a rising interest in supporting flexibility for process model activities. Most process design techniques lead to rigid processes where business policies are hard-coded into the process schema, hence reducing flexibility. Flexible process variants can be configured by using rules to a generic process template. This leads to a split the business policy and control flow. This structure can facilitate process variant configuration and retrieval [24], [25].

A multi-layered method for configuring process variants from the base layer is presented in [26]. The Provop approach [27] allows a user to manage and design to create process variants from a base process (i.e., a process template) with various options. Mohan et al. [28] discuss the automatic identification of inconsistencies resulting in the customisation of business process model and configuration procedure. The MoRE-WS tool [4] activates and deactivates features in a variability model. The changed variability model updates the composite models and its services that add and remove a fragment of WS-BPEL code at runtime. However, the tool uses services instead of direct code, but the dependency on programming and code is always associated with it. Lazovik et al. [29] developed a service-based process-independent language to express different customization options for the reference business processes.

Only a few rule language solutions consider the customization and configuration of a process model in a domain-specific environment. An exception is the work of Akhil and Wen [24] where the authors propose an template and rule for design and management of flexible process variant. Therefore, the rule template based configuration can adopt the most frequently used process. Since enterprise business processes change rapidly, the rule-based template cannot be adapted in changing situations. We need a solution that can be operated by non-technical domain experts without a semantic gap between domain expert design and development. The solution should be flexible, easy to adapt and easy to configure in terms of usability. Therefore, we have propose a domain-specific rule language, which resolves the domain constraints during the customisation process and a framework through which non-technical domain users can customise BPM with the generated set of domain-specific rules (DSRs).

D. Positioning the Approach

At the core of our solution is a process model that defines possible behaviour. This is made up of some frame of reference for the system and the corresponding attributes used to describe the possible behaviour of the process [30], [31]. 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. We use rules to combine a condition (constraint) with a resulting action [32], [33]. The target of our rule language (DSRL) is a standard business process notation (as in Figure 1). Rules shall thus be applied at the processing states of the process.

Our application case study is intelligent content processing. Intelligent content is digital content that allows 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 complexities governed by constraints.

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 [2]. 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 [34]. A domain specific language (DSL) is a programming or specification language that supports a particular application domain through appropriate notation, grammar and abstractions [35]. 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 [36], [37], [38], [39]. 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) [40] method. It represents all the configurations (called instances) of a system, focusing on the features that may differ in each of the configurations [41]. We apply this concept to constraints customisation for processes. The Feature Description Language (FDL) [42] is a language to define features of a particular domain. It supports an automated normalization of feature descriptions, expansion to

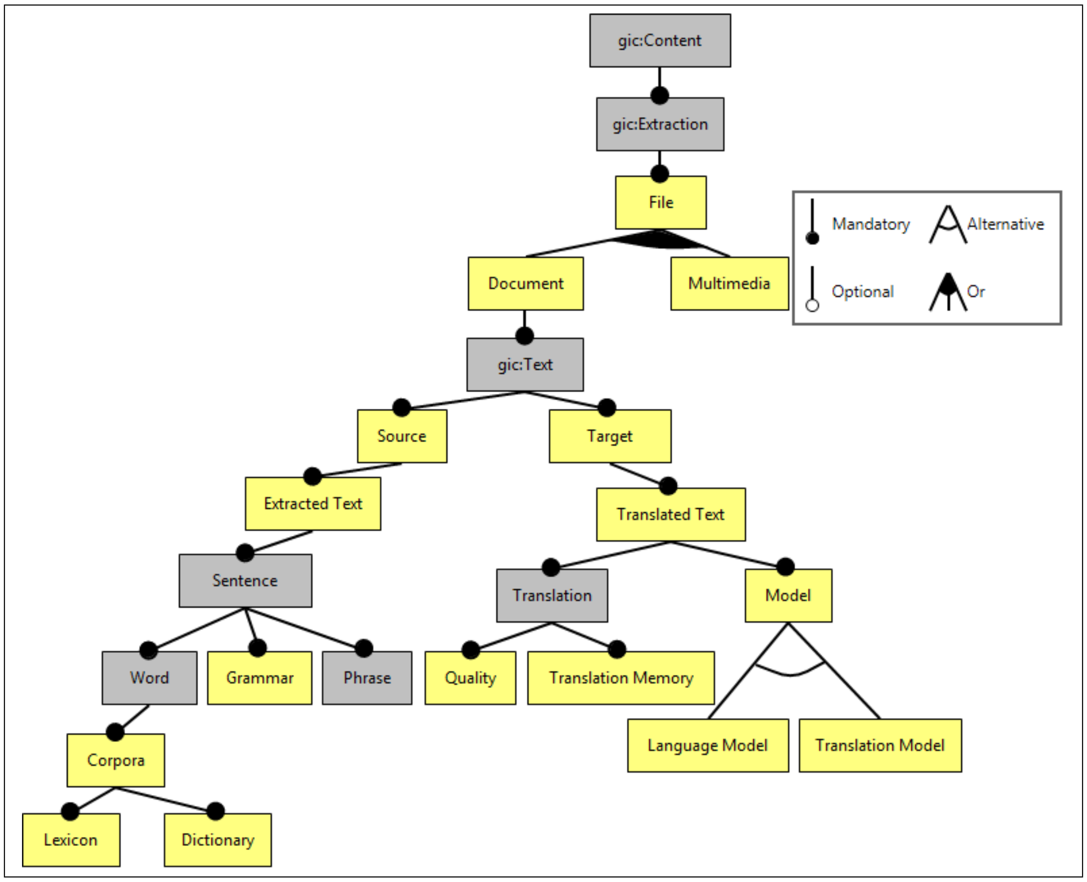

Figure 2. Feature model for intelligent content (note that the darker grey boxes will be detailed further in Figure 3).

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. Here all elements are mandatory. 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 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 could also be made 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 superclass.

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 [41]:

- 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 important for the construction of the rule language (and thus the process customisation) here. 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 [43], [44]. 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 [45], [46]. 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 by adding more details to the feature model. Note that there is an underlying domain ontology here, but we use the class aspects only (i.e., 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, which records at top level only the presence of source and target. The respective Source and Target text strings are included in the respective classes. 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). Extracted Text is text from source content for the purposes target translation. Translated Text is a text after translation. Corpora is a set of structured texts. It may be single or multi language. gic:Sentence is a linguistic unit or combination of words with linked grammar. gic:Translation is content generated by a machine from a source language into a target language. A Grammar is set of structural rules. gic:QualityAssessment is linguistic assessment of translation in term of types of errors/defects. A Translation Memory is a linguistic database that continually captures previous translations for reuse.

Both domain and feature model feed into the process customisation activity, see Figure 4.

IV. CONSTRAINTS RULE LANGUAGE

Rule languages typically borrow their semantics from logic programming [47]. 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 (Ontology Web Language). 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 and content processing activities (e.g., extraction or translation) that process content objects. ECA rules are then defined as follows:

- Event: on the occurrence of an event ...
- Condition: if a certain condition applies ...
- Action: then an action will be taken.

Three sample ECA rule definitions are:

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

The rule model is designed for a generic process. An example shall illustrate ECA rules for ‘extraction’ as the activity. Different cases for extraction can be defined using feature models to derive customised versions:

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

The example below illustrates rule definitions in more concrete XML syntax. Here the rule is that a document must be post-edited before sent for QA-Rating:

```
<p1:Policy policyId="QA-Rate-policy1" priority="0">
  <p1:Objects>
    <p1:ObjectsAnyOf>
      <p1:ObjectsAllOf>
        <p1:Activity>
          <Name>QA-Rate crowd-sourced</Name>
        </p1:Activity>
      </p1:ObjectsAllOf>
    </p1:ObjectsAnyOf>
  </p1:Objects>

  <p1:ActivityStates>
    <p1:ActivityState>Validating-Pre
    </p1:ActivityState>
  </p1:ActivityStates>

  <p1:Rule priority="0"
             ruleId="constraintRule-QA-Rate">
    <p1:Conditions>
      <p1:ConditionExpression
             type="Provenance-Context">
        <p1:Para>//Document/ID</p1:Para>
        <p1:Expr>constraintRule-QA-Rate-Query
        </p1:Expr>
      </p1:ConditionExpression>
    </p1:Conditions>

    <p1:Actions>
      <p1:Pa-Violate>
        <p1:Violation>
          <Type>Functional:Protocol</Type>
        </p1:Violation>
      </p1:Pa-Violate>
    </p1:Actions>

    <p1:FaultHandler>
      <p1:Ca-Log level="5"> </p1:Ca-Log>
    </p1:FaultHandler>
  </p1:Rule>
```

In the example of a rule above, there is one constraint rule and a fault rule (the fault rule details themselves are skipped in the code). The policy (combination of rules) targets the ”QA-Rate crowd-sourced” activity before it is executed. The

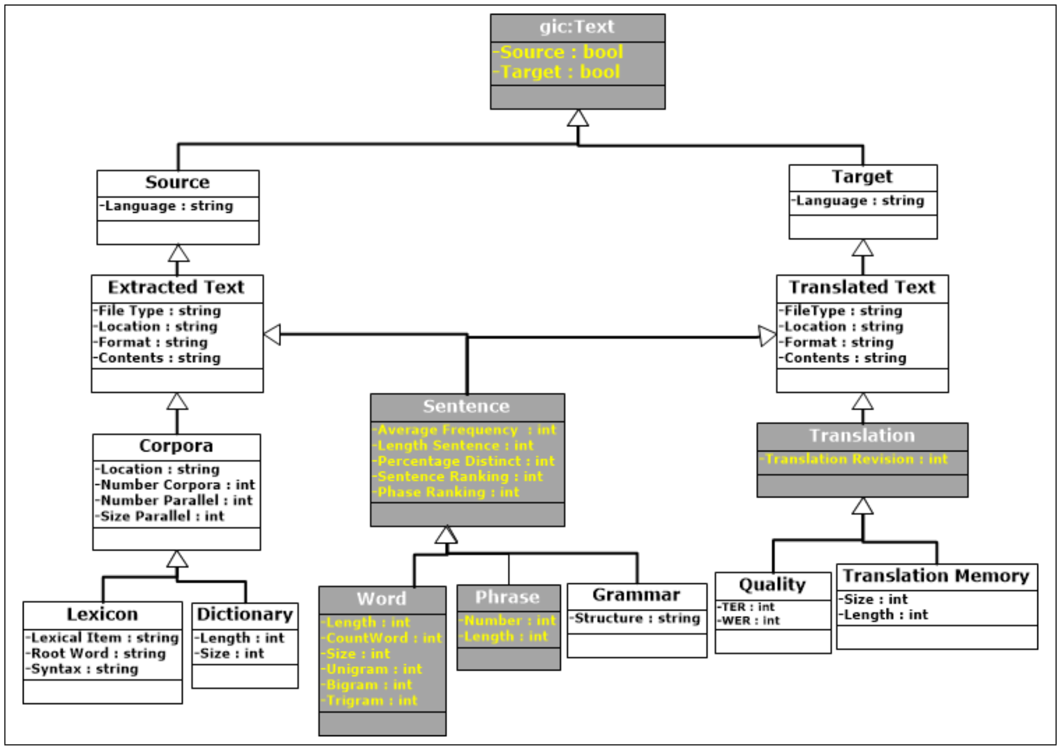

Figure 3. Domain model for intelligent content.

constraint rule has a condition on the provenance context or the document history. A parameterized query (e.g., in SPARQL – Semantic Protocol and RDF Query Language) could check if the current document (using the document ID as parameter) has NOT been post-edited. If the condition is true, then the rule results in a functional:Protocol violation. A fault rule can be defined for handling the violation. The policy will cancel the current process, if no remedy action was found in the fault rule for violation handling.

A. Rule Language Basics

We define the rule language as follows using GLOBIC concepts in the ECA-format with events, conditions and actions (to begin with, we use some sample definitions here to illustrate key concepts before providing a more complete definition later on). The core format of the rule is based on events, conditions and actions. Events are here specific to the application context, e.g., (file) upload, (text) translation or (information) extraction.

gic:Rule ::= [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 from the beginning of the section can be formalised using this notation. Important here is the guidance in defining rules that a domain expert gets through the domain model as a general reference framework and the feature model definition to understand and apply the variability points.

B. Rule Categories for Process Customization

To further understand the rule language, looking at pragmatics such as rule categories is useful. The rules formalised in the rule language introduced above are a syntactical construct. Semantically, we can distinguish a number of rule categories:

- Control flow rules are used for amending the control flow of a process model based on validation or case data. There are several customisation operations, like deleting, inserting, moving or replacing a task. In addition, they are moving or swapping and changing the relationship between two or more tasks.
- Resource rules depend on resource-based actions or validation of processes. They are based on conditional data or case data.
- Data rules are associated with properties or attributes of a resource related to a case.
- Authorisation rules and access control rules, i.e., the rights and roles defined for users, which is a key component in secure business processes that encourages trust for its contributing stakeholders [43].
- An authentication rule expresses the need to verify a claimed identity in an authentication process.
- Hybrid rules concern the modification of several aspects of process design. For example, they might alter the flow of control of a process as well as change the properties of a resource.

C. Control Flow Rule Examples

As an example for the rule language, a few control flow rules (first category above) shall be given for illustration.

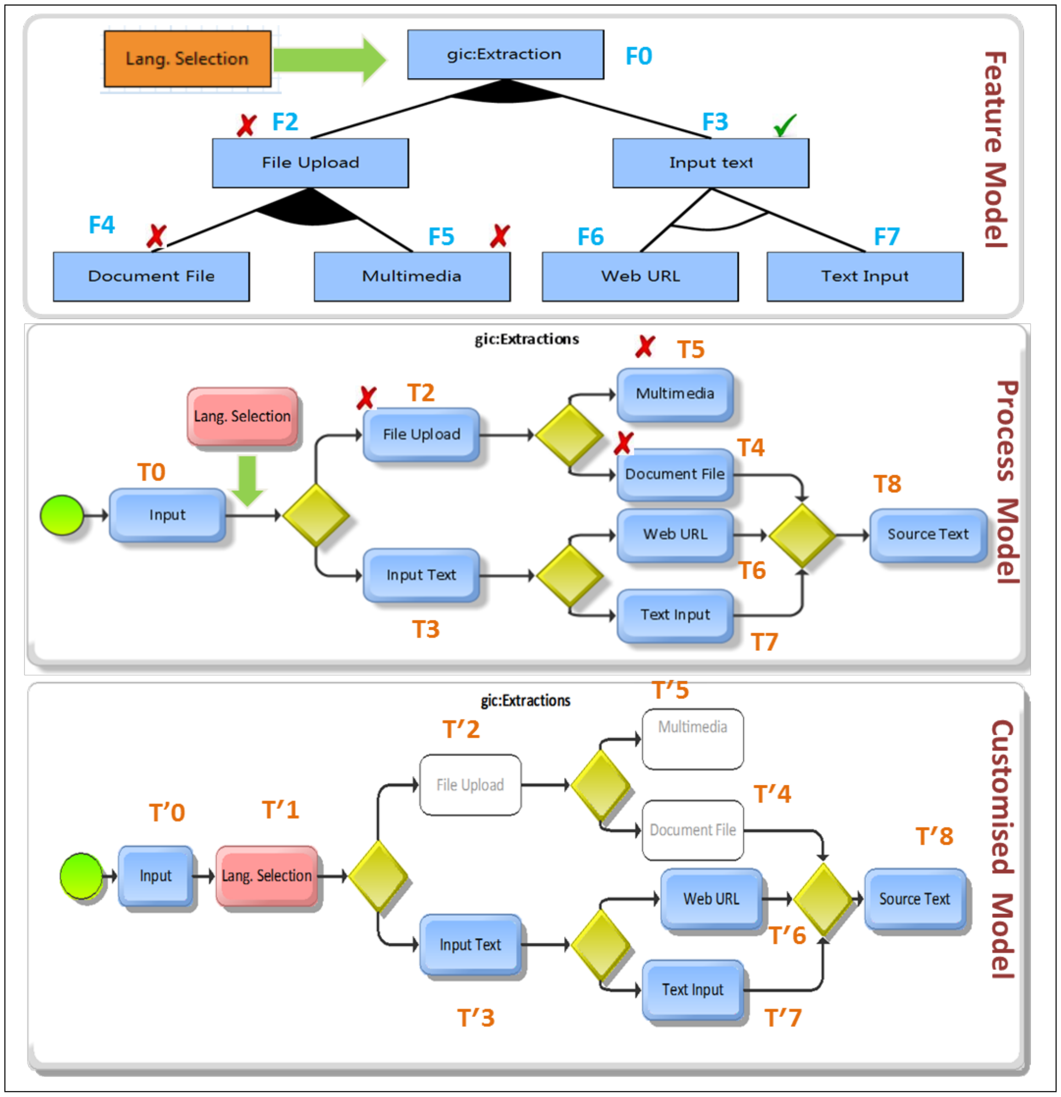

Figure 4. Content process customisation.

```
R1: T0.SourceLang==empty && T0.TargetLang=empty
        -> Insert [T1]
R2: T2.FileType == .txt||html||.xml||.doc||.pdf
        -> Delete [T3]/Deactivate [T5, T3, T6, T7]
R3: T3.TextLength < X
        -> Delete [T2, T4, T5, T6]
        -> Insert [Ttemp.LanguageDetection]
```

Rule R1, R2 and R3 are concerned with a control flow perspective. R1 inserts task T1 in a process model, when source and target language are missing at input (T0). The language selection is a mandatory input task for the Globic process chain and every sub-process has to use it in different aspects. R2 checks the validation of file, e.g., if a user or customer wants to upload a multimedia file, rather than plain text inputs. Therefore, it suggests to Delete T3 or Deactivate T5, T3, T6, T7 from the process model. Similarly, R3 deletes T2, T4, T5 and T6 from the process model, if users want to use input text instead of files, so there is no need for the file upload process. R4, R5, R6 and R7 below are resource flow-related rules and the tasks are based on data cases or validations:

```
R4: T2.FileSize < 5MB
     -> Validation [T2, NextValidation]
R5: T1.SourceLanguage==FR && T1.TargetLanguage=EN
     -> Corpora_Support(T1.SourceLanguage,
                        T1.TargetLanguage,Service)
R6: Ttemp.LanguageDetection != T1.SourceLanguage
     -> Notification (Source language and
                      file text language mismatched)
     -> BackTo([T2],R2)
R7: T6.WebURL != Valid(RegExpr)
     -> Alert(Web URL is invalid !)
     -> BackTo([T3],R4)
```

When the above set of rules is run with use case data, the corresponding rules are fired if their conditions are satisfied. Then, the actions are applied in form of configurations of variants and the process models are customised. Let us assume the sample use case data as follows:

```
fileSize<5 MB; fileType= .txt
```

Then, the rules triggered are R1, R2, R3, R4 and R5. The actions that become valid as a result of these rules are:

```
Action 1: Insert [T1]
Action 2: Delete [T3]/Deactivate [T5, T3, T6, T7]
Action 3: Insert [Ttemp.LanguageDetection]
Action 4: Validation [T2, NextValidation]
Action 5: Corpora_Support(T1.Source,T1.Target,Service)
Action 6: Notification (Exception/Validation message)
```

D. Rule Language Definition

Now we go back to the full rule definition. The Domain Specific Rule Language (DSRL) grammar is defined as follows. We start with a generic skeleton that will then be mapped to the Globic domain model.

```
<DSRL Rules> ::= <EventsList>
                 <RulesList>
                 <ProcessModelList>
<EventLists> ::= <Event> | <Event> <EventLists>
<Event>      ::= EVENT <EventName> IF <Expression> |
                 EVENT <EventName> is INTERN or EXTERN
<RulesList>  ::= <Rule> | <Rule> <RuleList>
<Rule>       ::= ON<EventName>
                 IF<Condition>DO<ActionList>
<ActionList> ::= <ActionName> |
                 <ActionName>,<ActionList>
<ProcessModelList> ::= <ProcessModel> |
                     <ProcessModel>,<ProcessModelList>
<ProcessModel> ::= PROCESSMODEL <ProcessModelName>
```

where a named process model PROCESSMODEL is defined by:

```
[TRANSITION_SEQUENCIAL (DISCARD|DELAY)],
[TRANSITION_PARALLEL (DISCARD|DELAY)]
[INPUTS(<InputList>)] [OUTPUTS(<OutputList>)]
```

TRANSITION_SEQUENCIAL and TRANSITION_PARALLEL are transitions of a workflow.

The description of the DSRL contains lists of events, rules and workflow states. An event can be internal or external (for rules generated as an action, it may be INTERNAL or EXTERNAL) or be generated when the expression becomes true. An event name is activated with the ON expression, which is a Boolean expression to determine the particular conditions that apply and the list of actions that have to be performed when event and condition are matched or true (preceded by DO expression). The workflow contains the state name, a certain policy to be activated in the workflow when sequential and parallel actions to perform. DISCARD allows discarding all the instructions, but the current one and DELAY allows delaying all instructions, but the current one.

```
EVENT gic:FileUpload::BOOL && gic:FileSelect::BOOL
IF    FileUpload_ON
ON    exists
        IF (gic:FileType ==True)
        DO
          ON exists
             IF (gic:FileSize <5MB)
             DO gic:Translate(File)
             ELSE Notification(File size < 5 MB)
        ELSE Notification(File format is invalid)
```

The grammar of the DSRL (in its form specific to the GLOBIC mapping, with events, conditions and actions that are domain-specific) is defined as follows:

List of Events:

```
Event_List ::=
   {gic:File->FileUpload, gic:Text->TextEnd,
    gic:Text->Parsing, gic:Text->MTStart,
    gic:Text->MTEnd, gic:Text-> QARating, ... }
Expr ::= gic:Content.Attributes
```

List of Conditions:

```
<Condition_List>::=<gic:Extraction.Condition>
   | <gic:Segmentation.Condition>
   | <gic:MachineTranslation.Condition>
   | <gic:QualityAssesment>
   | <gic:PostEdit>

<gic:Extraction.Condition> ::=   // EXTRACTION
 IF (<gic:File.FileType(X)::=FileList> )
 | IF (<gic:File.FileSize::=<Y>)
 | IF (<gic:Text.Length::=<L>)
 | IF (<Source.Language::=Language_List>)
 | IF (<Target.Language::=Language_List>)
 | IF (<MultiLanguageText(gic:Text)::=T|F>)
 | IF (<SingleLanguageDetect(gic:Text)::=T|F>)
```

where X is the file type, Y is the size of file (in MB) and L is the length of the text.

```
<gic:Segmentation.Condition)>::=  // SEGMENTATION
IF (<IsDictionaries(Source.Lang)::=T|F>)
| IF (<IsDictionaries(Target.Lang)::=T|F>)
| IF (<IsParCorpus(Source.Lang,Target.Lang)::=T|F>)
| IF (<IsParLexicon(Source.Lang,Target.Lang)::=T|F>)
| IF (<nic:Sentence.WordCount::=<WInteger>)
| IF (<IsTreeParsing(nic:Sentence)::=T|F)>)
```

where WInteger is the word count of the source sentence or target sentence.

```
<gic:Translation.Condition)> ::=  // TRANSLATION
 IF (<gic:Translation(Source.Lang, Target.Lang,
      gic:Text) ::= T|F>)
 | IF (<gic:Translation.Memory ::= <TM)(Mem Underfl)
 | IF (<gic:Translation.Memory ::= >TM)(Mem Overfl)
 | IF (<gic:Translation(gic:TxtSource,Source.Lang)>
        gic:Translation(gic:TxtTarget,Target.Lang)>)

<gic:Quality.Condition> ::=
 IF (<gic:Quality.TER(gic:TextTarget)::=<TERNo>)
 | IF (<gic:Quality.WER(gic:TextTarget)::=<WERNo>)
```

where TM is the specific memory size, TERNo is the Translation Error Rate number and WERNo is the Word Error Rate number.

Source language and target language are elements of a language list. We assume a Language_List(L) = {L1,L2,L3,,Ln} with Source.Language(Ls) ∈ Language_List and Target.Language(Lt) ∈ Language_List.

Actions can be state-specific constraint validations. The process can move to the next state. Acknowledgements are notifications messages to the user. The GIC process is embedded in the *Provenance* context that records Agent, Association and Activity.

List of Actions in the GIC domain:

```
ActionList ::= <Validation->Validation.Next>
               <Process->Next> ,
               <Acknowledgement>
               <Process->Provenance>

<Acknowledgement>::= <Ack_Msg>|<Ack_Msg_List>
```

```
< Ack_Msg_List> ::= <Validation.Msg_Display>
 |<System.Error>
 |<System.Delay>
 |<Process.Active>
 |<Process.Stop>
 |<Process.Abort>
 |<Process.End> (Finished)

<Process->Next> ::= <Process->gic:Text.Extraction>
 |<Process->gic:Text.GrammarCheck>
 |<Process->gic.Text.Name_Entity_Find>
 |<Process->gic:Text.Parsing>
 |<Process-> gic:Text.Segmentation>
 |<Process->gic:Translation>
 |<Process->gic:Quality Assessment>
 |<Process->PostEdit>

<Process->Provenance> ::= <prov:Agent> // PROVENANCE
                          <prov:Association>
                          <prov:Activity>

<prov:Agent> ::= <prov:InstantaneousEvent>
 |<prov:AgentInfluence>
 |<prov:Influence>
 |<prov:EntityInfluence>
 |<prov:Delegation>
 |<prov:Start>
 |<prov:End>
 |<prov:Derivation>
 |<prov:ActivityInfluence>
 |<prov:Quotation>
 |<prov:Generation>
 |<prov:Revision>

<prov:Association> ::= <prov:Role>
 | <prov:Invalidation>
 |<prov:Attribution>

<prov:Activity> ::= <prov:Entity>
 |<prov:Communication>
 |<prov:Plan>
 |<prov:Usage>
 |<prov:PrimarySource>
```

The Global Intelligent Content (GIC) semantic model is based on a abstract model and a content model. The abstract model captures the different resource types that are processed. The content model details the possible formats.

Abstract Model:

```
gic:Domain ->   gic:Resource
gic:Resource -> gic:Services |
                gic:Information Resource |
                gic:IdentifiedBy |
                gic:RefersTo |
                gic:AnnotatedBy
gic:InformationResource -> gic:Content | gic:Data
```

Content Model:

```
gic:Content -> gic:Content | cnt:Content
cnt:Content -> cnt:ContentAsBase64 |
               cnt:ContentAsText|
               cnt:ContentAsXML
cnt:ContentAsBase64-> cnt:Bytes
cnt:ContentAsText ->  cnt:Chars
cnt:ContentAsXML ->   cnt:Rest | cnt:Version |
                      cnt:LeadingMisc |
                      cnt:Standalone |
                      cnt:DeclaredEncoding |
                      cnt:dtDecl
cnt:dtDecl ->      cnt:dtDecl | cnt:DocTypeDecl
cnt:DocTypeDecl -> cnt:DocTypeName |
                   cnt:InternetSubset |
                   cnt:PublicId | cnt:SystemId
```

V. IMPLEMENTATION

While this paper focuses on the conceptual aspects such as models and languages, a prototype has been implemented. Our implementation (Figure 5) provides a platform that enables building configurable processes for content management problems and constraints running in the *Activiti* (http://activiti.org/) workflow engine. In this architecture, a cloud service layer performs data processing using the Content Service Bus (based on the *Alfresco* (https://www.alfresco.com/) content management system).

This implementation is the basis of the evaluation that looks at feasibility and transferability – see Evaluation Section VI. We introduce the business models and their implementation architecture first, before detailing the evaluation results in the next section.

A. Business Process Models

A business process model (BPM) is executed as a process in a sequential manner to validate functional and operational behaviour during the execution. Here, multiple participants work in a collaborative environment based on Activiti, Alfresco and the support services. Policy rule services define a process map of the entire application and its components (e.g., file type is valid for extraction, quality rating of translation). This process model consists of a number of content processing activities such as Extraction & Segmentation, NER, Machine Translation (MT), Quality estimation and Post-Edit.

One specific constraint, access control, shall be discussed separately. An access control policy defines the high-level set of rules according to the access control requirements. An access control model provides the access control/authorization security policy for accessing the content activities as well as security rights implemented in BPM services according to the user role. Access control mechanism enable low-level functions, which implement the access controls imposed by policies and are normally initiated by the model architecture.

Every activity has its own constraints. The flow of entire activities is performed in a sequential manner so that each activity's 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.

B. Architecture

The architecture of the system is based on services and standard browser thin clients. The application can be hosted on a Tomcat web server and all services could potentially be hosted on a cloud-based server. Architecturally, we separate out a service layer, see Figure 5. Reasons to architecturally separate a Service Layer from the execution engine include the introduction of loose coupling and interoperability.

The system has been developed on a 3-tier standard architecture: browser-based front-end thin clients, Tomcat Application server-based middleware, distributed database service as data service platforms. We follow the MVC (Model View

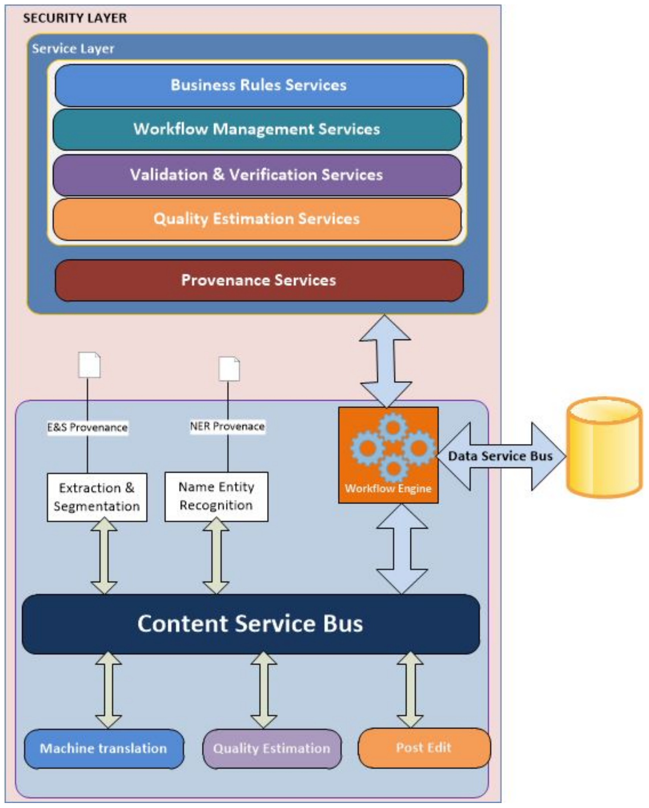

Figure 5. Prototype implementation architecture.

Controller) architecture. Multiple technologies are used to integrate each component of the content management solution:

- Common Bus/Hub: Alfresco is providing a common bus platform for all the activities.
- Application connectivity: Activiti and a cloud service layer play an important role to solve connectivity issues in the architecture.
- Data format and transformation: By using web services and other APIs, we maintain a common format for the entire application.
- Integration module: This module connects different sections of the application: Activiti, data and service bus, cloud service layer, Alfresco, and databases.

VI. Evaluation

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.

In this evaluation, we look at utility, transferability and feasibility. We balance this discussion by a consideration of restrictions and limitations.

A. *Utility*

The general utility is demonstrated empirically. The domain and feature models here specifically support domain experts.

Process and Cohort. We have worked with seven experts in the digital media and language technology space as part of our research centre. While these were largely researchers, their background was language technology and content management and all had development experience in that space, some also industrial experience. These experts were also from different universities. Despite being researchers, they act as application domain specialists in our case, being essentially language technology experts, but not business process experts. In total, seven expert have contributed to this process. However, we counted them as multipliiers as each of them had worked with other researchers, developers and users in industry.

Mechanism. The qualitative feedback, based on the expert interviews as the mechanism, confirms the need to provide a mechanism to customise business processes in a domain-specific way. We asked the participants about their opinion on the expected benefit of the approach, specifically whether this would lead to improved efficicency in the process modelling activities and whether the approach would be suitable for a non-expert in business modellling, with background in the application domain.

Results. The results of the expert interview can be summarised as follows:

- The experts confirm with majority (71%) that 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 (86%) in the evaluation have confirmed simplification or significant simplification in process modelling.

This confirms our hypothesis in this research laid out at the beginning.

B. *Transferability*

In addition, we looked at another process domain to assess the transferability of the solution [48]. Learning technology as another human-centred, domain-specific field was chosen.

Application Domain. In the learning domain, we examined learner interaction with content in a learning technology system [49], [50]. Again, the need to provide domain expert support to define constraints and rules for these processes became evident.

Observations. Here, educators act as process modellers and managers [15], 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.

C. *Feasibility Analysis*

From a more technical perspective, we looked at the feasibility of implementing a production system from the existing prototype. The feasibility study (analysis of alternatives) is used to justify a project. It compares the various implementation alternatives based on their economic, technical and operational feasibility. The steps of creating a feasibility study are as follows [41]:

Determine implementation alternatives. We have discussed architectural choices in the Implementation.

Assess the economic feasibility for each alternative. The basic question is how well will the software product pay for *itself?* This has been looked at by performing a cost/benefit analysis. In this case, using open-source components has helped to reduce and justify the expenses for a research prototype.

Assess the technical feasibility for each alternative. The basic question is the possibility to build the software system. The set of feasible technologies is usually the intersection of the aspects implementation and integration. This has been demonstrated through the implementation with Activiti and Alfresco as core platforms that are widely used in practice.

D. Restrictions, Limitations and Constraints

The concerns below explain aspects, which impact on the specification, design, or implementation of the software system. These items may also contribute to restrict the scalability and performance of the system as well. Because of either the complexity or the cost of the implementation, the quality or delivery of the system may suffer.

Constraints that have impacted on the design of our solution are the following:

- The information and data exchanging between two activities during workflow processing.
- User management and security of overall system including alfresco CMS, workflow engine, rule engine and CNGL2 challenges.
- Validation of different systems at runtime.
- Interoperability between features and functions.
- Major constraint utilisation of translation memory and language model through services.
- Process acknowledgement or transaction information sharing between different activities of workflow.
- Error tracking and tracing during transaction.

VII. Conclusions

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. This benefits as follows:

- Often, business processes take domain-specific objects and activities into account in the process specification. Our aim is to make the process specification accessible to domain experts. 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 technical contribution core is a rule generation technique for process variability and customisation. The novelty of our approach is a focus on process constraints and their rule-based management, advancing on structural variability. The result is flexible customisation of processes through constraints adaptation, rather than more intrusive process restructuring.

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) [51]. Furthermore, self-service provisioning of resources also requires non-expert to manage this configuration. BPaaS relies on providing processes as customisable entities. Targeting constraints as the customisation point is clearly advantageous compared to customisation through restructuring. For BPaaS, if a generic service is provided to external users, the dynamic customisation of individual process instances would require the utilisation of a coordinated approach, e.g., through using a coordination model [52], [53]. Other architecture techniques can also be used to facilitate flexible and lightweight cloud-based provisioning of process instances, e.g., through containerisation [54].

We also 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, 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] N. Mani and C. Pahl, "Controlled Variability Management for Business Process Model Constraints," International Conference on Software Engineering Advances ICSEA'2015, pp. 445-450. 2015.

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

[3] M. Asadi, B. Mohabbati, G. Groner, and D. Gasevic, "Development and validation of customized process models," Journal of Systems and Software, vol. 96, pp. 73-92, 2014.

[4] G. H. Alferez, V. Pelechano, R. Mazo, C. Salinesi, and D. Diaz, "Dynamic adaptation of service compositions with variability models," Journal of Systems and Software, vol. 91, pp. 24-47, 2014.

[5] J. Park, M. Moon, and K. Yeom, "Variability modeling to develop flexible service-oriented applications," Journal of Systems Science and Systems Engineering, vol. 20, pp. 193-216, 2011.

[6] M. Galster and A. Eberlein, "Identifying potential core assets in service-based systems to support the transition to service-oriented product lines," in 18th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS), pp. 179-186. 2011.

[7] R. Mietzner and F. Leymann, "Generation of BPEL customization processes for SaaS applications from variability descriptors," in IEEE International Conference on Services Computing, pp. 359-366. 2008.

[8] T. Nguyen, A. Colman, and J. Han, "Modeling and managing variability in process-based service compositions," in Service-Oriented Computing, Springer, pp. 404-420, 2011.

[9] C.-A. Sun, R. Rossing, M. Sinnema, P. Bulanov, and M. Aiello, "Modeling and managing the variability of Web service-based systems," Journal of Systems and Software, vol. 83, pp. 502-516, 2010.

[10] F. Puhlmann, A. Schnieders, J. Weiland, and M. Weske, "Variability mechanisms for process models," PESOA-Report TR17, pp. 10-61, 2005.

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

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

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

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

[16] 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.

[17] 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, IEEE, 2010.

[18] Y. Huang, Z. Feng, K. He, and Y. Huang, ”Ontology-based configuration for service-based business process model,” In: IEEE International Conference on Services Computing, pp. 296303. 2013.

[19] N. Assy, W. Gaaloul, and B. Defude, ”Mining configurable process fragments for business process design,” In: Advancing the Impact of Design Science: Moving from Theory to Practice, DESRIST’2014. LNCS 8463, pp. 209224. 2014.

[20] M. Javed, Y. Abgaz and C. Pahl, ”A Pattern-based Framework of Change Operators for Ontology Evolution,” 4th International Workshop on Ontology Content OnToContent’09. 2009.

[21] C. Pahl, ”A Pi-Calculus based framework for the composition and replacement of components,” Conference on Object-Oriented Programming, Systems, Languages, and Applications OOPSLA’2001 Workshop on Specification and Verification of Component-Based Systems, 2001.

[22] M. Koning, C.-A. Sun, M. Sinnema, and P. Avgeriou, ”VxBPEL: Supporting variability for Web services in BPEL,” Information and Software Technology, vol. 51, pp. 258-269, 2009.

[23] M. Colombo, E. Di Nitto, and M. Mauri, ”Scene: A service composition execution environment supporting dynamic changes disciplined through rules,” in Service-Oriented Computing, pp. 191-202. 2006.

[24] A. Kumar and W. Yao, ”Design and management of flexible process variants using templates and rules,” Computers in Industry, vol. 63, pp. 112-130, 2012.

[25] D. Fang, X. Liu, I. Romdhani, P. Jamshidi, and C. Pahl, ”An agility-oriented and fuzziness-embedded semantic model for collaborative cloud service search, retrieval and recommendation,” Future Generation Computer Systems, Volume 56, pp. 11-26. 2016.

[26] M. Nakamura, T. Kushida, A. Bhamidipaty, and M. Chetlur, ”A multi-layered architecture for process variation management,” in World Conference on Services-II, SERVICES’09, pp. 71-78, 2009.

[27] A. Hallerbach, T. Bauer, and M. Reichert, ”Capturing variability in business process models: the Provop approach,” Jrnl of Software Maintenance and Evolution: Research and Practice 22, pp. 519-546, 2010.

[28] 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.

[29] A. Lazovik and H. Ludwig, ”Managing process customizability and customization: Model, language and process,” in Web Information Systems Engineering, 2007, pp. 373-384.

[30] M. Helfert, ”Business informatics: An engineering perspective on information systems.” Journal of Information Technology Education 7:223-245. 2008.

[31] P. Jamshidi, M. Ghafari, A. Ahmad, and C. Pahl, ”A framework for classifying and comparing architecture-centric software evolution research,” European Conference on Software Maintenance and Reengineering, 2013.

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

[33] 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.

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

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

[36] 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.

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

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] C. Pahl, ”A Formal Composition and Interaction Model for a Web Component Platform,” Electronic Notes in Theoretical Computer Science, Volume 66, Issue 4, Pages 67-81, Formal Methods and Component Interaction (ICALP 2002 Satellite Workshop), 2002.

[44] C. Pahl, S. Giesecke, and W. Hasselbring, ”Ontology-based Modelling of Architectural Styles,” Information and Software Technology, vol. 51(12), pp. 1739-1749, 2009.

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

[46] 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.

[47] 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.

[48] M. Helfert, ”Challenges of business processes management in healthcare: Experience in the Irish healthcare sector.” Business Process Management Journal 15, no. 6, 937-952. 2009.

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

[50] 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.

[51] C. Pahl and H. Xiong, ”Migration to PaaS Clouds - Migration Process and Architectural Concerns,” IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems MESOCA’13. IEEE. 2013.

[52] E.-E. Doberkat, W. Franke, U. Gutenbeil, W. Hasselbring, U. Lammers, and C. Pahl, ”PROSET - a Language for Prototyping with Sets,” International Workshop on Rapid System Prototyping, pp. 235-248. 1992.

[53] F. Fowley, C. Pahl, and L. Zhang, ”A comparison framework and review of service brokerage solutions for cloud architectures,” 1st International Workshop on Cloud Service Brokerage (CSB’2013). 2013.

[54] C. Pahl, ”Containerisation and the PaaS Cloud,” IEEE Cloud Computing, 2(3). pp. 24-31, 2015.