A Configuration-based Domain-specific Rule Generation Framework for Process Model Customization

Neel Mani, B.Sc., M.Sc., M.Tech.

A dissertation submitted in fulfilment of the requirements for the award of
Doctor of Philosophy (Ph.D.)

Dublin City University
School of Computing

Supervisors: Dr. Markus Helfert
and
External Supervisor: Dr. Claus Pahl
Faculty of Computer Science
Free University of Bozen, Bolzano,
Italy

September 2018

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: [illegible] (Candidate) ID No.: 13211190 Date: 03/09/2018

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF PUBLICATIONS

	Publication Details	Chapters
Journal		
J1	**Mani, Neel**, Helfert, Markus and Pahl, Claus (2016), *Business Process Model Customisation using Domain-driven Controlled Variability Management and Rule Generation*, International Journal on Advances in Software, vol. 9, pp. 179 - 190, 2016.	Chapters-6, 8
Book Chapter		
B1	**Mani, Neel**, Helfert, Markus, Pahl, Claus, Nimmagadda, Shastri and Vasant, Pandian (2017), *Domain Models Definition for Rule Generation Using Controlled Variability Management*, Computational Intelligence, Innovative Computing, Optimization and Its Applications.	Chapters-5, 6, 7, 9
Conferences		
C1	**Mani, Neel**, Helfert, Markus, and Pahl, Claus (2017), *A Framework for Generating Domain-specific Rule for Process Model Customisation*, In, International Conference on Computer-Human Interaction Research and Applications (CHIRA), 31 Oct, 1-2 Nov 2017, Funchal, Maderia- Portugal.	Chapters-4,5
C2	**Mani, Neel**, Helfert, Markus and Pahl, Claus (2017), *Domain-specific Generation Using Variability for Business Process Model Constraint,* In, 21st International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, 06-08 Sep 2017, Marseille, France.	Chapters-7
C3	**Mani, Neel**, and Helfert, Markus *Domain Model Definition for Transformation of Rule Language*, In, *International* Conference on Communication, Management and Information Technology ICCMIT 2017, 3-5 April 2017, Warsaw, Poland.	Chapters-5,7
C4	**Mani, Neel** and Pahl, Claus (2015) *Controlled variability management for business process model constraints.* In, International Conference on Software Engineering Advances ICSEA'2015, 15-20 Nov 2015, Barcelona, Spain. ISBN 978-1-61208-438-1	Chapters-4,5
C5	Pahl, Claus and **Mani, Neel** (2014) *Managing Quality Constraints in Technology-managed Learning Content Processes*. In, EdMedia'2014 World Conference on Educational Media and Technology 2014 (pp. 985-990). Jun 23, 2014 in Tampere, Finland.	Chapters-4
C6	Pahl, Claus and **Mani, Neel** and Wang, Ming-Xue (2013) *A Domain-Specific Model for Data Quality Constraints in Service Process Adaptations*. In, 3rd International Workshop on Adaptive Services for the Future Internet (WAS4FI 2013), 11 Sept 2013, Malaga, Spain.	Chapters-2, 4

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
BNF	Backus-Naur form
BP	Business Process
BPM	Business Process Management
BPMN	Business Process Model and Notation
CIM	Computation Independent Models
DCT	Digital Content Technology
DSC	Domain-specific Constraint
DSPL	Dynamic Software Product Line
DSPLE	Dynamic Software Product Line Engineering
DSR	Domain-specific Rule
DSRG	Domain-specific Rule Generation
DSRL	Domain-specific Rule Language
EBNF	Extended Backus-Naur form
ECA	Event Condition Action
GPL	General Programming Language
DSRG	Domain-specific rule generation
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardisation
IT	Information Technology
IS	Information Systems
MDA	Model-driven Architecture
OMG	Object Management Group
OWL	Web Ontology Language
PC	Presence Conditions
PIM	Platform Independent Models
PLE	Product Line Engineering
PM	Platform Models
PSM	Platform Specific Models
QoS	Quality of Service

SDF	Syntax Definition Formalism
SLR	Systematic Literature Review
SOA	Service-Oriented Architecture
SPL	Software Product Line
SPLE	Software Product Line Engineering
UML	Unified Modeling Language
V&V	Validation and Verification
W3C	World Wide Web consortium
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XPATH	XML Path Language

ACKNOWLEDGEMENTS

These four years of journey has enriched my life in many ways. Intellectually, it has added a volume to my knowledge and emotionally, I have met some amazing persons and have gathered beautiful moments that I would cherish for life. This has been an opportunity for me, not only to explore a new field of knowledge, but also a wonderful country with distinguishing cultures.

When it comes to acknowledging the contributions of others in my thesis, I think the most valuable lesson a researcher learns in the process of research is how kind and helpful the people are around him. My deepest gratitude goes to my supervisors, Markus Helfert and Claus Pahl for being the constant source of motivation, support, knowledge and encouragement from the very beginning till date. Also, I would like to take this opportunity to thank all my colleagues at Centre for Next Generation Localization and ADAPT Centre, for supporting me throughout this study.

I was also very lucky to have received valuable input from accomplished individuals, like, Piyush, Shastri, Eva, and Dimi. Piyush has rendered help from the very first day and has enriched my understanding of the subject with his valuable comments, guidance, and encouragement at each phase of this research. Gopal supported at the initial phase, when I probably needed it most. Shastri, Eva and Dimi have done a great job by proofreading this thesis. Here goes a BIG thanks to the members of the Business Informatics Group (BIG).

I consider myself to be a blessed person. Destiny has bestowed upon me a life full of challenges and has eventually given me the power to conquer each of them, obtaining valuable lessons and immense knowledge in the process. I cannot thank enough my family, especially my wife Pallavi and my daughter Visalakhshi. They have been a constant source of inspiration and encouragement, making it possible for me to focus on my work, when it also meant defocusing my attention from them. I consider this accomplishment to be theirs as well. I hope to stand up to their sacrifices and fill their lives with joy as much as possible. I would also thank my parents for their unconditional love and support throughout my life.

ABSTRACT

"A Configuration-based Domain-specific Rule Generation Framework for Process Model Customization"

Neel Mani

In today's changing world, there is an ever-increasing demand and need for software reuse in applications, where the process model needs to be reused in different applications in a domain-specific environment. The process model is required to adapt and implement changes promptly at run-time, in response of the end-user configuration requirements. Furthermore, reusability is emerging strongly as a necessary underlying capability, particularly for customization of business in a dynamic environment where end-users can select their requirements to achieve a specific goal. Such adaptations are in general, performed by non-technical end-users which can lead to losing a significant number of person-days and which can also open up possibilities to introduce errors into the system. These scenarios call for - indeed cry out for - a system with a configurable and customizable business process, operable by users with limited technical expertise.

Research aims to provide a framework for generating the rule language and configuring domain constraints. This framework builds upon the core idea of Software Product Lines Engineering (SPLE) and Model-Driven Architecture (MDA). The SPLE provides a platform that includes the variability model. Variability models offer features where end-users can select features and customize possible changes in the domain template, which is the container for domain and process models. The user selects their requirements as a feature from feature models and generates rules from domain models using MDA. Then, the generated rules are translated from a high-level domain model, based on the requirements of the end-user. On the other hand, the weaving model is responsible for reflecting activation and de-activation of features of variabilities in the domain template.

The usability of the proposed framework is evaluated with a user study in the area of Digital Content Technology. The results demonstrate that usability improvements can be achieved by using the proposed techniques. The framework can be used to support semi-automatic configuration that is efficient, effective and satisfactory.

CHAPTER 1

INTRODUCTION

1.1 Motivation

In today's dynamic and competitive business environment, organizations must respond rapidly to changes in the way their businesses operate [1, 2]. The competitive edge of an organization will only remain intact if it can adapt quickly to new environments and any challenges they present. Such challenges and changes may be triggered through external entities or internal stakeholders. For example, changes in customer demands and preferences, or amendments to various laws, require modification of business strategies. In addition, enterprises face the challenge of the existence of extensive collections of process schemas [3, 4] (i.e., control flow description of a process) of the process model. One of the main challenges of the process model is that have the schema variants have only minor differences between them. For instance, language-based technologies have a different type of processes, to translate from one language (source) into another language (target). A new process model is required for a new language application system that can use a fixed source language, e.g. English, and can produce output in multiple languages (e.g., German, French, Mandarin etc.). However, for the type of process model[1] used, the process schema remains the same, only the process model activities may have minor changes in terms of update actions (move, replace or delete) or insert actions. Nevertheless, the multi-variants[2] approach such as single model and the multi-model create a variant, by duplicating a process model, and adjusting it to fulfill specific needs. The single model makes the process difficult to complex to understand.

[1] Process model refers to a structural representation, description or diagram, which defines a specified (data, control, process) flow of activities for a particular organizational and business unit
[2] The multi variants scenarios are handled with two approaches, i.e., single-model and multi-model. The single model captures multiple variants into single approach through conditional branch (IF-ELSE) and multi-model combined with all variants in multiple processes.

Another set of examples is from the Digital Content Technology (DCT) domain and relates to process customization. For instance, the domain experts may need to introduce a new data extraction source model, such as document, multimedia or both, which will be used as an input for a multilingual “machine translation process” without interrupting an old process model. Since there are certain activities in the process that are mandatory, and cannot be excluded from the overall process model, such as the data extraction source (i.e., upload the document or the input text) and its sub-processes which are working simultaneously without interruption, even if the new data source is significantly different from old service. Another example of domain process customization is when a user selects the wrong source for a language, i.e., one which is not compatible with source text. In this case, they may want to add a language identifier process before the source data selection.

Nowadays, researchers are focusing on process model customization [3, 5-10]. They provide solutions to customize process models in order to address the problem of an increasing number of changes in the business requirements. These changes force the organization to adapt these changes promptly [11-13]. However, the domain experts are only able to design high-level, functional parts of the process model. The major limitation of current research is the dependency on General-purpose Languages (GPL) (e.g., Java, C#, C, C++) because the domain experts often do not possess sufficient knowledge of high-level GPLs. Therefore, they become dependent on existing solutions which do not allow them to work independently.

Motivated by the aforementioned problems, this thesis proposes a framework for non-technical users to generate low-level rules from high-level domain models for process customization.

1.2 Problem Statement

Process model languages [14-18] provide expressive as well as multiple verification techniques, (e.g., Petri nets (PNs)) which ensure that processes [19] are reliable in terms of schema designs. However, the languages can restrict domain experts in the changes they can make, such as task control flow, data flow and work/process allocation schema. These are the pre-defined execution plans in process model language. The changes are made at the modeling stage or the design phase, which results in a rigid process model [19-21]. Due to process model rigidity, it becomes challenging to customize, adapt and maintain the process. The process model languages limit flexibility of enterprises [22-24] as they may not be suitable in a dynamic environment; the nature of organizations is often volatile and processes are excessively rigid [19-21, 24, 25].

Therefore, enterprises are looking for new configurable solutions. Such solutions should be domain-specific in order to make the process model dynamically adaptable in terms of the process execution plan. The configuration solution would also simultaneously reduce dependency on programming, and consequently software developers.

As discussed above, the research challenge lies in the rigidity of the process language and its dependency on GPLs, which limit their usage to technical users. The purpose of this research is to develop an extended version of rule language and Domain-specific language (DSL) to overcome GPL limitations [26-29]. The DSLs are tailored language for a particular domain or application [28].

1.3 Research Questions and Objectives

The critical problems discussed in the motivation and problem statement section lead to addressing the following research question:

How to develop an end-user usable framework to generate and configure a domain-specific rule (DSR) to customize process model dynamically?

The primary research question can be further subdivided into three sub-questions:

RQ 1. How to develop a rule generation and configuration framework to customize the process model dynamically?

The aim of this research question is to develop a framework in a dynamic environment, where the process model can be adapted by the end-users, with the dynamic generation of the rules and configuration of the domain. The design time customizations, required by single (multiple conditions applied- IF-ELSE) and multi-model (multiple time use) variant [3] are complex, difficult to understand and time-consuming for technical people. This compels a research goal, of a framework for handling the configuration and customization challenges of process models at run-time.

RQ 2. How to implement a framework to support a domain-specific rule language that is usable by non-technical domain experts?

The main goal of this research question is to define a rule language, which can help a non-technical user[3] configure the values of domain constraint parameters. However, it should be noted that the definition of rule language and configuration of domain constraint parameters, as discussed above, is complex and time consuming; even for technical experts at design time. As already discussed in RQ1, this is a challenge for the end-users, as they require technical expertise to configure domain constraints and customize the process model. This question allows us to develop a framework for implementing configuration and customization which can be used conveniently by non-technical end-users.

RQ 3. How can a framework be implemented that meets end-users usability requirements?

The primary aim of this research question is to develop a validating strategy for a framework that meets the usability criteria. Manual configuration and customization are complex and error-prone, as well as time consuming at run-time. This question prompts us to evaluate the usability of a prototype framework in terms of efficiency, effectiveness, and satisfaction.

[3] The user does not have software development skill, but they have expertise of process model and domain knowledge.

Consequently, a solution is developed for evaluating the usability of the framework and this usability evaluation can be defined by three main hypotheses.

- **Evaluation Hypothesis 1**- *Measuring Run-time configuration efficiency:* The approach of generating rule configurations can be compared with manual and semi-automatic configuration at the time of configuring the domain parametric constraints. The end-user must configure the generated rule in a time efficient way. Therefore, to ensure this approach, manual configuration must be compared to semi-automatic configuration with respect to the time taken for the end-user to complete the tasks. A semi-automatic approach is more efficient than manual rule configuration.

- **Evaluation Hypothesis 2**- *Measuring Run-time configuration effectiveness:* Semi-automatic approach is more effective in terms of preventing errors and improving qualities of rule configuration; the application of this approach includes customization and configuration.

- **Evaluation Hypothesis 3**- *Satisfaction evaluation:* The framework promotes a high-level of satisfaction to the end-user, as subjective scores show. Such scores are computed to incorporate different scales of end-user satisfaction. This is achieved with the inclusion of varying levels of questions to comprehend the satisfaction in terms of these scores, evaluated under the System Usability Score (SUS) [30].

These hypotheses fulfill usability criteria in general and specifically for the problems considered in this thesis. However, they do not explicitly encompass the entire spectrum of such criteria and therefore may take marginally different forms, depending on the problem under consideration. Fulfillment of these hypotheses is a challenging yet key task for validation of the solution, and therefore the research methodology itself.

A framework is proposed that allows a non-technical domain expert to customize the process model without knowledge of a technical (development and deployment) language. The framework addresses two challenges. The first consists of high-level, domain model knowledge transfer from a domain concept to a low-level rule language. The second challenge faced is the configuration of DSR in the process model languages. A domain-specific approach offers a dedicated solution for a defined set of problems. To address these challenges in relation to the three sub-research questions, the specific objectives of this research are formulated:

- To develop a framework which allows non-technical domain users to customize and configure the process model.
- To develop a Domain-specific Rule Generation (DSRG) prototype by using variability management.
- To define a domain-specific rule language based on the principle of DSL.
- To perform automated rule generation, configuration, and process model customization.
- To validate the framework based on scientifically established hypotheses.

This thesis facilitates the end-user to generate rules from a domain model to customize and configure the process model in a dynamic environment. It proposes a framework for rule generation which supports the domain-specific process model and its scope and implementation in the Digital Content Technology (DCT) area. The aim is to automatically generate the domain-specific rules (DSR) after the end-user selects the features, and configures and customizes the process model. The framework is expected to support the domain-specific process constraint management. Furthermore, we define a Domain-specific Rule Language (DSRL) for the domain-specific environment.

1.4 Research Contributions

The main contributions of this research are to propose a viable framework prototype, where (1) a non-technical domain expert can generate low-level rules from a high-level domain model, and (2) a DSR is generated to customize and configure the process model in a domain-specific environment. The DSR generation enhances the efficiency of development and customization of the business process applications. This opens an avenue to minimize the work of the domain experts so that they can focus on high-level design problems, while simultaneously meeting goals of the desired domain application. This can be achieved by selecting features from a feature model which bridges the gap between a domain model (ontology) and DSR. The feature model streamlines the end-user requirements to achieve the desired target process model. It acts as a bridge, capturing the requirements of the target process model (i.e. customizing the process model and generating the DSR) based on these requirements. Following are the major contributions of the thesis:

1. A domain-specific rule generation (DSRG) framework for translating a set of rules from high-level models (domain models) on an ad-hoc basis.
2. Defining a Domain-specific Rule Language (DSRL) to translate a high-level graphical domain model into a low-level text model.
3. Implementing the model-driven approach for generating or translating a set of domain-specific rules.
4. Applying controlled variability management to process model customization. This framework structures the components of the feature model to enable end-users in selecting their requirements, and customizing the domain template.
5. Finally, a comprehensive empirical study of efficient, effective, and satisfactory usability criteria has been completed for evaluating and validating the framework by the end-user experience.

We develop a framework in which a non-technical user can generate rules to customize and configure the process model. The rule generation helps to configure the constraint to solve a domain-specific problem. We intend to minimize the development efforts by avoiding the need to: (i) *write programming syntax*, (ii) *compile*, (iii) *customize*, and (iv) *redeploy*, by introducing automated rule generation, and configuration-based processes for non-technical domain experts. The research is aimed at increasing the efficiency and effectiveness of a configured rule. The generated rule, intended for enterprise software applications, can drastically reduce the development, testing and debugging time.

This research develops a systematic DSR generation framework, which could assist domain users focus on high-level modeling and understanding of the more abstract domain problems rather than focusing on low-level rules, executable programs, and programming code. This research is a feature based solution and additionally borrows a customization solution from product line engineering [31-33] and, has generated a new rule language as a DSRL coming from a model-driven approach.

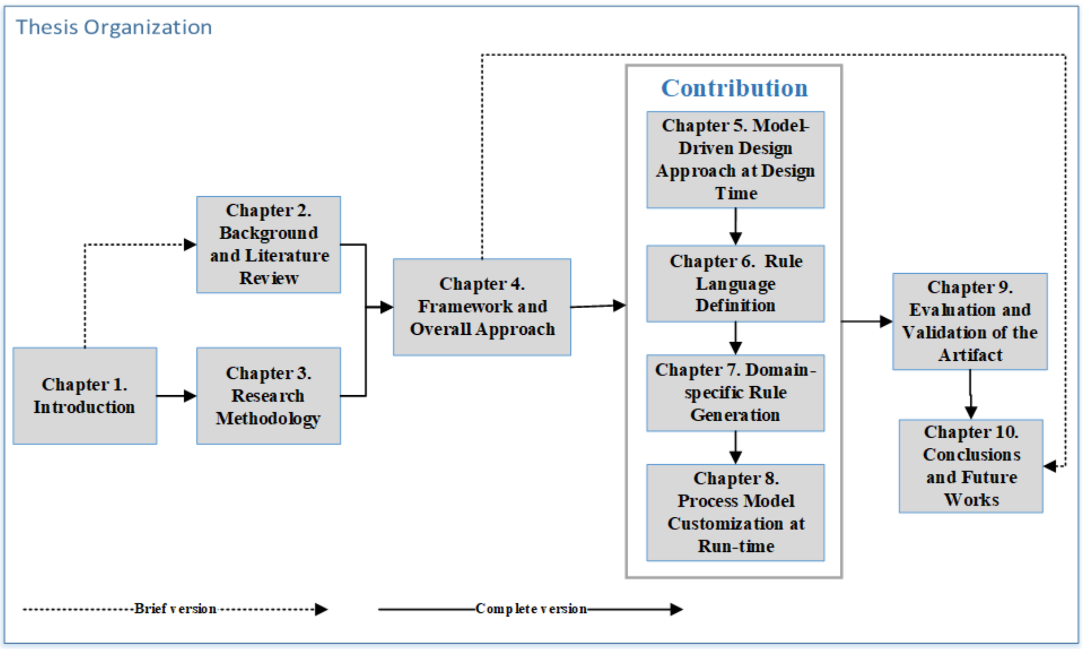

Thesis Organization
Contribution
Chapter 1. Introduction
Chapter 2. Background and Literature Review
Chapter 3. Research Methodology
Chapter 4. Framework and Overall Approach
Chapter 5. Model-Driven Design Approach at Design Time
Chapter 6. Rule Language Definition
Chapter 7. Domain-specific Rule Generation
Chapter 8. Process Model Customization at Run-time
Chapter 9. Evaluation and Validation of the Artifact
Chapter 10. Conclusions and Future Works
Brief version
Complete version

Figure 1.1: An Overview of the Thesis Organization

1.5 Thesis Outline

The structure of the thesis is illustrated in Figure 1.1. An overview of the contribution of each chapter is provided, followed by a summary of the objectives and the envisaged outcome for each chapter:

Chapter 2 - *Background and Literature Review:* This chapter covers the main concepts and various features of the approaches, providing a basic background and understanding of the overall thesis. Specifically, the chapter discusses Model-driven Development and Software Product Lines. State of the art analysis is presented, providing the most recent and relevant approaches that have been proposed to achieve process model customization.

Chapter 3 - *Research Methodology*: This chapter covers the design science research methodology on which the research objectives have been defined and validated. Various approaches and processes are investigated based on the DSR generation artifact and their development. As a follow-up, the design science approach and its components are explained. Furthermore, this chapter also describes how the artifact is designed and built; based on the literature review of Chapter 2 and in in tandem to select the appropriate research methodology.

Chapter 4 - *Framework and Overall Approach*: This chapter introduces the proposed approach to development of software reuse systems with variability models at run-time. This overview covers the key components of the framework and briefly describes each process and how to apply it in our research. This chapter also clarifies how the approach has been evaluated using the case study of a Digital Content Technology.

Chapter 5 - *Model-Driven Design Approach at Design Time*: This chapter discusses how the knowledge captured in variability models is used to provide autonomic behavior during rule generation. The chapter also outlines the efficiency of the MDD approach and conducts a thorough analysis of variability models.

Chapter 6 - *Rule Language Definition*: This chapter illustrates the structure of rule language in terms of abstract and concrete syntax with language descriptions of the DSR. It also describes the DSR and its depiction in Event Condition Action (ECA) Language. Additionally, we present the ECA model and rules in XML format.

Chapter 7 - *Domain-specific Rule Generation*: This chapter describes a Model-driven Architecture (MDA) approach to prepare conceptualization of DSR generation. An overview that covers all the meta-levels of MDA with a model of text translations is also presented. In addition, the implementation of rule generation from a high-level model is explained.

Chapter 8 - *Process Model Customization at Run-time:* This chapter presents the mechanisms for customization of a process model in the DCT domain. The domain templates created at domain engineering in SPLE (design time) are used to manage the domain template components to activate and de-activate on feature events triggered by the end-user (run-time).

Chapter 9 - *Evaluation and Validation of the Artifact*: Experimental work is evaluated to test and validate the domain-specific rules. This chapter evaluates and demonstrates the proposed research and its usability at run-time configurations. The framework is evaluated in terms of effectiveness, efficiency, and satisfaction based on both the statistical and the system usability score evaluation.

Chapter 10 - *Conclusions and Future Works*: This chapter presents the main contributions, results, limitations and future work.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

This chapter presents definitions of key terms and concepts, and a discussion of related research. Section 2.1 discusses *software product line engineering (SPLE)*, Section 2.2 focuses on *variability management*, Section 2.3 describes *feature models and their components* and Section 2.4 presents *rule language*. In Section 2.5, current approach and limitations of the *customization and configuration business process model* are discussed. Furthermore, Section 2.6 is a critical discussion of the literature review of process model customization and where gaps in research are identified. Finally, Section 2.7 summarizes the chapter.

2.1 Software Product Line Engineering

In the highly competitive business environment, it is important to understand the domain knowledge, rapidly adapt changes and systematic implement of reuse [34]. The systematic reuse uses to build a new system from old single-family system. The systematic reuse is core assets of mass-production environments that use the SPL. The SPL is easy to use a variable development of software paradigms. The SPLE support mass customization with improvements in time to the market quality of production, productivity, the satisfaction of the customer, cost schedule [35].

> "A SPL is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [35].

The SPLE is the foundation for this proposed research that provides a platform (in Figure 2.1) for customizing the models and configuring of the rule generation of process model customization, in the current approach. The SPLE helps to adapt the product and fulfill the customer needs as well as enormous reductions in time-to-market, the production costs and engineering overhead. The principal objective of

SPLE is to design and develop inherent features with in variety of systems. This section also discusses three SPLE basic components: Domain Engineering, Core Asset, and Application Engineering. It consists of two components: system input and system process.

The system input defines who is using the framework. There are two different type of users: *Domain Expert* and *Domain User* (end-user). Domain Expert provides domain and process model as a template, and Domain User selects features of the process model as system input based on his needs.

This section explains the conceptual view of the research. We discuss the background and existing platform of the proposed approach. This section covers the relevant area such as, the SPLE, how the SPLE outlines the problem solution in the framework including the rule language and how it works in the framework.

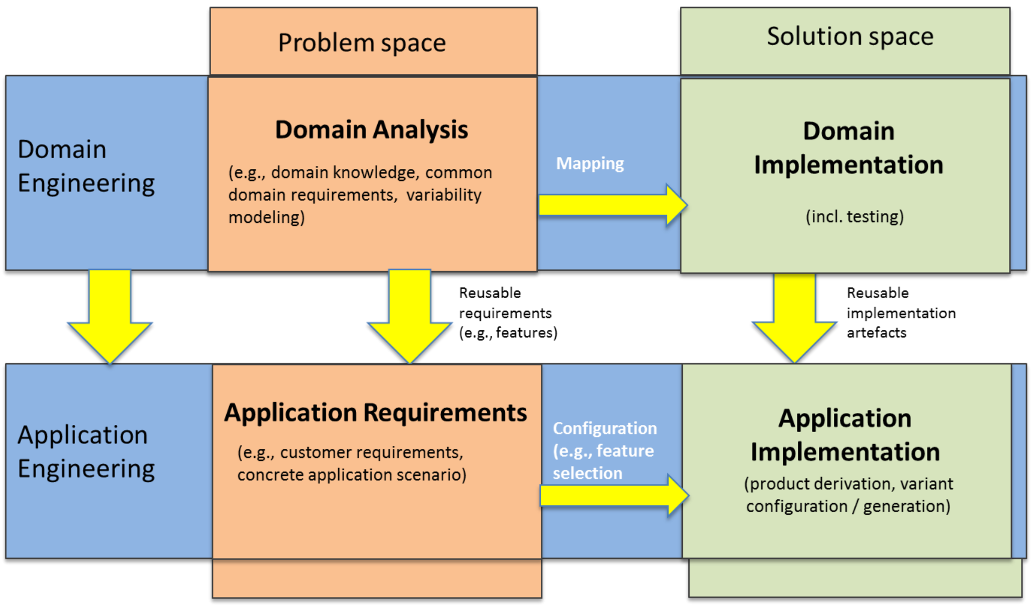

Problem space
Solution space
Domain Engineering
Domain Analysis
(e.g., domain knowledge, common domain requirements, variability modeling)
Mapping
Domain Implementation
(incl. testing)
Reusable requirements (e.g., features)
Reusable implementation artefacts
Application Engineering
Application Requirements
(e.g., customer requirements, concrete application scenario)
Configuration (e.g., feature selection
Application Implementation
(product derivation, variant configuration / generation)

Figure 2.1: Framework for SPLE: Problem and Solution Space [36]

The goal of Software Product Lines Engineering (SPLE) is developing a set of software components and systems with similar characteristics and catering to the requirement of the domain through the management of specific features [37]. SPLE effectively turns down the development cost and market time of the software, enhances and overall quality of the engineering by reusing assets strategically within the domain. It uses adopted techniques to manage reusability with commonality and variability model that effectively categorizes the common assets and their variabilities.

The software product line (SPL) framework has two spaces and phase (see Figure 2.1): (i) *the Problem Space describes 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 the development and the implementation of products take place. In SPLE, the variability modeling technique is known as “Feature Models” which are used to show the variability in hierarchical manner that differentiates or simplifies the features in the hierarchy of products belonging to the software family. A feature may be conceived as a logical unit, having requirements; both functional (what the system should do) and non-functional (how the system works, should behave and quality attributes) [36, 38]. iv) *Application engineering phase* derives the requirements of the target application by analyzing the needs of the consumer. Through the use of the variability model in the configuration process, a concrete product is derived and served it up to the consumers.

2.1.1 Domain Engineering

Domain engineering is described as a process of SPLE, which establishes reusability platform and defines the commonality and variability of a product line [39]. The Domain engineering phase accumulates all the data available related to a specific domain to develop reusable software assets. In this phase, a domain expert is deployed to find out the commonalities and the variabilities existing between the members of a product family SPL, utilized to design reference architecture of SPL [40]. All those family components, which are common and reusable artifacts, are thus encompassed in the *reference architecture* (*components, test cases, requirements, etc.*). The *product variability* (*such as mandatory, optional, alternative, etc.*) and the configuration process is included in the architecture. The configuration step of the different process is formally contained in the production plan. The assets which would be used for making software products are kept in the baseline. The baseline refers to a specialized database which contains software assets and enables their recovery and maintenance. The important purpose of a baseline is to make the core assets available whenever they are required for creating all software artifact requirements such as design, test, realization, etc. and track-able links between these artifacts, thus making systematic and consistent reuse possible [41].

2.1.1.1 Domain Analysis

The method analyzes, identifies and represents commonalities between related domains, which may be reused in other software systems of the same domain [42]. The method aims at categorizing the thought processes employed in designing and developing a software system in a specific domain through capturing the wisdom and experiences of experts. Along with software reuse elements, domain analysis also facilitates communication, training, tool development, the software specifications and design through utilization of domain expertise. The outcome of domain analysis is the domain model. However, inconsistencies exist in the literature regarding the process and artifacts of the domain model. Domain analysis aims at identifying the common elements of a product family so that the main component or ingredients of domain models can be expressed as follows [37]:

- Domain scope (domain definition, context analysis) – find the boundary of domain
- Commonality analysis – identifies the application’s commonalities and variabilities
- Domain dictionary (domain lexicon) - defines the terms, vocabulary, and keyword of the domain

- Notations (concept modeling, concept representation)- express a uniform way to represent the concepts used in domain modelling (object diagrams, state-transition diagrams, entity-relationship diagrams, and data-flow diagrams)

- Requirements engineering (feature modeling) – consists of defining, gathering, documenting, verifying, and managing the need of applications that are stated in the particular domain

2.1.1.2 Domain Implementation

This includes the implementing tools, components and architecture designed in earlier phase. It may include the document preparation and implementing generators and languages that are typical of a particular domain. Domain engineering aims at producing assets which can be reused and implemented in this phase. Hence, the outcome of domain engineering phase consists of components, feature models, analysis and design models, frameworks, production plans, architecture, domain specific language and generators [42].

Core asset development - The core assets refer to the reusable building blocks that are designed and implemented in this phase. Along with functionality of the domain, it also defines the process of extension of the core assets [40].

Production Plan- As mentioned earlier, production plan describes how the individual products are to be assembled through the use of core assets.

2.1.2 Application Engineering

This phrase refers to a specific SPLE process that tries to reuse domain assets and explore variabilities and commonalities to define and creating a product line application [43]. In order to develop SPL products, reference architecture acts as a reference model. The baseline meets the need for assets in a new product. There are two core activities in: (i) configuration of particular products within limits of valid variation points (referred ad product) (ii) Developing product line members through the use of existing domain assets (product derivation).

Product Configuration- This is the process of selection and deselection of the variability of valid combination that was established in the process of domain engineering. This selection is referred to as "binding time of variability" [41].

Product Derivation- This is a concrete process of building the SPL application, which may either be done automatically or manually. Product configuration acts as a primary input and the artifacts involved can be traced through the domain engineering process.

Product characterization- The characteristic feature of the product selected.

Product synthesis- Query is made to the baseline, and required core assets are retrieved therein from developing desired products.

Product construction- Following the production plan, the required core assets are processed as per the proper specification mentioned in the product plans regarding the particular task to be carried out (compilation, code generation, program execution, etc.).

2.1.3 Problem Space and Solution Space

The contrasting term problem and solution highlights the sharp distinction between the systems and their application domain [44]. Several terms are in used to reference these: "*problem model*" and "*solution model*"; "*problem domain*" and "*solution domain*"; "*problem space*" and "*solution space*"; "*problem analysis*"; and "*solution design*".

The distinction between problem and solution spaces are illustrated below with respect to a number of paradigms, and the two spaces are separated at the time of software development (for instance SPL) (see Figure 2.1).

The abstraction specific to a particular domain is primarily contained in the problem space which illustrates the software requirement and with their intended behavior. For example, problem space is involved in domain analysis, and its outcomes are recorded as features. The solution space, on the other hand, contains abstractions that are implementation-related, like code artifacts. Abstractions in solution space, are employed in numerous programming languages, spanning from assembly language to object-oriented language which facilitates the programmer in organizing structural as well as behavioral information making up the software.

There exists a mapping among problem space elements with the solution space elements, which identify the feature to which a particular implementation artifact belongs. Mapping may come with different levels of complexities, ranging from simple implicit one to complex rules of generators. The form and complexity depend on the level of automation and implementation approach [45].

A simplified prototype of a framework has been depicted in Figure 2.1. There exists a chain in the mapping between the problem space and solution space. Two or more could be mapped down to one (or more) solution space (a common phenomenon where representations of different aspects of a system are made). A problem space may be implemented into multiple solution spaces through mapping [45].

2.2 Variability Management

The variations between products in SPL is called variability. It sets off as a proper variability management where many items can be distributed from a set of reusable assets. Pohl et al. [39] explains the concept of variability *subjects* and *objects* for describing the variability. *Variability subjects* are variability features or practical world items which do vary. In software product line engineering, *variation points and* *variants are mostly used to denote variability subjects and objects*. A *variability point* defines a variability subject like application usually gives a particular kind of user interface. A variant expresses a single option for a variation point. As an example, for "Machine Translation": Data Extraction Source is a variability subject and the ways of the source like file, text, multimedia and web URL as a variant.

Variability in software product line can be divided in two dimensions: *space* (e.g., software artifacts) and *time* (i.e., software artifacts changing over time) [39]. Variability may be of different types: functional variability which refers to a certain function appears in certain products but not in others; non-functional variability that takes place when products have same kind of functionality, differing only in quality; and data and control flow variability that occurs when certain patterns of interaction or data vary between products.

As per the domain problem, product line variability and commonality may be adopted for problem space as well as solution space in many ways depending on perspectives [45, 46]. Domain concepts defining the proposed software requirements can be found at problem space along with stakeholders' focus, quality parameters, the objectives of application programmers and so on. High level of functional and operational abstraction and product line quality requirements are used for expressing the objectives or goals. Depending on context, the functionalities and qualities vary among themselves.

Variability abstractions derived from variability model are called features, which are realized at the time of creating the artifact, where mapping exists between features and artifacts. Since variability occurs in multiple artifacts involving several levels of abstractions as well as variation point can get released through multiple variation points spread at multiple elements. Therefore, problem space (features, requirements etc.) and solution space (through artifact space) elements are mapped with information on implementation artifact and their core requirements. Plausible relationships that the mapping can have vary from one-to-one to many-to-many, including one-to-many. According to the level of automation as well as the method of implementation, mapping may take several forms or it may be a simple implicit one, depending on naming conventions or complex rules [45, 47]. Therefore, tractability between requirement and implementation, feature configuration, and product derivation [47-50] is possible in mapping models.

Kang et al. [20] explains three important dimensions of variability modeling: (i) usage in multiple context, by different market segments as well as by stakeholders with different objectives (ii) multiple objectives or usage contexts might require several quality features or functionalities (iii) same functionalities can vary in method of implementation that explains quality properties. The focus of Variability modeling lies with managing and modeling variability space through such dimensions.

2.2.1 Variability Modeling

Multiple modeling and domain analysis approaches exist (as mentioned in Section 2.1.1.1) among which Feature-Oriented Domain Analysis (FODA) [37] is considered to be the baseline for variability modeling. FODA emphasizes identifying the unique features of software systems. Here, "feature" is considered to be a distinctive or prominent user-visible aspect, quality or characteristic of a software system or systems for the purpose of abstracting away functionalities and concepts effectively in different levels, support communications between multiple stakeholders of a product line, as well as achieving maximum reusability. The feature model analysis process involves the following activities: (1) collecting and identifying features source; (2) abstracting and classifying the identified features in a feature model; (3) defining the features; and (4) validating the model [37]. Being a component of domain analysis, FODA brings up feature modeling, a conceptual modeling technique, for identifying as well as representing common and variant features, expressing relations among features and properties of features, and providing expression of the permissible configurations of features in a given domain.

FODA serves as the basis for development and extension of several methodologies like ODM [51], FORM [52], FeatuRSEB [53], PLUSS [54], Generative Programming [45] and GP-Extended [55]. Much effort has been directed towards presenting the diverse viewpoints as well as extending feature modeling. Example of these extensions are structure, binding, configuration, operational dependency, and traceability perspectives [46].

Several variability modeling approaches have been proposed and developed to support the variability management and product derivation [56, 57], e.g., cardinality-based feature modeling [58, 59], Orthogonal Variability Modeling (OVM) [39], COVAMOF [60], CONSUL / Pure:: Variants [61]. General Variability Modeling techniques are discussed in [57, 62].

2.3 Feature Model

Feature models can be defined as a graphical representation of commonalities and variabilities between products of SPL. This is a hierarchical organization of features within the framework of models. They illustrate a set of relationships among parent features and child features. A feature may have one of these relationships with its children: - mandatory (representing shared design), OR optional relationships with different kinds of groups. Another common form is a cross-tree relationship which illustrates inclusion or exclusion constraints.

Variable points are associated with a number of different feature models. Hence families of the domain model and process model may be described as compositions of feature models. A compositional technique gives provision for reasoning about compatibility among connected models to ensure congruency of the whole model template, as well as facilitating automatic propagation of variability choices whereas feature model selects the feature.

Notwithstanding, there are several other extensions and variants available in the different literature, we now choose to move on to the most important proposals relevant to this study.

FORM Feature Model [52] brings up a different analysis viewpoint of commonality among applications in a particular domain with respect to services, operating environments, domain technologies and implementation techniques.

FeatuRSEB Feature Model [53] is domain analysis process, an extended part of RSEB process. RSEB models include variability and aspects of domain engineering. A combination of RSEB and standard FODA process models, the FeatureRSED is simpler than FODA, with its focus lying on feature orientation of domain engineering. These feature models serve as a convenient index for commonality and variability present in use case and object models, simplifying the task for the reuse: (i) Using notational change: alternative features variation point feature and variant features; (ii) Introducing constraint (e.g., require) notation; (iii) Introducing binding time notation: reuse-time and use-time binding.

Van Gurp et al. Feature Model [63] is an external features framework that comes up with notions related to variability. The framework of terminology provides three recurring patterns of variability tools for detailing variability with respect to variability point and variants in a software system. This feature model is refining the relationship of generalization/specialization to OR-specialization and XOR-specialization relationships of variability in recurring patterns and this generalizes binding time notation: compile-time, link-time, and run-time binding.

PLUSS Feature Model (Product Line Use case modeling for Systems and Software engineering) [54] is an industrial case study to be applied and evaluated in the specific target domain which is based on data of case study that helps to determine the performs of PLUS with other models. The whole process is following the guidelines given by IBM - Rational Unified Process (RUP). The RUP provides: (i) Using rational changes with alternative features in a group in single and multiple adaptors; (ii) Providing constraint notation.

Hein et al. Feature Model [64] may be considered from the initial experimental results run by Bosch. The experiment has been applied on car periphery supervision (CPS) domain with its vision on the practicality of variability modeling with feature oriented domain analysis (FODA). The experiment also enlightens us on FODA model not facilitating with required expressiveness for different kinds of cross links in a specific domain. This provides UML-based modeling language and introducing secondary structure for constraint (e.g., require) dependencies.

Generative Programming (GP) Feature Model [45] - Redefining the alternative relationship to XOR and OR relationships

Riebisch et al. Feature Model [65] – The existing feature models fall short of providing support to a complete description of the semantics of relationships and dependencies between features. They come up with new concepts of feature diagrams enabling a multiplicity of features sets. The annotation of the multiplicity of feature subsets is realized in BNF, which is accepted by developers, who are aware of the UML.

GP-Extended Feature Model [55] – Bringing up the notion of feature modeling notation, detailing a domain-independent system configuration editor, describing tool support for feature modeling and taking out the applications of a static configuration in the field of embedded systems.

2.3.1 Feature Description

A language feature model includes both the aspects of software family members like commonalities and variabilities and along with it, it also identifies and showcases the dependencies between variable features. The feature diagram is a core element of the language feature model, which graphically represents dependencies between a variable feature and its components.

The presence of mandatory features in a concept instance may be inferred from its presence at the core. Optional features may be present if their parent is present. Alternative features is a set of features from which one is present if their parent is present. Groups of features and a set of features are a subset which are present in their parent. 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. The mutex operator means that once we choose a feature, the other feature must be excluded (mutual exclusion).

A DSL feature model can cover languages, transformation, tooling, and process aspects of DSLs. The feature model diagram specification starts with a method like the Feature Oriented Domain Analysis (FODA) [37] method. All configurations (called instances) of a software system are represented by them, focusing on the features that may differ in each of the configurations [66]. The notation of model features diagrams is now described. The FDL (Feature Diagram Language) is a feature definition notation for DSLs.

2.3.1.1 Feature Description for Content Processing

The Feature Description Language (FDL) [67] may be described as a textual language to define features of a particular domain. It supports automated normalization of feature descriptions, variability computation, expansion to disjunctive normal form, and constraint satisfaction. Here, it has been applied to digital content processing. The foundation is a domain ontology called GLOBIC[4] (global intelligent content). GLOBIC elements are prefixed by 'gic'. Feature diagrams are a graphical notation of the FODA method. They are used for capturing structuring, communicating, documenting and annotating the features of applications in specific domains as well as a tool for describing the properties of applications from an end-user perspective. This is the first step in a systematic development of a Domain-Specific Rule Language (DSRL) for GLOBIC-based DCT processing use case (Figure 4.3).

[4] GLOBIC having been developed for the ADAPT centre and used here as a real-world use case

2.3.2 Core Asset Development

Core assets (also called platform) include many aspects of software development including reusable software components, requirements analysis, architecture, performance and analysis, testing (test case, test plans, test suits, etc.) and documentation. The development process of core assets follows an iterative process or waterfall model. Core asset development makes use of product constraints, architecture style, pattern and framework, production constraints, production strategy, and pre-existing assets which are discussed in the following sections. The results of core development are the product line scope, the core assets, and the production plan.

Product constraints describe the commonalities and variabilities among the products in the product line regarding the action of features, standards observed, performance limit, interface and environment constraints, quality, and security constraints. Architecture style, pattern, and framework that influence the development of core assets are described. The architecture may have a design constraint as to the way its components interact. Patterns and framework force the development paradigm on developing the core assets. Production constraints specify what commercial, military, or company standards apply to the products, its infrastructure on which the products must be built if any, from which legacy and off-the-shell components could be reused. Production constraints may adversely impact the core asset development.

2.3.3 Product Development

The second activity in SPL is product development (also called application engineering) which is the process, from which the products are created using the developed core assets. This process may affect the core asset development, for example, a product may require or introduce new core assets. Producing a new product that has an unexpected commonality with another product may lead to creating a core asset that can be shared with future products. Product development makes use of the output from core asset development: product line scope, requirements, core assets and production plan. The product line scope determines whether the product under consideration can be included in the product line or not. This is also referred to as product space. The production plan describes how the product needs to be built from the core assets. The ultimate goal of product development is generating product spaces that accommodate the core assets.

2.3.4 Feature Model Variability

Recently, researchers have started applying product line concepts in service-oriented computing [5, 6, 68, 69]. We focus on approaches that used the SPL technique for process model configuration. There are several approaches for process based variability services, which enable reuse and management of variability and also support the Business Processes [70, 71]. Chang-ai Sun [71] has proposed an extended version of a COVAMOF framework, based on UML profile for variability modeling and management in a web service based system of software product families. PESOA [72] is a unique variability mechanism, which is represented in UML (activity diagram and state machines) and BPMN for a basic process model with non-functional characteristics, like maintenance of the correctness of syntactical process. Mietzner et al. [73] propose a variability descriptor that can be used to mark variability in the process layer and related artifacts of a SaaS application. The SaaS application template provides to customize processes. In this approach, the customization processes are significantly and less robustly in their applications.

2.4 Rules Language

Rules can be classified into three types: deductive or derivation rules, reactive rules or active rules, and normative rules or integrity rules [74]. Deductive rules allow deriving knowledge for rule engines in both directions, allowing forward and backward reasoning. Normative rules check constraints and any obligations in data or business logic to maintain consistency in databases or knowledge bases. Reactive rules are used for rule-based programming applications, which are responsible for updating the databases or knowledge systems.

Reaction RuleML is a standardized rule markup/serialization language. It has semantic interchange format for reaction rules and rule-based event processing [75]. It has been developed for reaction rules where most parts evolve separately and define their domain and platform specific language. Reactive rules are further subdivided into five sub-rule types: Derivation Rule (if-then), KR Rule (if-then or on-if-do), Production Rule (if-do), ECA Rule (on-if-do) and CEP Rule (arbitrary combination of on, if, do).

- ECA Rule (ON-IF-DO) - Event, Condition, Action, - (event or action algebra operators)
- Derivation Rule (IF-THEN) - Spatial, Time, Situation, Interval (plus algebra operators)
- Production Rule (IF-DO) - Assert, Retract, Update, Action
- KR Rule (IF-THEN or ON-IF-DO) - Initiates, Fluent, Happens, Holds, Terminates
- CEP Rule (arbitrary combination of ON, IF, DO) - Message, Send, Receive

As the nature of the business is volatile, we require a rule language, which is more than or extended ECA rule to improve the process model in terms of flexibility and verification. For XML, there exist a some more ECA rule languages. However, research studies do not focus on analyzing the behavior of the rule. It may be noted that based on the SQL3 triggers standard [76], Active XQuery [77] is one of the ECA rule languages for XML. This language allows full XPath in parts of the rules, and full XQuery in the condition and action parts and hence may be considered as somewhat more complicated than our ECA rules. However, in the present study, it is excluded from careful analysis as described next. It differs from our language in terms of rule execution model. In our language, inserting, or updating, or deleting XML fragments is treated as atomic updates with the corresponding process model activities.

In order to share rules among ECA rule processing systems of different types, ARML [78] came up with an XML-based rule description. It differs from an Active XQuery as well as the language descriptions that are made in the current research. The definitions of actions and conditions are abstract like XML-RPC method. Similar functionality is provided by Active XML [79] like XML ECA rules by embedding calls to web services via special tags, striving to integrate distributed data and distributed computations done in P2P architectures.

Most of the prominent relational DBMS and limited XML repository vendors now support triggers on XML data. However, it confirms within document-level trigger with insert, deletes or update of an XML document. Moreover, XML documents may be decomposed in relational DBMS as sets of relational tables, which allows developers to exploit existing relational triggering functionality for defining fine-grain triggers over XML data structures.

2.5 Process Model Customization and Configuration

This section covered a literature review which examines the studies of the current approach and limitations to allow to consider proposed approaches through design science and applied systematic literature review (SLR) in Section 3.3.1 regarding customization process mode including process model customization and configuration in SPL contexts. The primary research areas are described in the following sections. The research subareas described are: *process model customization* (Section 2.5.1), *process modeling languages and adaptation* (Section 2.5.2), *configuration of process model, and rule template* (Section 2.5.3) and *Business Process Modeling and Configuration* (Section 2.5.4).

2.5.1 Process Model Customization

The concept of variability descriptor for modeling variation points in the process layer of a service-oriented application was proposed by Mietzner and Leymann [73] The process models are represented through BPEL in this approach. Also, here process model configuration is done based on inputs from a customer for variation points. Further, the customization is validated in terms of variability constraints defined in the feature models. For the purpose of modeling variability and transferring feature model into BPEL process models, an eclipse plug-in is created.

In business processes, requirements of centralized design protocol with decentralized execution plan may occur, keeping the design part intact. RosettaNet PIPs [80] for this purpose developed a BPEL solution. It consists of a three-level approach to describe the protocols of templating in case of high-level patterns, specialization, in case of certain protocols, and implementation in case of certain protocol realizations.

The model families of business process models as a variant-rich business process model. The family is configured through direct selection of business process elements of variant-rich processes. To support this, the extension of BPMN has been created by Schnieders et al. [81] with concepts for modeling variation. Consumer knowledge of business process modeling is essential for this.

Boffoli et al.[82] and La Rosa et al. [83] [84] is used different techniques for explaining differentiate between business process models and variability models. While La Rosa et al. provide variability by questionnaires, Boffoli et al. model use variability table for problem space. While these researchers aim at obtaining valid configuration, we strive to establish that every valid feature configuration has a valid process model which ensures well-formed constraints.

During the literature review process, primary research in the field of customization of the process model are discussed. Table A1 (*Appendix A*) shows the research that has been studied in recent years. It compares processes and outcomes from a number of studies regarding different techniques in the field of process model customizations.

2.5.2 Process Modeling Languages and Challenges

In this section, we present the process modeling language and adaptation through language level. Process modeling languages [14-16, 18, 85], providing expressively various verification techniques (e.g., Petri nets, PNs) thus ensuring processes [19] that are reliable in terms of schemas. However, the languages restrict domain experts to make changes such as defining the process execution plan explicitly as pre-defined: task control flow, data flow, and work/process allocation schema. The changes reflect at the modeling stage or design phase, which make the process model rigid [19-21]. Due to process model rigidness, it becomes challenging to customize, adapt and maintain the process. These languages limit the flexibility of enterprises [22-24] and suitability for the dynamic environment. As the nature of organizations is volatile (policies of business) and processes are often excessively rigid [19-21, 24, 25], enterprises are looking for a new configurable language solution. They can enable domain-specific solutions to make process model more dynamically adaptable in terms of the process execution plan, reducing the dependency on programming, and software developers. Table 2.1 gives a comparison overview of these processes and modeling languages in different criteria or features.

A process modeling language has its own syntax and semantics for defining and specifying business process needs and its service composition correctly. Several graph and rule-based languages have emerged for business process modeling and development, which rely on these formal backgrounds, for example, Business Process Modeling Notation (BPMN) [86], Yet Another Work flow Language (YAWL) [87], Business Process Execution Language (BPEL)/WS-BPEL, UML Activity Diagram Extensions [88], Event-Driven Process Chains (EPC) [89], WebSphere FlowMark Definition Language (FDL) [90], XML Process Definition Language (XPDL) [91], Java BPM Process Definition Language (jPDL) [92], and Integration Definition for

Table 2.1: Literature Comparison of Process Modeling Languages

Language	Feature Descriptions	Syntactic Correction	Semantic Correction	Language Independent	Control Flow	Object Flow
Event-Driven Process Chains (EPC) [89]		+	-	-	+	-
UML Activity Diagram Extensions [88]	• Control-flow • Action nodes and control nodes. • Object-flow • Activities to denote inputs and outputs • Signal sending and receiving at the conceptual level • Waiting states and processing states • Handling activity interruption by decomposition	-	-	-	+	+
BPMN [86]	control-flow of a BPMN process means of tasks, representing activities, and gateways, representing splits and joins (like EPCs- OR, XOR and AND) Associations of data objects with tasks to denote input and output artifacts (UML AD)	-	-	+	+	-
BPEL/ WS-BPEL [97]	Extends imperative programming languages (e.g. C) with constructs for the implementation of Web Services.				+	-
Yet Another Work Language (YAWL) [87]	• Hierarchical structure of tasks corresponding to atomic or composite work items, and conditions, to explicitly • Represent the notion of state • Multiple instance tasks, control-flow semantics and advance features. • Global variables to capture the data-flow	+	+		+	-
XML Process Definition Language (XPDL) [91]	Structural representation of a process and the semantics of its execution					

Function Modeling (IDEF3) [93]. These languages focus on the different level of abstraction, ranging from business to technical levels and have their own weaknesses and strengths for business process modeling and execution. Mili et al. [94] survey the significant business process modeling languages and provide a brief comparison of the languages, as well as the guidelines to select such a language. Recker et al. [48] present an overview of different business-process modeling techniques. Among the existing languages, BPMN and BPEL are widely accepted as de facto standards for business process design and execution respectively.

Analyzing and combining literature on the dynamic adaptation of process model customisation and configuration facilitated in implementing variability constructs at the language levels. For instance, VxBPEL [95] an extended version of BPEL language, identifies variation points and configurations for defining processes in a service-centric system. SCENE [96], a further extension of WS-BPEL describes the primary business logic and Event Condition Action (ECA) rules to guide the execution of binding and to rebind self-configuration operations. Rules are applied for associating a WS-BPEL with the declaration of the policies or business rules to be reused at time of (re)configuration.

2.5.3 Configuration of Process Model and Rule Template

The present era witnesses a growing interest on incorporating flexibility in process model activities. A large number of process design techniques results in strict processes in which "hard coded" business policies are embedded in process schema thereby compromising on flexibility. Flexible process configuration is done through the use of rules to a generic process template. It results in a division among business policy and control flow. The artifact may help to configure or retrieve process variant [3] efficiently.

Mohsen et al. [5] discuss identifying inconsistencies automatically which leads to customize the business process model and configuration procedure [6] helps features become activated or deactivated in a variability model. The composite models and their services become updated by the altered variability model, which leads to addition or removal of WS-BPEL code fragments at runtime. Though the tool uses services instead of direct code, the dependency on programming and code is always associated with it. Lazovik et al. [9] has come up with a language, which relies on service rather than process, and represents customization options for reference business processes.

The CAptLang [98] is an extension of traditional workflow languages (e.g., BPEL) in which adaptation and execution are parts of a dynamic environment with classical workflow language. It uses the java-based business process engine within the ASTRO-CAptEvo [99] adaptation framework.

2.5.4 Business Process Modeling and Configuration

The proposed approach can be related to the research on: (1) *Business Process Variability Modeling and Configuration;* (2) *Process Model Configuration in SPL;* and (3) *Business Process Model Product Line* with respect to a source model. Process configuration and customization have been of interest to researchers in Information Systems (IS) and Software Products.

2.5.4.1 Business Process Variability Modeling and Configuration

As discussed in Section 2.5.3, a promising approach to rule-based configuration is to support the integration of functionality and operationality (including the interoperability) of application systems in terms of loosely-coupled configurable rules across organizations and computing platforms. The development of process-driven rule-based configuration is an emerging business software development method that supports the business logic from the domain-specific technologies and implementation rule languages. The process models are the key element in the complex composited applications because they allow the relationship between high-level abstraction and logical, implementation-independent designs the concrete implementation of a system.

Business processes can express the structure of composite applications, where a process includes a set of activities realized and implemented by services. Business process models (process models for short) specify the workflow (control flow) execution order of the activities to achieve a specific business goal. Processes are deployed and executed by a process engine, where a process can also be exposed as a service with a standard interface, and therefore, can be invoked by end-users or other services in enterprise scenarios, such as business-to-business processes to name one. Business processes have been one of the active research areas in the field of business process management including the concepts, methods, and techniques to support analysis design, development, management, and configuration (or customization) of business processes [100, 101]. In the following sections, we focus on providing an overview of approaches for business process variability modeling and configuration.

2.5.4.2 Process Model Configuration in SPL

In the present era, application of product line concepts in service-oriented computing [69, 102-104] may be noted in many research studies. Our focus lies on approaches that apply SPL techniques for process model configuration.

PESOA: The Process Family Engineering in Service-Oriented Application (PESOA) project [81] describes a variability mechanism in order to model variability and define the apparently variant-rich process models. Stereotype annotations are employed for extending such models to incorporate variability and provide configuration options. At the time of configuration, variation points are bound with one or more variant as per its type. PESOA fails to present a concrete guidance on configuring feature models as per target application’s needs.

2.5.4.3 Business Process Model Product Line

Software product line development employs several approaches that make a distinction between domain engineering and application engineering. Domain engineering is concerned with the creation of product line itself while application engineering strives to produce new products from a common base of core assets. Manual programming efforts are usually needed in both [35, 39]. As is the case with tools like pure:: variants [105] or Big Lever Software Gears [106], this research approach also has cut back application engineering to an automatic configuration step. A valid configuration in FeaturePlugin is all that is to be created in this approach. Then this configuration is utilized by tool MODPL Feature Plugin for automatically generating to the configuration of the required product.

The V-BPMI framework [107] address up variability with respect to process lines and process variability. The framework creates a "*by composition*" approach of variability where process models are designed through the assembly of process fragments and reuse of process variants. Goal fulfilment and contextualization guide process model design throughout adaptation of process lines and reuse of process variants.

Herzog et al. 2013 [108] developed a product line simulator model for an aircraft to analyze and define in a Product Variant Master. A configurator system is implemented for creation, integration, and customization of stringent simulator model configurations. Its goal is accelerating reusability while combining models for usage in a range of development and training simulators. Though Design Automation and Knowledge-Based Engineering solutions do exist, their use in SPLE and reuse of simulation models have serious limitations.

2.6 Critical Discussion on Process Model Customization Literature

Research relating to process model customization, with specific attention to SLR has been systematically identified. The results of the comparative analysis are summarized in Table 2.2.

Table 2.2: Literature Comparison of Primary Research Work of Approaches and Techniques Applied for Process Model Customization from 2009 to 2014

Year	**Authors**	**Solution Approaches/ Techniques**	**Rule**	**Rule Generation**	**Variability Model**	**SOA**	**Configuration**	**Knowledge**	**SPL**	**UML**	**SaaS**
2011	Liang et al. [109]	Ontology	-	-		-		-			
2012	Kumar and Kanagaraj [110]	Ontology	-	-		-		-			
2013	Huang et al. [111]	Ontology	°	-							
2009	La Rosa and van der Aalst [84]	Questionnaire driven	-	-	+	-	+	-		+	
2012	Kumar and Yao [3]	Rule templates	+	-	+	-		-			
2014	Asadi et al. [5]	Feature model	-	-	+	-	+	-	+		
2014	Alférez et al. [6]	Feature model	-		+	°	+	-	+		°
2014	Wang et al. [8]	Service-Oriented/ Feature model	-	-	+	+	+	-	+	+	+
2009	Schleicher et al. [112]	Business Process Template	-	-	+	-	+	-	-		-
2017	**Our Work**	**Configurable Rule**	+	+	+		+	+	+	+	

Notations are as follows: "+" indicates a fulfilled criterion, "-" indicates the criteria are not fulfilled, "°" indicates partial fulfillment, and blank spaces indicate that is not applicable or undiscussed.

The first column identifies the year of publication and the second column lists the research reference. The third column specifies the solution approaches or techniques adopted by each researcher to customized and configured process models. This configuration is a process which is intentionally narrated with continuous change to accommodate during a BPM application. The remaining columns indicate the feature or principle for each criterion. For the first feature criterion, the rules are processed as a set of instructions or command/protocol from abstract and detailed representation language for general and domain-specific perspectives. For the second feature criterion, rule generation is considered as transferring knowledge from the domain model to a configurable set of domain-specific rules.

The core technique is rule generation by model translation, i.e. the generation of a structured representation (model) of the target program (compiling or code building) instead of a configurable XML rule on which the application running should support such continuous changes. This technique can also be applied to the high-level of a domain model for translation of high-level extensions of the metamodel to lower-level constructs of DSRL using model-to-text translation (details in Chapter 7). In the third feature criterion, variability models enable us to describe the variants in which a system can evolve.

2.6.1 Research Gap

As summarized in Table 2.2, the customization of the process model has been discussed in a different context, which was defined in Section 2.6 such as *Rule, Rule Generation, Variability, SOA, Configuration, Knowledge Transfer (High-level to Low-level), SPL, UML, and SaaS*. A research gap can be highlighted from a careful analysis of the process model customization literature in this chapter.

The table demonstrates the results of the systematic comparison of solution approaches and studies, which enables us to analyze which approaches and topics were pursued in each process customization. None of the features address all criteria. Rule, and rule generation approaches in particular are largely neglected. Indeed, there are no such mechanisms where customizable and configurable process models are adapted to the requirements of end-users capturing the product line variabilities in a specific domain. For example, in approaches [109], [110] and [113], ontologies are used without fulfilling the list of criteria or the feature which is mentioned in the table. The only approach which gets closer to rule criteria is a rule template [3]; where customization issues are resolved through a rule-based approach. This thesis is the only research referred to in the table which covers the configuration rule by using the feature model, includes rule generation and knowledge transfer from high-level model to low-level rule, and computes the process model customization and configuration. The research objective is to capture features from the feature model, based on the feature activate, and de-active, corresponding domain model and process model, which in turn is based on that rule generation, rule configuration, and process model customization. The solution approaches considered in the heterogeneous set of models are associated with the overall approach.

2.7 Summary

This chapter presents the foundations of the thesis with a conceptual view of the research, foundations and definitions. It includes a literature review which discusses *current approaches and limitations*, in which we describe the solution platform of the approach: *SPLE*, the *framework for SPLE*, Variability Management, Feature Model and Rule Language. This discusses adapting of process model customization and configuration in terms of different research, this includes the customization of the process, the process modeling languages and adaptations, the configuration the process model and rule template, and business process modeling and configuration. A more detailed focus and analysis of customization and configuration of the process model allows the formalization of the research gap and highlight addressing the main research question of this thesis. The next chapter is dedicated to a selection of research methodologies, their suitability, and how this research is validated in terms of philosophy, reasoning, approach, methodology, analysis and evaluation methods.

CHAPTER 3

RESEARCH METHODOLOGY

In Chapter 2, a review of the literature, in particular *current approaches and limitation* of process model customization was discussed. This chapter describes the methodology that has been chosen to help answer the questions posed by the research. The discussion includes justification for the chosen methodology, a brief description of other methodologies at, and their possible limitations and drawbacks, along with a description of endeavors to overcome limitations with our chosen methodology. In order to efficiently address research challenges which are formulated by literature. This chapter starts with the selection of methodology which will fulfill the requirements of this research. Then, the analysis of the selected research methodology is carried out, so most appropriate methodology is chosen for this research. This chapter is presented the most suitable research methodology to be employed in this study.

3.1 Methodological Requirements

The goal of any research is to travel from the unknown to the known, in relation to a specific question. The question may emerge from any field, like, the natural, artificial, or the behavioral world as long as it is scientifically answerable and aims to fill a gap in our existing knowledge. This research revolves around seeking an answer in a scientifically rigorous manner. Hence, research is a problem-solving approach, wherein the knowledge-gap or the question is the problem, and the researcher seeks to solve this problem in a scientific way. From a philosophical viewpoint, life can be seen as problem solving [114]. Therefore, the research interest cannot be separated from the problem solving or seeking out solutions to a specific problem at hand, though there may be different vested interests in the outcome of a particular challenge solving. For instance, academics and researchers, may not find a practitioner- driven software development project to be specifically of interest, as they might be more interested in the theoretical work rather than a fully practical or operational work [115].

As stated earlier, the output of this research is a prototype framework, which is realized as a software tool that solves a reusability requirement in the volatile environment of business strategies, by the process model customization in the domain-specific environment. However, during the process of building such a framework in a scientific and rigorous way, it is important to proceed methodically. In the Information System (IS) field, there exists a wide range of methods. This research is expected to have a direct impact on business by adapting the regular changes in their existing or new strategies and policies in the day to day life. The non-technical domain user can handle the changes in terms of customization and configuration. This research aims to contribute in both the academic and the industrial context. A participative way of exploring and solving complex socio-technical problems with key stakeholders is E*ngaged Scholarship* (ES) [116]. *Action Research* (AR) and *Design Science* (DS Research) are two variants of ES that have caught much attention and there is even a combination of two methods (AR and DR) which have recently been devised as *Practice Research* (PR) [117].

3.2 Methodology Selection

The identification of a methodology that is appropriate for solving the problem at hand is a crucial phase of the current research. Instead of devising a new methodology, it was decided to utilize an existing technique provided by some of the IS methodology. Consequently, a literature survey revealed four plausible research methodologies that seem to be suitable for conducting this research. These are Action Research [118], Grounded Theory [119], Case Study [120] and Design Science [114, 121, 122].

3.2.1 Action Research

Action Research (AR) [118], as it appears in IS strategy literature, is a unique methodology that allows a blending of the academic and the industrial knowledge.

Hence this methodology was appealing to the present research since validating the rule generation and configuring processes for the process model customization in the DCT case was one of the major requirements of this research.

AR was first used by Kurt Lewin [123] during the 1940s and is often described as an approach that “combines theory and practice (and researchers and practitioners) through change and reflection in an immediate problematic situation within a mutually acceptable ethical framework” [124]. Therefore, AR may be taken as a methodology that aims at contributing to both knowledge and practice by providing a solution to a specific entity (usually represented by an organizational setting). Therefore, Action Research “is highly contextually dependent, while attempting to address the specific client’s concerns” [125]. AR focuses on solving a socio-technical problem by developing a new solution and evaluating it in an organizational context. However, due to its flexible and innovative nature, it becomes liable to biases and influences of the hosting organization.

The current research does not only aim at providing solutions to an immediate organizational problem, but also towards instantiation of the artifact. This suggests that it may be suitable as a part of the overall methodology, i.e., evaluating the outcome of this research through experiments. However, though AR seems to fit from a holistic point of view, it fails to meet the rigorousness required to undertake this research. Moreover, this study is designed to contribute towards both theoretical knowledge in academia and to its practical application in industry. AR might limit the findings to the targeted organization, making it difficult to achieve generic results [126].

3.2.2 Grounded Theory

When the objective of rigorous theoretical development comes into question, Grounded Theory (GT) [118] presents itself as an efficient approach. GT has been defined as “a systematic methodology involving the discovery of theory through the analysis of data” [119]. This theory appears as ‘*grounded’* in the analysis of actual settings and processes [127]. Therefore, the process of theory development begins from selecting participants who have experienced the phenomenon under investigation. Hence GT is essentially qualitative research, which provides for a detailed entailment (i.e., theory) of a process, action, or interaction, occurring among a large number of participants. The research questions that would suit GT, involve understanding individual experiences on the process of interest, and identifying the steps in the process (i.e., what was the process? How did it unfold?).

However, this approach might not be the most suitable because it aims at understanding a phenomenon, and not at developing a *prototype oriented framework*. And hence, it is discarded as the primary methodology for this research. However, certain techniques of GT might be adopted for a better understanding of the phenomenon involved in the current study. For example, one of the features that can be taken from GT is to refer to the theory in order to establish a common vocabulary of an area and define it with different levels of formality, as well as the meaning of the terms and the relationships among them. As a consequence, an examination of the existing theories would contribute (to some extent) towards developing the artifact, which in this case, is a framework for the process model customization through the rule generation and configuration.

3.2.3 Case Study Research

One of the most popular research methods in the field of qualitative/quantitative research is the Case Study (CS) [120] method. CS has been used and adopted in various socio and socio-technical fields like psychology, political science, education, clinical science, social work, and administrative science information systems [128]. There is a wide range of literature available for this methodology. Yin et al. [120] suggest that “a case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident”. A case refers to a subject of inquiry, and the subject may comprise of people, organizations, events, places, institutions, or Information Systems (IS) that are studied by one or more methods. IS include people and computer applications that process or interpret information. Furthermore, CS may refer to either single or multiple case studies. It may consist of either qualitative or quantitative approach or a mixed approach for data collection. The key aspects of case study research include exploring, generating and finally testing the hypothesis.

The goal of CS is developing an understanding of the concerned issue, problem, or phenomenon using the case as a specific illustration [129]. Hence, the researcher with CS approach explores a bounded system or multiple bounded systems (cases in specific settings/contexts) over time, through detailed, in-depth data collection, involving multiple sources of information and reports.

The Case Study methodology, therefore, is seemingly the best suited methodology, keeping in mind the goal of the project that allows for an empirical evaluation of the problem at hand in a practical scenario. Furthermore, through this approach, an artifact (theory, concept framework or process) may either be proved or disproved. In the present study, the artifact is a framework for process model customization through rule generation and configuration. At the beginning, a case is selected that suits the business process model (BPM). This may be a requirement for proper framework evaluation. Once, the correct case is selected, the framework should be implemented. Since, in this phase, we would deal with an abstract or general artifact, it is required to gather some details regarding the specific case (i.e. digital content technology, a case of BPM). In the next step, implementation of the instantiated process as a framework prototype application can measure and analyze the framework usability in configuring the generated rule for process model customization. This usability is evaluated in terms of efficiency, effectiveness, and satisfaction. At the end, the framework is used (through generation, configuration, and customization) by end-users such as *a domain expert, domain engineer, customer, or stakeholder*.

However, the case study research methodology cannot be compared to testing and evaluating of the artifacts in a practical situation, as this methodology does not allow a method for building these artifacts. Therefore, this methodology alone would not fully suffice for the present study, though it could be employed for the purpose of assessment of the artifact.

3.2.4 Design Science Research

Design Science (DS) methodology is increasingly popular in Computing and Information Systems research in the current study [114, 122]. This methodology allows knowledge generation– descriptive and prescriptive strategies for providing a solution to problems arising from both literature and practice, and makes provision for assessment through the collaboration of academia and industries. As a result, the imparted knowledge can be used as IS artifacts. Thereby, artifacts can be either (or combination of) constructs, models, methods, or instantiations [122]. The literature varies when it comes to opinions and recommendation of DS [130, 131]. A major challenge in DS is that guidelines provided from the precursors are seldom 'applied' [131] indicating lack of clarity or inadequate operationality of the existing methodologies due to the level of abstraction being too high [114]. The activities (procedures, tools, techniques) pertinent to the study are briefly described here. DS may appear similar to the Action Research (already described), though there are subtle differences among the two [125].

If a paradigmatic comparison is made, the DS allows greater variability. Moreover, AR may be considered as a special case of DS though as opposed to AR, the focus of DS research lies especially on building new IT artifacts. However, these two approaches may be combined, as AR method may be incorporated in DS to evaluate the research. The practical relevance appears to be the focal point in both AR and DS research approaches [131] which makes the two approaches suitable for the present study.

Table 3.1 presents a summary of findings on the suitable research methodologies in the present research. The parameters that have been used for comparing the different research methodologies are *research output*, *main activities*, *problem solving* and *framework building*. The taxonomy forms the criteria in the research methodology which was identified in the literature. The research output plays a critical role in the choice of the methodology. Our approach is problem-driven, meaning that we are solving a research problem, and fulfilling the business needs – e.g., domain experts need a technology free tool that would allow them to change quickly their business processes. The reason for selecting DS research over other approaches lies in the fact that the design of IT artifacts, is more focused on the artificial creation of solutions to problems encountered and to fulfil industrial business requirements. In this regard, DS research provides defined research outputs in the form of IT artifacts.

Table 3.1: Overview of Research Methodologies [132]

Research Approach	**Main Activities (Phases)**	**Research Output**	**Problem Solving**	**Framework Building**
Action Research	• Diagnosing • Action Planning • Action Taking • Evaluating	Specific Organizational solution	Yes	No
Ground Theory	• Theory Generating • Theory Evaluation	Abstract Knowledge Theories	No	No
Case Study	• Environment • Analysis • Observation • **Evaluation**	Phenomenon investigation Generalization Tests	No	No
Design Science	• **Analysis** • **Design** • **Evaluation** • **Communication**	Construct Models Methods(processes) Instantiations	Yes	Yes

The DS research seems to offer a coherent approach in building constructs (e.g., rule generation and customization). As mentioned, the primary objective of the present study is to provide a rule generation and configuration for the process model customization. The DS research is found to be the most suitable methodology due to its ability to support building and evaluation of such constructs.

3.3 Design Science as Research Methodology

The processes outlined in Figure 3.1 show the Design Science process used in this research. Moreover, it expresses appropriate techniques and methods to the DS process. Additionally, an overview of the research outcome at each step is presented.

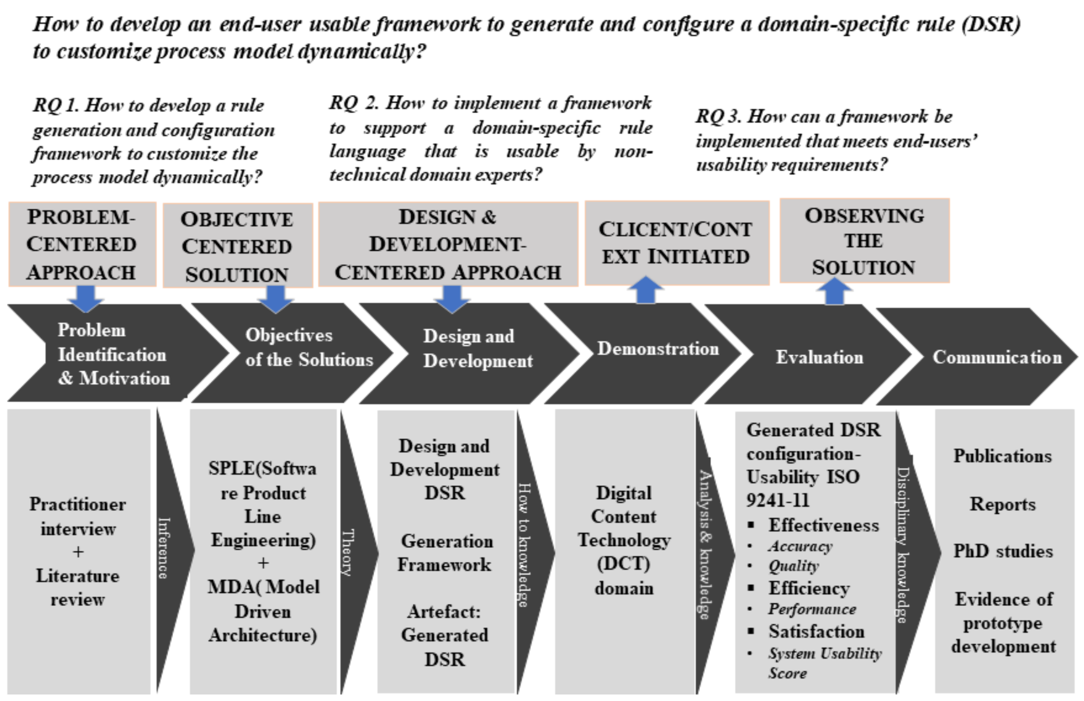

How to develop an end-user usable framework to generate and configure a domain-specific rule (DSR) to customize process model dynamically?
RQ 1. How to develop a rule generation and configuration framework to customize the process model dynamically?
RQ 2. How to implement a framework to support a domain-specific rule language that is usable by non-technical domain experts?
RQ 3. How can a framework be implemented that meets end-users' usability requirements?
PROBLEM-CENTERED APPROACH
OBJECTIVE CENTERED SOLUTION
DESIGN & DEVELOPMENT-CENTERED APPROACH
CLICENT/CONTEXT INITIATED
OBSERVING THE SOLUTION
Problem Identification & Motivation
Objectives of the Solutions
Design and Development
Demonstration
Evaluation
Communication
Practitioner interview + Literature review
Inference
SPLE(Software Product Line Engineering) + MDA(Model Driven Architecture)
Theory
Design and Development DSR
Generation Framework
Artefact: Generated DSR
How to knowledge
Digital Content Technology (DCT) domain
Analysis & knowledge
Generated DSR configuration-Usability ISO 9241-11
Effectiveness
Accuracy
Quality
Efficiency
Performance
Satisfaction
System Usability Score
Disciplinary knowledge
Publications
Reports
PhD studies
Evidence of prototype development

Figure 3.1: DSRM Process Model (adapted from [114]).

3.3.1 Problem Identification and Motivation

At this phase of research, we identify the motivation and develop the problem statement. It can be considered as a process for preparing, accumulating knowledge, or developing a foundation on which the construction of an artifact is possible. This requires gathering sufficient knowledge through the available literature. Systematic Literature Review (SLR) [133] has been employed for extracting context specific results.

The SLR aims at identifying topics from reliable and high-quality sources. The topics must come with a precise description as well as the rationale for selection. The search material should be kept as transparent and replicable as possible [134]. At this stage, the aim is to identify all concepts that are someway related to the variability model and process model, and might act as enabling factors. Also, it attempts to identify the only plausible ways for configuring the generated DSR from domain model by using the variability model, and it helps to customize the process model. However, since manual DSR configuration is a complicated and challenging assignment job, and chances of committing mistakes is very high (error prone), a few parameters were set for selection of literature: like the relevance in the present study, the year of publication and the reputation of the journal or the conference where the article has been published. All the available literature was scored by these parameters.

Essentially, a review of literature attempts to frame and solve the research question selected while finalizing the scope (scoping). There are two types of tasks performed during Systematic Literature Review – broad, and advanced, literature search. In the beginning of scoping, a vast study of literature was performed, where a wide array of abstracts, conclusions, prefaces, and references were considered, which were found suitable for the present study. The SLR may be differentiated from other approaches by the kind of rigor it involves, which, in turn, definitely provides an edge while gathering the reviews, evaluating the amount of plausible relevant materials and identifying the numerous search themes in all their combinations.

In the context of the present study, the motive behind this broad SLR was to collect every kind of information regarding process model rigidity and hardcoded processes and data flow. Afterwards, the findings were related to the domain variability. The sources of literature were primarily journals, periodicals, reports and a few reliable forums and professional blogs available on the Internet. The search was directed by the following keywords - "*process model customization*", "*business process model customization*", "*process model in SPLE*", and "*process model using variability*". Then, the common articles from various search terms were chosen for containing information about the appropriate solutions. This resulted in an exhaustive list of literature that might be worth reviewing. Afterwards, on the basis of the abstracts, the number of relevant articles was whittled down to about 40, which required careful study of all the papers. These papers were checked for their customization and configuration consistency with the defined research questions available in Chapter 1.

In order to check the customization and configuration of the business process model application, a questionnaire survey was also independently carried out along with the literature findings. The questionnaire that was employed for this purpose is available in *Appendix A* (*Section A1 BPM Survey Questions* and *A2 Approval from DCU Research Ethics Committee*). The survey included participants who design solutions in the process model customization. The total number of participants was 22, which included: 3 project managers, 4 process model engineers, 4 BPM experts, 5 domain experts and 6 software developers. The survey questions were framed on the basis of issues that had emerged during the review of the literature. *Process model customization and its adaptation at run-time is challenging and error-prone due to the rigidity of the process model*. It was found that the results were in accordance with the literature. Particularly, 76% of participants and their organizations encountered challenges of *rapid changes* due to *external and internal sources*, and the dynamic adaptation was also difficult for domain experts who lacked the technical skills.

Therefore, the survey outcomes above, and the literature analyzed, unanimously indicated, that due to the complexity of the process model (discussed in Chapter 1), there exist issues with the customization and adaptation of process model in a dynamic environment. The functional and operational parts are difficult to handle, at the time of process model customization, if end-users are lacking the technical skills. Hence, the research objective was to develop a framework, which would enable the end-user to generate DSR and configure the domain constraint value for process model customization. Moreover, the framework would be evaluated on usability parameters like efficiency, effectiveness, and satisfaction in a dynamic environment.

3.3.2 Objective of the Solution

This objective is to develop a framework for rule generation which supports the domain-specific process model and its scope and implementation in the Digital Content Technology (DCT). The aim is to generate the domain-specific rules (DSR) from a domain model to customize and configure the process model in a dynamic environment based on the end-user requirements. Moreover, we define a Domain-specific Rule Language (DSRL) for the domain-specific environment.

a) *For objectives, I and II*: SPLE use for mass customization and a model-driven approach to translation of low-level rule from high-level domain model as the core of the framework. The customization of the process model and the rule generation development uses the SPLE lifecycle. The customization process model development lifecycle and domain engineering deal with analyzing the domain and customizable process model, as well as variabilities and configuration options. Here, a domain expert can design and develop the high-level abstract domain template (process model and domain model) which contain all possible features (to be covered in Chapter 5). We also implement the core assets of the customizable models. This lifecycle provides customization models and the DSR generation using the MDA model as well as the implementation of the core models in terms of the activation and the deactivation features (to be further discussed in Chapter 7 and 8). The development lifecycle and application engineering deal with capturing the requirement of the target application and deriving target models based on these requirements. The lifecycle is delivered to the customized BPM with a corresponding set of the generated DSR, based on a target feature selection by non-technical domain user or stakeholders (also discussed further in Chapter 8).

b) *For objectives III and IV:* We define a rule language with the *abstract syntax*, the *concrete syntax,* and definition with *ECA language model*. Subsequently, the high-level conceptual model of DSRL is defined in the form of high-level BNF[5] grammar. Additionally, the abstract grammar of DSRL as well as each component/activity of the DCT process model is expressed in BNF. Both the abstract and concrete syntax follow the literature consolidation and industrial observations (discussed in Chapter 6, Section 6.3).

[5] Backus-Naur Form

c) *For objective V:* The scenarios of the case study involve processes in the DCT domains, that include user studies with a prototype. The rules are automatically generated based on feature selection by the end-user. In this case study, we evaluate the usability evaluation of the main artifact, as the configuration of the generated rule in terms of efficiency, effectiveness, and satisfaction (discussed in Chapter 9).

3.3.3 Design and Development

As discussed in Chapter 1, the objective of this research is to investigate how to generate and configure the DSR from a high-level domain model based on end-user requirements (variability model) to customize the process models. In order to achieve this goal, the prototype of a framework for generating and configuring the rule is designed and developed. More specifically, the end-users get the flexibility to customize and configure the process model as per their requirements. The final output is a generated low-level DSR, from a high-level domain model for configuring and customizing the process model, which is ultimately implemented as a prototype framework. The component of the artifact of this research has been defined in Chapter 4 and each component and its process are discussed in Chapter 5, Chapter 6, Chapter 7 and Chapter 8 of this thesis, as well as in other research works [135-140].

The design and development of the artifact are performed during the information/process synthesis stage. This is the phase where the knowledge obtained from the systematic literature review and discussions with practitioners in the form of methodologies, principles, and concepts (e.g., Software Reuse [141]) is fused.

3.3.4 Evaluation

In the subsequent case study, the measurement is used to configure the DSR in terms of usability criteria used in the DCT domain case study. In Chapter 9, we evaluate the usability of the framework in terms of efficiency, effectiveness, and satisfaction. The efficiency is measured by the comparison between manual and semi-automatic generated rules. The effectiveness is measured by the quality of the rule configuration by error prevention and accuracy with error correction. The SUS use for achieving the score from the end-user and calculating the satisfaction of the prototype framework, as well as the *cross-validating of the efficiency and effectiveness of the framework* by a set of questionnaires in a controlled environment (discussed in Chapter 9), makes the satisfaction more subjective. In this research, we have used a number of evaluation methods: a *case study, controlled experimental design, and user feedback as questionnaires (SUS)*. Each evaluation has some benefits and drawbacks, due to the fact that we use the *combination of evaluation methods* that gives a proper validation of the research contribution in terms of rule generation and configuration prototype to DS research.

3.3.5 Communication

The outcome of the evaluation may be disseminated with proper allocation of resources like time, human resources, and monetary resources. For community members with time resource limitations, it may appear difficult to produce and present academic papers. Therefore, the key outcomes of the study should be made available in a way that can be applied in technical and managerial fields. For the purpose of technical implementation, descriptiveness might be necessary, while from a managerial viewpoint, a brief description of its utility is essential. The goal of the present research is to lay out a concise methodology for both design and description, that would be relevant in a technical as well as a managerial context. Our work has been presented to the research community [135-140].

3.3.6 Contribution

The study results in artifacts that consist of the design and the evaluation of the rule generation from a high-level model in a systematic approach of variability management to allow the end-user to customize their process model (already discussed in Section 1.4). These artifacts provide a measure for use in the development practice at the organizational and project level for the evaluation, the assessment of the effectiveness and the performance of software reuse and the process model. The proposed artifacts would be a valuable measurement in the area of software reuse.

3.4 Summary

This chapter described the methodologies used in this research and gave an overview of a number of research methodologies in the context of current research. The design science philosophy was deconstructed, and arguments put forward as to why it was selected as the primary research approach. Consequently, an illustration of adapted research framework, as well as an introduction of the research process in the form of an IT artifact ‘build’ cycle was presented. Furthermore, an elaboration was provided on the applicability of DS research to examine the context of various ways industry practitioners, and academic researchers, can best collaborate. Finally, the importance of DS research in clarifying the answers to research questions was discussed.

CHAPTER 4

FRAMEWORK AND OVERALL APPROACH

A conceptual foundation to address the research questions is provided by the literature review in Chapter 2 and from the research methodology employed in this study described in Chapter 3. The aim of this Chapter 4 is to describe a framework for the DSR generation and configuration for process model customization; which aims at enhancing rule generation and configuration at the functional and the operational level of process models. Introducing the components of this framework; the design science methodology presented in Chapter 3 (see Figure 3.1) is followed, and the core design science activities (i.e., background and literature review, involvement of domain participants and framework design and development) is identified. This chapter aims to answer the following research question:

RQ 1. How to develop a rule generation and configuration framework to customize the process model dynamically?

First, the main components and architecture of this framework in terms of conceptual construction are defined. The components address the challenges identified in Chapter 1. To allow non-technical users to generate and configure rules, the next step is to prototype the framework. In order to achieve the customization of a process model, we argue that the models that are produced as artifacts from SPLE and MDA methodologies can be used during the DSR generation and configuration, to drive the process automatically.

In SPLE, the use of the common variability of the product's family provides a platform for mass customization of products. MDA is used for abstraction or reduction of the level of participation in a system; it allows the translation of the high-level model to a set of low-level rules or text (i.e., programming code) and vice versa.

4.1 Introduction

The main contribution of this research is an approach for generating rules and evaluating rule configuration for process model customization. The proposed approach covers design and an application phase, and utilizes three modeling paradigms, namely, (i) *domain modeling*; (ii) *process modeling*; and (iii) *feature modeling*. A *domain model* represents the domain requirements in terms of the main actors, their intentions and tasks, and their dependencies in reform of *class ontologies*. A *process model* represents the control-flow of activities within a process-aware information system. And a *feature model* depicts the variability options in both requirements (i.e., adapted process model) and design models (i.e. domain template). As shown in Figure 4.1, all three models are related via traceability links (i.e., mappings) with the *weaving model* [142]. The weaving model represents the relations between the other models. We present the weaving model in detail in Chapter 5.

The proposed approach consists of a domain and an application engineering lifecycle (as shown in Figure 4.2). The domain engineering lifecycle concentrates on developing and implementing customizable process models. In order to implement rule generation and configuration of the process model customization, SPLE and MDA paradigms can be employed where models realize activities. In the application engineering lifecycle, a new application system is built by adapting the customizable process model and by tailoring implementations to fit the customers' requirements.

In the domain engineering (domain template development) lifecycle, the overall possible process requirements of a domain are captured in a domain template that covers all variability points. The domain model template combines a domain model and a process model. It is an extension of the standard goal model that combines software variability and product line variability in a single space. A single space is a particular domain that meets the objectives and requirements of different end-users.

The feature model and domain template communicate via the weaving model that contains commonality and variability relations. Weaving models take care of software and product line variability tasks. Such a model creates a virtual table of relations between different object models. In addition to connecting the feature model and the domain template, a weaving model helps to activate and deactivate features in the domain template, which were selected in the feature model, based on the end-user requirements.

In the application engineering, - the rule generation process development-lifecycle, and its configuration are adapted based on the requirements of customers. First, objectives and preferences of customers are captured and the product line variabilities in the family goal model, are resolved. Next, according to the objectives, the feature model is preconfigured by deselecting the features, which are mapped to undesirable goals. Afterwards, the customer preferences and constraints are used to reach a fully configured feature model and based on the mapping between feature model and the customizable process model, activities corresponding to deselected features, are removed from the customizable process model.

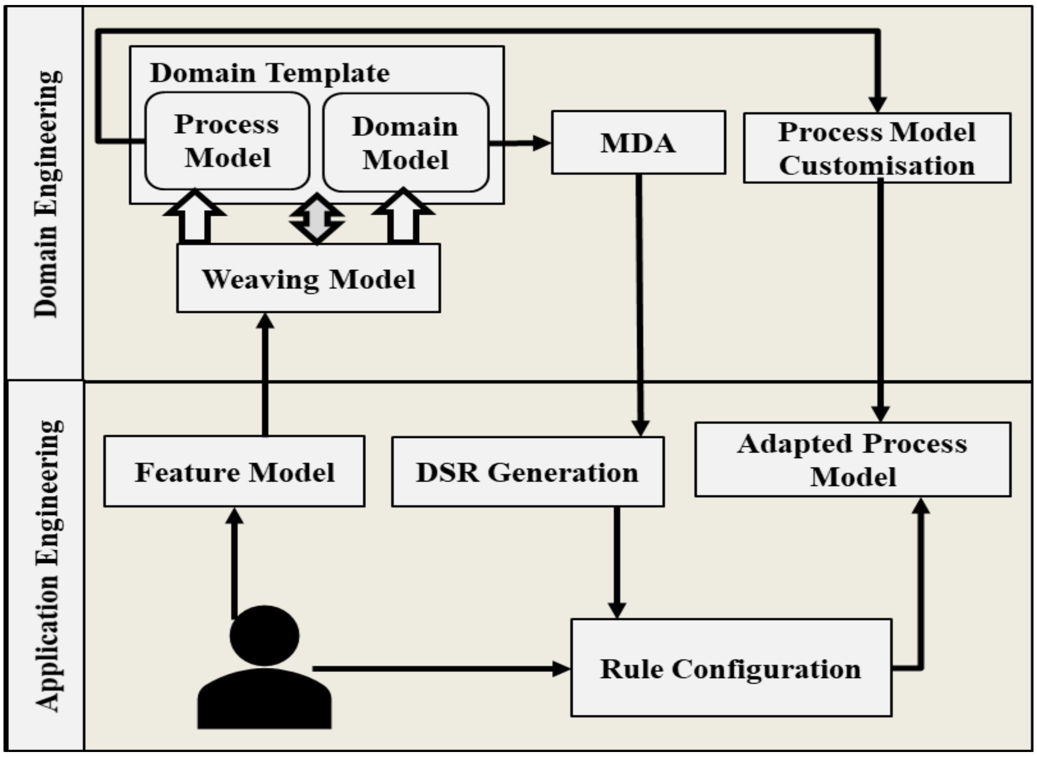


Figure 4.1: Overview of Proposed Approach

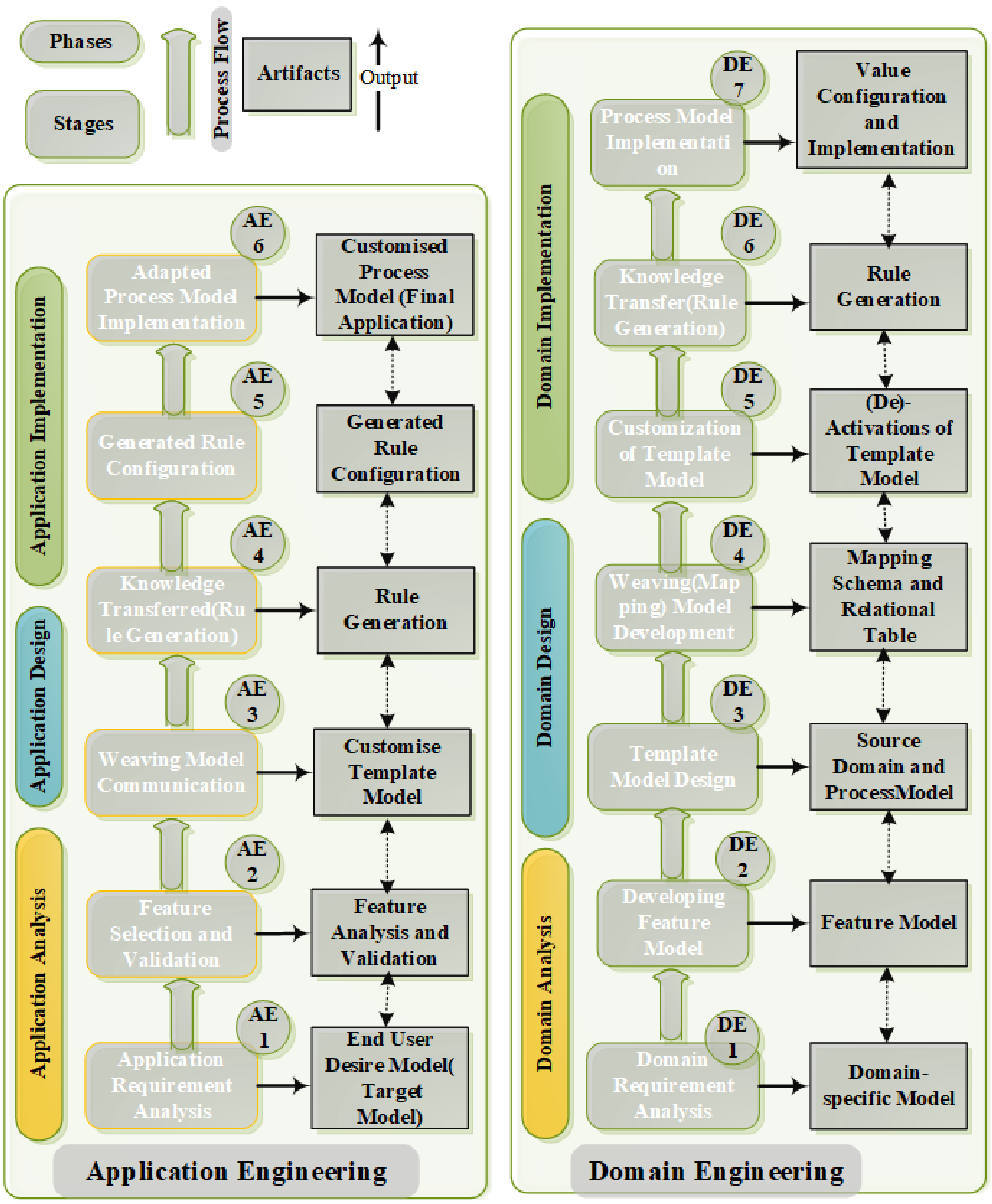
Finally, the process model is further adapted to satisfy any requirements that are not covered in the customizable process model. In the remaining parts of this section, we explain the phases and artifacts of this approach to develop and customize process models.

The present research can be understood from two viewpoints. First, a conceptual viewpoint where we describe the essential components of the framework (see Figure 4.4). Second, an architectural viewpoint where the framework to carry out the dynamic rule generation and configuration is explained (see Figure 4.5). The remainder of this chapter is structured as follows. Since this approach is developed in the context of the ADAPT Centre for the Digital Content Technology (DCT) at Dublin City University, we use DCT as the main use-case to study the implications and applicability of this framework which is detailed in Section 4.3. The DCT case study is used throughout this thesis to demonstrate several key aspects of this approach. Section 4.4 presents the main conceptual components of the thesis. Section 4.5 introduces this framework for achieving rule generation and configuration for process model customization. The conclusion of this chapter is presented in Section 4.6.

4.2 Conceptual Approach of Framework Under SPLE

Figure 4.2. illustrates process sequence of our proposed approach method. This approach consists with domain engineering and application engineering lifecycle of SPLE. Here, DE and AE are domain engineering and application engineering respectively. Every life cycle consists with three different software phases like *Analysis, Design and Implementation*. These phases are standard phases of SPLE that allows domain and end user to work in different phases based on their need and expertize. Every phase has sequential flow with different stages which express as E1, E2, E3, …etc.

The proposed approach method is illustrated in Table 4.1. The table is expressed the proposed method lifecycle, every phase, stage, stage name and outcomes of every stages in terms of artifacts. We explain first row of table, the DE1 is associated with *Domain Analysis* that are focused on *Domain Requirement Analysis* and the artifacts of this stage is *Domain-specific Model*.



Phases
Stages
Process Flow
Artifacts
Output
Application Implementation
Application Design
Application Analysis
AE 6
Adapted Process Model Implementation
Customised Process Model (Final Application)
AE 5
Generated Rule Configuration
Generated Rule Configuration
AE 4
Knowledge Transferred(Rule Generation)
Rule Generation
AE 3
Weaving Model Communication
Customise Template Model
AE 2
Feature Selection and Validation
Feature Analysis and Validation
AE 1
Application Requirement Analysis
End User Desire Model(Target Model)
Application Engineering
Domain Implementation
Domain Design
Domain Analysis
DE 7
Process Model Implementation
Value Configuration and Implementation
DE 6
Knowledge Transfer(Rule Generation)
Rule Generation
DE 5
Customization of Template Model
(De)-Activations of Template Model
DE 4
Weaving(Mapping) Model Development
Mapping Schema and Relational Table
DE 3
Template Model Design
Source Domain and ProcessModel
DE 2
Developing Feature Model
Feature Model
DE 1
Domain Requirement Analysis
Domain-specific Model
Domain Engineering

Figure 4.2:The Stages and Phases of the Proposed Method

In the application engineering lifecycle, the end-user selects all the features, and consequently, the rules are generated and configured. The application engineering lifecycle takes place during *run-time*. This allows for *dynamic rule generation and configuration*.[6]

[6] This feature is in correspondence with the requirements set by RQ1.

The decision to use the domain template at run-time to achieve DSR stems from two reasons: (i) if the domain template deactivation or activation reflects into both the domain models and process models, then customizations can be made at the domain engineering level on the domain template; (ii) since generation and configuration of the rules work on the operational and the functional context, any changes in these rules will affect the process model and drive the business application at application level.

Table 4.1: The Proposed Method in Sequence Process

Lifecycle	Phases	Stage	Stages Name	Artifacts
Domain Engineering	*Analysis*	DE1	Domain Requirement Analysis	Domain-specific Model
		DE2	Design Feature Model	Feature Model
	Design	DE3	Domain Template Design	Source Domain and Process Model
		DE4	Weaving (Mapping) Model Design	Mapping Schema and Relational Table
	Implementation	DE5	Customization of Domain Template	(De)-Activations of Domain Template
		DE6	Knowledge Transfer (Rule Generation)	Rule Generation
		DE7	Process Model Implementation	Configuration of Parameter
Application Engineering	*Analysis*	AE1	Application Requirement Analysis	End User Desire Model (Target Model)
		AE2	Feature Selection and Validation	Feature Analysis and Validation
	Design	AE3	Weaving Model Communication	Customize Domain Template
		AE4	Knowledge Transferred (Rule Generation)	Rule Generation
	Implementation	AE5	Generated DSRL	Configuration of Generated DSRL
		AE6	Customized/Adapted Process Model	Customized Process Model (Target Application)

Our thesis provides a product line and model-driven approach framework to allow a non-technical domain user to dynamically generate and customize the processes. According to the proposed framework, a domain template that contains both the domain and process models is developed at design time. This domain template is created by the domain expert or engineer, who has knowledge of the domain as well as an understanding of high-level modeling tools. The proposed product line, together with the model-driven framework forms an operational platform. This operational platform uses SPLE features to express the variability models and dynamic adaptations. These dynamic adaptations employ the rule generation and configuration for the process model customizations at run-time.

At run-time, in order to address problematic feature selection events, the variability model provides for decision-making and validation of the features selected by the user. The activation and deactivation of features in the variability model result in changes to the domain template. The change in the domain template is reflected at every level (e.g., activities, classes, etc.) in each of its components (the process model and the domain model). The translation of models from high-level to low-level rules takes place at run-time, i.e., based on the requirement model defined by the end-user according to their feature selection, the framework covers the dynamic rule generation, and the configuration of the process model.

4.3 Case Study

To illustrate the need for automatic rule generation and configuration, we introduce a rule generation and configuration process for process model customization that supports a digital content machine translation (MT) process[7]. The example is specified according to Business Process Model Notation (BPMN) in Figure 4.3. BPMN task expresses the functional and the operational activities such as data extraction for machine translation; the sub-activities or sub-processes express source to target data

[7] This process reflects the MT process conducted at the ADAPT centre.

translation. Every sub-activities have transition phase which is expressed like Transition between Login and Extraction sub-activities as a T-LG-EX.

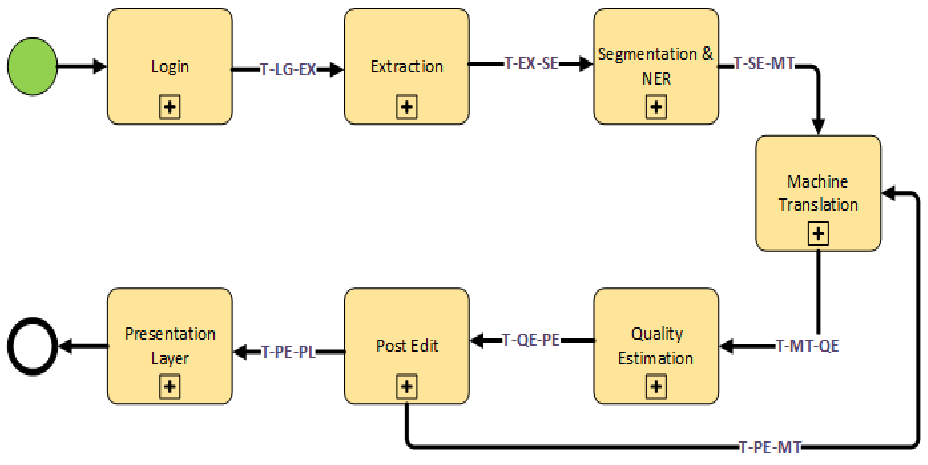


Figure 4.3: A BPMN Model for Digital Content Technology

The process model starts when an end-user is looking for an MT system. First, the end-user selects the source text and target language for an MT system. The source input operation, like text input or upload, is provided by the `gic:Extraction` (see definition of gic in Chapter 2, Section 2.3.1.1) sub-process (or step) of the digital content system. Once the source is validated by a validation constraint, then the next process can take place. This process is described by a process model that defines possible behaviors based on a number of references form the corresponding system. The set of behaviors constitute a process referred to as the extension (of that process) while the individual behaviors in the extension are referred to as instances. The constraints can be applied at states of the process to continuously determine its behavior depending on the current situation. The rules combine a condition (a constraint) on a resulting action. The target of this rule language (DSRL) is a standard business process notation (as in Figure 4.3) where the rules are applied at the processing states of the process.

This case study is of 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 while the data is exchanged and annotated with meta-data via web resources. As the content is delivered from creators to consumers, it follows a particular path. This path consists of 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.

The content process workflow is given in Figure 4.3, as an example for the rule-based instrumentation of processes. This process is governed by constraints. For instance, the quality of an MT system dictates whether further post-editing is required or not. Generally, these constraints are domain-specific, e.g., referring to domain objects.

4.4 Main Components of the Framework

In this section, we present a conceptual view of the proposed framework and approach, all components of the approach are illustrated in Figure 4.4 as features. This approach follows the basic principles of SPLE and has consists of two essential and elemental components such as *Domain Engineering at design time* and *Application Engineering at run-time*. At domain engineering level, we propose a domain template that has carried all possible and practicable a set of features in a domain and process model. Adding on to this, at Application Engineering, we base the foundation of all the possible set of selected features on the requirements and provisions of the end-user. In fact, During Design Time, we suggest and advocate projecting a set of models as a domain template. Hence, Domain experts and/or domain engineers design these models and at run-time, the featured model assembles the characteristics and attributes of the projected models provided by the end-user.

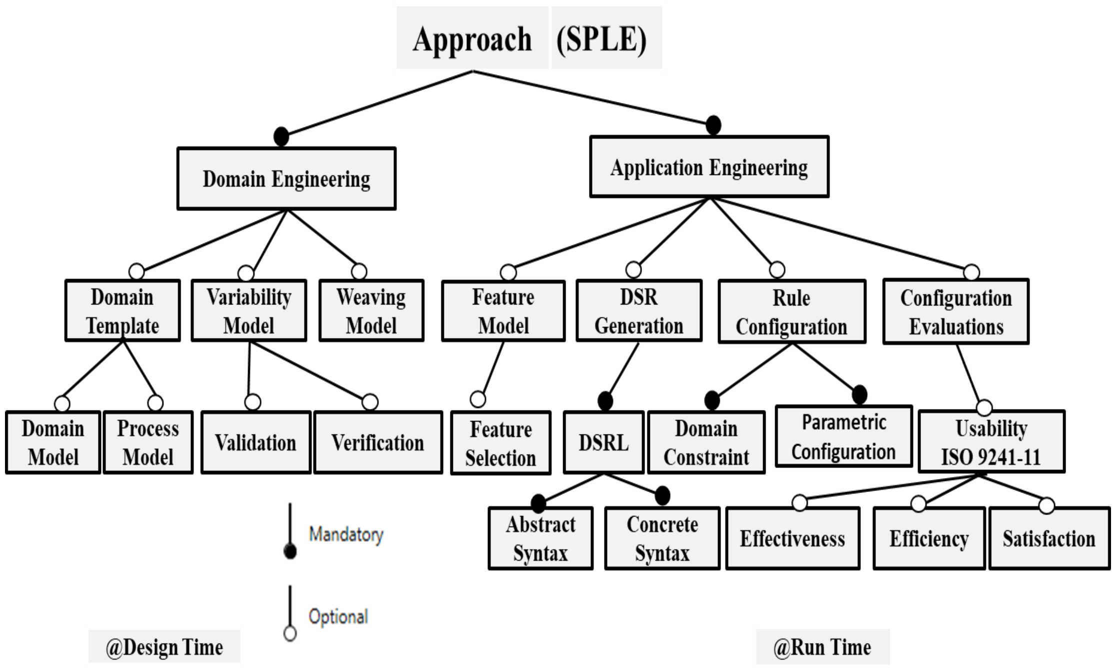


Figure 4.4: Main Components of the Framework.

4.4.1 Model-Driven Design Approach (Domain Engineering)

The Domain Engineering component supports the high-level design of models at design time (discussed in Chapter 5).

- **Domain template**: The domain template is the combination of a process and domain model. Domain experts and/or engineers design and develop the domain template at design time during domain engineering. It covers all the possible components and combinations of a particular domain. Now, we discuss various models involved in domain template at the domain engineering (design time):
  - **Domain Model**: Semantic models have been widely used in process management [67]. This ranges from normal class models for capturing structural properties of a domain to full ontologies to represent and reason about the knowledge regarding the application domain or the technical process domain [143]. Domain-specific class diagrams are the next step of 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, behavior,

and relations. A class diagram is based on a feature diagram model and helps in stabilizing relation and behavior definitions. Note that we use the class aspects (subsumption hierarchy only) despite having an underlying domain ontology here. The domain model is described in detail in Section 5.5.1.2 of Chapter 5.

- **Variability Model:** Variability models are known for managing variability information space from the recognized features and identified constraints in product line engineering and can control this customization.
    - **Validation**: Validation is the task of demonstrating that the model is a reasonable representation of the expected objectives and performance of the system. There are three different aspects of validation: (i) assumption, (ii) input parameter values and distributions, and (iii) output values and conclusions. They are detailed in Chapter 8 (Section 8.4 Analyzing the Validation).
    - **Verification:** Verification is the task of ensuring that the model is a reasonable representation of the intended outcomes. This is usually a set of programs, rules or conditions which are predefined for different tasks, as detailed in Chapter 8 (Section 8.4 Analyzing the Validation).
- **Weaving Model**: Weaving models are models which capture various kind of relationships among models [142]. They work by finding a similar type of patterns between model elements. These patterns are then integrated into a tabular form. We discuss them in detail in Section 5.5.1.3.

4.4.2 Adaptation of Process Model (Application Engineering)

The Application Engineering component supports the feature selection from feature model, rule generation and configuration for adaptation of the process model at run-time (discussed in Chapter 8).

- **Feature Model:** The feature model is the most important result of domain analysis. A feature model covers the aspects of software family members, like commonalities and variabilities, and also reflects the dependencies between variable features. The dependencies between the variable features are depicted by a graphical representation known as a feature diagram. In a feature diagram, the *mandatory features* are present in a concept instance if their parent is present while the *optional features* may be present as well. A feature diagram also consists of *alternative features* which have a set of features of where only one is present. Whole *groups of features* are a set of features from which a subset is present, only if their parent is present. Between two features there may exist a *Mutex* or a *Requires* relation. If two features are in a *Requires* relation, that means that when one feature is selected, the other feature must also be selected. In contrast, *Mutex* means that once one feature is selected, the other feature must be excluded (mutual exclusion). We discuss the details in Section 5.5.1.1.
    - **Feature Selection:** The system elements, their functionalities and their dependencies on each other are contained in the feature model. Further, it contains mappings between the stakeholders' goal and the pertinent SPL feature that achieves such goals.
- **DSR Generation**
    - **DSRL**: The DSRL is a combination of rules and BPMN. A DSRL process, based on BPMN and ECA rules is the main focus of the operational part of the DSRL system (i.e., to check conditions and to 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 to discuss business process variability, which primarily forms a structural customization perspective. However, it also uses an ontology-based support infrastructure. This is further detailed in Chapter 6. The DSR generation has the following components:
        - **Abstract Syntax** In the context of rule language, abstract syntax refers to the data structure (possibly an abstract data type), which

is free of any specific representation or encoding. The texts are mostly represented in this way and typically stored as abstract syntax trees. Abstract syntax, which concerns itself only with the structure of data, differs from concrete syntax, since it also contains information about its representation. We define and discuss the abstract syntax in Section 6.3.1.1.

    - **Concrete Syntax:** Concrete syntax contains concrete syntactic features like parentheses (for grouping) or commas (for lists), which are not part of the structure and hence never appear in the abstract syntax. We define and discuss the concrete syntax in Section 6.3.1.2.

- **Rule Configuration**

    - **Domain Constraint:** Domain constraints are considered as the most basic level of integrity constraints. Once the data is given, domain constraints may easily be verified as the attributes possess specific values under practical conditions.

    - **Parameter Configuration:** The set of the domain constraints are configured in the generated DSR with some specific value based on the end-user requirement.

- **Configuration Evaluations**

    - **Usability ISO 9241-11:** In this overall framework, the usability criteria is defined as effectiveness, efficiency, and satisfaction. It specifies that end-users achieve specified goals in domain-specific environments. We define each of these goals below:

        - **Efficiency:** The comparison between the time taken to configure domain constraints in the manual and semi-automatic process, based on that identifying and using which process is more efficient (Section 9.7).

- **Effectiveness:** The generated rule configuration in terms of accuracy to prevent or protect errors to achieve the configuration of domain constraints goal (Section 9.8).
- **Satisfaction:** The measure of end-user comfort and acceptability of the overall framework (Section 9.9).

The configuration evaluation criteria define present experiments and discusses results in Chapter 9.

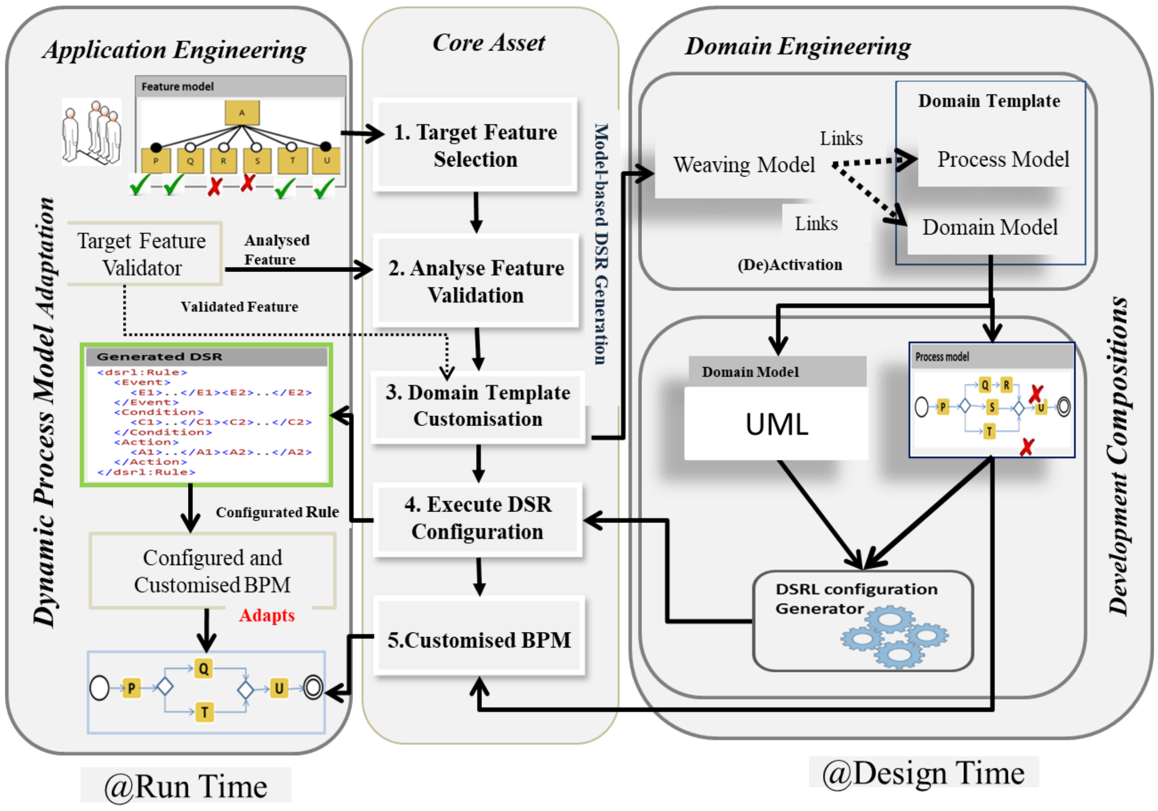
4.5 A Framework for DSR Generation and Configuration

This research proposes the following 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 to the feature model and the domain template (via a virtual relational table) at design time, the customization and adaptation may take place at run-time.

Customization and adaptation process begin when the feature selection is initiated in the feature model. The process is highly abstract, which should make it easy to understand and apply to end-user. At run-time, we provide a web-based infrastructure that detects the feature selection in the context and enables dynamic activation or deactivation of features at the domain engineering level, with the help of core assets[8].

An overview of this approach is shown in Figure 4.5. As illustrated, the approach covers both dynamic process model adaptation and core assets. The aim is to design an appropriate architecture that models the feature selection based on end-user requirements and generates the configurable rule for the process model customization.

[8] Core assets are defined in Section 4.5.2.



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

Figure 4.5: A Framework for Dynamic Customization of Process Model

At run-time, users select their requirements via the feature model and configure the rules. The framework facilitates the quality configuration in terms of effectiveness, efficiency and user satisfaction.

Every selected target feature in the feature model is continually analyzed and validated by the target feature validator, in accordance with the basic criteria of the feature model. The selected feature immediately gets reflected in the domain template environment as activation or deactivation of components via the weaving model. In case of violation of feature selection, the target feature validator prompts a proper error message to the end-user. For example, if the user removes a mandatory feature of the system, the target feature validator prompts the user with an error message. Next, we briefly discuss each phase and describe their activities.

The framework examines the following approach to offer a dynamic solution for rule generation and process model customization, consisting of two sections:

Development Composition and Dynamic Process Adaptation. The development composition encapsulates elements such as Weaving Model, Process model, metamodel and a DSR configuration generator. This method comprises of two building blocks that utilize the domain model variability and commonality at run-time: (i) a domain model customizer using a weaving model (see Table 5.2.); and (ii) the feature model and feature selection (Section 5.5.1.1). The weaving helps in execution of the customization process based on valid features. The valid features are analyzed and verified features that are captured from the end-user requirements. In the Rule Generator (see Figure 4.5), the models and metamodels, created in the domain model customizer are used in MDA at the different level, i.e., M0, M1, M2, and M3 as illustrated in Figure 5.5 (described in section 7.5.2 Meta levels and implemented in Section 7.6). The generated rule configuration has two different types: pre-configured which is defined at the design time and post-configurable metamodel which has a dynamic definition.

We propose the following approach for dynamic adaptation of rule generation and configuration for process model customization. The processing of mandatory and optional features (see Chapter 5, Table 5.1 for more details) of the process model and metamodel are carried out by the variability model. While the features of a process can be activated or deactivated at any moment (at design or run-time), we propose to perform activation/deactivation at run-time (i.e., at the application level). In this research, we consider two different groups of users: (i) experts in the domain with modeling knowledge, and (ii) users that have a functional domain knowledge, but who are non-technical. Therefore, we select a Software Product Line Engineering (SPLE) platform where users from both these groups can perform their tasks efficiently and independently. The SPLE is a standard model to develop software applications using the platform and mass customization [41]. In our framework, a domain expert can design high-level solutions for a domain. Based on that solution, the end-user can modify and customize model elements (activities) in any process over time where the activation and deactivation of model elements depend upon the variability model and the end-user requirements.

4.5.1 Dynamic Process Model Adaptation

In the Dynamic Process Model Adaptation, the proposed structure carries out the following steps. First, the model-based configurator collects the information from various models. If the captured requirements of target features have violated the feature selection or customization activities such as rename, move, update and delete without selecting the parent feature, then the activation and deactivation are not processed. After this, the customization model passes the captured requirements to the weaving model for further processing. Finally, the generated and configured set of rules are obtained from the Development Composition (Figure 4.5).

4.5.2 Core Asset

A core asset allows connecting, providing resources, coordinating, and supervising the domain and application engineering activities. To obtain the final product, the selected core assets are processed by following the production plan. The production plan specifies several tasks such as code generation, compilation, and execution of programs.

4.6 Summary

This chapter has answered the *RQ1,* defined in beginning of the chapter. The chapter has introduced the main components of our approach and presented an overview of our framework and generate the DSR and configure it for customizing process model adaptation. SPLE platform and its levels introduced. A DCT business process presented as a case study. The main components of the framework were brief introduced, and the domain engineering and application engineering lifecycles explained. Finally, this framework for domain-specific rule generation and configuration over design and run-time was defined and discussed in Section 4.5.

CHAPTER 5

MODEL-DRIVEN DESIGN APPROACH AT DESIGN TIME

5.1 Overview

The previous chapter outlined the research approach and framework components overview. This chapter will introduce the design-time modeling of the proposed approach during domain-engineering (as per Figure 4.4 in Chapter 4). The contribution of this chapter is functional view of model-drive approach at design time and to define the definition of domain model language. In this chapter, the variability model, domain template, and artifacts are designed and developed at design time i.e.

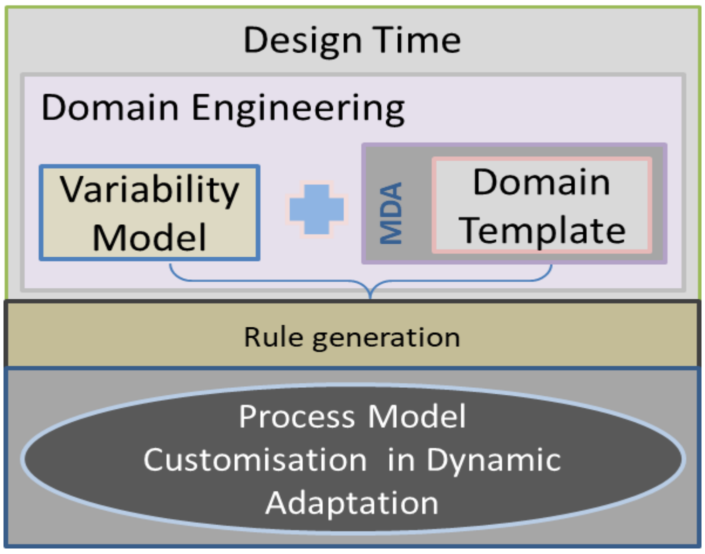


Figure 5.1: Scope of Chapter 5

during the domain-engineering phase, to support the DSR generation and configuration for process model customization in a dynamic environment (Figure 5.1). This chapter aims to illustrate a conceptual view of the prototype, including domain adaptations, the definition of domain model language and its syntax. This syntax is used in model to text translation and it is discussed the DSRL automatic generation in Chapter 7.

5.2 Introduction

The implementation of this approach facilities, the design of models is organized in a domain and process model that associations a variability model and a domain template at design time. Due to the variation points in a domain-specific environment being fixed, a traditional SPLE will be used instead of a DSPLE (Dynamic Software Product Line Engineering). Following the principles of traditional SPLE, the software products are developed by selecting features and configuring a set of rules the shared core assets at design time. The domain templates are defined, designed, and developed by domain experts at design time though adapted at run-time.

With this approach, the end-users select features based on requirements, resulting in a customized process application. It is therefore proposed that the software process model is based on the domain template and features models allowing us to map a relationship between models at run-time.

Among the set of models, the variability model describes the variants which are essentially the representations of variability objects within the model artifacts [39]. The first step is the customization of the domain template, which is followed by generation of configurable rules at run-time. This allows the variability model in representing the features and makes it easier to understand the possible features within the domain under consideration. In addition to the above, we propose the customization of the process model carrying at run-time, a variation point similar to the domain and feature models.

At run-time, the DSR is a configurable rule and is used to support *critical applications* such as stock exchange, banking or transportation, etc. The main advantage of using such a rule is that it eliminates the requirement to compile or build the rules. It is automatically generated at specific locations (changed by the end-user) and does not require republishing, redeployment or rebooting of the application server (changes reflect on the server).

The chapter is structured as follows. Section 5.2 introduces variability and software product line engineering relating to the implementation of customization of models. Section 5.3 defines the basic component of variability and SPLE principles. The model-driven approach applies into SPLE in Section 5.4. In Section 5.5, the processes involved in designing dynamic customization of models describe. Section 5.6 defines the abstract syntax of the domain model and its syntax definition. Section 5.7 presents the summary of this chapter.

5.3 Variability and Software Product Line Engineering

The SPLE supports the systematic reuse of similar software products that create and maintain software assets with the help of various tools and techniques. In software product lines (or systems families), the products are derived by selecting the features that are needed while rejecting those which need not be part of the product. Therefore, SPLE exploits commonalities among a set of systems in a particular domain, while managing the variabilities among them. This results in improvement in time and quality when the product reaches the market while achieving the goal of systematic reuse. Commonalities are the elements with the highest reuse potential and variabilities represent capabilities to change or customize a system [144].

The SPLE consists of three core components such as Domain Engineering (known as reference model development), Application Engineering (known as Product Development) and Core Asset (discussed in Section 4.5). At Domain Engineering, the variability of an SPL is defined with common and variable domain artifacts. At Application Engineering, the development of software products is carried out by selection and configuration of shared artifacts [145]. Core Asset provides for connecting, allocating resources, coordinating and supervising the domain and application engineering activities.

The SPLE supports systematic reuse of the set of similar software products which offered to share a collection of by software companies. The SPL share the method, technique, and tools for developing and maintaining a common software collection to for software systems to share the common resource or set of software assets. This can be expressed in four concepts, as shown in Figure 5.2

- **Software asset (Feature Model)** shows ways to compose and configure the software input assets with test cases using source code components, including requirements, documentation and architecture for creating any product in Software Product Line (SPL) product. In order to bring up some product variation, a few assets can contain internal variation points which is configurable in multiple ways for producing the difference in behavior.

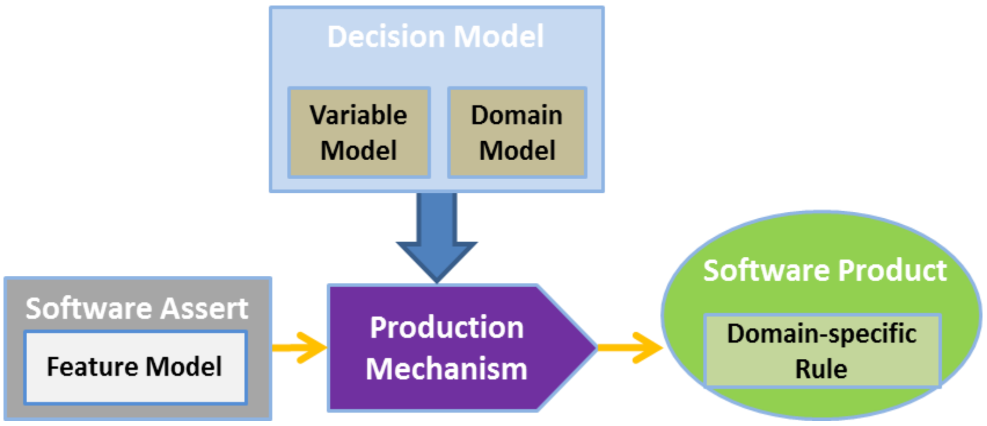


Figure 5.2: Basic SPL Concepts

- **Decision model** refers to the optional and variable features of SPL products. Each SPL product (variable model and domain model) is defined in a unique way through product decisions (i.e., choosing the variable and optional features for a particular domain).

- **Production mechanism and process** refers to describing the means of composing and configuring products from feature model as software asset inputs. With the help of product mechanisms, the decision is taken on determining the particular software asset input as well as well as variation point configuration in assets.

- **Software output products (DSRs)** is the collection of all DSR set of products which might be used in SPLE platform and translation of the graphical models into text used in the MDA technique. The SPLE scope may be assessed through a set of software product outputs which may be produced by the feature model as software assets or domain model as a process artifact.

5.4 Model-driven Software Product Line for Rule Generation

It is proposed to abstractly support the different SPLE aspects from MDA. The MDA aims to capture 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. This assists in avoiding accidental implementation details and is also more suitable for results and analysis. Models in MDA context have a greater value than just being supportive artifacts; rather, they are actually source artifacts which can be utilized for automated analysis and/or rule generation.

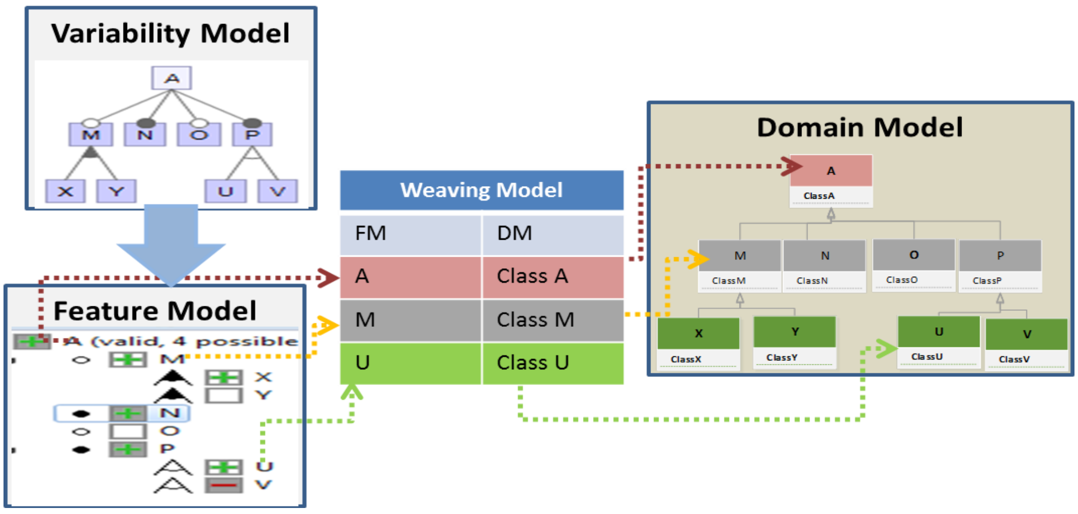


Figure 5.3: Models-driven Support to the SPL for Rule Generation

The model-driven approach aims to provide 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. As the domain model is the primary source of the rule generation, the definition of a bridge between the elements in the variability and domain models could be used to support the dynamic rule generation in the underlying domain-specific environment.

To this end, a weaving model is used as an extra software asset input to the domain template changes in the features of the variability model. The weaving model in effect works as a bridge or object relational table between the elements in domain template models.

The domain template customization is depended on activation or deactivation of features in the feature model (Figure 5.3) to manage the requirements of a domain user at feature selection level. Therefore, DSRs generation depend upon the features of the feature model. The rule generation on the variability model produces an adaptation space with 1) all the possible domain constraint configurations of the process model specifically in terms of active and inactive features in the feature model with parametric values, and 2) customization of the process model in terms of functional and operational tactics. To avoid problems or interruptions (e.g. error, system halt, and malfunctioning, wrong interpretation) during rule generation and configuration in critical service application, the feature model is therefore used. Relevant feature configurations should be validated and verified at runtime (Figure 4.5, second flow “Analyzer Feature Validation” at core assets). For these reasons, model adjustment is properly performed in the domain template. The customized domain template further uses MDA at domain engineering level of SPLE.

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

- Metamodeling refers to type of systems with 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 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.

The advantages of using together, the MDA and variability of SPLs are: (i) Ability to capture the similarities and variabilities in a set of systems and (ii) automation of repetitive tasks.

Figure 5.4 shows how to combine modeling and model transformations to develop a framework for rule generation and configuration for Digital Content Technology. First, the assets of the Domain template are considered as model elements, describing a family of DCT. These model elements conform to the metamodel of the domain model, which is a DSL for the DCT. Second, characteristics of a DCT are specified by a decision model. Third, the scoping of the domain model is performed by projecting features on DSL by using the weaving model. Ultimately, the translation of model provides output of the system as a text.

Given such Domain Engineering of SPLE, we argue that the modeling effort put in to define the SPLE is useful for two reasons: (i) It is responsible for realizing the system, and, (ii) provides autonomic behavior during execution. Describing variants using the knowledge captured from earlier variability models helps evolve a system. Additionally, the necessary steps in order to reconfigure software system can be assisted by variability models. Next, the models that conform approach are described (see Figure 5.5), while also discussing systems that can be enabled using these models.

5.5 Process for Design Dynamic Rule Generation

We describe the Design Time (Domain Engineering) components of our approach as per Section 4.4.1 (Chapter 4) while the rest (at Run-time- Application Engineering) is discussed in subsequent chapters. The active events are described in the following sections.

5.5.1 Creation of the Initial Domain Template

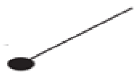
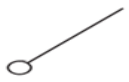
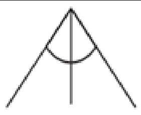
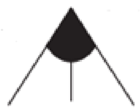
In our approach, the initial domain template is a generic template for a domain; it covers all the possible combinations carried out with commonality and variability in terms of variability points. It is designed during domain engineering lifecycle in the domain-specific environment by the domain experts. The domain template represents the following: (i) the variability operations, which are involved in the adjustment of domain template for generating the rules and process model customizations; and (ii) the sequence flow of a model among the design models and their operations that state the order in which operations are performed. The domain engineering lifecycle is accessed by a domain expert or stakeholder to define, design and develop the generic domain template. This template consists of two different models: domain and process models with variability model and its points (the variability notations). Both models are tightly coupled to each other so that if a domain expert changes one model, then the other model should be changed as well.

The model notations can be used to illustrate the sequences of the models with their dependencies in both models. In Figure 5.3, the links among feature model and an activity diagram of UML (domain model) is represented with different color doted lines. In this work, a domain model is represented as a UML class diagram in a domain template. The UML express the hierarchical relation of child-parent classes of models in terms of sequential and dependable components.

5.5.1.1 Feature Model

A feature model refers to a set of features arranged hierarchically and is most commonly used in software product line engineering to represent the variability and commonality in a variety of development artifacts. This includes requirements, design, and test [146]. The main idea behind feature models is the set of features in hierarchical structure. The feature models, over features of the product line, define variability relations, constraints, and dependencies while also formally and graphically representing features. The feature is defined by Bosch et al. [31] as "*a logical unit of behavior specified by a set of functional and non-functional requirements*." This work corroborates with this definition by basing the DSR generation on both functional and non-functional requirements. The hierarchical set of features in a feature model are composed.

Table 5.1: Types of Feature Models

Mandatory	
Optional	
Alternative	
OR	

In Table 5.1, types of features are summarized which are represented as a tree-like structure [147]. These features are described below:

- **Mandatory feature:** In case of selection of a parent feature, the corresponding child feature, if mandatory, must be selected along with it.

- **Optional feature:** In case of selection of a parent feature, the corresponding child feature, if optional, may or may not be selected.

- **Alternative feature group:** In the alternative (XOR) feature group, only one feature must be selected exclusively in the feature model.

- **OR feature group:** More than one features in the OR feature group must be selected in the feature model.

Example: `gic:Extraction` Feature Model

Figure 5.4 illustrates the feature model for `gic:Extraction`(F0 represents as a Feature Model) activity of DCT domain We use as our case study. For instance, the *Document File (F4)*, *Multimedia (F5)*, *Web URL (F6)* and *Text Input(F7)* features are variants that can be used during execution to accomplish the data extraction functionality in the variability model point. The features model contains two different features as express in OR (*Document File* and *Multimedia*) and alternate features (*Web URL* and *Text Input)* of the `gic:Extraction.`

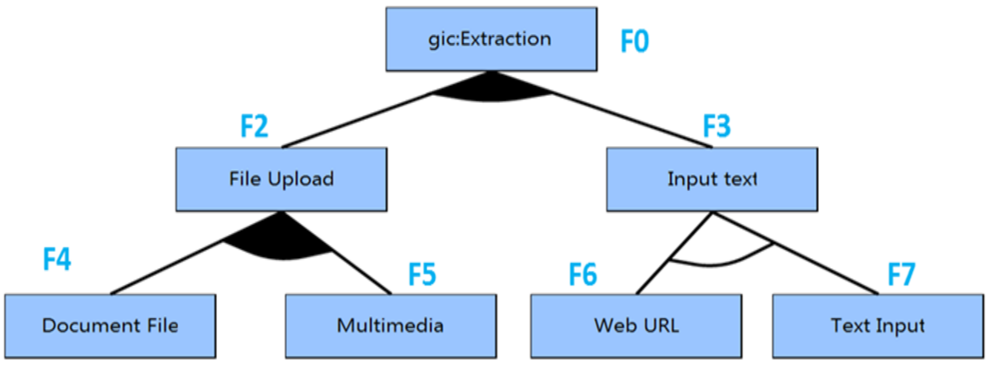


Figure 5.4: Feature Model of `gic:Extraction`

5.5.1.2 Domain Model

The similarity of a domain model to UML class paradigms may be noted in a modeling standpoint. This visual formalism is widely known in the data-modeling community and is language independent [148]. A few of the elements of UML class diagrams can be separated although domain model is not a visual language, which is briefly described in a later section. Leveraging known mechanisms or formalisms for a domain in a relevant DSL is generally a good practice. Thus, intuitive knowledge of

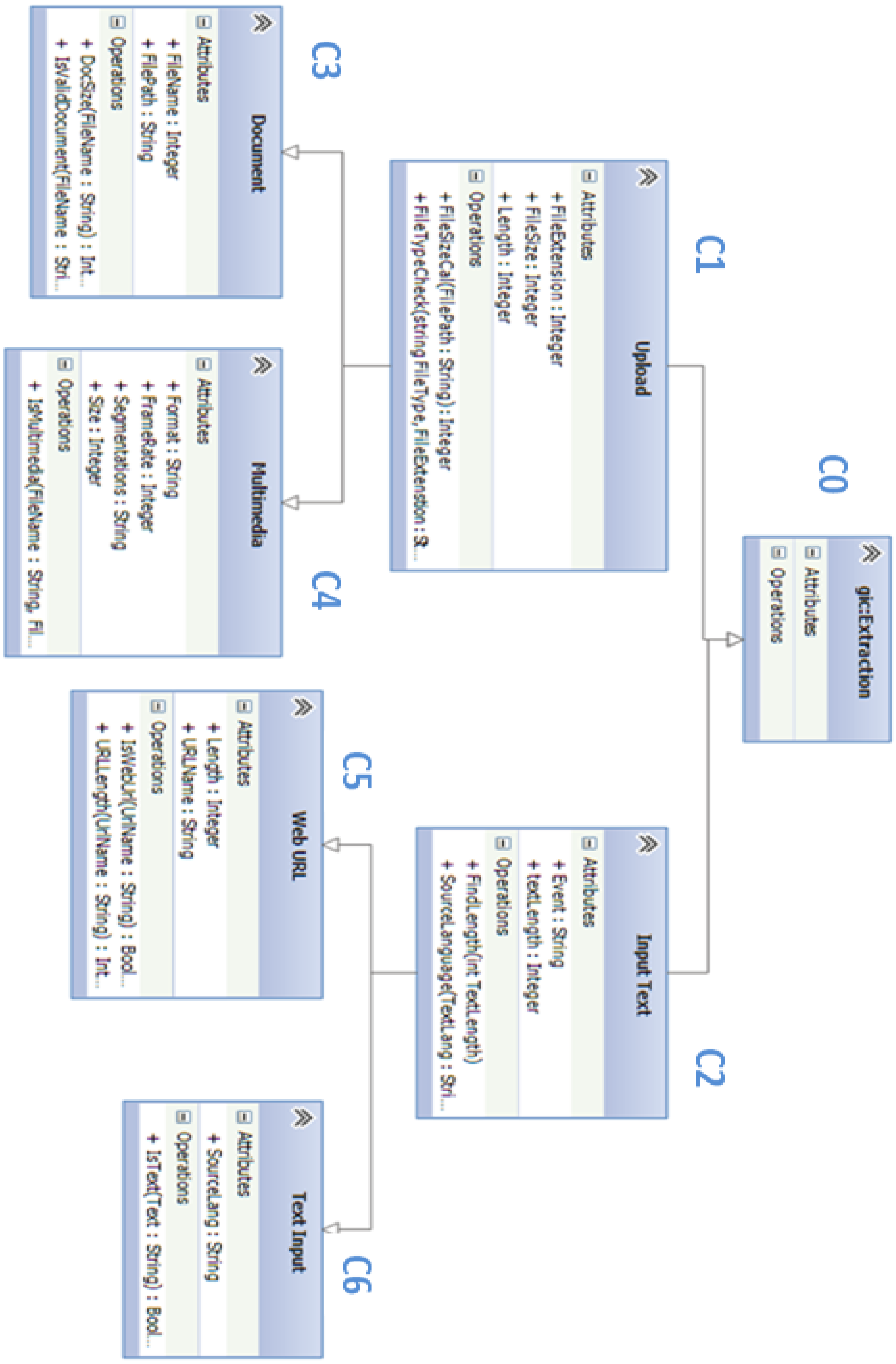


Figure 5.5: Domain Model of `gic:Extraction`

users of a domain can be leveraged while using or learning the DSL with the help of UML class diagrams, which are frequently used as formal input for Database Schema design from a functional point of view.

It extents or spans from the level of rule language to the level of the configurable process model. For describing its structure and inspecting a concrete example of a domain model, the domain-specific rule language is introduced. Subsequently, the implementation of the DSL in digital content technology is analyzed, where the translated rule of the configurable parameterized rule language is provided, solidifying the semantics of the language.

In the content use case, class diagrams of `gic:Extraction` (C0 represents as a Class of `gic:Extraction)` and its components based on common properties, are shown in Figure 5.5. The class diagram focuses on `gic:Extraction`. The two major classes are *Upload* (Document or multimedia file) and *Input Text* (Web URL or Text Input), consisting of a different type of attribute like content: string, format: string, or frame rate: int.

5.5.1.3 Static Weaving model

A product line feature model represents variabilities and commonalities. These are represented in a concise taxonomic form. Additionally, it must also be noted that simply put the features are symbols in a feature model. Further, the semantics are obtained by feature mapping of other models (feature model with *F*, domain model with *Class*, and process model with *Activity*) (see Table 5.2). Next, with the weaving model [144], we show how to perform the mapping. We use static weaving model for managing the variability relationships among all models. The principle argument for using the static weaving model is domain-specific environment. When the domain experts have significant knowledge of domain and its features, 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.

Table 5.2: Example of Weaving Model

	Feature Model	**Process Model**	**Domain Model**
	Fgure 5.4	Figure 8.4	Fgure 5.5
1.	Document File (F3)	Document File (T4)	Document File (C3)
2.	Multimedia (F4)	Multimedia (T5)	Multimedia (C4)
3.	Web URL (F5)	Web URL (T6)	Web URL (C5)
4.	Text File (F6)	Text File (T7)	Text File (C6)

The weaving models are used during definition and capture of relationships among model elements. The relationships between model elements are present in many different application scenarios [142]. The relationships among model elements and features are defined by using the weaving models. Let us consider a weaving model with the following specifications:

- It is defined among a feature and domain models.
- The model is represented by <WM, FM, DM> where WM, FM and DM are weaving model, Feature Model and Domain Model respectively.
- It consists of elements linking set of elements in FM and DM.

The link management is supported by elements of WM as follows:

- **WElement:** All elements inherit from this, i.e., it is the base element.
- **WModel:** It is the root element. It comprises of elements of weaving model and the references to the corresponding woven models.
- **WLink** refers to the link that is present among the model elements.
- **WLinkEnd**: It refers to link endpoint types, which denotes a linked model element allowing creation of N-ary links.
- **WElementRef** elements are related to a dereferencing function. This input to this function is ref attribute value, while in retrospect the function returns the linked element.

The string element (WElementRef) discussed above allows creating N-ary links. Here, the links amongst elements of feature and domain models are specified. The features in the feature model are obtained from the K set (active or deactivate features) while DM comprises of domain model elements. In the present scenario, a `WLink ∈ WM` signifies inclusion of an element `d1 ∈ Domain Model` to the resultant Domain Model configuration, if `f1 ∈ FM` is active. Consequently, the configuration of the domain model can be instantiated by deactivating or activating features in the feature model, through the weaving model. Finally, the weaving model is able to access the concrete elements through a reference, which is contained in feature and domain models.

5.6 Definition of Domain Model Language

Domain model serves as the very basis of all types of business applications that run on a Domain, both individual as well as enterprise applications. The objective is to define the language for domain model in order to determine the internal data structures or when a schema is used. It is easy to transform or translate the graphical domain model into textual rule language for a particular domain. In this scenario, the objective of data structure refers to the representation of a domain model in a changing environment, as in the case with a rule language. This is because the target of a language is to map the translated domain model knowledge into XML schema of DSRL following the rule paradigm.

The main objective is to come up with a flexible rule language translation from a domain model in order to overcome the domain constraints and also to configure and customize the process model activities in a volatile environment.

The general language definition is a mapping between a collection of models along with the models themselves. Usually, concrete and abstract syntaxes along with the semantic domain constitute a language. Mappings are nothing but the association of model elements. The correlations among various syntaxes, such as abstract and concrete, abstract semantic domain are inherent characteristics of a language.

5.6.1 Language Description

Metamodeling is used to accomplish specifications for an abstract syntax. We introduce the Domain Model language by analyzing its syntax definition (Figure 5.6 shows in EBNF notation). The language with its basic notions and their relations are defined with structural constraints, multiplicities, precise mathematical definition and relationships which are implicit. The visual appearance of the domain-specific language is accomplished by syntax specifications, which is done by assigning visual symbols to the language elements that are to be represented on diagrams.

1	Domain	::=	<Domain Model>	Domain definition
2	Concept	::=	<Concept>	Concept definition
3	Class	::=	<Attributes>, <Operation>, <Receptions>, <TemplateParameters>, <Component>, <Constraints>, <Tagged Values>	Class Definition
4	<Relations>	::=	<Association>| <DirectedAssociation>| <ReflexiveAssociation>| <Multiplicity>| <Aggregation>| <Composition>| <Inheritance/ Generalisation>| <Realisation>	Class relationships
5	<Association>	::=	'→' | '✸' | '::'	Structural relationship between objects (classes) of different type
6	<Type>	::=	<BuiltinType>| <UCaseIdent >| <EnumType>	Domain Model type concept type or extended type enumeration type list type
7	<PrimitiveTypes>		<String>,<Integer>,<Bool ean>, ...,<Date>	Domain Model primitive (built-in) types

Figure 5.6: Syntax definition for Domain Model Language

The process of defining or expressing a language in general, or specifically, a rule language encompasses a variety of activities which primarily involve using the concepts of abstract and concrete syntax. This is achieved by designing semantics and grammar. These activities are performed through conceptualization, delineation and evolving a systematic domain specific rule language system, elucidating the functions and its frameworks, priorities of operators and its values, naming convention procedures for internal, and external, etc. A set of rules, conforming to BNF or EBNF grammars which are processed by rules or process engines to produce the set as an output is used to express syntaxes. Since, abstract syntax and grammar are manifested in concrete syntax, the rules, so generated, obey them to define the domain models and concepts.

Propositions for the operationalization of the association between concrete and abstract syntax definitions are elaborated in [149-151], which suggest that concrete syntax definitions are adapted into abstract syntax definitions. Actually, the idea is based on the concept that concrete and abstract syntax definitions might be incomplete but by themselves complete each other based on certain heuristics and mapping name [149]. However, the concrete syntax [150] claims that it is mapped operationally to the abstract syntax which is based on grammar transformation. The abstract and concrete syntax definitions are linked through annotations, (An exemplary MDA approach is the one of TCS for KM3 [152].) However, these propositions do not offer potential automatic solution for change propagation.

An attribute grammar is used to map concrete syntax to abstract syntax which is the traditional approach. There are several viewpoints to combine grammar and attribution variations [153], but they are not appropriate in integrating abstract and concrete syntaxes. We think that coordinated editor model collection problems are greater than the challenge faced in concrete or abstract syntax integration.

5.6.2 Semantic Checks

The input model is assumed to be well-formed in the presented transformations, and this is the basis of the information provided earlier. Hence a separate phase is assigned in order to check the formation of the input included before the actual compilation. The following constraints are checked in this phase:

- These standardized checks bear some resemblance with the traditional semantic checking phase of a GPL compiler. Besides, more domain specific issues continue to exist for reporting to the user domain model. The primary reason of these issues is the usage of annotation on concept members.

- Moreover, we avoid the usage of the name annotation on a concept member having list type. Also, as the library used for implementation is not capable of supporting such behavior, using a unique constraint on such a member is not allowed (it can be checked only at run-time by the .net library). Adding more domain-specific checks at compile time is not possible as the checks enumerated earlier have no cost in the .net compiler [154]. Therefore, according to us, this makes a strong feature for DSL.

5.7 Summary

This chapter introduced a set of models (template model) are created in the Design Time of this framework to support the rule generation and configuration in the process model. An approach to achieve an automatic rule generation and configuration by combing SPLE and MDA is introduced then the variability and the software product line engineering principle is applied for customizing the domain template. The model-driven approach is applied in the software product line to customize the models for generating DSRs. A process approach is then presented to engineer the dynamic rule generation at run-time and define the domain template to provide and define models systematically. The domain model definition and language description were defined with semantic check.

CHAPTER 6

RULE LANGUAGE DEFINITION

6.1 Overview

In this chapter, the abstract and concrete syntax definitions are formalized along with rule language formulation. In other words, the aim of this chapter is to define the DSR syntax to help the translation of a domain model. Further, we utilize this rule language to represent the syntactic and semantic properties of the domain model, as a set of generated rules, by using a model-driven architecture (MDA). Here, the syntactical properties consist of the name, while the semantic properties are defined as a data type of attributes and functions. In simple terms, it is a knowledge transfer from a high-level domain model, and the transferred knowledge is then in the form of a set of DSRs that are expressed as a domain-specific rule language.

The technical contribution of this chapter is to define the domain-specific rule language. The abstract and concrete syntaxes are defined for generating DSRL, which help in translating low-level text as a DSR, from the high-level graphical domain. In chapter 9, the configuration of generated DSR is validated and evaluated on a usability criteria. In this chapter, A Domain-Specific Rule Language (DSRL) is defined and discussed the DSRL automatic generation in the next chapter (Chapter 7). This chapter partially answers the following research question (RQ2) in terms of What is a rule language? what is the structure of a rule language and how it able uses by non-technical domain experts.

RQ2. How to implement a framework to support a rule language that is usable by non-technical domain experts?

While solving this problem, we focus on a domain-specific language, i.e., a rule language providing the schema to represent the translated graphical model of a domain model into text form in a domain-specific environment where the domain is '*digital content technology*'. The main contribution is to define the foundation of DSRL in terms of syntax (abstract syntax and concrete syntax), structure and grammar of the rule language.

6.2 Introduction

Rules are standard statements that express execution plans, procedures, and policies while defining terms and governing the overall operations and functions of a business stand-alone units, in a declarative manner [137, 155-157]. A rule approach is a methodology—and possibly special technology—by which you highlight challenges and change policies, from strategic functions and operational perspective of the system. Therefore, this chapter focuses on a rule language, the structure of language, and the language definition (as discussed in Chapter 3, Section 3.2.2 (b)). We now endeavor to provide a formal definition of the rule language solution.

In this context, a defined rule language model includes the information about the structure of rule formulation. The definition of the model is used for defining the abstract and concrete syntax, which are essential for model translation from a high-level domain model to a low-level rule design and generation process. These syntaxes capture and represent the structure of graphical model elements into the domain-specific language in form of a set of rules. The definition of a graphical domain model is defined and presented in Section 5.3.

The rest of the chapter is structured as follows: the structure of the rule language is defined in Section 6.3 in terms of syntax and language description; The abstract and concrete syntax are introduced in technical spaces in Section 6.3.1.1 and 6.3.1.2 respectively. In Section 6.4, the DSRL definition and description in ECA models is discussed. The definition of DSR is detailed in Section 6.4.1 is the definition of DSR as an *event condition action rule language*. This section defines the ECA rules and general expressions. The ECA description and its model definition are described in Section 6.4.2. It details upon the ECA Model and rule express in XML format. The chapter closes with a summary of the ECA rule approach and fulfillment of the research objective.

6.3 The Structure of Rule Languages

For defining the structure of a language in general, or a rule language, containing several kinds of activities such as need to specify the concept of the syntax (concrete and abstract), develop the grammar and then design the semantics for delineating the meaning of the language. These activities require one to conceptualize, design and develop a systematic domain – specific rule language systems, defining the functions and parameters, precedence or priorities of operators and its values, naming convention systems for internal and external uses. Certain rules are used to express these syntaxes. These rules conform to BNF or EBNF grammars and can be processed by process engines to generate or transform the set of rules as an output. These generated rules are in accordance with the abstract syntax and grammar, defined by the domain models since both (abstract syntax and grammar) are reflected in the concrete syntax.

6.3.1 Rule Syntax

To define or express a language in general or a rule language, that contains various kinds activities.

> "When describing or implementing a language, it is customary to distinguish between its abstract syntax, i.e. the hierarchical structure of the language, and its concrete syntax, i.e. what the language looks like as it is read and written" [158].

The first activity, or the primary requirement, is to specify the concept of the syntax (i.e. abstract and concrete syntax), develop the grammar and then the semantics to suit the meaning of the language. The activities are completed by conceptualizing, designing and developing a systematic domain-specific rule language. The functions and its parameters, priorities or precedence of operators and their values, the naming convention for internal and external uses are defined. The syntaxes are expressed with certain rules, conforming to BNF or EBNF grammars that can be processed by rules or process engine to transform or generate the set of rules as an output.

The generated rules follow the abstract syntax and grammar to define the domain concepts and domain models because both the artifacts (abstract syntax and grammar) are reflected in a concrete syntax. For a definition of the syntax, we use the syntax definition formularization (SDF) [159, 160]. The SDF integrates the definition of the lexical and context-free syntax (Figure 7.1 at Meta-metamodel and 7.3 at M3 part of CIM).

6.3.1.1 Abstract Syntax

The abstract syntax refers to a data structure, which contains only the core values of data in a rule language, with the semantically relevant data contained therein. It excludes all the notation details like keywords, symbols, sizes, white space or positions, comments and colours of graphical notations. The abstract syntax may be considered as more structurally defined by the grammar and meta model which signifies the structure of the domain. The grammar of the rule language can be expressed using BNF grammar. The grammars define the syntax of the language when analysis and downstream processing of rule language are the main usages of *abstract syntax*. The stream of characters are derived the abstract syntax by grammar and mapping rules.

The resemblance between abstract syntax and meta model is that both of these refer to the data structure, while ignoring the notation. However, they differ in the sense that a meta model is generally defined first, disregarding any kind of notation while the abstract syntax typically derives itself automatically from the grammar. Hence, it may be concluded that although the abstract syntax is structurally affected by the grammar, a meta model can be considered as clean because it purely represents the structure of the domain. However, in practice, the meta model is affected by editing standard tool considerations.

The BNF definition of a language (rule or programming) facilitates in recognizing the physical text of a program and hence can be considered as the concrete syntax of the language. A rule-based program/engine is taken as a file of characters, by the application or software utilities, since the context-free syntax for the language is satisfied by it, and it also produces a derivation tree exhibiting its structure (parsing in tree form). This software may generally be decomposed into two parts: The first part is a syntactic analyzer or parser which is based on the definition provided by BNF derivation tree from the token list, and the second part can be described as a lexical analyzer or scanner which can read as text and create a list of tokens.

6.3.1.2 Concrete Syntax

Rule languages generally use textual concrete syntax, which implies that a stream of characters expresses the program. Modeling languages have traditionally used graphical notations that have primarily been used in modeling languages. Though textual domain-specific languages (and mostly failed graphical general-purpose languages) have been in use for a long time until recently, and the textual syntax has found a prominent use in domain-specific modeling. The textual and concrete syntax is traditionally used to represent programs, and this character stream is transformed through the use of scanners and parsers into an abstract syntax tree for further processing by programming languages. In the modeling languages, the editors are able to ascertain a major usage, as it directly manipulates the abstract syntax and uses projection to express the concrete syntax in the form of diagrams.

The concrete syntax of DSL is expected to be textual by default. In the scenario where a tool support is available, textual support has long been found to be adequate for large and complex software systems. The programs (code) are shorter in a DSL as compared to a GPL for expressing the same functionality because the available abstractions are generally quite similar to the domain keywords and functionality. An additional language module suitable for a domain can always be defined easily by the domain or technical expert. The DSRL is the extended version of the DSL because it is translated from the high-level model and no need to recompile or rebuilt, it is easier to adapt by a non-technical domain expert.

The graphical model is usually abstract, it hides the functionality and certain cases, graphical editing may seem appropriate. Editing the program in graphical form is not necessary in order to see graphical notations in the structures. To knowing and exploring the detailed overview of complex structures, visualizations signifies an important role as well.

In order to find an acceptance in the target user community, a DSL has to make a wise choice of concrete syntax. This holds even for the business domains. In order to be successful, the DSL has to use notations that specifically fit the domain and may even have to reuse the existing established notations. The expression of common concerns become simple and concise with the use of a good notation and the latter also provides sensible defaults. In case of the less common concerns, a little more verbosity in the notations is acceptable. In particular, the following language descriptions are focused while designing a concrete syntax.

6.3.2 Rule Language Description

We introduce the rule language by analyzing its abstract syntax. Figure 6.1 shows a condensed version of the abstract syntax, implemented to translate a domain model into ECA type rule language for configuration and customization process model as well as handle the domain-specific process constraint. The syntax is expressed in EBNF notation. The descriptions and skeletons of DSRL contain list of processes, the list of events, events in terms of transition of process model, list of rules, list of actions, and postcondition and event list. The events can be internal (raised by rules as an action) or external (raised by process translation handler) or be generated if the described expression becomes true (condition IF is used). In the case of the process model, many types of transitions are available during the transition states including:

- `[TRANSITION_ (SEQUENTIAL (DISCARD|DELAY))],`
- `[TRANSITION_PARALLEL (DISCARD|DELAY)],`
- `[TRANSITION_ CHOICE (DISCARD|DELAY))],`

• `[TRANSITION_LOOP (DISCARD|DELAY)],`

Every process model has at least two states where each transition works between the two states. The flow moves from one state to other state, when multiple transitions

1	`<DSRL Rules>`	`::=`	`<ProcessModelList> <EventsList><RulesList>`	Skeletons
2	`<ProcessModelList>`	`::=`	`<ProcessModel> | <ProcessModel>, <ProcessModelList><ProcessModel >`	Activities and its list
3		`::=`	`PROCESSMODEL<ProcessModelName>`	Process Name
4	`<EventLists>`	`::=`	`<Event> | <Event><EventLists>`	Array of Event List
5	`<Event>`	`::=`	`EVENT<EventName>IF<Expression>| EVENT<EventName> IS INTERNAL OR EXTENAL [TRANSITION_ (SEQUENTIAL (DISCARD|DELAY))], [TRANSITION_PARALLEL (DISCARD|DELAY)], [TRANSITION_ CHOICE (DISCARD|DELAY))], [TRANSITION_LOOP (DISCARD|DELAY)], [INPUTS (<InputList>)][OUTPUTS (<OutputsList>)]`	Transition of BPM
6	`<RulesList>`	`::=`	`<Rule> | <Rule><RuleList>`	Array of Rules
7	`<Rule>`	`::=`	`ON<EventName>IF<Condition> DO <ActionList> CHECK <PostCondition>RAISE<PostEvent>`	Rule syntax
8	`<ActionList>`	`::=`	`<ActionName><ActionName>, <ActionList>`	Array of Actions
9	`<Acknowledgement>`	`::=`	`<Error>|<Message>|<Next Activity>|<Process>| etc`	Array of acknowledgement

Figure 6.1: Abstract Syntax Definition for Domain-specific Rule Language

are taking place in terms of policies and conditions, for validating the starting (initial) and ending (target) states. In the above, we define the transition as a condition, for example, if a transition is SEQUENTIAL and it is invalidated (conditions do not apply or fulfill) during the state change, either DISCARD or DELAY is automatically processed. A DISCARD is used when system functionalities do not work properly in for deadlock and live lock situation. And, DELAY is used for getting information from different source or waiting for input. In the absence of any such condition, the entire process might be left hanging or crash due to transitions. Therefore, we applied the transition with a basic pattern of process model: Sequence; Parallel; Choice; and Loop; in our rule language to avoid and prevent the deadlock situation. We assume that during configuration of the DSR, if the end-user commits any mistake, it will help in preventing a deadlock. Hence, we validate the effectiveness of the configured rule in terms of accuracy and quality in Chapter 9 (Table 9.1 and Figure 9.2).

Further, the DSRs contain more than one process name `<ProcessModelList>` where each process has multiple sub-processes or activities. Every activity performs a specified task. The event name should activate (in terms of **ON/OFF** or **True/False**) a Boolean expression to determine if a certain condition applies or condition fulfills (preceded by term **IF**), and the list of actions that have to be performed when event and condition are true (proceeded by terms **DO**).

The rule language definition contains the *`<Acknowledgement>`* as a process which delivers a message to the initial state. This is defined at the end of the abstract rule (Sequence number 9 in Figure 6.1). It can be handled or managed by the message notification services: like an error, alert, prompt; during the transition of the process. If multiple functions are executing at the same time or another rule is used by a different system or application; and deadlock situation or livelock situation is probable, reset or prompt signals are communicated to the particular system which wants to configure the rule. The rule language definition is designed to manage the critical conditions and contains key parameters for handling any unforeseen circumstances during the transitions.

As an example, we consider processing activities from the case domain - Digital Content Technology: Extraction; and Machine Translation (MT). The list of processes, events, and conditions are expressed as:

List of Process

```
        <ProcessModelList>:: =<gic:Extraction>
```

List of Event

```
    EventList::={gic:File→FileUpload, gic:Text→TextEnd, gic:Text→Parsing ,
gic:Text→MTStart, gic:Text→MTEnd, gic:Text→ QARating, … }
```

List of conditions

```
  <ConditionList>           : =<gic:Extraction.Condition>
                  |<gic:MachineTranslation.Condition>


<gic:Extraction.Condition> ::=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)::== True|
      False>)
                  | IF (<SingleLangugeDetection(gic:Text)::== True|
      False>)
```

Where X is file type, Y is size of file (in MB) and L is length of text.
Source and Target languages are the elements of Language_List.

Language_List(L)={L1,L2,L3,…Ln} ,Source.Language (Ls)∈ Language_List and Target.Language(Lt) ∈ Language_List

For an example, the list of condition is considered `gic:Extraction` activities or sub-process of DCT domain. The `gic:Extraction` focus on data extraction from different data sources. The core condition of data extraction is source validation. Thus, the main condition is associated with file validation in terms of `FileType (X)` from `FileList.` The `FileType` is validated then Other validation conditions is checked such as `FileSize` should less than and equal to `Y;` `TextLenth` should less than equal to `L;` and `SourceLanguage and TargetLanguage` always as sub set of *`Language_List`*.

6.4 Domain-specific Rule Language Definition and Description

The DSRLs subscribe to the definition of rules which handle the domain-specific constraint through the configuration of the parametrical value of the constraint. The domain constraints are configured in the DCT domain. This achieves the required action dynamically and automatically. It must be noted that the definition is valid in case the mentioned events occur in the presence of the specified conditions.

6.4.1 Definition of Domain-specific Rule Language

A DSRL is formalized as an extended ECA language. It is represented in the DSRL as a 6-tuple $<P, E, R, C, A, A_k>$, where P is process model activities, E is an event identifier, R is a set of rules, C is validation of conditions, A is executing action, and A_k is an acknowledgement (error, message, next process, etc.).

Event Condition Action Rule Language

These rules are called Event-Condition-Action (ECA) rules and have three parts: the event, the condition, and the action, specified by the general expression:

ON *event* IF *conditions* DO *actions*

In an ECA rule, the *event part* describes a rule response on a specific event. It indicates when and where the response of the rule occurs, and accordingly the rule is raised. The *condition part* describes a certain condition, which either holds rules to triggers or fires. The *action part* deals with the specific actions which rule performs in case the event is triggered as well as the condition has fulfilled. In this way, the actions are followed by further events, which in turn, trigger more ECA rules, thus making a cycle of event, condition and action of rules until the process ends.

6.4.2 ECA Language Description

The DSRL is now being introduced by which all the subsequent rule languages (expressions, formulas, etc.) are defined. The symbols in this signature of ECA model will be described in next section. Moreover, we proposed a constancy of theory approach and a system that depicts domain-specific aspects of rule programs inherited from the process model-based applications.

6.4.2.1 ECA Model for DSRL

The active ECA language rule has become a major research interest in the present era. ECA rule is found in a wide variety of applications, from enterprise applications to electronic applications. In order to organize the applications, the ECA uses event trigger and event condition mechanisms. For a traditional ECA System, events are captured by event detectors, and the latter then decides which condition is satisfied according to ECA and takes appropriate action. However, such research finds limited attention in applications [161]. Thus, we present a new general model for ECA system. Once the event happens, the event identifier or the event detector first identifies the event. Based on the identified event, it selects a rule from the rule list that fulfill the process activities and further validates the condition list. Based on the condition, appropriate action is selected, and the rule program is executed.

Event, Condition, and Action

1. **Definition (Event)** An event is a meaningful entity that happened at a particular time or space. Event definition indicates that an event can be triggered by an object state and a status change, triggering a particular condition at a particular time while maintaining the same communication between objects and invocation.

2. **Definition (Condition)** The condition is Boolean in nature. When an event occurs, condition validates an object attribute in terms of true/false (0/1) and is thus marked as evaluated. If the value is true (condition satisfied), then the corresponding action is activated; otherwise action is not valid, and the event is lost.

3. **Definition (Action)** Action is an executable program or set of computation decision. Action provides methods or function invocation, creating, modifying, updating, communicating or destroying an object, etc.

6.4.2.2 ECA Rules express in XML

A language is introduced in order to define ECA rules based on XPath and XQuery on XML data [162, 163]. XPath fragment is used within the event and condition parts of ECA rules for selecting and matching XML's nodes in sub-documents. When construction of new XML sub-documents come up for processing, a fragment of XQuery is used. Moreover, we develop techniques for studying the relation of trigger and activation among rules that can be 'plugged into' existing generic frameworks for ECA rule analysis.

A number of issues evolve due to the semi-structured nature of XML in ECA rule context:

- Event semantics: The data manipulation in events semantics is simple for relational data as update, insertion or deletion of events occur when a relation witnesses a particular event. It is more complex to specify in case of XML document for insertion or deletion of data. For this, path expressions become crucial to identify locations within the document.

- Action semantics: Data manipulation actions for relational data are simpler, because, update, insertion or deletion have only a single relationship impact upon tuples. However, in XML, it becomes more complex since the whole sub-document is manipulated by actions and a set of events is triggered by the insertion or deletion of subdocuments.

- Rule analysis: For relational data, determining trigger and activation of relationships between ECA rules is simpler as compared to XML. The actions and events are more implicitly associated to XML and the semantic comparisons between sets of paths and expressions require more sophistication in the latter.

The DSR execution semantics and syntax of the XML are described in detail in [163]. A prototype implementation of the language is described in [162] where the new ECA rules are parsed and checked by the parser component for semantic validity.

Furthermore, the set of valid rules are processed to rule processor for a functional and operational aspect of the application. The rule processing functionality is encapsulated by an execution engine, comprising of an *Event Dispatcher,* a *Condition Evaluator* and an *Action Scheduler* [164]. A *Wrapper* is used to interface with these three components. The wrapper either transmits or receives data from the ECA rules. The updates resulting from rules that have been fired at the head of an Execution Schedule are listed one by one by the Action Scheduler. In case more than one rule is fired, then the resulting updates are prefixed to the schedule in accordance with the priorities attached to the rules.

6.5 Summary

In this chapter, we introduced a DSRL and its artifacts (abstract syntax, grammar and concrete syntax) . The description of rules introduced and how they worked as well as the structure of a rule language and are importance. In the structure of rule language, we define the syntax of the DSRL. Furthermore, we describe the basic components of rule syntax: abstract syntax; and concrete syntax, in terms of syntax and language description. The syntax section describes the abstract and concrete syntax. We presented a DSRL definition and description and defined the conceptual model of DSRL as well as defined DSRL in terms of ECA language, model. We discuss expressing it in XML format as an implementable rule. In summary, we define a rule language whose primary goal is to represent the knowledge translated from a high-level domain model into a low-level rule language. This helps non-technical end-users to configure the domain constraints and to run their customized business or enterprise application promptly based on their business needs and strategies without any technical knowledge. This was also the main objective of this research. In the next chapter, we implement the rule language and its syntax to represent the extracted or translated text model from the high-level domain model with process model activities.

CHAPTER 7

DOMAIN-SPECIFIC RULE GENERATION

7.1 Overview

This chapter focuses on issues relating to generating low-level rules from the high-level domain model by using a model-driven approach. A conceptual model based on model approach for DSRL generation is chosen as the most appropriate approach for the anticipated challenges. The aim of this chapter is to implement a model-driven architecture (MDA) as the platform of relative metamodel, and the translation of *the graphical model into the text model (rule)* at application level. The technical contribution of this chapter is implementation of an automatic systematic domain-specific rule generation which uses variability management (discussed in Chapter 5). Finally, the translation, or serialization of the platform specific implementation models (metameta model), into a set of DSR is discussed.

7.2 Introduction

The rule generation is an automatic approach that accesses customized domain models as per Figures 4.5 and 5.3 (de-activation of feature based on the end-user). The domain models are input for translation process to put output in a specific rule syntax and introducing concrete syntax (discussed in Section 6.3.1.2). This translation process is dependent on and guided by the metamodel with the modeling language with its concepts, semantics and rules, and the input syntax required by the domain framework for a target environment. In this chapter, we implement the process of domain model translation in a target rule environment. The abstract view of the implementation is illustrated in Figure 7.1.

The model-driven approach is used in different settings, e.g. Web engineering uses methodologies such as [165], WebML [166] and WebDSL [167] ; web application on MVC architecture using a JSF framework Ribarić et al. (2007) [168]; UML models and ontologies for business rule Dioufet al. (2007) [52]. In the context of Rule language, Rule Markup Initiative (RuleML) [169], the REWERSE Rule Markup Language (R2ML) [170], and the Semantic Web Rule Language (SWRL) [171] are the important measures in standardization and exchanging of rules. In this approach, the principle of MDA is used for extracting the knowledge of high level domain model into low-level domain-specific rules.

The most important factor whilst generating a rule concerns: the source model is the domain model and its syntax definition discussed in Section 5.6.1; and target model is DSR which syntax definition is already described in Chapter 6. Both of these are artifacts of the model translation [172]. If target artifacts are non-executable but configurable programs (i.e., rules, source code, byte code, or machine code), one uses the term translation [173]. In this research, as the source artifacts are high-level models, we use the one-ways or unidirectional model-to-text translation. “Unidirectional transformations can be executed in one direction only, in which case a target model is computed (or updated) based on a source model” [174]. If the artifacts have different models (graphical to text) in terms of structure, syntax, semantic and, grammar, the term *model translation* is used. The former can range from abstract analysis representations of the system to very concrete models of rule. Subsequently, model translations also include translations from more abstract to more concrete models (e.g., from design to rule) and vice versa (e.g., in a reverse engineering context). Model translations are clearly required in common framework components, such as rule generating, rule interchanging and parsing.

All translations or transformations (including DSR) perform as model translations, the source and target models of translation are related to their own structure[9]. Explicitly, the representation of model can be represented in tree and graph

[9] The child-parent class relationship (generalization association is relationship between base class and super class or parent class) as well as syntactic and semantic containing

(ontology). Every graph contains a class with specific nodes to connect with other nodes which differ from their child nodes (class hierarchy). However, when traveling a graph as a class, it is more complex or difficult to visit each and every node specifically with joint operations. Defining a graph in a relational data structure (schema or grammar) leads to further joint operations as the relations between nodes of class and their linked (child) node require being represented using references of a parent as well. Therefore, class should select from the domain model that matches the

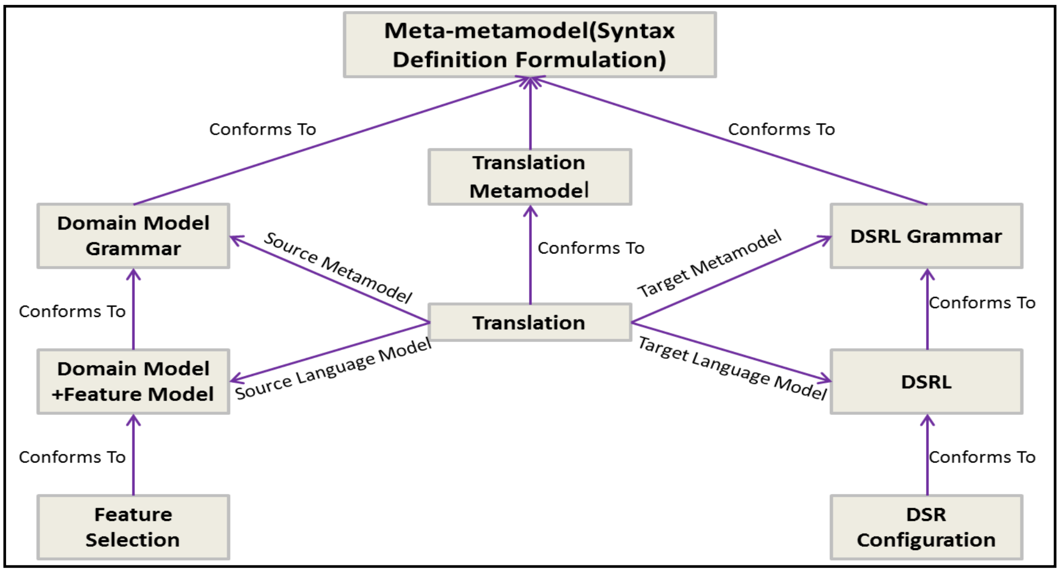


Figure 7.1: Domain Model to DSR Translations

schema of all properties, such as attributes and its association, functions and its parameters, as closely as possible without losing (semantic and syntactical) too many identities during translation of the domain model.

These models express in a modeling language (e.g., domain model as a design model and DSRL as a target model language,) to translate domain models. The modeling language syntax and semantics is expressed by a metamodel. For example, the Domain Model (expressed as language description in Section 5.6.1) syntax metamodel is expressed using class diagrams, whereas its semantic abstractions [172] is described by a mixture of well-formed rules (expressed as OCL constraints [175]), OWL language [176] and natural language [177, 178].

7.3 Model Translation

In the model translation, the translations of source and target models are distinguished on the basis of language. The target model is the *customized domain template*. The translations among the same language are called endogenous, while exogenous refers to translation between models using different languages. In the model translation context, endogenous translation is referred to as rephrasing and exogenous translation is referred to as translation [185].

Some common instances of translations (exogenous transformation) are:

- An abstract (higher level, such as a design model) to concrete (lower level, such as a model of a rule language) synthesis rule generation may serve as an instance, in which translation is made from source model to text (rule, code) or configurable file or executable code.

- The opposite of synthesis is reverse engineering where low-to-high level conversion is made.

- Migration where the abstraction level remains the same, but the language of the program gets altered.

It is proposed that a better approach would be to pass through an intermediate model (MOF at M3 in Figure 7.2). The intermediate models are carried out at every level of model-driven approach like in M0, M1, M2 and M3 Level (Figure 7.3). In the example of a rule generation from a domain model, the proposed approach begins by translating a domain model into DSRL model using a model translation. It is followed by model synthesis into text by model-to-text approach means of a rule generator ontology-based concept model (in Figure 7.2). The advantage of this approach is that the semantic domain translation is achievable by a model translation, which is a dedicated conceptual model, while the rule generators need to be in alignment with the concrete syntax of the target language. The process separates two distinct tasks (translation and synthesis) that are performed using appropriate tools.

Model Type Translations In this type of translation, the source metamodel (stating abstract syntax, types, grammars, etc.) instances are mapped and related to targets (types). The connections among metamodels, models and translation is shown in Figure 7.1. Among the translations, the model to text transformation is crucial and occurs in translations metamodel. It involves DSR generation and thereby higher order translation. The meta-metamodel (MOF) is commonly shared by each of the models. Though it allows multi-directional translations, in practice mostly only unidirectional translations are supported.

7.4 A Conceptual Model for Domain-specific Rule Generation

A metamodel should be used to explain the relationships and important notions while the domain model has to be adapted for a domain. The concrete syntax, abstract syntax and static semantics of the derived notation are included in definitions of the meta model, as depicted in Figure 7.2, which is an extended version [179]. While the domain model or any graphical notations such as box, arrows etc. are contained in concrete syntax, constructs such as classes (nodes), attributes, associations (relations) and between elements [180, 181] are included in abstract syntax. Sometimes multiple diagrams or views are used to represent a model and each of the views again consists of many diagrams. Permissible modeling element type instances (model elements) generally form a diagram and a set of different model elements are used to refine them. The modeling element types are categorized primarily into two groups: concepts, and relations. Relations can be further subdivided into generalization or association. To form a conceptual model which operates according to rules described in a static semantic or abstract syntax, the modeling element types are often employed. All are defined for the purpose of framework design; which can identify features of UML based notations.

Now, we discuss the fundamentals of conceptual domain modeling, DSL context along with application in the intelligent content context. In system development analysis stage, domain conceptual models (DCM) establish as a crucial component, allowing views and relating certain domains. It facilitates in understanding the needs of the problem domain while forming an important the foundation stone of ontology. Many tools, terminologies, techniques and methodologies are employed for DCM as they facilitate in conceptualizing, representing and communicating a problem situation in certain domains. In the present study, DCM is used to obtain domain-specific rule language (*Appendix B*).

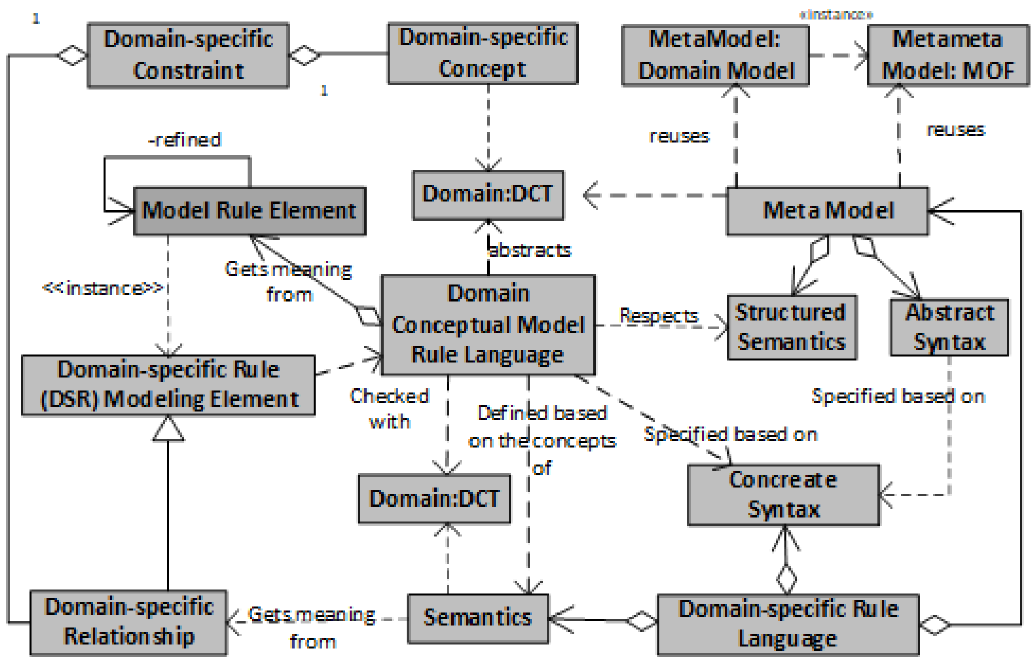


Figure 7.2:Domain Model based DSL Concept Formalization

DCM defines constructs based on UML for a DSL, as depicted in Figure 7.2. In the case of a specific domain, a UML-based language is used for describing important notions and connections (intra or inter-model) with a metamodel. An abstract syntax (defined in Section 6.3.1.1), concrete syntax (defined in Section 6.3.1.2), and static semantics of the DSL form the metamodel. The modeling elements; nodes, classes, aggregation, association and generalization, and relationships between the modeling elements [179] are described in abstract syntax.

7.5 Model-to-Text Translations

A generator is able to create or alter a model instead of aiming to produce only textual output. Thus, we are still left with a model, which requires another generator for production of textual output needed by compilers as well as other tools. Recognizing the challenges of editing generated rule, it is impractical to create model transformations for producing models which are required to be edited by the domain expert at domain engineering level in SPLE platform. In the context of MDA their exist a few concerns about the generated rule language and grammar as covered in Section 6.3.

Hence, when model-to-text translations form one part of a chain of translations, it eventually results in textual output, and the intermediate stages are invisible to the modeller indicating the importance of model-to-text translations. The metamodels are in best position to take a decision on whether to generate from models to text in a single step or in several steps with intermediate transient models, based on the available tools. For example, in case the generation tools do not possess enough power to support the mapping from models to the required text in one step, the intermediate phases become useful. Similarly, it may be useful to translate Domain Model to a certain DSRL model format if it is tested from that model format to the rule translations.

7.5.1 Model Driven Architecture (MDA)

Model Driven engineering is the key focus area of Model Driven Architecture (MDA). The starting point can be assumed as the Object Management Architecture (OMA), that allowed a schema for distributed systems [182]. The major priorities of MDA is found in interoperability, portability and re-usability [183]. Therefore, specification is exclusively required for the system, without taking the supporting platform into consideration. The transformation of the system specifications to a certain platform depends on platform selection. In a common pattern of MDA, translation occurs to platform specific model (PSM) from platform independent model (PIM), potentially with SPLE approach, as shown Figure 7.4. Each model can potentially be transformed to the same type (e.g., PIM to PIM) or to any other rule or model. Moreover, abstract platform serves as the basis of PIM [184], e.g. virtual machines, which also require transformation, along with the PIM, to the platform.

Model Types - In the field of MDA, examples of commonly used models are the computation independent model (CIM), platform independent model (PIM), platform specific model (PSM) and platform model (PM). Among these, CIM, PIM and PSM represent diverse viewpoints and abstraction levels, with respect to analysis, design and implementation views when compared to traditional software engineering.

Computation Independent Models (CIM) - CIM provides a "computation independent viewpoint" to the system according to MDA Guide [185]. This refers to details of system structure instead of computation abstraction. The structure details are called Analysis model, domain model or business model, based on the selected MDA approach. In order to fulfill the needs of the domain, the domain experts and the design experts must work in alignment and this alignment is achieved through the independent computed model [185].

Platform Independent Models (PIM) – PIM implies that no particular platform properties are inherent to these types of models, indicating that this model is extremely general. Examples of PIM targets are technology-neutral virtual machines, a general kind of platform or abstract platforms [186].

Platform Specific Models (PSM) – PSMs are designed keeping a certain platform in view. Since they are derived from PIMs, they contain both platform independent specification as well as platform specific details. Depending on its goal, PSM can give more or less details. When PSM is loaded with the required information, which would allow automatic generation and implementation, it may be regarded as representative of a platform specific model. Hence, it is possible to derive the DSR by serializing this model. In case, the PSM requires additional automatic or manual alterations for deriving the platform specific implementation model, it is representative of the implementation model.

Platform Models (PM) PM is a concept which is not defined with detailed clarity in the MDA guide [185]. It may be broadly described as a combination of concepts containing various parts and forming a platform and its intended services. Therefore, it may be regarded as a model, in a general platform metamodel, though at the same time, it also disseminates concepts regarding parts of a platform, to be utilized in platform specific model, therefore, providing metamodel for that platform specific model. A platform model concept [187] is described as per descriptive logistics. It can be used for automatic selection as well as configuration of some reusable model translations for a concrete platform.

Table 7.1: `gic:Extraction` of MDA Metalevel

METALEVEL	DESCRIPTION	ELEMENTS
M3	MOF, i.e., Meta-metamodels to define metamodels, the set of constructs used to describe metamodels	MOF Class, MOF Attribute, MOF Association, etc.
M2	Metamodels, consists of instances of MOF constructs	Class, Association, Attribute, State, Activity, etc.
M1	Models, consisting of instances of M2 metamodel constructs	Class "Upload," Class "File" or "Multimedia", "Input Text", "Web URL" or "Text Input" etc.
M0	Objects and data, i.e., instances of M1 model constructs	Upload File1.txt or Audio1.mp3, Input text google.com, "this is a book"

7.5.2 Metalevels

MOF architecture is based on four "metalevels". We define these levels as M3, M2, M1, and M0, as shown in Table 7.1.

M3 Level

M3 refers to MOF, whose components are constructs supplied by MOF in order to describe metamodels. The components are Class, Attribute, Association etc. Essentially there exists a single MOF concept. Many regards MOF as the *Meta-metamodel* as MOF is essentially a model describing metamodel. Though using the meta twice may sound unusual, it is correct in the actual essence of the term. To elaborate further, MOF may be considered as a collection of constructs.

M2 Level

M2 Level is populated by metamodels described through MOF constructs. We have already encountered few such metamodels, like the UML or a few not so standardized such as the simple instances that served as our examples. The definition of such constructs involves MOF Class, MOF Attribute, MOF Association etc. Hence, the examples of M3 constructs are regarded as M2 constructs in the true sense. Figure 7.3., describes these metamodel defined constructs that serve as examples of MOF constructs.

M1 Level

This consists of models, which contain examples of M2 constructs. A UML object diagram has been depicted in Figure 7.3 showing a simple data metamodel that can be an instance of showing involvement of instances M1 and M2 model constructs. A data model is depicted in this instance model which describes the Text Input table having two columns: *Event* and *Text Length*. Figure 5.5 highlights, as instance of metamodel M2, while the basically instance of metamodel M2. The links between the *Web URL* class and its child are instances of the M2 association between metalevel and its metamodel. So, *Text Input* (M1) is an instance (M2), which is an instance of MOF Class (M3). Event (M1) is an instance (M2), which is an instance of MOF Class (M3). The link between Extraction and data source (M1) is an instance of the metamodel's association between parent and child (M2), which is an instance of MOF Association (M3).

M0 Level

The link between File Upload and Text File or Multimedia (M1) is an instance of the metamodel's association between MDA metalevel and M2, which is an instance of MOF Association (M3).

7.6 Implementing DSR Generation by MDA

The architecture of the DSR generator follows the MDA four-level model organization presented by Bézivin [188] as illustrated in Figure 7.3. At top level, the M3 is the Syntax Definition Formalism (SDF) meta-metamodel, which is the grammar of the SDF. This level is also known as Computational Independent Model (CIM)

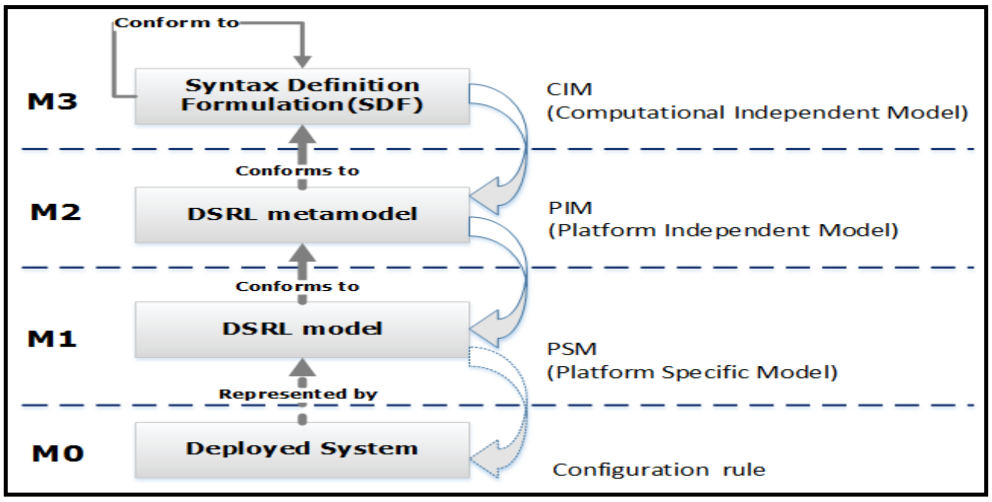

Conform to
M3
Syntax Definition Formulation(SDF)
CIM
(Computational Independent Model)
Conforms to
M2
DSRL metamodel
PIM
(Platform Independent Model)
Conforms to
M1
DSRL model
PSM
(Platform Specific Model)
Represented by
M0
Deployed System
Configuration rule

Figure 7.3: MDA Organization View of the Model Approach and DSR Artifact

where the meta-metamodel is defined (and thus conforms to itself) [159]. A BNF notation can represent the rule in one line. Thus, BNF use as a self-representation notation. This notation allows defining infinite number of well-formed grammars. A given grammar allows defining the infinitely many syntactically correct DSR configuration.

At the M2 level, we define the DSRL metamodel, i.e., the grammar of DSRL with ECA defined in SDF and this level is called Platform Independent Model (PIM). The metamodel conforms to the meta-metamodel at level M3. At the M1 level, we define DSRL models of configuration applications. This is known as Platform Specific Model (PSM), consisting of entity and definitions. The model conforms to the metamodel at level M2. The bottom level is called M0, where we define the configuration of process model customization consisting with DSR in the XML format, which represent the models at the M1 level.

Domain Engineering	M3	Domain Variability	Domain Model (MOF)	Process Model	Weaving Model
	M2	Feature Metamodel	DSRL Metamodel		Relational Metamodel
Application Engineering	M1	Feature Model	DSRL Model		Relational Model
	M0	Feature Selection	DSRL	Configuration	Tuples

Figure 7.4: DSR Generation and Process Model Configuration in SPLE Aspect

7.6.1 Technical Space

A model management of framework which consists of tools, concepts, mechanisms, languages, techniques, and formalisms linked with specific technology is introduced in technical space [99]. The used metamodel (M3 level) actually determines this space. As per Section 6.4.2, we defined the ECA rules in XML, therefore XML acts as in a *technical space*, that involves *XML Schema* as meta-metamodel. Languages like XML, XMI, XSLT, and XQuery are supported by this space. Also, Object Management Group (OMG) endorses the *MDA as a technical space* that employs *MOF* as meta-metamodel, supporting domain model for language. There are a number of technical spaces, a few of which are depended on abstract and concrete syntax; grammars; and semantics of domain models and its translations, rule and code technology, or ontologies.

The source and target models of a model translation might be associated with either same or different technical spaces. If they belong to different spaces, specialized tools are required for defining translations for connecting the technical spaces. A potential solution is to have model knowledge transfer as exporters at the time of execution of the translation in the technical space. The solution could be used by the either the source or the target model.

If we take the translation of domain model into XML rule, for an example, we can either opt for XML or MDA technical space for executing the translation of models and opt for XML presenting the translated or generated rules. If we choose XML technical space, XSLT or XQuery programs can be used for translation of XML rule into XML document obeying syntax of XMI standard (XML metadata interchange) as well as semantic abstractions of MOF 2.0/XMI [189] for UML profile standard. Next XMI parser[190] may be employed for importing the resultant XMI document in a UML profile tool, in MDA technical space.

7.6.2 Solution Space

The purpose of rule generation framework is to use a Asp.NET 5.0[10] web prototype from a domain model that is customized based on end-user requirement. Our framework is specific for our domain model, but it is reused for different application within a specific domain. The customized domain model (after de-activation of feature) provide as input for MDA. The generated artifacts are in the form for low-level text. These artifacts are an output as a domain-specific rules with the syntactically concrete and semantically abstract syntax. The customization domain template is an input, generated rules is an output and configuration are an operation and functional requirement of domain in terms of constraints. A MOF metamodel will be needed for translation into MDA technical space. When rules have been parsed into instances of metamodel, MOF (as per Figure 7.2) translation is considered as completed. We propose to provide standardization of rule generation and implement this type of model translation.

To implement the configuration of the domain constraint at the running process model, the execution component is divided into two steps: 1) DSR is generated

[10] https://msdn.microsoft.com/en-us/magazine/dn879354.aspx

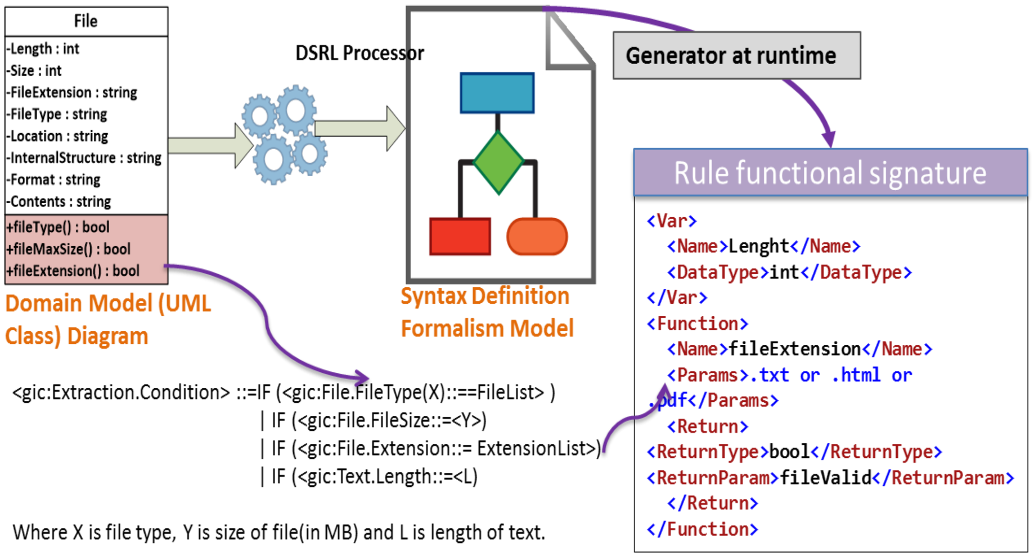
through customized domain model (activated or deactivated classes) which is adapted by the parametric domain constraint value 2) DSR is configured for adapting the customized process model operation in terms of functionality. These steps are described below in detail.

Step 1: DSR Generation

The DSR is generated from a customized domain model by using the MDA. This is the principle step as the entire configuration and customization of the process model is executed by generating the DSR. We have implemented a DSR generation with customization of domain template with dynamic selection of the required feature.

Step 2: DSR Configuration

The adaptation of customized process is configured by the end-user selecting information at the time of input. when the end-user. In this step, an approach to reflect the changes is proposed, which is made at the input of DSR generation by (de)-activating the domain template. Specifically, DSR generation carries out the three actions to adapt the customized process model shown in Figure 7.5.



```
<gic:Extraction.Condition> ::=IF (<gic:File.FileType(X)::==FileList> )
                             | IF (<gic:File.FileSize::=<Y>)
                             | IF (<gic:File.Extension::= ExtensionList>)
                             | IF (<gic:Text.Length::=<L)

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

```
<Var>
  <Name>Lenght</Name>
  <DataType>int</DataType>
</Var>
<Function>
  <Name>fileExtension</Name>
  <Params>.txt or .html or
.pdf</Params>
  <Return>
<ReturnType>bool</ReturnType>
<ReturnParam>fileValid</ReturnParam>
  </Return>
</Function>
```

Figure 7.5: DSR Generation and Signature

The principle argument is that customization of process models can be carried out at run-time, which is made possible, by SPLE. An SPLE facilitates mass customization and satisfies different stakeholder requirements [39] (discussed in Chapter 4). The SPLE can be implemented in two steps: domain engineering and application engineering. The domain engineering is responsible for reusable platform and defining the variability and the commonality of the product line.

The application engineering provides a platform for managing the product line applications which are utilized or accessed in domain engineering. Therefore, we consider the SPLE as an enabler for mass customization. Rule generation is challenges when it is translated from high-level domain model to low-level rule. The MDA [156, 191, 192] concept can be used to generate rules (DSRL), providing definition and composition (discussed in Chapter 6) of domain challenges.

MDA approach can give us a multiple conceptual platform, which helps the end-user in developing application models, business logic concept and generating rule for target platform through translations. With the help of MDA, one can actually put the emphasis on creating models specifically for application domain, even while maintaining independence from platform, rather than being obligated to use high-level language for writing platform-specific rule [193]. Therefore, MDA assists in increasing the abstraction level in software development.

7.7 Summary

The overview of the implementation of the framework for generating a set of DSRs was described. The DSRL generator principle with model to text transformation was defined, the model translation and Metalevels are described. The conceptualization of Domain-specific rule generation was formulated. The model to text transformation was presented. Finally, the implementation of DSRL generation was defined and discussed. This section divided into two subsections: (i) Technical Space; and (ii) Solution Space. The technical space defines the XML and its components.

CHAPTER 8

PROCESS MODEL CUSTOMIZATION AT RUN-TIME

8.1 Overview

In the last chapter (Chapter 7), the implementation of DSR generation from the domain template was discussed, along with MDA methods and its components, to translate the models into rule language. This chapter presents our approach to achieve the process model customization at run-time. And, to illustrate this approach using four common steps to manage a MAPE-K loop [194] : (i) *Monitor*, (ii) *Analyze,* (iii) *Plan,* and (iv) *Execute, Knowledge*. In our case the *knowledge* is both the domain template, and the feature model that manages the loop as a flow at run-time. The loop components (MAPE-K) are given in Figure 8.1. The technical contribution of this chapter is to customize the process model, based on end-user requirement, and to adapt the new process model. As discussed in Chapter 5. Therefore, it is essential that selected

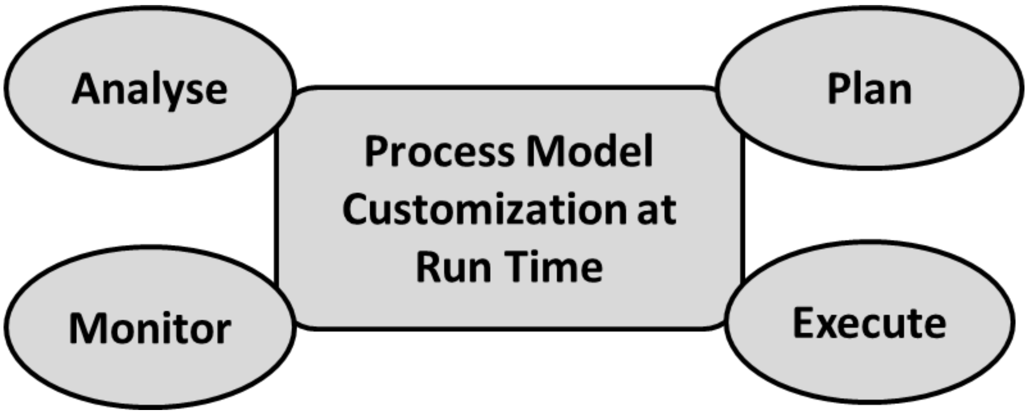


Figure 8.1: Scope of Chapter 8

features and configurable values are verified and validated both prior and post generation of the set of rules. In absence of such verification and validation, the output of customization and configuration may be incorrect and error prone, leading to undesirable results. Therefore, this chapter discusses validation techniques which aim to protect against unwanted feature selection.

Accordingly, the feature model has complementary artifacts that use the variability points to capture requirements as input. This requires mapping between the feature model and the domain template where the weaving model is used as a mapping bridge between both models (discussed in Chapter 5 during the design phase). Here, in this chapter, the feature model serves as an input of the framework, that is responsible for activation or de-activation of the domain template components to customize the whole domain template.

Section 8.2 discusses dynamic adaptation and the implementation of an overall approach; Section 8.3 clarifies how the target feature is monitored and Section 8.4 explains the analysis of the collected user input information in order to request activation or de-activation, if target features are valid. Section 8.5 explains the plan and how the domain template is customized. Section 8.6 explains the execution of the adaptation on the generated DSR configuration for customization process model. Finally, Section 8.7 presents the summary of the whole chapter. The next Chapter 9 discusses the evaluation to validate the research claim.

8.2 Dynamic Adaptation Process

This section discusses the implementation of the customization process model based on the end-user requirements. Chapter 6 shows an ECA based DSRL has already been provided which allows the end-user to configure the domain constraints. These configured domain constraints provide functional and operational aspects of business applications. This framework allows the end-user to select the features, from the feature model, based on the requirement of process models. The end-user customizes the process model application and configures domain constraints, at the same platform, without requiring technical knowledge.

Whenever changes need to be applied in an existing business application, a new process model needs to be developed. Following the change requests, domain experts are always dependent on developers and process engineers. The developer each and every time encounters the huge challenge of developing, managing, and maintaining the regular change requests from the end-user; consistently changing, re-compiling and redeploying the code on the server.

Introducing these changes is error-prone for the developer. This proposed framework is an automated solution for a DSR generation to configure domain constraints in generated rules and customizes the process model in a single platform. The validation of feature selection and its mappings describes the realization of target models by the activities in the process model application.

An ad-hoc approach to building customize process model applications is to use existing variability mechanisms[11] (e.g., if-statements or method dispatch) directly in the architecture. Therefore, in the dynamic adaptation of our framework, we use the captured requirement of end-users, in the form of features to help the dynamic adaptation of process model. In response to changes in the context; the system, based on selection, can activate, or deactivate, these models to determine the necessary adjustment modifications in the domain template.

This approach achieves the dynamic adaptation of process models through the run-time generation of DSR and configuration with the help of feature selection (see Figure 8.2). The infrastructure of this framework is based on the components of the SPLE and MDA, for achieving the mass customization of the domain template and generation of rules from the domain model respectively. In this approach, the DSR manages the configuration (new system) or reconfiguration (old system) based on models at run-time.

The Figure 8.2 follows the MAPE-K loop to achieve dynamic adaptation of the process model. First, the collection of selected feature information is gathered from the feature variability models by the end-user, which is called Target Feature Selection. The second component is the *Monitor* component that monitors the feature

[11] Single and multi-model approach discussed in Chapter 1

selection information. The *Analyze* component analyzes and validates selected features from variability models. The feature and its relationship (in Table 5.1) like

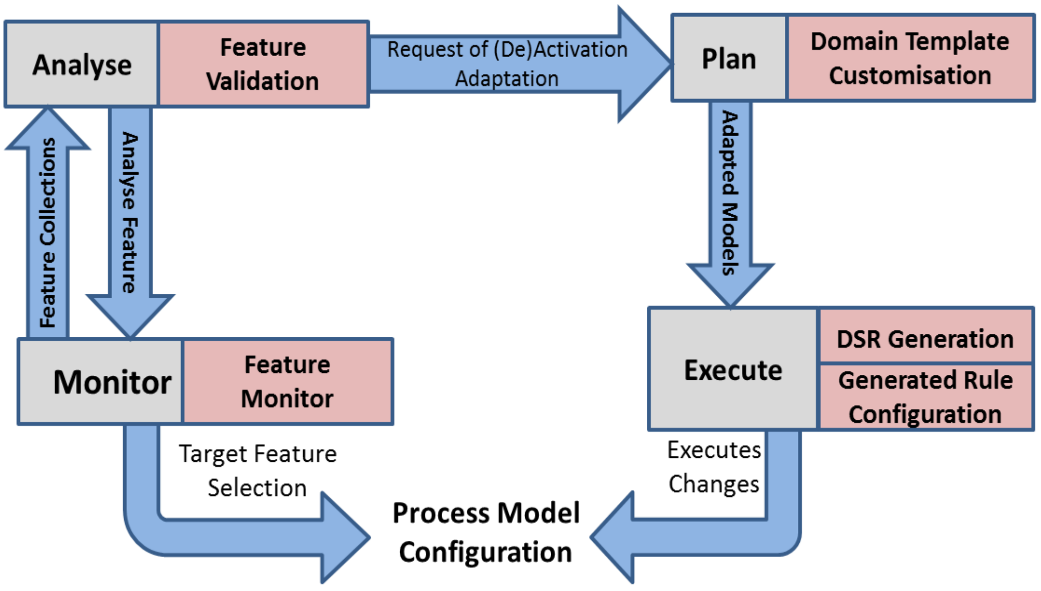


Figure 8.2: Dynamic Adaptation of Process Model Customization

mandatory, optional, OR and alternative types are observed, and the information is analyzed by Feature Validation. Once the features are validated, the request of activation or deactivation has been processed, then customization is processed. The domain template activities or de-activities the models and its component based on feature selections.

The variabilities of the domain template are activated or deactivated at run-time and then the model changes. The SPLE architecture supports the dynamic customization of models for the generation of DSR and adaptation of process model. The adapted domain template is used to automatically generate a domain-specific rule plan with adaptation actions to carry out the process model customization simultaneously. The customized process models are processed on server to Execute component.

In the Execute component, the generated DSR is mapped with the adapted process models and configured domain constraints. The configuration DSR and other required artifacts are deployed or published in the business process/enterprise Execution Server or Engine at run-time.

8.3 Feature Monitoring

The monitoring component in this approach involves capturing basic requirements of the end-user in terms of the features. The main component in the monitoring is the feature monitor that observes and gathers the information of target model in terms of operational requirements of the end-user at the time of feature selection. The Feature Monitor works as an application that is connected with the domain template at run-time as an input, or as behavioral monitoring. Therefore, the Feature Selection can be used or implemented by different mechanisms according to specific needs or requirements. The feature monitor observes every input from the end-user like selection or optional input performance. Exceptions can arise when an operation fails to meet its time constraints in terms of waiting time exceeding a specific value (session time-out or connection pooling or load pooling in web applications). We have implemented a Feature Monitor that observes the feature-by-feature selection triggering, an automatic request for the validation and verification response. Our motivations to implement, Feature Monitor is as follows: (i). To rely on complete flexibility in the feature selection information being observed and the frequency of its observations; (ii). To gather the information on end-user selection type monitoring of the feature for understanding the automatic feature selection analysis.

The Feature Monitor follows request/response message pair between nodes [195], in our case, between the Feature Monitor and the Feature Validation. The request/response approach can be used to determine selected feature availability in actual feature (domain template) and to find the feature miss matching in the communication. The Feature Monitor counts the feature selection to monitor the context and to get measures for the basic feature model types (Mandatory, Optional, Alternative, etc.) the validation and parametric value verification[12] for input information quality attributes.

This prototype has a feature selection interface where the end-user can select features, and based on the features, domain-specific constraint parameters can be

[12] end-user configures constraint values and validate the type data type like, string, integer, and date time as input value for configuration

selected, based on the rule generation, and then the configuration of domain constraints values may be given. Several features can be chosen at the same time, and parametric values for new or updated old process model corresponding to its features can be added simultaneously.

8.4 Analyzing the Feature Validation

The list of feature collection is processed with parametric values where the input by the end-user has to be made a model-based composition. This schema flow task is validated by the feature requirement and verifies the parametric values of every feature. This is done in order to count the parametric value of features with respect to every feature collection, which has been monitored by the Feature Monitor. Next selecting the feature in the feature model, the analyzer evaluates the information values to find out if any validation and verification (V&V) condition has been violated.

Here, we have implemented a model view approach for V&V operation for understanding the feature selection, and its corresponding constraint values, whether the condition is fulfilled or not. The model is based on the data type corresponding to domain variabilities in the specific domain. For example, the end-user selecting the web crawling as a translation memory in the particular condition; in that case, the regular expression or keyword matching, the web link analysis. Based on features, a data type of parametric value, which is matched against the analyzer model and the result obtained from the matching, is processed to validate the information. The feature can be used as a simple query or string matching or regular expression model. The same validation is also used for a configured rule, based on this regular expression, the prototype can then be identified by the number of error preventions during rule configuration by the end-user.

8.5 Planning the Model Customization

When model customization has been requested (i.e., after a feature validation and parametric value verification condition has been fulfilled), DSR generation carries out the following steps (8.5.1 - 8.5.2) to plan the adaptation of the process model customization and domain constraint configuration.

The target model, in domain engineering lifecycle, capture the variability [196, 197] and motivations of present features in SPL. Therefore, with the target model, it ensures that the present features and the relationship of variability in feature models has to be in line with the variability of the target model. The product differences can help us trace back the differences in end-users or stakeholders' motivations.

8.5.1 Model Process Customization Plan

In this step, the end-user bases the template and feature models and their variability used for customizing the domain template on target feature selection. Following the feature selection, the activations or deactivations takes place in the domain template. In the following case, we specify the target feature to be released from a domain template by the feature of a feature model in terms of mappings. Based on these mappings, we protect any type of inconsistencies that might occur due to contradicting relationships of the target model and their mapped features. This flow schema is the part of domain engineering lifecycle and the mapping relationship designed by the domain expert.

The target model is used in a domain template to resolve product line variability in the intentional space. Hence, the contribution links might meet the expected amount of satisfaction of domain template to target models in terms of rule generation and customization. The process model configuration with domain constraints features selected, and their constraints value could help in achieving the adaptation of process model based on the selection of appropriate product line variant-based intentional variability. For instance, the challenge of the file upload is the size of the file as it cannot be modified by end-user from *X* MB to *Y* MB. Goals and tasks facilitate in choosing a proper variant of product lines based on intentional variability. In order to reduce cost delivery, the target model may serve as a criterion for resolving product line variability in the item delivered goal. Target models may be described as below:

Customize Model *A domain template DT=* D, *P, F, f extends a target model DT=* D_f, P_f *as follows: The decomposition relation D is extended by decompositions that cover product line variability* $D_f \subseteq D \times P_f \subseteq P \times$ *{OR, XOR, AND, AND-O, OR-VP, XOR-VP}.* Here DT is Domain Template, D is Domain Model, P is Process Model F is Feature Model and f is selected feature.

The explicit mapping (those coming from domain expert) may be represented in this model by developing a mapping relation for each mapped task. To illustrate this, consider, the `gic: Extraction` where File Upload task is mapped to the Document File and Multimedia features, therefore, a mapping relation *(Extraction, (File Upload, {Document File, Multimedia}), (Input Text, {Web URL}, {Text Input}))* comes into existence. Once there is explicit mapping among features in a feature model and tasks in domain template, the intermediate tasks, and target features may be implicitly mapped by present relations in templates and feature models. For instance, it can be inferred that the target `gic:Extraction` managed in the templates model is implicitly mapped to the feature `gic:Extraction` management (see Figure 5.4).

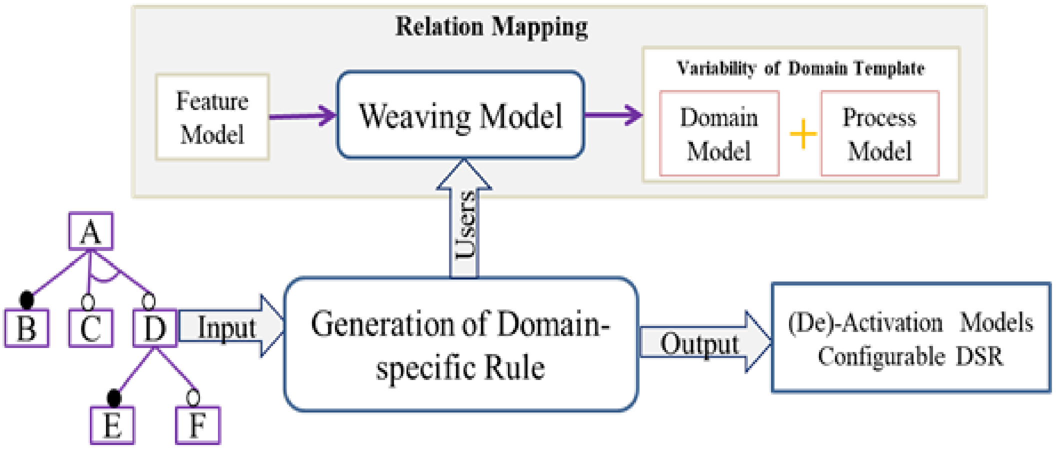

Relation Mapping
Feature Model
Weaving Model
Variability of Domain Template
Domain Model
Process Model
Users
A
B
C
D
E
F
Input
Generation of Domain-specific Rule
Output
(De)-Activation Models Configurable DSR

Figure 8.3: Customization of Domain Template Plan

8.5.2 Adapt the Domain template

In this step, DSR generation plan and configuration contains a set of action for (De)-activating components and adapting the domain template according to the target model (end-user feature selection). The customization actions are stated as domain template activation (DT_a) and domain template deactivation (DT_d). These operations take variability models as input, and they calculate the modification to the domain template by activating (adding) DT_a or deactivating (removing) DT_d variant in domain template (Figure 8.4). The customized domain template eventually causes the activation or deactivation of the domain template that orchestrates the generation of the DSR and the process model's operations adaptation.

In order to perform activation and deactivation actions, the domain template associated with the weaving model mapping between features is responsible for activating the new customization of the domain model and related process model. In this way, the adaptive domain template only active when it is related to the selected feature in the domain template customization. The domain template is customized through the activation or deactivation of features. The currently active features, which have not been deactivated in the customized domain template of the variability model, remain still active.

Example: Customization Plan of gic:Extraction

When activation and deactivation are applied to the domain template during the initial stage of the case study (Figure 8.4), the resulting customization plan forthcoming as:

DT= {File Upload {Document File, Multimedia}, Input Text {Web URL, Text Input}}.

DT_a= {Input Text {Web URL, Text Input}}.

DT_d= {File Upload {Document File, Multimedia}}.

These actions express how to reorganize the variable elements in the domain template to move from one state (domain template) to other state (required models) when these feature elements are activated or deactivated in Extraction sub-process of DCT.

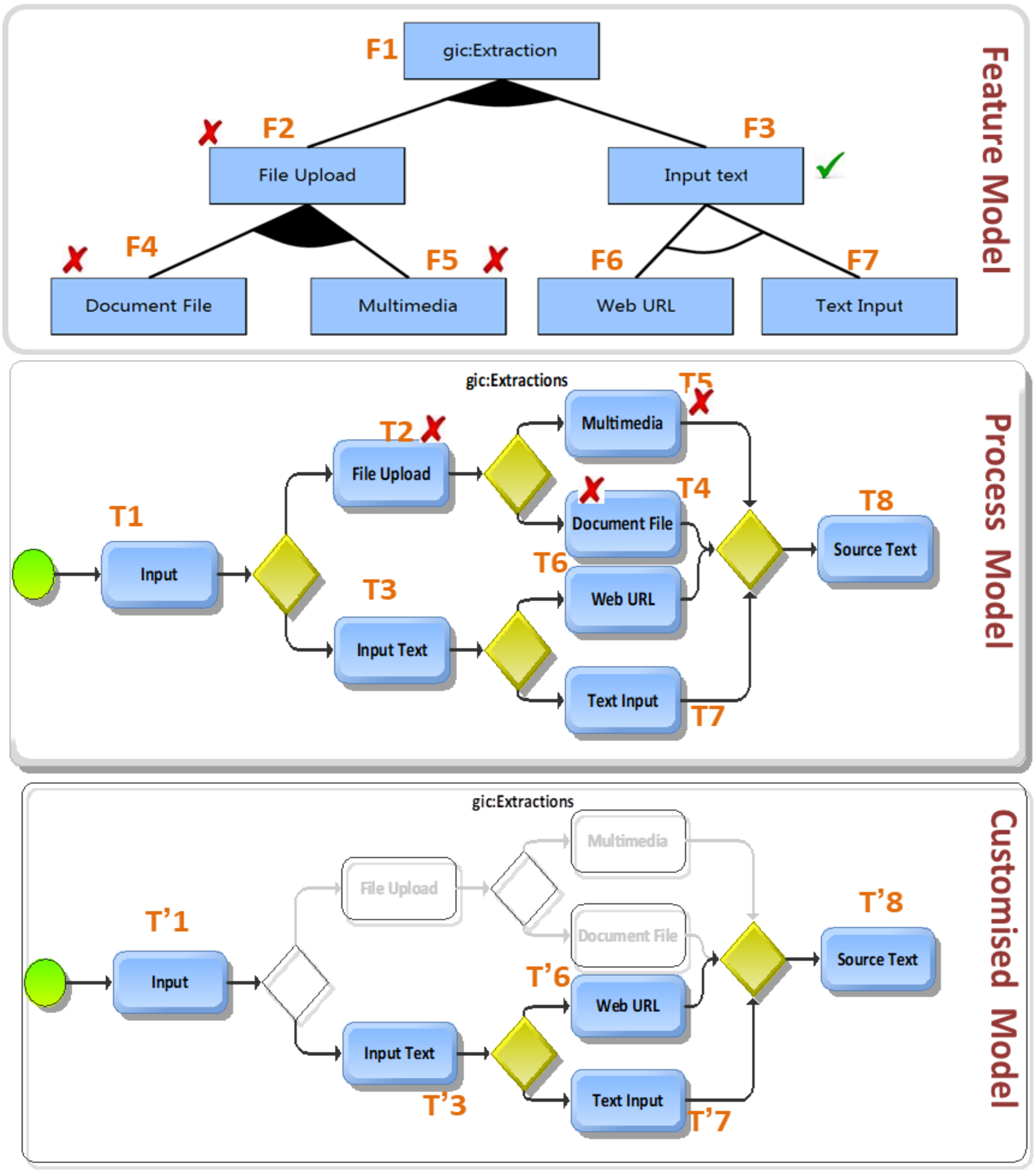


Figure 8.4: Overview of Process Model Customization

8.6 Executing the Customization Adaptation

This section discusses the implementation of customization process model through feature selection in feature model at run-time. The selected features are reflected as an activation and de-activation of a feature component in the domain template.

8.6.1 Adapt the Process Model Customization

In this step, the target model applied the customization plan (with DT_a and DT_d actions) on the domain template (as per Figure 8.5). Specifically, it carries out the following actions:

1. Fresh target model is planned to be created at domain engineering and design time. The default domain template is loaded, and the features are selected in application engineering at run-time. There is no need, any extra effort to customize the running instance version due to the flexibility of the DSR configuration (XML format). The loaded models are activated or deactivated in the feature according to needs of the end-user.

2. Domain template deactivates all the possible modeling elements except mandatory features at the variation point while the running both models, and is adjusted by DT_d actions.

3. Domain template activates all the modeling elements in the variable model at the variation point of the running both models according to DT_a actions.

4. The new version of target model is deployed automatically and is correspondingly configured rule in the running model.

A summary of example

a) Initial feature model arrangement.

b) Feature selection in terms of feature activate (green with check sign) and deactivate (red with cross sign) form the feature model

c) After customized process model based on feature selection by the end-user has been executed.

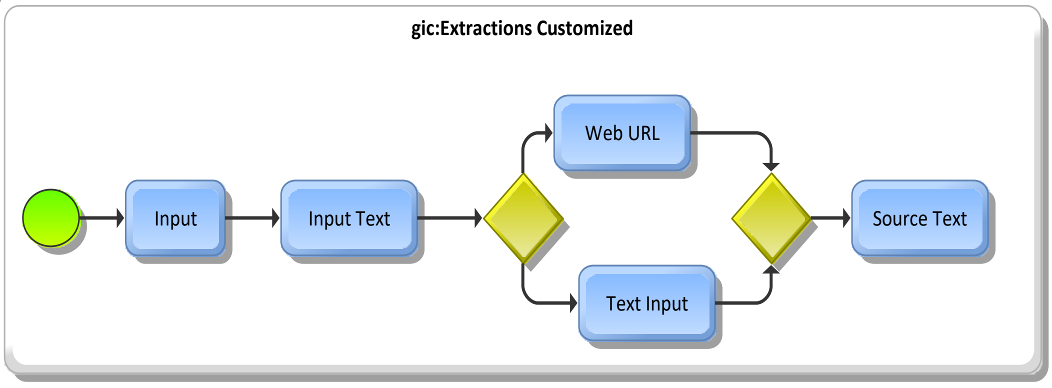


Figure 8.5: Adaptation of Customized Process Model

The section shows the initial feature model (variability model) of DCT domain, we select a sub process as a `gic:Extraction` for our case study. Here, the feature model contains the possible feature of machine translation sub-process. In Section 5.4, the activation and deactivation of feature in domain template are based on the variability model, which is selected by the end-user. Finally, in this section, process model activates and deactivates the feature plan and execute the DSR generation and configure the process model.

8.7 Summary

This chapter presented an approach to achieve the dynamic adaptation of customization process model at run-time. An approach to the dynamic adaptation of the process customization was introduced at run-time and discussed the need for process model customization. Then, the collection of information was monitored in terms of feature selection and constraint value as a parameterized input. Afterwards, the analysis of collected information and verification of the condition that was fulfilled were described. Here we are discussed the validation of the feature model. The customization was described in terms of activation and deactivation plan and adaptation of domain template, i.e., specifically, the DSR generation, a configuration plan and customization of the process model. Finally, the action that is carried out by DSR generated and customized process models, to execute the adaptation were described.

CHAPTER 9

EVALUATION AND VALIDATION OF THE ARTIFACT

9.1 Overview

In the previous chapters, we presented a DSR generation and configuration solution for process model customization. These rules are the final solution elements for configuring the domain constraints in the domain-specific environment. The final solution helps in adapting changes in various applications, such as BPM. In this chapter, we describe the process of validation of the research claims, which were presented in Chapter 1. In particular, this includes a certain set of generated domain-specific rules, to meet the user requirements. The objective of this chapter is to:

- Evaluate the efficiency and effectiveness of the rule configuration process and establish to what degree the semi-automatic configuration is better than the manual configuration.
- Demonstrate the user satisfaction and usability of the framework in a real-world scenario.

This chapter presents the results of the experiments, and their analysis. This validates the research claims and answers the research question (RQ3). It also partially answers RQ2 with regards to the usability of the developed prototype by non-technical domain experts (as addressed in Section 1.3). Among the different types of investigation, we use a combination of evaluation techniques, which are defined in the Chapter 3. In the next Chapter 10, we will focus on the conclusions.

9.2 Introduction

Nowadays, business applications struggle with rapid changes in the business itself due to the dynamic and competitive environment [198]. These changes are continuous, and process models have to be continually customized to meet the new requirements of end-users, in that particular domain (see Chapter 1). Incorporation of all changes in business applications are time consuming, because of the complexity of the business application (e.g. heterogeneous data sources, hundreds of software libraries, etc.) and rigidity of the process model (hard-coded components). Therefore, the application passes through several stages such as development, deployment on the test environment: testing, test reporting, redeveloping and subsequently testing when needed. In many cases, the required changes are incorporated rapidly and implemented without following any processes which increase the time to deliver the product/service to the market.

In the standard set-up, the changes should be first understood by a domain expert, who should then be able to explain it, to a program developer. Afterwards, the developer should develop the application according to his interpretation. This development process takes time and additional human resources, with beginning to end and end-to-end loop reparative process. The more iterations, the more prone to errors the code becomes, to build/compile the code and to install it on the application server. With every code change, we need to go through the same process and sometimes reboot the server.

In contrast, the proposed solution allows a non-technical expert to introduce the changes without knowing about the technical details. We discussed the solution and its components in Chapter 4 and explained each separate component in Chapters 5, 6, 7, and 8. The design of a domain-specific system, or an application, that aims to resolve domain related issues, influences the functional and operational quality (FOQ) of the final framework solution. We discuss several challenges that impact the FOQ: the first part of this work concerns the evaluation of the proposed generated DSR configuration in terms of efficiency and effectiveness. The second part of the evaluation focuses on the SUS satisfaction as well as the efficiency and effectiveness of the framework as judged by the end-user experience. For user experience evaluation, we use usability as defined according to ISO 9241-11 [199]. Broadly, usability is defined as “the extent to which a product can be used by specified users to achieve specific goals with effectiveness, efficiency, and satisfaction in a specified context of use". Usability has been considered as an obvious requirement for all genres of technology [200] and is evaluated in terms of effectiveness, efficiency and satisfaction with which specified users achieve the stated goals in particular environments. These usability objectives have been considered independent of any specific domain of human activity, so they have been taken into consideration for design and evaluation of various software systems, including digital technology [201]. We present our finding on usability (Evaluation Criteria) in Section 9.4.1.

The main concerns for evaluating the configuration of the rule generation framework are its usability, efficiency, and effectiveness. These help us analyze how useful the proposed solution is, and how effectively it solves the problems faced by the end-users in the *real-world settings*. The objective here is to make the solution appropriate, and suitable for answering the real-world problems, faced by the end-users. *The efficiency* of using the semi-automatic (proposed) versus manual (traditional or baseline) configuration, is measured by the difference in time, required by both approaches for execution. *The effectiveness* of rule configuration is measured in terms of error prevention and correction in the proposed solution. *The satisfaction* is completely subjective, and the SUS helps to clarify the answer to it. Our experimental results show and prove that our approach can be validated with statistical significance.

9.3 Evaluation Process and Planning

The objective is to evaluate the generation of rule based on verification of the feature model and its configurable parameters. In this regard, first, we need to confirm that the framework is valid and represents real-world changes. Second, we evaluate how useful the framework is for the rule configuration by end-user and how easily it could be understood, learned and adapted by the users.

The steps for evaluation process are as follows:

1. Define the evaluation strategy and criteria

2. Evaluate the rule configuration change process using empirical case studies in particular domain

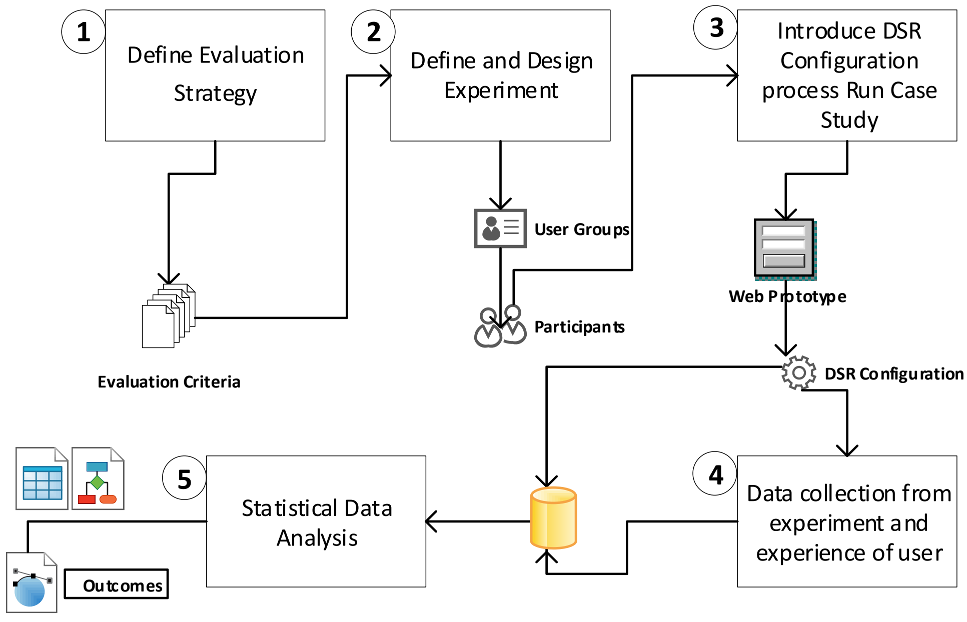

1
Define Evaluation Strategy
2
Define and Design Experiment
3
Introduce DSR Configuration process Run Case Study
User Groups
Participants
Web Prototype
Evaluation Criteria
DSR Configuration
5
Statistical Data Analysis
4
Data collection from experiment and experience of user
Outcomes

Figure 9.1: Evaluation Process and Planning

3. Conduct the user experience evaluation

4. Collect data from different modes[13] of tasks (experiments) and assigned tasks as a participant activity such as configuration time

We now describe these steps in more detail:

The first step (Figure 9.1) of the evaluation process is to accurately define criteria used to determine suitability of an artifact. The evaluation criteria are derived from the primary question of this research, i.e., *how to configure the generated rule with minimal technical knowledge* and *research output – a configurable domain-specific rule that customizes the process model*. The evaluation strategy is described in Section 9.4.

In the second step, we evaluate the rule configuration change process using empirical case studies in particular domain. Below, we discuss the experimental setup

[13] Mode of tasks like semiautomatic, manual and SUS

(in terms of domain selection and rule configuration development), empirical results and analysis of the empirical results. For empirical findings and statistical analysis of the configuration of the rule generated in the framework using empirical case studies with two different groups in Digital Content domains, we conduct a web-based user experience usability evaluation between manual and semi-automatic configuration. We designed tasks and divided them into two different categories. The first category of the task is to configure the generated rule which is divided into two modes, i.e., the manual and the semi-automatic. The manual model is defined as a simple text editor where the rule can be can be configured by the participant. Semi-automatic model is defined as a text box corresponding to each constraint parameter which are needed to be configured in the rule. Each participant had two manual and two semi-automatic tasks which were assigned automatically at the time of registration. The second category of the task is SUS. It is a completely subjective task based on five positive and negative sentiment questionnaires. Each sentiment has a different type of calculation (illustrated in Table D1 in *Appendix D*). Each user has independently to finish 5 different tasks which are pre-assigned on their dashboard in the web interface, after login. The user needs experience as an experimental setup is required in data collection for evaluating the performance of rule configuration in experiments under a controlled environment.

The third step of the evaluation process of this research study is to conduct the user experience evaluation. Normally, the prototype evaluation takes 20-30 min during registration; we ask participants about domain knowledge[14], skills, and technical knowledge. The objective of organizing the evaluation is compared to the manual and semi-automatic configurations. Additionally, it also allows to determine which system is better in terms of usability.

At the end of the evaluation, we collected the data of all the tasks and participant activity such as configuration time, what was configured, how much time was taken for configuring the tasks. A number of errors while the tasks were being configured,

[14] Domain knowledge about Digital Content Technology – Domain knowledge is part of the design. It means, the selected participants have certain domain knowledge (Specifically, how to extract the data from different source).

during feedback pertain to the concerned parameters mentioned in the last stage of the evaluation process. This phase may be interpreted as a collection of participant's data. Further, a combination of qualitative and quantitative approaches (with a focus on the qualitative one) was chosen to evaluate the present study. SUS was adopted in this step to collect the quantitative data as it provides a mechanism for measuring the usability satisfaction of the end-user [202].

Table 9.1: Summary of Challenges, Proposed Solutions, and Evaluation Methods

Challenge	Evaluation Criteria	Evaluation sub-criteria		Solution Artifacts	Evaluation method	
					Experiments	*System Usability Scale*
C1	Efficiency	Performance	Processing time	DSR Configuration	√	√ √
C2	Effectiveness	Accuracy	Error detection		√	√ √
		Quality	Error prevention		√	√
C3	Satisfaction	Effectiveness		Overall Framewo		√
		Efficiency				√
		Learnability				√

Analysis of raw data and transforming it into the practical and meaningful outcome was the last phase in this evaluation process. The analysis aimed at retrieving some relevant data which facilitate in gauging the issue or specific circumstances by examining the situational perspectives and behavior of individuals within the context [203].

9.4 Evaluation Strategy

We use both quantitative and qualitative research methods and analysis throughout this evaluation. The evaluation strategy of the present study broadly encompasses the following methods, selected and based on the design science research as discussed in Section 3.3.4:

- *Case study*
- *Controlled user study experiments*
- *End-user opinions and feedback analysis (SUS)*

9.4.1 Evaluation Criteria

The solution should be effective, efficient and satisfactory according to the end-user when they configure the domain constraints with specific conditions. We evaluate our contribution in usability of the following aspects which are illustrated in Table 9.1 and Figure 9.2. One component of the ISO standard 9241, highlighting to the usability specification, applies equally to both hardware and software design. In Chapter 4 of this thesis, the following definition of usability was provided:

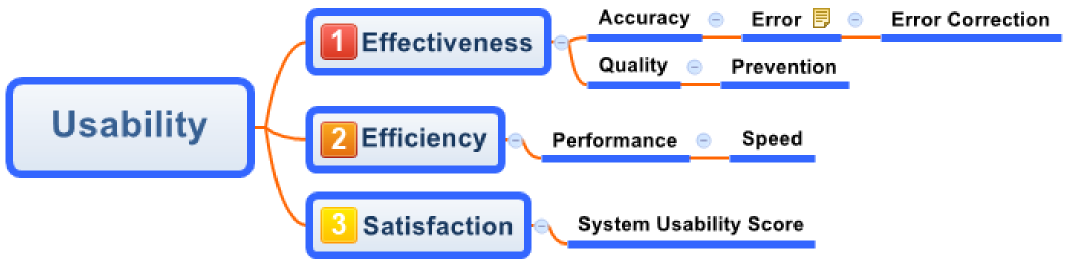


Figure 9.2: Usability Criteria

- Effectiveness: Each activity contains certain parameters (see Tasks and their Distributions for details in Section 9.6.4). An evaluation of the effectiveness of the technique is done in terms of accuracy and quality in the obtained results when using the generated rules as stated in Chapter 6 and 7. The manual and semi-automatic configuring of the rule are while studied, the accuracy is measured based on monitoring the errors, in terms of protection (text field validation), message quality, and error correction (discussed in Section 9.8).

    - Accuracy of configuration: the capability of the solution to produce error-free rule configuration and deployment on the server. An error-free configuration helps to run the process model application smoothly, i.e., without interruption while producing accurate output. We evaluate the accuracy by analyzing the system's capability for error prevention, error correction and error message.

    - Quality of configuration: the process of validation and verification of the information value before finalizing the configuration. Quality of configuration refers to the system's capability to prevent functional, operational and data errors (such as type-, semantics-, syntactic-mismatches). In our experiments, we consider the data type of the input value as a quality parameter where we calculated how many errors were prevented through dynamic validation, at the time of semi-automatic configuration.

- Efficiency: refers to ensuring that the attributes of the generated rule require minimum configuration and processing time. The processing time is estimated based on an evaluation of configuration of the constraints, and feature parameters. Afterwards, using randomized tasks for generating rules and parametric values of different sizes, we determine the time needed to configure the rules. The configuration time is judged by the time for assigning values to the parameters by the individual participants (discuss in Section 9.7).

    - Performance - The performance is measured based on configuration time, that includes run-time semi-automatic and manual configuration of the rule, domain constraints and their validations. In other words, it refers to the capability of the solution to be able to provide the required performance (in terms of time), relative to the number of resources used under stated conditions.

        - By comparing the time for rule configuration between manual and semi-automatic mode, we measure the time improvement of the semi-automatic configuration over the traditional or manual one.

- Satisfaction: the support provided by the tool that allows end-users to select features and generate rules. This includes the implementation of the rule configuration. We use the SUS from end-user intervention to evaluate the satisfaction of the end-users (discussed in Section 9.9).

9.5 Case Study – Process for Data Extraction Digital Content

In Chapter 7, we presented a domain-specific rule generator which we refer to as DSRL. We evaluate our DSRL in a case study, through which we validate the research claim (RQ3) of this thesis (i.e., research and industrial applicability, see Chapter 1). This case study considers a scenario of customizing Digital Content Technology (DCT) service for machine translation. The DCT domain is a significant domain that contains multiple activities. The main process activities are: *data extraction*, *segmentation* and *Name Entity Recognition*, *Machine Translation*, *quality estimation*, and *post-editing*. These are illustrated in Figure 4.3; an overview of this case study is given in Section 4.3. For demonstration purposes, we focus on the `Extraction` sub-process which is a part of the DCT business process that extracts the data from different sources like from text, web, document and multimedia sources (see Figure 9.3). The data extraction is an initial and fundamental operation for retrieving data for machine translation. The objective of this case study is to validate the research and to prove which mode of configuration is better in the overall framework, i.e., the usability evaluation. In Chapter 3, we already explained, why this case study (Section 3.2.2) is relevant for our research.

We now conduct a comparative analysis of the manual and semi-automatic modes for the extraction activity. After carrying out the literature review and the interviews with BPM industries (Section 3.3.1 Problem Identification and Motivation), we considered a manual configuration as a baseline (or traditional) system to compare against the proposed semi-automatic. The emphasis is laid on analyzing the relative benefits of the proposed framework in a manual approach regarding the efficiency, effectiveness, and satisfaction with function and operational compliance support. The feature selection and configuration scenarios involve modifications resulting from the improvement of the complex process activity that affected the function and operation of the process.

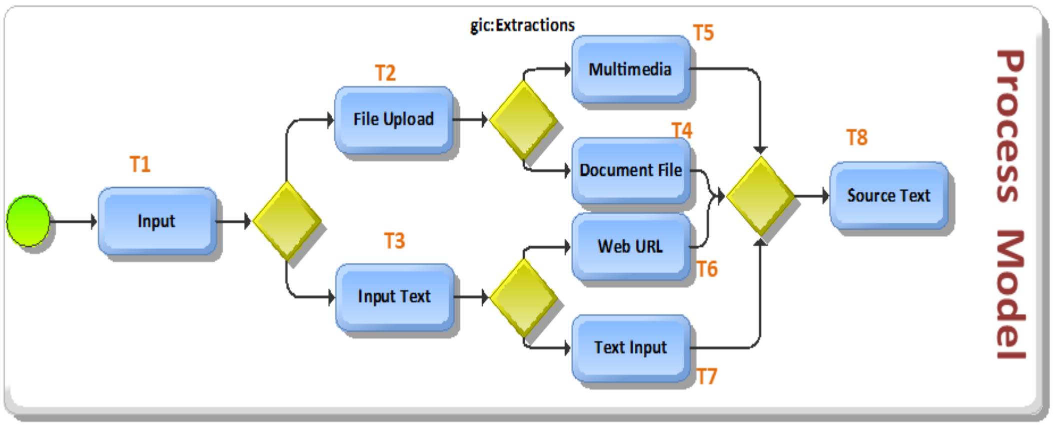


Figure 9.3: Extraction Sub-Process of the Digital Content Process.

The experiment was performed on an extraction sub-process of DCT of a real business process model. As per Figure 5.5 and 9.2, there are 8 classes and 8 activities (T1-T8) respectively in this case study: illustrating 27 class attributes are used in the entire experiment.

9.6 Experimental Design

The experiment was set up as a user *experiment* remotely through a web portal[15],with a *between-subjects design*. The user experiment was chosen remotely to be able to reach a wider audience of the domain and non-domain users in the domain of the DCT. An advantage is that this is a controlled experiment, with fixed tasks having different modes of settings (manual and semi-automatic). There is no control over the configuration value in manual setting. We use both analytical evaluation and experimental evaluation to evaluate the manual and semi-automatic configurations for performance in terms of efficiency and effectiveness. The analytical evaluation is used

[15] http://dsrl.nlplabs.org/

to evaluate the performance in speed/time, accuracy in error, correctness, and satisfaction. We implement a prototype to experimentally evaluate the manual and semi-automatic performance at process run-time for performance.

9.6.1 Definition and Planning

The experimental evaluation described in this chapter considers the strategy for a single product and team [204], where a researcher is interested in a better understanding of the quality of a product (software/technique), using an a priori set of variables for observation. The evaluation considers a number of experiments in

Table 9.2: User Experience Evaluation Methods

Evaluation Factor	Evaluation context
Lab tests	Prototyping- Framework
Field tests	Competitive evaluation of prototypes in the manual and semi-automatic environment
Field observation	Experiment result statistical analysis and observation
Evaluation of groups	Evaluating collaborative user experiences
Instrumented product	TRUE Tracking Real-time User Experience
Domain	Digital Content Technology
Approach	Evaluating UX jointly with usability
Evaluation data	Focus groups (multiple groups or measures, participants) evaluation (Quasi-Experiment)
User questionnaire	System Usability Scale
Human responses	PURE - preverbal user reaction evaluation
Expert evaluation	• Expert evaluation • Perspective-Based Inspection

the DCT cases to investigate the effectiveness, and efficiency of the proposed framework in different conditions with the domain constraint (as per section 9.4.1) and its values. Table 9.2 gives a brief overview of the entire experiment.

The semi-automatic performance measure and quality assurance processes are needed as the manual rule configuration by different end-users for statistical analysis of the efficiency in terms of configuration time, and the effectiveness in terms of error propensity, accuracy, and correctness.

A semi-structured questions survey was used for collecting qualitative data. The survey consisted of several open-ended questions relating to the usability of the system.

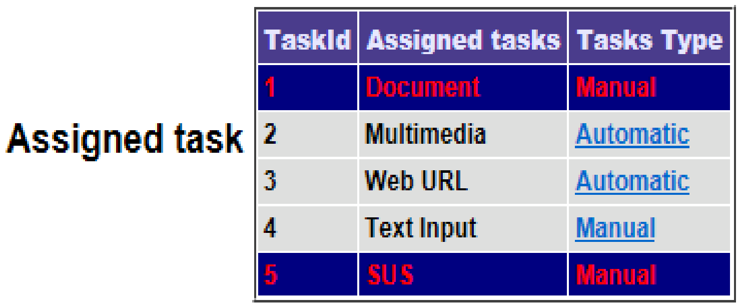
Initially, we developed a web-based prototype for rule generation and configuration with different tasks for participants in the same application. This assignment is called SUS in Figure 9.4 -with blue back color and red foreground color.

The SUS contains a set of pre-defined questions on system usability. Those participating in the experiments are asked to select the score based on their own experience, and then give critical feedback in terms of their satisfaction with the system in terms of the scale of effectiveness and efficiency.

9.6.2 Experimental Procedure

The experiment was performed by distributing a web link (URL) among domain users at the DCT Centre as well as at other organizations. The web link directed and asked the users to register the user with a consent form. After registration, the user can login and access the prototype. The welcome page defines the user-assigned tasks and explains the process of experiments as a power-point presentation tutorial. The PPT tutorial of the prototype was shown, followed by a short practice session.

After watching the tutorial, users can choose the task, and each task mentions the task type: manual, or semi-automatic. The users were instructed to select the feature model based on their assigned tasks and generate the rule. The configuration of the rule was carried out with two interfaces: first was the manual implementation and



Assigned task

TaskId	Assigned tasks	Tasks Type
1	Document	Manual
2	Multimedia	Automatic
3	Web URL	Automatic
4	Text Input	Manual
5	SUS	Manual

Figure 9.4: Assigned Tasks on Dashboard of Participants

the second was the semi-automatic. After generating the rule, participants configure the rules according to their requirements. The final steps consist of a SUS with ten different questions. A session takes about 15-20 minutes per user (sequential steps of experiment in *Appendix C*). This user experience experiment was a voluntary task, and no payments were made for participation.

9.6.3 Group and User Selection

In our research, more than one task was evaluated with different parameters, while having participants from a single group, as the tasks for participants were not randomly assigned. It was a controlled experiment, where each participant had five certain tasks in a different mode. This type of experiment is called a *quasi-experiment* [205]. The structure of the study follows factorial design because there are two or more independent variables in our research. The factorial design determines the number of conditions, so we consider adapting *between groups, and within group, or split-plot*. The *between groups* participants are only exposed to one experiment, and *within groups* participants are exposed to multiple experimental conditions. We used the *within group participants* for multiple tasks and conditions.

9.6.4 Participants and Tasks

We selected participants from the digital content domain organization and institution. The participants worked in the area of *web mining*, *machine translation*, *information retrieval* and *other digital contents*. There were *twenty-four* participants in this experiment: all the participants were divided into two different experiments using, manual and semi-automatic configuration. The experiment results were compared in terms of individual performance as well as experimental performance, and it included SUS score and the feedback of each participant.

Table 9.3 : Tasks Details of DCT Case Study

S.No.	Activities of Process Model (Tasks Names)	Configuration Parameters	Appendix
1.	Multimedia	8	C6
2	Document File	7	C8
3	Web URL	6	C5
4	Text Input	6	C7

Tasks and their Distributions

The total number of combinations of the features involved in experiments is ***4! (4x3x2=24)***. As each task has been divided into two categories like manual and semi-

Table 9.4 : Example of Web URL Task

S.No.	Tasks Names	Name of Configuration Parameters	Example
1.	Web URL	Input Text	This is a source text for English to German translation.
2.		Text (Source) Language	English
3.		Event	KeyPress
4.		Text Length	Word count
5.		*URL Name*	https://translate.google.com/
6		Target Language	German

automatic, therefore, the total number of tasks is *24x2=48*. In Table 9.3, we mentioned the tasks performed by the user which are *Multimedia, Document File, Web URL, and Text Input*. We also provide the number of parameters configured for individual tasks. In Table 9.4 present an example of Web URL task. Common errors occur in parameter configuration for example, the participants wrongly entered invalid Web URL.

Table 9.5: Tasks Distributions of DCT Case Study

Experiment Mode	Number of Participants	Number of Tasks Assigned to User	Total Tasks
Manual	24	2	48
Semi-automatic	24	2	48
SUS	24	1	24

automatic. Therefore, the total possible tasks (from Table 9.5) in one mode is 48 and total possible tasks assigned is 96. The distribution of tasks is listed in Table 9.5. Therefore, we considered 50% of the total possible combination of the tasks as distributed among the 24 participants in our experiment.

9.7 Evaluation of DSR Configuration: Efficiency

The objective is to evaluate the efficiency of the framework, based on the configuration time or processing time of the artifact or solution. First, we need to compare that the proposed framework is efficient with respect to the configuration time. Second, we evaluate how useful the framework is for a non-technical domain expert and non-domain user, and how easily it could have been understood, configured, and adapted by the end-users. We evaluate the configuration of the generated rule framework, using an empirical case study between the manual and semi-automatic modes.

The performance is measured at the time of completion of tasks (the configuration time), in manual and semi-automatic experiments, which are assigned to end-users.

We evaluate a considerable number of parameters involved in every task during experiments, based on which, the performance of the tasks is calculated. The assumption behind these criteria is to compare the time and effort required to implement the configuration of the generated rule that has a large number of constraints, or parametric values. For each evaluation strategy, we count the number of configuration parameters and how much time is taken for the configuration of these parameters. This measure is useful when there is a need to compare strategies using a number of parameters, irrespective of their configuration time.

9.7.1 Result and Analysis

The output of the manual rule is considered as a baseline. The standard manual configuration must be measured and compared to establish a baseline. In order to control the experiment, the configuration time needs to be kept within a set limit, and has to remain relatively constant. This allows for a comparison of the manual (baseline), and semi-automatic values, once we compare both the values and measures to indicate the performance.

The configured manual rule is considered as a baseline or standard for measuring and comparing tasks with a semi-automatic configuration. Our experiment is a controlled experiment where the set of limited parameters configured, are dependent on the tasks. We recorded the configuration time for both modes (manual and semi-automatic) and compared them.

Table 9.6: Paired t-Test Sample Statistics of Manual and Semi-automatic

Paired Samples Statistics					
		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Manual	2.0102	48	.39446	.05694
	Semi-automatic	1.1423	48	.38007	.05486

Statistical Evaluation efficiency of Manual v Semi-Automatic Configuration (Paired t-Test)

The following 3 tables illustrating a *paired t-Test* are: Paired Samples Statistics (9.6), Paired Samples Correlations (9.7), and Paired Samples Test (9.8) which present the statistical paired t-test evaluation on the (manual and semi-automatic) configuration time. Table 9.6 (Paired Samples Statistics) gives univariate descriptive statistics such as *mean, sample size, standard deviation, and standard error* for each of the entered variables. Table 9.7 (Paired Samples Correlations) shows the bivariate Pearson correlation coefficient with a two-tailed test of significance for each pair of variables entered. The Table 9.8 (Paired Samples Test) gives the hypothesis test results.

The Paired Samples Statistics result recurrences what was configured before the tests were executed. The Paired Samples Correlation table highlights the information that the manual configuration and semi-automatic configuration times is significantly positively correlated (r =. 457).

Table 9.7: Paired Samples Correlations of Manual and Semi-automatic

Paired Samples Correlations				
		N	Correlation	Sig.
Pair 1	Manual & Semi-automatic	48	.457	**.001**

Interpretation of Paired Sample Test is as follows (reading from left to right):

- First column: the pair of variables being tested, ordered, and then the subtraction is carried out.
- Mean: the average difference between the two variables.
- Standard deviation: the standard deviation of the difference scores.
- Standard error mean: the standard error, i.e., the standard deviation divided by the square root of the sample size.

Table 9.8: Paired t-Test Samples of Manual and Semi-automatic

Paired Samples Test									
		Paired Differences					T	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	Manual – Semi-automatic	.86792	.40371	.05827	.75069	.98514	14.895	47	.000

- T: the paired t-Test statistic (denoted T).
- df: the degrees of freedom for this test.
- Sig. (2-tailed): the p-value corresponding to the given test statistic t.

Result and Discussion of Pair t-Test

Manual and Semi-automatic scores were statistically signification correlated (r = 0.457, p < 0.001). There was a significant average difference between manual and semi-automatic configuration time (t47 =14.895, p < 0.001). On average, manual configuration times were .86792 higher than semi-automatic time (4.0371% CI [0.75069, 0.98514])

It shows that semi-automatic configuration is more efficient than manual configuration. We measured and compared the configuration time between the manual and the semi-automatic modes.

Comparison of Manual vs Semi-Automatic Mode

In Figures 9.5 and 9.6, we show a comparison between the manual and semi-automation configuration of rules with respect to comparative configuration time, i.e.,

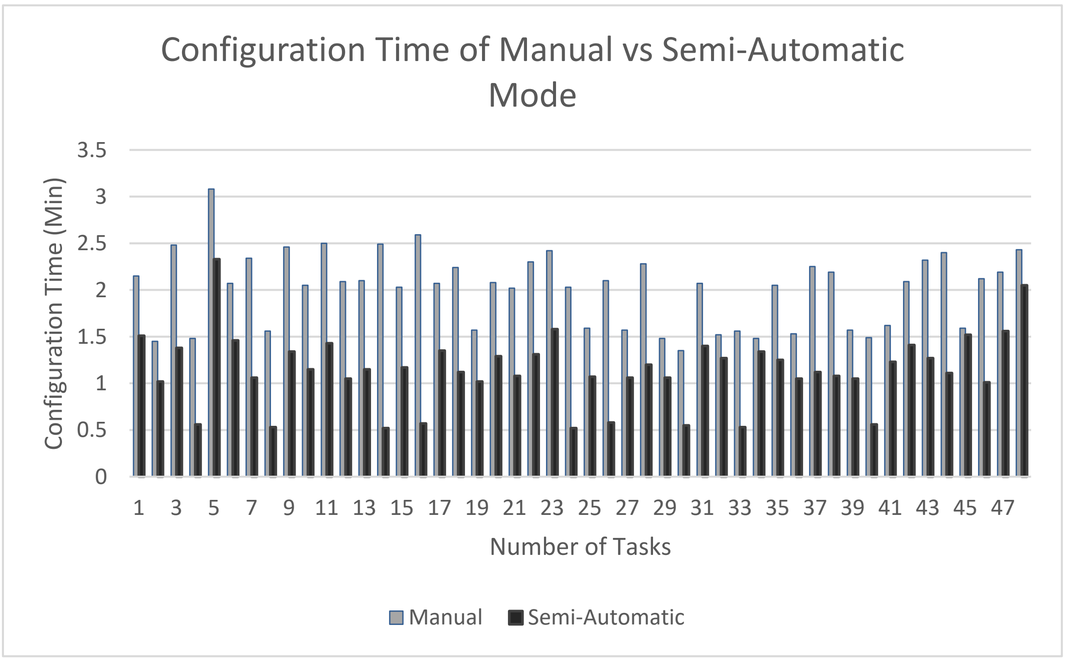


Figure 9.5: Comparison between Semi-automatic and Manual Configuration Time

the time to finish each task is given in minutes. Figure 9.5 illustrates the time taken for each of the 48 tasks. It shows how regular changes affect the configurationtime for

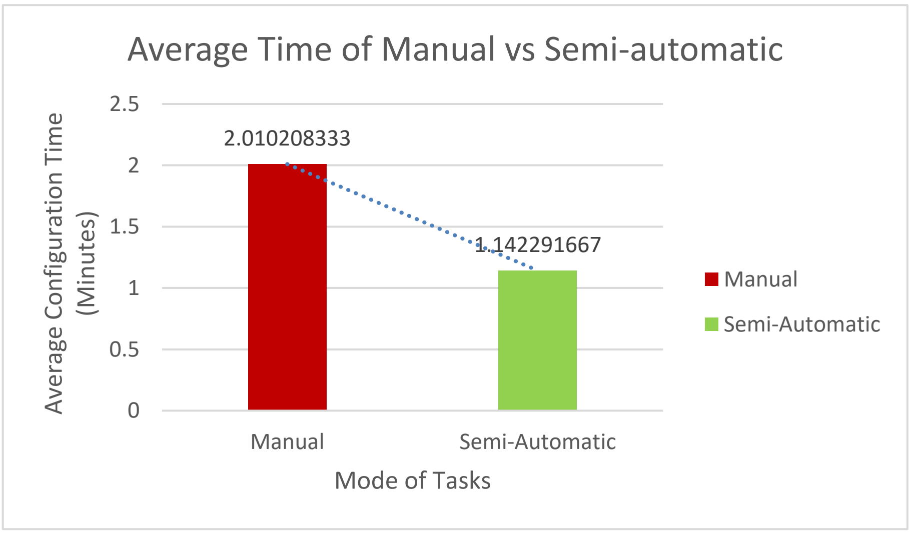


Figure 9.6: Average Time Taken in Manual and Semi-Automatic Configuration

both the manual and the semi-automatic modes. We observe that the semi-automatic mode is more efficient than the manual one. In Figure 9.6, we calculated the average time taken in each mode: the $Avg_{Manual} = 2.010$ and $Avg_{SemiAuto} = 1.142$. That is, the semi-automatic mode is *56.97%* more efficient than the manual one.

9.7.2 Discussion

In the DCT domain, the end-user can change their machine translation system, input data type, source language, target language, file type, file format, and the length of the text. The empirical observation shows that parametric semi-automatic configuration of the generated rule, takes lesser time than the manual configuration from *t Pair Test*, and as evident is lesser than the manual rule configuration, as illustrated in Figure 9.5. The average time taken in manual and semi-automatic configuration is then illustrated in Figure 9.6 and it shows the significant changes between them.

The objective of the DCT domain-specific rule configuration is to facilitate ease of the end-user with the execution of activities within the domain. In this case, the process model of the DCT is customized and configured with many sub-processes and activities with a large number of parameters in business applications. The requirements for change come from each business level, based on the requirements of the end-user, rapidly and regularly at the feature level. The list of *Web URL* tasks and its parameters are given in Table 9.4. The table is based on the user requirement of the participants.

9.8 Evaluation of DSR Configuration: Effectiveness

The main research question focuses on finding the usability of the overall framework and analysis of the configuration of the generated rules. In this section, we focus on the effectiveness of the configuration of the generated rules, in terms of error prevention and error corrections. This includes identification of the impact of accuracy and the quality of the configured rule with individual participants.

In this section, we discuss looking at the accuracy and quality impact of the overall framework by user experience. As discussed, a prototype is built to implement the proposed method. In this experiment, we analyze the performance of the semi-automatic configuration data, for finding the error, and a number of errors prevented. By using the prototype, we evaluate the accuracy of the proposed method to identify the impact of errors occurred and the errors prevented during the configuration of rules. We conducted experiments to evaluate the accuracy and adequacy of the proposed solution and further compared the effectiveness of the configuration with both the experiments.

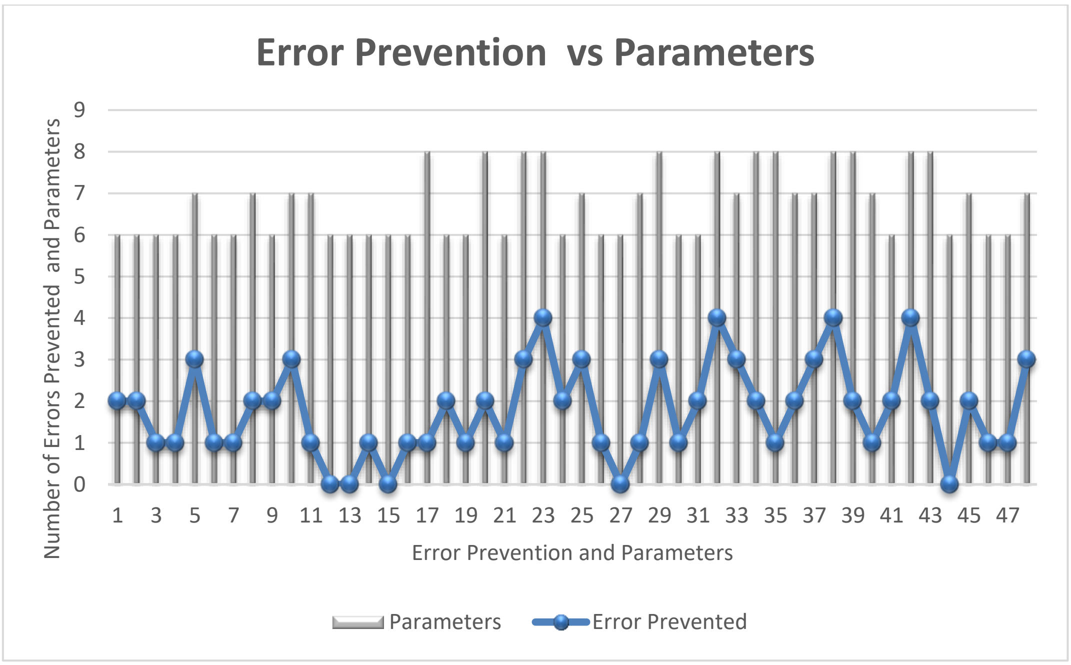


Figure 9.7: Error Prevention based on Numbers of Parameters

The objective is to evaluate the effectiveness of the proposed artifacts in finding the number of error preventions and assessing the quality during rule configuration based on the usability. Participants made mistakes during the rule configuration, but in system design, it is usually assumed that they should not make errors in the first place. However, if they do so, the semi-automatic configuration should detect these earlier, and it becomes easier to prevent mistakes. It helps participants to recognize, diagnose and recover from errors with proper error messages. The proposed approach offers error prevention and simple error handling that, in case, participants make mistakes, they can recover by providing clear and informative instructions.

9.8.1 Result and Analysis

The number of correct configured values and number of incorrect attempts is identified by using input from participants, and the number of errors is analyzed. Precision in the error measures the number of correctly configured rules, identified the impact compared to the number. In Figure 9.7, the graph shows the bar chart of the number of parameters and line chart shows the number of errors prevented during the semi-automatic configuration. As per graph and Table 9.9, the percentage of errors prevented during configuration is 26.24%. It indicates, the semi-automatic mode prevented significant amount of errors. Therefore, it improves the quality of the configured rules.

Table 9.9: Number of Error Prevented in Semi-Automatic Configuration

Total No. of Parameter	No. of Error Prevented	% of Error Prevented
324	85	26.24 %

9.9 Evaluation of Overall Framework by System Usability Score (SUS)

The comments and feedback by the participants, in accordance with the criteria set in the process of evaluation, may be considered as the final step in the list. Figure 9.1 describes this phase as a collection of data based on the feedback of the participants. As mentioned earlier, both qualitative and quantitative techniques have been employed for the purpose of validating the present study. An elaborate account of the logic and motives of the choice of this approach has been provided in Section 9.4 as evaluation strategy. The respondents of the scale (SUS) expressing their opinion in the last section with feedback and comments on a 5-point scale (by selecting a radio button), along with any particular comment in a text box if they wish so (presented in Figure D.1 at *Appendix D* as a user interface of SUS form[16]) is required.

9.9.1 System Usability Score Process

The first step of the evaluation starts with the identification of parameters or criteria. Next, data collection from the participants begins, as per the steps described in Figure 9.1. Both qualitative and quantitative techniques (with an emphasis on qualitative approach) have been employed for the purpose of evaluating the present study. An elaborate account of the logic and motives behind the choice of this approach has been

[16] Web interface of SUS http://dsrl.nlplabs.org/UsabilityScale.aspx

provided in Section 3.3.4. The Post-Study System Usability Questionnaire (PSSUQ) [206] survey tool was adopted and employed for collecting the quantitative data at this juncture and SUS served as the evaluative instrument since it allows capturing the usability factor [207]. A total of 10 questions is presented on this scale, where the participants are required to choose from five response categories ranging from strongly disagree to agree strongly (*Appendix D*). Along with this, to understand the logic of the respondents' response-selection, qualitative data was gathered from them. The data was collected by the researcher himself and the outcome in this phase was raw data – like SUS answer sheets that can be scored to obtain quantitative results and the open-ended text comments for each SUS question, that are qualitative in nature.

9.9.2 SUS Calculation and Measurement

The administration of SUS usually takes place after the participants have received enough exposure to the rule generation prototype system but not before a discussion or debriefing has occurred about the same. Participants are required to choose the immediate response that comes to their mind after reading each specific question, instead of dwelling long about each item. Another requirement is that each question is to be answered. In case a participant is undecided regarding a particular question, the centre point of the scale is to be checked for that item. After the scale is properly filled, it has to be scored to obtain a single score for the entire scale that provides a composite measure of the system usability as a whole. The individual item scores are irrelevant in SUS. In order to obtain the final score for the SUS, the individual item scores are to be determined first, which ranges from 0 to 4. For items 1, 3, 5, 7, and 9 the score contribution is the scale position minus 1. For items 2, 4, 6, 8 and 10 the contribution is 5 minus the scale position. In order to calculate the total score for the scale, the sum of all scores is to be multiplied by 2.5. The total score should always be between 0 and 100. An example of SUS scoring is presented in next Section 9.9.3.

The Evaluation tools differ in terms of their effectiveness, on the basis of particular features of the environment and goals of evaluation [208]. Few popular evaluation methods are heuristic evaluation [209], field studies and observations [210, 211], filling questionnaires based on the usability of the overall framework and participant performance in web-based environment.

In the context of the present study, involving evaluation of rule configuration systems, a primary focus lies on identifying a design science methodology which involves adequate tools to evaluate the usability of different framework components in the digital content domain, along with ascertaining the satisfaction and effectiveness of the prototype of the framework on the basics of usability. For this purpose, an elaborate evaluation research of a digital content domain framework has been carried out in a controlled environment by a group of domain experts. The particular framework prototype was developed and is currently being in operative, allowing a platform in which customization of the process model and the configuration of its operational part can be done by non-technical domain experts through generating a rule from the customized domain model. The evaluation of usability for overall framework SUS component of the prototype is carried out in the context of this study.

9.9.3 Experimental Results

Analysis of statistical performance in terms of efficiency is provided in Section 9.7 where processing time of configuration is described and, in Section 9.8 the effectiveness is assessed on the basis of the accuracy and quality of the configured rule. The SUS has been used for the validation of the statistical analysis. In order to assess the overall performance of the framework, the statistical score has been compared with the subjective score. The subjective score model is described in the previous Section 9.9.2.

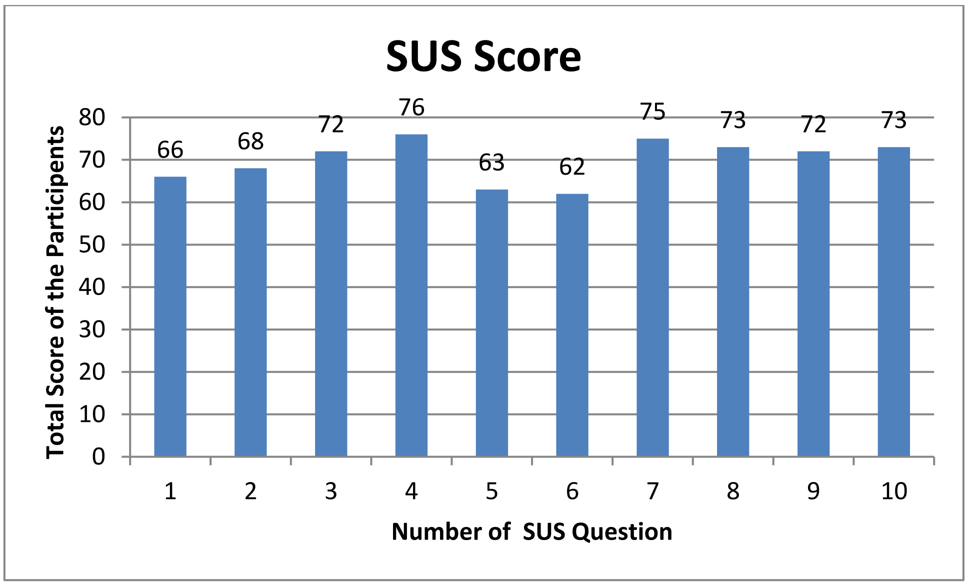


Figure 9.8: SUS Score Individual Questions

In addition to the quantitative analysis outlined above, a pre-defined question approach was employed with in tandem the SUS to satisfy the usability evaluation criteria

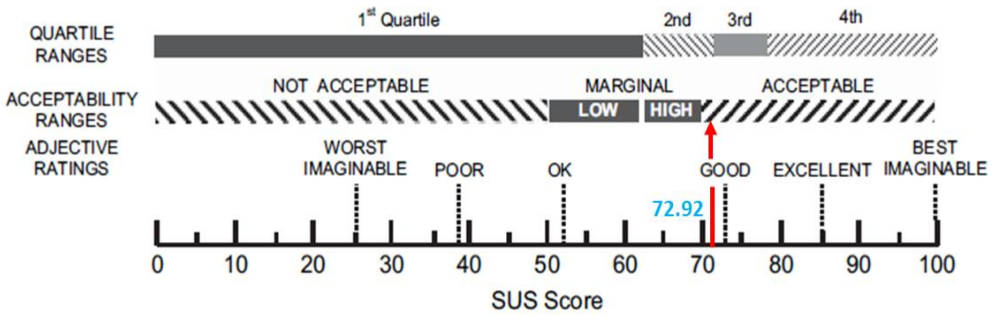


Figure 9.9: SUS Normal Scale

discussed in Table 9.1, and Figure 9.2. Reflecting the research criteria, there were three main areas of concern: efficiency, effectiveness, and satisfaction. The sub criteria of efficiency are performance, i.e., processing time or configuration time. As per Table D1 and Section D1 (in *Appendix D*), the question number 8 “I found the system very cumbersome to use” is associated with efficiency and its SUS score is 73, it shows that the prototype is efficient. This is presented in *Appendix D*.

A per Figure 9.8 and 9.9, it is evident that the semi-automatic configuration is more efficient, effective, and satisfactory than the manual configuration in terms of performance, accuracy, quality, learnability, user-friendliness, and reliability. The horizontal axis represents 5 positive and 5 negative questions, and in the Figure 9.8, the vertical axis of indicates the total number of points corresponding to each question.

9.10 Summary

The aim of this chapter was to answer the third research question RQ3 – "*How to evaluate the proposed prototype of the framework in terms of its usability?*". The focus was to evaluate the usability of the artifact from two points of views: (i) Effectiveness and Efficiency of rule configuration and, (ii) Satisfaction of the overall framework in the DCT domain. We discussed the principal findings, and results of this research evaluation.

In summary, we evaluated the usability of the framework for adaptation of the configured rule for process model customization. We assessed the efficiency, effectiveness, and satisfaction of the proposed solution. Specific to this research, we evaluated, the usability of the framework for non-technical domain experts where the rules were generated and configured with specific claims, which we made in Chapter 1 (discussed in Section 1.3 as evaluation claim):

- *Evaluation of Hypothesis 1* (run-time efficiency): The configuration activities that need to be semi-automatically configured in the generated rule are required to be time efficient in comparison to manual (traditional) configuration. Therefore, we compared the time taken by participants in the manual versus semi-automatic tasks. The semi-automatic configuration turned out to be more efficient than manual configuration (see Section 9.7). The difference in performance was *statistically significant*.
- *Evaluation of Hypothesis 2* (run-time effectiveness): Although the number of parameters that needed to be configured are modest, the framework yielded a *26.24 %* reduction of error when using the semi-automatic, as opposed to manual evaluation. We only considered the data validation (data type error prevention like string, integer and date time) (as per Section 9.8).

- *Evaluation of Hypotheses 3* (overall satisfaction): Results from SUS and data analysis showed that participants' experience was positive based on SUS Likert scale with regard to satisfaction Table 9.1 of use in terms of: (i) *efficiency*; (ii) *effectiveness*; (iii) most of them do not need a technical person to run the system, (iv) *learnability* and (v) *overall framework* (as per Section 9.9).

CHAPTER 10

CONCLUSIONS AND FUTURE WORKS

10.1 Overview

This chapter outlines key conclusions, the significance of the research and discusses its limitations. It identifies possible areas for future research as follow: Section 10.2 provides a summary of the research contribution of the thesis; Section 10.3 discusses limitations; and Section 10.4 examines possibilities for further research. Section 10.5 presents closing remarks which encapsulate the essence of this thesis.

10.2 Summary of Solution Approach and Thesis Contribution

As per the thesis introduction and the acknowledged inherent problem and challenges that end-users in business face challenges, caused by day to day changes from internal and, external sources. The users in business require a platform where they can customize and configure changes to business strategy and adapt these changes smoothly to their business processes.

This research aims to present an appropriate framework where *non-technical domain users* can customize and configure a process model with the assistance of rule language generation and domain constraint configuration. No doubt, variability is an important concept: variability analysis, modeling, and management have been the fundamental research in software product line engineering (SPLE). SPLE addresses the challenges of planning and developing systems with the foundation of large-scale reuse in development. In fact, SPLE offers effective and efficient methods and techniques for variability and systematic reuse in software development, in order to: (i) enable mass customization, and (ii) support configurable software architectures.

This approach of *configuration and customization* to meet the requirements of individual customers, based on modifications, is a desirable model. In parallel, model-driven approaches are one of the most promising paradigms in software engineering. The model-driven approach enables a systematic use of models in the engineering lifecycle. Models are first class entities, foremost, to replace the code as a primary artifact.

10.2.1 Contribution of the Research

The thesis contributes a DSRG framework where by the process model customizations and configuration of the domain constraint at run-time is configured by end-user dynamically. Rule generation, configuration, and validation, as well as validation of feature selection by the end-user, are presented in this thesis. The process model customization based on a domain template is included. The contribution of the research is summarized as follows:

- A DSRG framework that allows the generating a set of rules from a high-level of abstract models (domain model) to a low-level configurable rule language on an ad hoc basis.

- Controlled variability management models to customize the domain template for generating the DSR.

- A DSR language definition, explicitly in ECA language, which is based on XML.

- A set of domain-specific rules are generated from the domain model, and configuring the domain constraint for process model customization.

- A prototype implementation of dynamic configuration of generated rules and customization of process models.

10.2.2 A DSRG Framework Solution

The primary research question focuses on the development of a usability framework where end-users can generate and configure domain-specific rules to customize process model dynamically. This research addresses the problem by dividing it into three research questions and proposing a solution for each question. The first sub-research question is '*RQ 1. How to develop a rule generation and configuration framework to customize the process model dynamically?*'.

The first step of the solution is to design the essential components and the framework as they are defined in Chapter 4. The next step is the development and implementation of designed components as a prototype framework where end-users can perform their tasks.

The design-time challenges were discussed in Chapter 5. The basic components were validated conceptually and theoretically in line with other research in this area. This chapter defined the abstract syntax, in order to discuss the semantic checks for the domain model.

10.2.3 Domain-specific Rule Language Definition

The primary purpose of the research is to define the rule language that can represent the translation of a low-level rule to a high-level domain model. The second sub-research question '*RQ2 How to implement a framework to support a domain-specific rule language that is usable by non-technical domain experts?*' primarily focuses on the rule language and definition in terms of abstract and concrete syntax, as well as an adopted model of ECA language for the fulfillment of operational and functional requirements. Domain-specific rule language are defined in terms of basic artifacts such as syntax, structure, definition, and model in Chapter 6.

Chapter 6 provided a partial answer to the RQ2, covering the main objective of the question; is it 'usable for non-technical domain experts.' Section 6.4.2 describes how the rule language in the XML format including the ECA language is expressed, allowing a non-technical person to configure the rule or implement the usable criteria.

10.2.4 Domain-specific Rule Generation

Another important element of this research is generating domain-specific rules from the high-level domain model. Accordingly, the model-driven approach (MDA) was used to generate the rule as a text model from the high-level model. It is also relevant to RQ2 as it relates to the implementation side of the framework where non-technical domain experts where able to generate the desired configurable rules, and to configure them according to their need or the domain constraints of the application. The implementation of rule translation from the domain model in terms of conceptual, theoretical and technical implementation aspects are discussed in Chapter 5 and Chapter 7.

10.2.5 Framework Implementation

This research uses the SPLE as a platform to combine all the components which were defined in Chapter 4. At the application level, the framework takes input from the end-user and processes it at the domain engineering level. After processing, the final output comes back to the end-user with the appropriate changes applied dynamically at application engineering level. The implementation part of a framework has two main components: implementation of the DSR generation, and customization of the process model. The DSR generation is covered in Chapter 7 and process model customization is implemented in Chapter 8.

10.3 Limitation

This research does not address all the problems associated with rule definition, generation, and configuration. There are areas that are not covered, and areas that are covered are not necessarily covered in the required depth. Consequently, we present the work on configuration-based DSR generation for process model customization with the following limitations:

1. The prototype does not provide a facility to end-users to add new features and update the feature from the application end. As discussed in Chapter 2 and 4, domain expert and process engineer work on the domain engineering level. End-users have only access to the application engineering level, they cannot access the domain engineering. Therefore, our framework has a dependency on the domain expert as they can only add new features in a domain template.

2. The SPLE is used as a platform of the framework. This framework restricts adding, editing and removing of the features from the feature model at run-time. End-users can only activate and deactivate the feature in domain template at run-time. Whereas the domain expert can change the domain template at design time (domain engineering level). Hence, the framework limits adding and updating the features at application engineering level or run time. As we have mentioned, in the first limitation above i.e., during domain engineering, the domain template can be changed, which is generic for the particular domain. It has the limitation that it works in a specific domain. Therefore, the current research is focused on the Digital Content Technology domain (DCT). In this thesis, we only apply our framework in one domain, but it could be used in another domain.

3. We considered the two levels of classes which comprised one level of parent or superclass class and second level of child class at the time of rule generation. Therefore, a framework is restricted for implementing the domain model to n-level of a tree or an ontologies structure of classes. This is the limitation of the framework, i.e., it covers only the above two levels.

10.4 Future Work

The proposed work can be extended in several ways. The directions in which the work can be extended are summarized as follows: For the approach to be general, i.e., applicable to other domains, research focusing on how to adapt the DSRL across multiple domains and how to convert conceptual models into universal DSR languages will be required.

Thus far, the framework supports semi-automatic translation, however, it can be improved with a system that learns from existing rules and domain models. This improvement is driven by the feature approach and results in an automated DSRL generation. However, this thesis addresses many of the problems identified, through the research, and discovered areas that would benefit from further investigation in the future.

1. The proposed rule generation and configuration approach are based on the requirements of the end-user. Based on past activities of the end-user and work patterns, the feature model provides feature recommendations in real time. Additionally, the system recommends features based on previously generated rules and vice versa. These approaches can be utilized for mining the previously configured rules, which can be applied to customization and configuration of process models. The system also recommends the future steps during rule generation and configuration on the case information.

2. In the proposed rule generation, we consider only two level of the classes (discussed in Section 10.3). This limitation needs more attention. Also, attention is given to the value of the parameters that should be automatically recommended to end-user based on their previous configuration values.

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

customizable entities. It targets constraints as the customization point and is advantageous compared to customization through restructuring. For BPaaS, if a generic service is provided to external users, the dynamic customization of individual process instances would require the utilization of a coordinated approach, e.g., through using a coordination model. Other architecture techniques can also be used to facilitate flexible and lightweight cloud-based provisioning of process instances, e.g., through containerization.

4. For both digital content technology and machine translation systems, users require more autonomic functionality. We consider the rule generation and configuration techniques for process model customization of the DCT domain that can achieve similar results using other techniques such as Service Oriented Architecture or Method Engineering or Service-as-a-service as described here.

5. Further research which focuses on adapting the DSRL across different domains and converting conceptual models into generic DSR language, applicable to other domains, is required. Thus far, this translation is semi-automatic, but shall be improved with another domain as mentioned in third limitation (Section 10.3).

10.5 Conclusion

The contribution of this thesis is that people who have not much technical knowledge can easily create and customize business application to deal with rapid changes in the business world. This research work demonstrates a prototype framework for generating the rule language and configuring domain constraints. This framework builds upon the core idea of Software Product Lines Engineering (SPLE) and Model-Driven Architecture (MDA). We evaluated the usability of the framework for adaptation of the configured rule for process model customization. We assessed the efficiency, effectiveness, and satisfaction of the proposed solution. The novel approach of this research is to generate domain-specific rule language from a high-level domain model by only using variability management.

REFERENCE

[1] Y. Antonucci, "Using workflow technologies to improve organizational competitiveness," *International journal of management,* vol. 14, pp. 117-126, 1997.

[2] R. Lenz and M. Reichert, "IT support for healthcare processes–premises, challenges, perspectives," *Data & Knowledge Engineering,* vol. 61, pp. 39-58, 2007.

[3] A. Kumar and W. Yao, "Design and management of flexible process variants using templates and rules," *Computers in Industry,* vol. 63, pp. 112-130, 2// 2012.

[4] A. Jiménez-Ramírez, B. Weber, I. Barba, and C. Del Valle, "Generating optimized configurable business process models in scenarios subject to uncertainty," *Information and Software Technology,* vol. 57, pp. 571-594, 1// 2015.

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

[6] G. H. Alférez, 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.

[7] C. Ayora, V. Torres, B. Weber, M. Reichert, and V. Pelechano, "VIVACE: A framework for the systematic evaluation of variability support in process-aware information systems," *Information and Software Technology,* vol. 57, pp. 248-276, 2015.

[8] J. Wang, Z. Feng, J. Zhang, P. C. Hung, K. He, and L.-J. Zhang, "A Unified RGPS-Based Approach Supporting Service-Oriented Process Customization," in *Web Services Foundations*, ed: Springer, 2014, pp. 657-682.

[9] A. Lazovik and H. Ludwig, "Managing process customizability and customization: Model, language and process," in *Web Information Systems Engineering–WISE 2007,* ed: Springer, 2007, pp. 373-384.

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

[11] N. Assy, W. Gaaloul, and B. Defude, "Mining Configurable Process Fragments for Business Process Design," in *DESRIST*, 2014, pp. 209-224.

[12] T. Morgan, "Business process modeling and ORM," in *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops,* 2007, pp. 581-590.

[13] L. Aldin and S. de Cesare, "A literature review on business process modelling: new frontiers of reusability," *Enterprise Information Systems,* vol. 5, pp. 359-383, 2011.

[14] O. Standard, "Web services business process execution language version 2.0."

[15] B. List and B. Korherr, "An evaluation of conceptual business process modelling languages," in *Proceedings of the 2006 ACM symposium on Applied computing*, 2006, pp. 1532-1539.

[16] S. A. White, "Business process modeling notation," *Specification, BPMI. org,* 2004.

[17] M. Reichert and P. Dadam, "ADEPTflex—Supporting dynamic changes of workflows without losing control," *Journal of Intelligent Information Systems,* vol. 10, pp. 93-129, 1998.

[18] W. M. Van Der Aalst and A. H. Ter Hofstede, "YAWL: yet another workflow language," *Information systems,* vol. 30, pp. 245-275, 2005.

[19] M. Boukhebouze, Y. Amghar, A. c.-N. Benharkat, and Z. Maamar, "A rule-based approach to model and verify flexible business processes," *International Journal of Business Process Integration and Management,* vol. 5, pp. 287-307, 2011.

[20] M. E. Rangiha and B. Karakostas, "Goal-driven social business process management," in *Science and Information Conference (SAI), 2013*, 2013, pp. 894-901.

[21] A. Gromoff, N. Kazantsev, K. Evina, M. Ponfilenok, and D. Kozhevnikov, "Modern era in business architecture Design," *Far East Journal of Psychology and Business,* vol. 9, pp. 15-34, 2012.

[22] T. van Eijndhoven, M.-E. Iacob, and M. L. Ponisio, "Achieving business process flexibility with business rules," in *Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE,* 2008, pp. 95-104.

[23] C. Ayora, V. Torres, M. Reichert, B. Weber, and V. Pelechano, "Towards run-time flexibility for process families: open issues and research challenges," in *Business Process Management Workshops*, 2013, pp. 477-488.

[24] S. W. Sadiq, M. E. Orlowska, and W. Sadiq, "Specification and validation of process constraints for flexible workflows," *Information Systems*, vol. 30, pp. 349-378, 7// 2005.

[25] B. Weber, S. Sadiq, and M. Reichert, "Beyond rigidity–dynamic process lifecycle support," *Computer Science-Research and Development*, vol. 23, pp. 47-65, 2009.

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

[27] M. Fowler, *Domain-specific languages*: Pearson Education, 2010.

[28] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," *ACM computing surveys (CSUR)*, vol. 37, pp. 316-344, 2005.

[29] P. Hudak, "Domain-specific languages," *Handbook of Programming Languages*, vol. 3, pp. 39-60, 1997.

[30] J. Brooke, "SUS-A quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, pp. 4-7, 1996.

[31] J. Bosch, *Design and use of software architectures: adopting and evolving a product-line approach*: Pearson Education, 2000.

[32] F. Stallinger and R. Neumann, "A Framework for Innovation System Customization for Product Line-based Software Businesses," in *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, 2013, pp. 94-97.

[33] N. Anquetil, B. Grammel, I. Galvao Lourenco da Silva, J. Noppen, S. Shakil Khan, H. Arboleda, *et al.*, "Traceability for model driven, software product line engineering," 2008.

[34] W. B. Frakes and S. Isoda, "Success factors of systematic reuse," *IEEE software*, vol. 11, pp. 14-19, 1994.

[35] P. Clements and L. Northrop, "Software product lines: practices and patterns," 2002.

[36] M. Acher, "Managing, multiple feature models: foundations, languages and applications," Nice, 2011.

[37] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," DTIC Document1990.

[38] S. Soltani, M. Asadi, D. Gašević, M. Hatala, and E. Bagheri, "Automated planning for feature model configuration based on functional and non-functional requirements," in *Proceedings of the 16th International Software Product Line Conference-Volume 1*, 2012, pp. 56-65.

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

[40] M. Hindawi, L. Morel, R. Aubry, and J.-L. Sourrouille, "Description and implementation of a UML style guide," in *International Conference on Model Driven Engineering Languages and Systems*, 2008, pp. 291-302.

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

[42] M. Harsu, *A survey on domain engineering*: Citeseer, 2002.

[43] J.-C. Royer and H. Arboleda, *Model-Driven and Software Product Line Engineering*: John Wiley & Sons, 2013.

[44] G. Génova, M. C. Valiente, and M. Marrero, "On the difference between analysis and design, and why it is relevant for the interpretation of models in Model Driven Engineering," *Journal of Object Technology*, vol. 8, pp. 107-127, 2009.

[45] A. Kalnins, L. Lace, E. Kalnina, and A. Sostaks, "DSL Based Platform for Business Process Management," in *International Conference on Current Trends in Theory and Practice of Informatics*, 2014, pp. 351-362.

[46] K. C. Kang and H. Lee, "Variability modeling," in *Systems and Software Variability Management*, ed: Springer, 2013, pp. 25-42.

[47] K. Czarnecki and M. Antkiewicz, "Mapping features to models: A template approach based on superimposed variants," in *Generative programming and component engineering*, 2005, pp. 422-437.

[48] H. Gomaa, "Designing software product lines with uml 2.0: From use cases to pattern-based software architectures," in *Reuse of Off-the-Shelf Components*, ed: Springer, 2006, pp. 440-440.

[49] K. Schmid and I. John, "A customizable approach to full lifecycle variability management," *Science of Computer Programming,* vol. 53, pp. 259-284, 12// 2004.

[50] P. Sochos, I. Philippow, and M. Riebisch, "Feature-oriented development of software product lines: mapping feature models to the architecture," in *Object-Oriented and Internet-Based Technologies*, ed: Springer, 2004, pp. 138-152.

[51] M. A. Simos, "Organization domain modeling (ODM): Formalizing the core domain modeling life cycle," in *ACM SIGSOFT Software Engineering Notes*, 1995, pp. 196-205.

[52] M. Diouf, S. Maabout, and K. Musumbu, "Merging model driven architecture and Semantic Web for business rules generation," in *International Conference on Web Reasoning and Rule Systems*, 2007, pp. 118-132.

[53] M. L. Griss, J. Favaro, and M. D. Alessandro, "Integrating feature modeling with the RSEB," in *Software Reuse, 1998. Proceedings. Fifth International Conference on*, 1998, pp. 76-85.

[54] M. Eriksson, J. Börstler, and K. Borg, "The PLUSS approach–domain modeling with features, use cases and use case realizations," in *Software Product Lines*, ed: Springer, 2005, pp. 33-44.

[55] A. Mos and M. Cortes-Cornax, "Business Matter Experts do Matter: A Model-Driven Approach for Domain Specific Process Design and Monitoring," *arXiv preprint arXiv:1606.04287,* 2016.

[56] J. Bayer, S. Gerard, Ø. Haugen, J. Mansell, B. Møller-Pedersen, J. Oldevik, *et al.*, "Consolidated product line variability modeling," in *Software Product Lines*, ed: Springer, 2006, pp. 195-241.

[57] M. Sinnema and S. Deelstra, "Classifying variability modeling techniques," *Information and Software Technology,* vol. 49, pp. 717-739, 2007.

[58] K. Czarnecki, S. Helsen, and U. Eisenecker, "Formalizing cardinality-based feature models and their specialization," *Software process: Improvement and practice,* vol. 10, pp. 7-29, 2005.

[59] K. Czarnecki, S. Helsen, and U. Eisenecker, "Staged configuration through specialization and multilevel configuration of feature models," *Software Process: Improvement and Practice,* vol. 10, pp. 143-169, 2005.

[60] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch, "Covamof: A framework for modeling variability in software product families," in *Software Product Lines*, ed: Springer, 2004, pp. 197-213.

[61] D. Beuche, H. Papajewski, and W. Schröder-Preikschat, "Variability management with feature models," *Science of Computer Programming,* vol. 53, pp. 333-352, 12// 2004.

[62] L. Chen and M. Ali Babar, "A systematic review of evaluation of variability management approaches in software product lines," *Information and Software Technology,* vol. 53, pp. 344-362, 4// 2011.

[63] J. Van Gurp, J. Bosch, and M. Svahnberg, "On the notion of variability in software product lines," in *Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on*, 2001, pp. 45-54.

[64] A. Hein, M. Schlick, and R. Vinga-Martins, "Applying feature models in industrial settings," in *Software Product Lines*, ed: Springer, 2000, pp. 47-70.

[65] L. Lace, A. Kalnins, and A. Sostaks, "Process DSL Transformation by Mappings Using Virtual Functional Views," *Baltic Journal of Modern Computing,* vol. 3, p. 133, 2015.

[66] A. Van Deursen and P. Klint, "Domain-specific language design requires feature descriptions," *CIT. Journal of computing and information technology,* vol. 10, pp. 1-17, 2002.

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

[68] 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.

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

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

[71] X. Zhou, S. Chen, B. Liu, R. Zhang, Y. Wang, P. Li, *et al.*, "Development of traditional Chinese medicine clinical data warehouse for medical knowledge discovery and decision support," *Artificial Intelligence in Medicine*, vol. 48, pp. 139-152, 2010/02/01/ 2010.

[72] F. Puhlmann, A. Schnieders, J. Weiland, and M. Weske, "Variability mechanisms for process models," *PESOA-Report TR*, vol. 17, pp. 10-61, 2005.

[73] R. Mietzner and F. Leymann, "Generation of BPEL customization processes for SaaS applications from variability descriptors," in *Services Computing, 2008. SCC'08. IEEE International Conference on*, 2008, pp. 359-366.

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

[75] A. Paschke, H. Boley, Z. Zhao, K. Teymourian, and T. Athan, "Reaction RuleML 1.0: standardized semantic reaction rules," in *Rules on the Web: Research and Applications*, ed: Springer, 2012, pp. 100-119.

[76] K. Kulkarni, N. Mattos, and R. Cochrane, "Active database features in SQL3," in *Active Rules in Database Systems*, ed: Springer, 1999, pp. 197-219.

[77] A. Bonifati, D. Braga, A. Campi, and S. Ceri, "Active XQuery," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002, pp. 403-412.

[78] E. Cho, "ARML: an active rule mark-up language for heterogeneous active information systems."

[79] S. Abitrboul, O. Benjellourn, I. Manolescu, T. Milo, and R. Weber, "Active xml: Peer-to-peer data and web services integration," in *Proceedings of the 28th international conference on Very Large Data Bases*, 2002, pp. 1087-1090.

[80] R. Khalaf, "From RosettaNet PIPs to BPEL processes: A three level approach for business protocols," *Data & Knowledge Engineering*, vol. 61, pp. 23-38, 2007.

[81] A. Schnieders and F. Puhlmann, "Variability Mechanisms in E-Business Process Families," *BIS*, vol. 85, pp. 583-601, 2006.

[82] N. Boffoli, D. Caivano, D. Castelluccia, and G. Visaggio, "Business process lines and decision tables driving flexibility by selection," in *Software Composition*, 2012, pp. 178-193.

[83] M. La Rosa, W. M. van der Aalst, M. Dumas, and F. P. Milani, "Business process variability modeling: A survey," 2013.

[84] M. La Rosa, W. M. van der Aalst, M. Dumas, and A. H. Ter Hofstede, "Questionnaire-based variability modeling for system configuration," *Software and Systems Modeling*, vol. 8, pp. 251-274, 2009.

[85] M. Lin, J. Malec, and S. Nadjm-Tehrani, "On semantics and correctness of reactive rule-based programs," in *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, 1999, pp. 235-246.

[86] S. A. White, "Introduction to BPMN," *IBM Cooperation*, vol. 2, p. 0, 2004.

[87] W. M. P. van der Aalst and A. H. M. ter Hofstede, "YAWL: yet another workflow language," *Information Systems*, vol. 30, pp. 245-275, 6// 2005.

[88] M. Dumas and A. H. Ter Hofstede, "UML activity diagrams as a workflow specification language," in *≪ UML≫ 2001—The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, ed: Springer, 2001, pp. 76-90.

[89] R. Davis, *Business process modelling with ARIS: a practical guide*: Springer Science & Business Media, 2001.

[90] IBM, "WebSphere©MQ Workow FlowMareket©Definition Language (FDL)," December 2010.

[91] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *Software Engineering, IEEE Transactions on*, vol. 30, pp. 311-327, 2004.

[92] JBoss, "jBPM Process Denition Language (jPDL)," January 2008.

[93] R. Maker, T. Cullinane, P. deWitte, W. Knappenberger, B. Perakath, and M. Wells, "IDEF3–process description capture method report," *Information Integration for Concurrent Engineering (IICE), Armstrong Laboratory, Wright-Patterson AFB, OH,* 1992.

[94] H. Mili, G. Tremblay, G. B. Jaoude, É. Lefebvre, L. Elabed, and G. E. Boussaidi, "Business process modeling languages: Sorting through the alphabet soup," *ACM Computing Surveys (CSUR)*, vol. 43, p. 4, 2010.

[95] 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.

[96] M. Colombo, E. Di Nitto, and M. Mauri, "Scene: A service composition execution environment supporting dynamic changes disciplined through rules," in *Service-Oriented Computing–ICSOC 2006*, ed: Springer, 2006, pp. 191-202.

[97] C. Barreto, "Web services business process execution language version 2.0," 2007.

[98] A. Bucchiarone, C. Mezzina, and M. Pistore, "CAptLang: a language for context-aware and adaptable business processes," in *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, 2013, p. 12.

[99] J. Bézivin, G. Dupé, F. Jouault, G. Pitette, and J. E. Rougui, "First experiments with the ATL model transformation language: Transforming XSLT into XQuery," in *2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture*, 2003.

[100] M. Reichert and B. Weber, *Enabling flexibility in process-aware information systems: challenges, methods, technologies*: Springer Science & Business Media, 2012.

[101] M. Weske, *Business process management: concepts, languages, architectures*: Springer Science & Business Media, 2012.

[102] B. Mohabbati, D. Gašević, M. Hatala, M. Asadi, E. Bagheri, and M. Bošković, "A quality aggregation model for service-oriented software product lines based on variability and composition patterns," in *Service-Oriented Computing*, ed: Springer, 2011, pp. 436-451.

[103] O. Bubak and H. Gomaa, "Applying software product line concepts in service orientation," *International Journal of Intelligent Information and Database Systems,* vol. 2, pp. 383-396, 2008.

[104] B. Mohabbati, M. Asadi, D. Gašević, and J. Lee, "Software Product Line Engineering to Develop Variant-Rich Web Services," in *Web Services Foundations*, ed: Springer, 2014, pp. 535-562.

[105] D. Beuche, "Modeling and building software product lines with pure:: variants," in *Proceedings of the 16th International Software Product Line Conference-Volume 2*, 2012, pp. 255-255.

[106] C. W. Krueger, "The biglever software gears unified software product line engineering framework," in *Software Product Line Conference, 2008. SPLC'08. 12th International*, 2008, pp. 353-353.

[107] R. Angles, P. Ramadour, C. Cauvet, and S. Rodier, "V-BPMI: A variability-oriented framework for web-based business processes modeling and implementation," in *Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on*, 2013, pp. 1-11.

[108] H. Andersson, E. Herzog, and J. Ölvander, "Experience from model and software reuse in aircraft simulator product line engineering," *Information and Software Technology,* vol. 55, pp. 595-606, 3// 2013.

[109] J. Xiong, Y. Hu, G. Li, R. Tang, and Z. Fan, "Metadata distribution and consistency techniques for large-scale cluster file systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. 22, pp. 803-816, 2011.

[110] P. K. Kumar and R. Kanagaraj, "Harmonizing the business process customization using ontology," in *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, 2012, pp. 788-791.

[111] Y. Huang, Z. Feng, K. He, and Y. Huang, "Ontology-based configuration for service-based business process model," in *Services Computing (SCC), 2013 IEEE International Conference on*, 2013, pp. 296-303.

[112] D. Schleicher, T. Anstett, F. Leymann, and R. Mietzner, "Maintaining compliance in customizable process models," in *On the Move to Meaningful Internet Systems: OTM 2009*, ed: Springer, 2009, pp. 60-75.

[113] M. V. Hecht, E. K. Piveta, M. S. Pimenta, and R. T. Price, "Aspect-oriented code generation," *Simpsio Brasileiro de Engenharia de Software*, 2005.

[114] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, pp. 45-77, 2007.

[115] G. Goldkuhl, "The research practice of practice research: theorizing and situational inquiry," *Systems, Signs & Actions*, vol. 5, pp. 7-29, 2011.

[116] H. Van de Ven Andrew, "Engaged scholarship: A guide for organizational and social research," 2007.

[117] G. Goldkuhl, "From action research to practice research," *Australasian Journal of Information Systems*, vol. 17, 2012.

[118] C. Cassell and P. Johnson, "Action research: Explaining the diversity," *Human relations*, vol. 59, pp. 783-814, 2006.

[119] P. Y. Martin and B. A. Turner, "Grounded theory and organizational research," *The journal of applied behavioral science*, vol. 22, pp. 141-157, 1986.

[120] R. K. Yin, "Discovering the future of the case study method in evaluation research," *Evaluation practice*, vol. 15, pp. 283-290, 1994.

[121] S. Gregor and D. Jones, "The anatomy of a design theory," *Journal of the Association for Information Systems*, vol. 8, p. 312, 2007.

[122] A. Hevner and S. Chatterjee, "Design science research in information systems," in *Design research in information systems*, ed: Springer, 2010, pp. 9-22.

[123] K. Lewin, "Action research and minority problems," *Journal of social issues*, vol. 2, pp. 34-46, 1946.

[124] R. N. Rapoport, "Three dilemmas in action research: with special reference to the Tavistock experience," *Human relations*, vol. 23, pp. 499-513, 1970.

[125] J. Iivari and J. Venable, "Action research and design science research-Seemingly similar but decisively dissimilar," in *ECIS*, 2009, pp. 1642-1653.

[126] J. Nandhakumar, M. Rossi, and J. Talvinen, "The dynamics of contextual forces of ERP implementation," *The Journal of Strategic Information Systems*, vol. 14, pp. 221-242, 2005.

[127] C. Urquhart, H. Lehmann, and M. D. Myers, "Putting the 'theory'back into grounded theory: guidelines for grounded theory studies in information systems," *Information systems journal*, vol. 20, pp. 357-381, 2010.

[128] W. J. Orlikowski and J. J. Baroudi, "Studying information technology in organizations: Research approaches and assumptions," *Information systems research*, vol. 2, pp. 1-28, 1991.

[129] R. E. Stake, *Multiple case study analysis*: Guilford Press, 2013.

[130] J. Venable, J. Pries-Heje, and R. Baskerville, "A comprehensive framework for evaluation in design science research," in *International Conference on Design Science Research in Information Systems*, 2012, pp. 423-438.

[131] R. H. Von Alan, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, vol. 28, pp. 75-105, 2004.

[132] P. Petkov, "Assessing and analysing data quality in service oriented architectures; developing a data quality process," Dublin City University, 2016.

[133] C. Okoli and K. Schabram, "A guide to conducting a systematic literature review of information systems research," 2010.

[134] Ł. Ostrowski, "Detailed Design Science Research and Its Impact on the Quality of Design Artefacts," in *European Design Science Symposium*, 2011, pp. 60-70.

[135] C. Pahl, N. Mani, and M.-X. Wang, "A Domain-Specific Model for Data Quality Constraints in Service Process Adaptations," in *Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2013, Málaga, Spain, September 11-13, 2013, Revised Selected Papers*, C. Canal and M. Villari, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 303-317.

[136] N. Mani, M. Helfert, and C. Pahl, "Business Process Model Customisation using Domain-driven Controlled Variability Management and Rule Generation," *International Journal on Advances in Software*, vol. 9, pp. 179 - 190, 2016.

[137] N. Mani and C. Pahl, "Controlled variability management for business process model constraints," in *ICSEA 2015, The Tenth International Conference on Software Engineering Advances*, 2015, p. 445 to 450.

[138] C. Pahl and N. Mani, "Managing quality constraints in technology-managed learning content processes," in *EdMedia'2014 Conference on Educational Media and Technology*, 2014.

[139] N. Mani, M. Helfert, and C. Pahl, "A Framework for Generating Domain-specific Rule for Process Model Customisation," presented at the International Conference on Computer-Human Interaction Research and Applications (CHIRA),, Funchal, Maderia- Portugal, 2017.

[140] N. Mani, M. Helfert, and C. Pahl, "Domain-specific Generation Using Variability for Business Process Model Constraint," presented at the 21st International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Marseille, France, 2017.

[141] A. Miller, "Engineering design: its importance for software," *IEEE Potentials,* vol. 8, pp. 14-16, 1989.

[142] M. D. Del Fabro and P. Valduriez, "Semi-automatic model integration using matching transformations and weaving models," in *Proceedings of the 2007 ACM symposium on Applied computing*, 2007, pp. 963-970.

[143] C. Pahl, "Semantic model-driven architecting of service-based software systems," *Information and software Technology*, vol. 49, pp. 838-850, 2007.

[144] L. Geyer and M. Becker, "On the influence of variabilities on the application-engineering process of a product family," in *Software Product Lines,* ed: Springer, 2002, pp. 1-14.

[145] J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, J. Obbink, and K. Pohl, "Variability Issues in Software Product Lines," *Software Product-Family Engineering,* pp. 303-338, 2002.

[146] D. Batory, *Feature models, grammars, and propositional formulas*: Springer, 2005.

[147] K. Czarnecki, "Generative programming," *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen,* p. 15.

[148] M. Fowler, *UML distilled: a brief guide to the standard object modeling language*: Addison-Wesley Professional, 2004.

[149] B. M. Kadhim and W. M. Waite, "Maptool—supporting modular syntax development," in *Compiler Construction*, 1996, pp. 268-280.

[150] D. S. Wile, "Abstract syntax from concrete syntax," in *Proceedings of the 19th international conference on Software engineering*, 1997, pp. 472-480.

[151] J. L. Overbey and R. E. Johnson, "Generating rewritable abstract syntax trees," in *Software Language Engineering,* ed: Springer, 2008, pp. 114-133.

[152] F. Jouault, J. Bézivin, and I. Kurtev, "TCS:: a DSL for the specification of textual concrete syntaxes in model engineering," in *Proceedings of the 5th international conference on Generative programming and component engineering*, 2006, pp. 249-254.

[153] W. L. G. Riedewald and M. Stoya, "Semantics-preserving migration of semantic rules after left recursion removal in attribute grammars."

[154] J. J. Gough and K. J. Gough, *Compiling for the. Net Common Language Runtime*: Prentice Hall PTR, 2001.

[155] W. May, J. Alferes, and R. Amador, "Active Rules in the Semantic Web: Dealing with Language Heterogeneity," in *Rules and Rule Markup Languages for the Semantic Web*. vol. 3791, A. Adi, S. Stoutenburg, and S. Tabet, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 30-44.

[156] J. D. Poole, "Model-driven architecture: Vision, standards and emerging technologies," in *Workshop on Metamodeling and Adaptive Object Models, ECOOP*, 2001.

[157] J. Cano, G. Delaval, and E. Rutten, "Coordination of ECA Rules by Verification and Control," in *Coordination Models and Languages,* 2014, pp. 33-48.

[158] A. Ranta, "Grammatical framework," *Journal of Functional Programming,* vol. 14, pp. 145-189, 2004.

[159] E. Visser, *Syntax definition for language prototyping*: Eelco Visser, 1997.

[160] J. Heering, P. R. H. Hendriks, P. Klint, and J. Rekers, "The syntax definition formalism sdf—reference manual—," *ACM Sigplan Notices,* vol. 24, pp. 43-75, 1989.

[161] Y. Wei, S. Zhang, and J. Cao, "Coordination among multi-agents using process calculus and ECA rule," in *Engineering and Deployment of Cooperative Information Systems*, ed: Springer, 2002, pp. 456-465.

[162] J. Bailey, G. Papamarkos, A. Poulovassilis, and P. T. Wood, "An event-condition-action language for XML," in *Web Dynamics*, ed: Springer, 2004, pp. 223-248.

[163] J. Bailey, A. Poulovassilis, and P. T. Wood, "Analysis and optimisation of event-condition-action rules on XML," *Computer Networks,* vol. 39, pp. 239-259, 2002.

[164] G. Papamarkos, A. Poulovassilis, and P. T. Wood, "Event-condition-action rule languages for the semantic web," in *Proceedings of the First International Conference on Semantic Web and Databases*, 2003, pp. 294-312.

[165] N. Koch, A. Knapp, G. Zhang, and H. Baumeister, "UML-based web engineering," in *Web Engineering: Modelling and Implementing Web Applications*, ed: Springer, 2008, pp. 157-191.

[166] S. Ceri, A. Bongio, P. Fraternali, M. Brambilla, S. Comai, and M. Matera, *Morgan Kaufmann series in data management systems: Designing data-intensive Web applications*: Morgan Kaufmann, 2003.

[167] D. M. Groenewegen, Z. Hemel, L. C. Kats, and E. Visser, "WebDSL: a domain-specific language for dynamic web applications," in *Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications*, 2008, pp. 779-780.

[168] M. Ribarić, D. Gašević, M. Milanović, A. Giurca, S. Lukichev, and G. Wagner, "Model-Driven engineering of rules for web services," in *Generative and Transformational Techniques in Software Engineering II*, ed: Springer, 2008, pp. 377-395.

[169] H. Boley, S. Tabet, and G. Wagner, "Design rationale of RuleML: A markup language for semantic web rules," in *Proceedings of the First International Conference on Semantic Web Working*, 2001, pp. 381-401.

[170] G. Wagner, A. Giurca, and S. Lukichev, "A usable interchange format for rich syntax rules integrating OCL, RuleML and SWRL," *Proc. of WSh. Reasoning on the Web,* 2006.

[171] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A semantic web rule language combining OWL and RuleML," *W3C Member submission,* vol. 21, p. 79, 2004.

[172] Z. Meliš, J. Žáček, and F. Huňka, "Comparison of MDA and DSM Technologies for the REA Ontology Model Creation," *e-Informatica Software Engineering Journal,* vol. 7, 2013.

[173] T. Mens and P. Van Gorp, "A Taxonomy of Model Transformation," *Electronic Notes in Theoretical Computer Science,* vol. 152, pp. 125-142, 3/27/ 2006.

[174] K. Czarnecki and S. Helsen, "Classification of model transformation approaches," in *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, 2003, pp. 1-17.

[175] J. B. Warmer and A. G. Kleppe, "The object constraint language: Precise modeling with uml (addison-wesley object technology series)," 1998.

[176] D. L. McGuinness and F. Van Harmelen, "OWL web ontology language overview," *W3C recommendation,* vol. 10, p. 2004, 2004.

[177] I. S. Bajwa, B. Bordbar, and M. G. Lee, "OCL constraints generation from natural language specification," in *Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International*, 2010, pp. 204-213.

[178] I. S. Bajwa, M. G. Lee, and B. Bordbar, "SBVR Business Rules Generation from Natural Language Specification," in *AAAI spring symposium: AI for business agility*, 2011, pp. 2-8.

[179] Ö. Ö. Tanrıöver and S. Bilgen, "A framework for reviewing domain specific conceptual models," *Computer Standards & Interfaces,* vol. 33, pp. 448-464, 2011.

[180] M. Völter, T. Stahl, J. Bettin, A. Haase, and S. Helsen, *Model-driven software development: technology, engineering, management*: John Wiley & Sons, 2013.

[181] J. Žácek and F. Hunka, "Reusable Object-Oriented Model," *e-Informatica Software Engineering Journal,* vol. 7, 2013.

[182] R. M. Soley and C. M. Stone, *Object management architecture guide: revision 2.0*: Object Management Group, Incorporated, 1993.

[183] F. Truyen, "The fast guide to model driven architecture the basics of model driven architecture," 2006.

[184] F. Truyen, "The Fast Guide to Model Driven Architecture The Basics of Model Driven Architecture," *URL: http://www. omg. org/mda/presentations. htm, January,* 2006.

[185] M. Belaunde, C. Casanave, D. DSouza, K. Duddy, W. El Kaim, A. Kennedy, *et al.*, "MDA Guide Version 1.0," 2003.

[186] J. P. Almeida, R. Dijkman, M. Van Sinderen, and L. F. Pires, "On the notion of abstract platform in mda development," in *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International*, 2004, pp. 253-263.

[187] D. Wagelaar and V. Jonckers, "Explicit platform models for MDA," in *International Conference on Model Driven Engineering Languages and Systems*, 2005, pp. 367-381.

[188] J. Bézivin, "On the unification power of models," *Software & Systems Modeling,* vol. 4, pp. 171-188, 2005.

[189] T. O. M. G. OMG, "MOF 2.0/XMI Mapping Specification," OMG Specification (formal/05-09-01)2005.

[190] L. A. Lanceloti, J. C. Maldonado, I. Gimenes, and E. A. Oliveira Jr, "Smartyparser: a xmi parser for uml-based software product line variability models," in *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, 2013, p. 10.

[191] R. Soley, "Model driven architecture," *OMG white paper,* vol. 308, p. 308, 2000.

[192] S. J. Mellor, K. Scott, A. Uhl, and D. Weise, "Model-driven architecture," in *Advances in Object-Oriented Information Systems*, ed: Springer, 2002, pp. 290-297.

[193] G. A. Lewis, B. C. Meyers, and K. Wallnau, "Workshop on Model-Driven Architecture and Program Generation," CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST2006.

[194] A. Computing, "An architectural blueprint for autonomic computing," 2006.

[195] M. Torres and G. H. Alférez, "Software Architecture Evolution in the Open World through Genetic Algorithms," in *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, 2014, p. 1.

[196] Y. Yu, J. C. S. do Prado Leite, A. Lapouchnian, and J. Mylopoulos, "Configuring features with stakeholder goals," in *Proceedings of the 2008 ACM symposium on Applied computing*, 2008, pp. 645-649.

[197] S. Liaskos, A. Lapouchnian, Y. Yu, E. Yu, and J. Mylopoulos, "On goal-based variability acquisition and analysis," in *14th IEEE International Requirements Engineering Conference (RE'06)*, 2006, pp. 79-88.

[198] E. Papulova and Z. Papulova, "Competitive strategy and competitive advantages of small and midsized manufacturing enterprises in Slovakia," *E-Leader, Slovakia,* 2006.

[199] I. 9241-11, "Ergonomic requirements for office work with visual display terminals (VDTs)–Part 11," *International Organization for Standardization,Geneva,* 1998.

[200] A. Dix, *Human-computer interaction*: Springer, 2009.

[201] N. Tselios, N. Avouris, and V. Komis, "The effective combination of hybrid usability methods in evaluating educational applications of ICT: Issues and challenges," *Education and Information Technologies,* vol. 13, pp. 55-76, 2008.

[202] J. Sauro, "Measuring Usability with the System Usability Scale (SUS)," 2011.

[203] B. Kaplan and D. Duchon, "Combining qualitative and quantitative methods in information systems research: a case study," *MIS quarterly,* pp. 571-586, 1988.

[204] V. R. Basili, "The role of experimentation in software engineering: past, current, and future," in *Proceedings of the 18th international conference on Software engineering*, 1996, pp. 442-449.

[205] J. Lazar, J. H. Feng, and H. Hochheiser, *Research methods in human-computer interaction*: John Wiley & Sons, 2010.

[206] J. R. Lewis, "IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use," *International Journal of Human-Computer Interaction,* vol. 7, pp. 57-78, 1995.

[207] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Intl. Journal of Human–Computer Interaction,* vol. 24, pp. 574-594, 2008.

[208] R. Molich and J. Nielsen, "Improving a human-computer dialogue," *Communications of the ACM,* vol. 33, pp. 338-348, 1990.

[209] J. Nielsen, "Heuristic evaluation," *Usability inspection methods,* vol. 17, pp. 25-62, 1994.

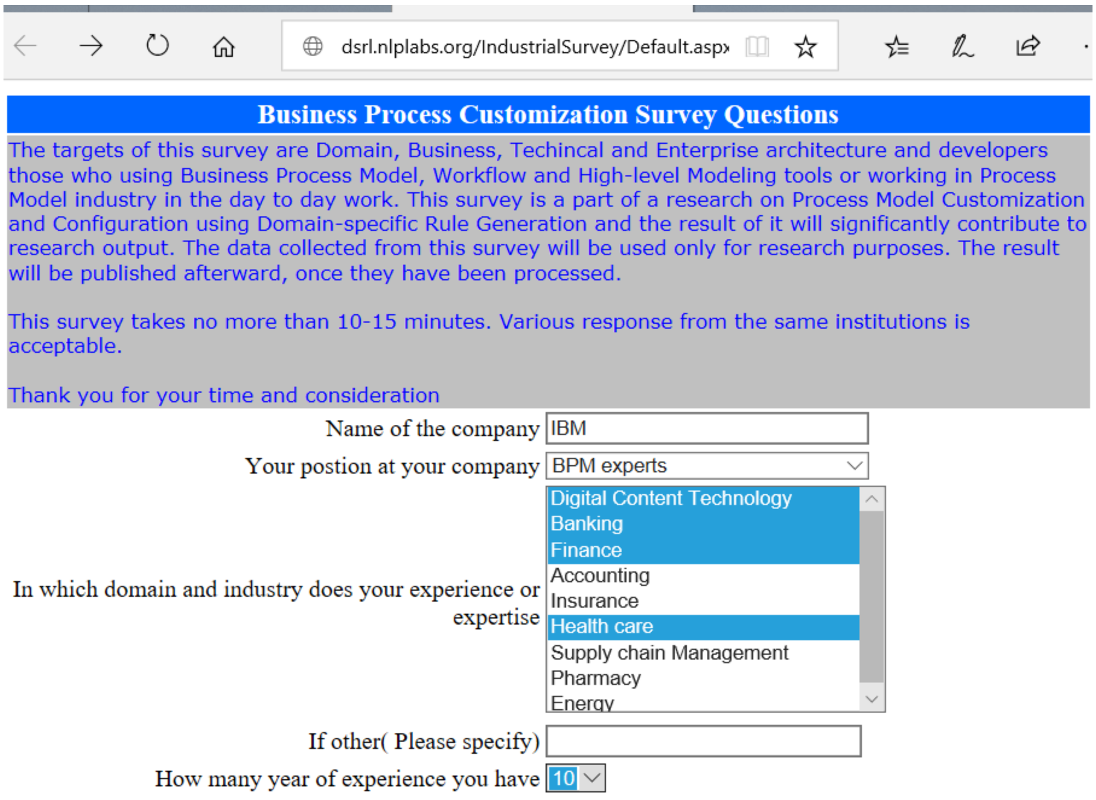
[210] B. Tognazzini, *TOG on Interface*: Addison-Wesley Longman Publishing Co., Inc., 1992.

[211] J. Preece and H. D. Rombach, "A taxonomy for combining software engineering and human-computer interaction measurement approaches: towards a common framework," *International journal of human-computer studies*, vol. 41, pp. 553-583, 1994.

APPENDIX A

INDUSTRIAL SURVEY FOR PROCESS CUSTOMIZATION RESEARCH

A 1. BPM Survey Questions



Business Process Customization Survey Questions

The targets of this survey are Domain, Business, Techincal and Enterprise architecture and developers those who using Business Process Model, Workflow and High-level Modeling tools or working in Process Model industry in the day to day work. This survey is a part of a research on Process Model Customization and Configuration using Domain-specific Rule Generation and the result of it will significantly contribute to research output. The data collected from this survey will be used only for research purposes. The result will be published afterward, once they have been processed.

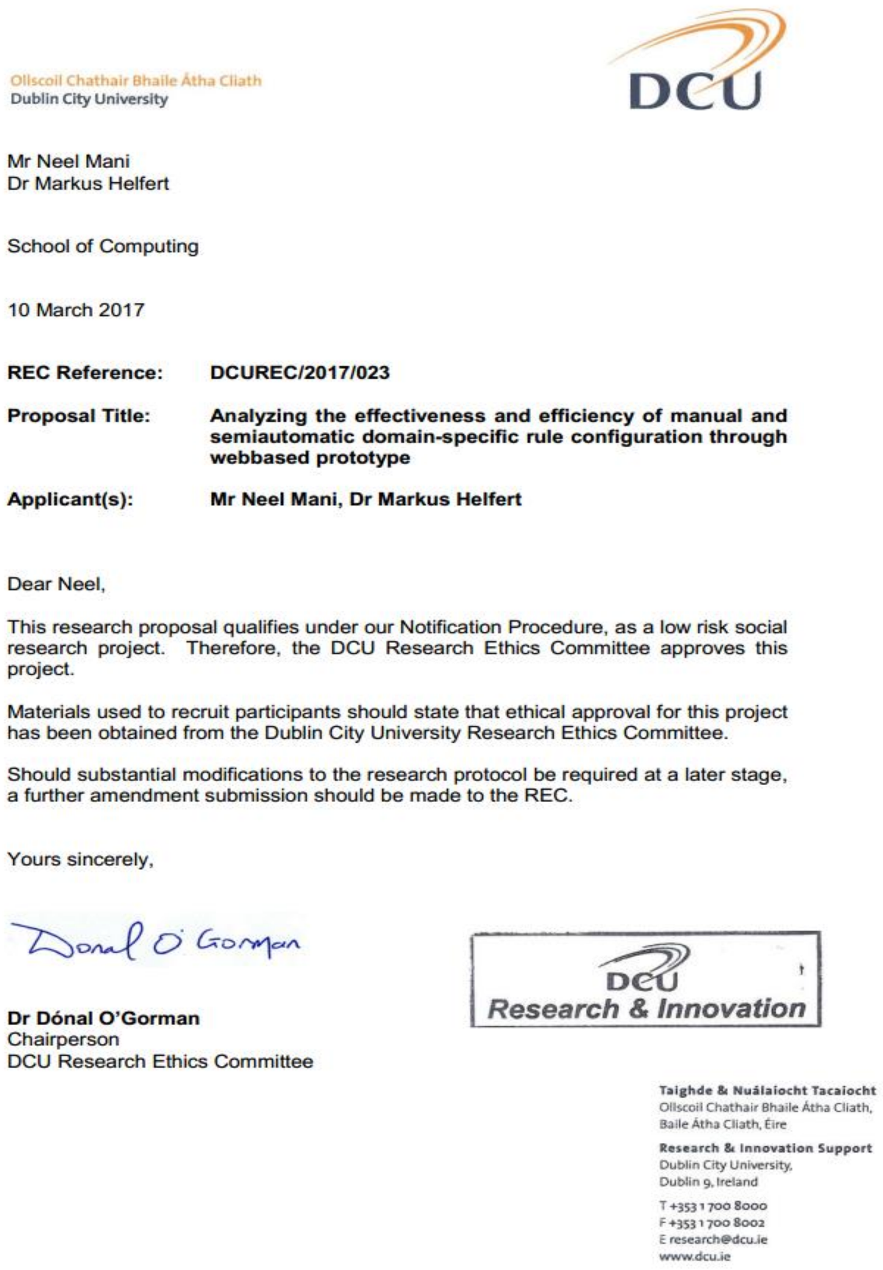
This survey takes no more than 10-15 minutes. Various response from the same institutions is acceptable.

Thank you for your time and consideration

Name of the company	IBM
Your postion at your company	BPM experts
In which domain and industry does your experience or expertise	[✓ Digital Content Technology] [✓ Banking] [✓ Finance] Accounting Insurance [✓ Health care] Supply chain Management Pharmacy Energy
If other(Please specify)	
How many year of experience you have	10

Figure A.1: Research Applicability Survey on Process Model Industry Participant

A 2. DCU Research Ethics Committee Approval



Ollscoil Chathair Bhaile Átha Cliath
Dublin City University

Mr Neel Mani
Dr Markus Helfert

School of Computing

10 March 2017

REC Reference:	**DCUREC/2017/023**
Proposal Title:	**Analyzing the effectiveness and efficiency of manual and semiautomatic domain-specific rule configuration through webbased prototype**
Applicant(s):	**Mr Neel Mani, Dr Markus Helfert**

Dear Neel,

This research proposal qualifies under our Notification Procedure, as a low risk social research project. Therefore, the DCU Research Ethics Committee approves this project.

Materials used to recruit participants should state that ethical approval for this project has been obtained from the Dublin City University Research Ethics Committee.

Should substantial modifications to the research protocol be required at a later stage, a further amendment submission should be made to the REC.

Yours sincerely,

Dónal O' Gorman

Dr Dónal O'Gorman
Chairperson
DCU Research Ethics Committee

DCU Research & Innovation

Taighde & Nuálaíocht Tacaíocht
Ollscoil Chathair Bhaile Átha Cliath,
Baile Átha Cliath, Éire

Research & Innovation Support
Dublin City University,
Dublin 9, Ireland

T +353 1 700 8000
F +353 1 700 8002
E research@dcu.ie
www.dcu.ie

A 3. Research Work on Process Model Customization

Table A1: Compendium Research Work on Process Model Customization

Authors	Comparisons	Summary	Solution	Implementation component	Domain
Liang et al. [109] [114]	Ontology	Service-based for business processes customization 1. Semantic inconsistencies 2. Behavioral mismatches 3. Misaligned rendezvous		OWL-BPC[17]	
Kumar and Kanagaraj [110]	Ontology	Knowledge based human semantic web for customizing process model. The framework with both the customization detection and enactment; the two techniques are used to solve both the semantic and behavioral mismatch.	Dynamic adaptation and accuracy	OWL-BPC	
Huang et al.[111]	Ontology	The dependencies are captured between variation points via variation point ontology and utilize SWRL rules encoded in ontology		SWRL	Urban logistics distribution
La Rosa and van der Aalst [84]	Questionnaire driven	Questionnaire models capture system variability based on questionnaire models that include order dependencies and domain constraints			Film production

[17]Web Ontology Language-Business Process Customization (OWL-BPC)

Authors	Comparisons	Summary	Solution	Implementation component	Domain
Kumar and Yao [3]	Rule templates	Template and configuration rules for customizing the process model Designing flexible business processes based on combining process, generic process template with business rules. Template- and rule-based Approach		Java and used the Drools Rule Language (DRL) conflict resolution component Variant configuration algorithm Transformation between a Process template in BPEL.	Insurance process
Asadi et al. [5]	Feature model	Post validation Inconsistencies resulting from the customization of business process models, which were originally derived from references process through configuration procedures.	Variability complexity, Modeling complexity, Delta requirements, and Customization validation	Propositional logic, first-order logic and temporal logic, Traversal algorithms to compare process graphs, Compare execution traces of process models and Description logics (DL)	Health care
Alférez et al. [6]	Feature model	WS-BPEL and service activation & deactivation MoRE-WS tool activates and deactivates features in a	Automatizes the creation of variability model configurations at design and run time.	Constraint logic programming, WS-BPEL code	Online book

		variability model at runtime.	Dynamic adaptations on an enterprise orchestration engine		

APPENDIX B

RULE LANGUAGE AND SYNTAX

B1. Rule Language Concrete Syntax

In this appendix, we give the concrete syntax of DSRL (discussed in Chapter 6) that has been implemented in form of XML. The abstract syntax and language description are being used to formulate and construct the DSR representing the configured rule of gic:Extraction with document upload with different validation level of rule.

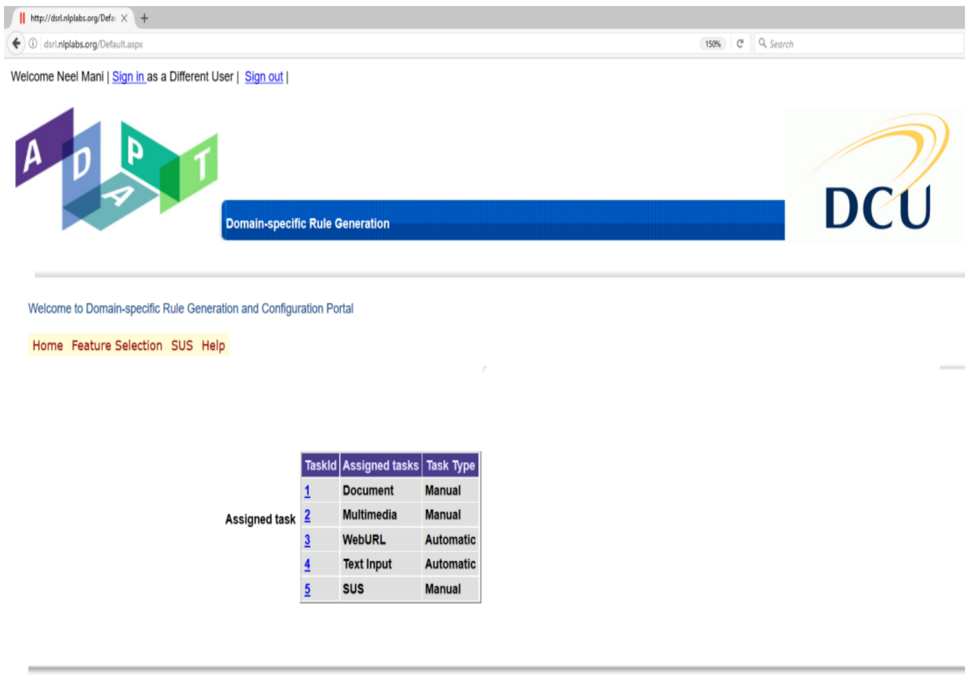
```
<DSR>
  <Head>
  </Head>
  <Domain>Globic Intellegent Content</Domain>
  <FeatureModelName="File">
    <FeatureType>Mandatory</FeatureType>
    <Inherited>gic:Extraction</Inherited>
  </FeatureModelName>
  <Functions>
    <LevelId="1">
      <Functionname="FileSizeCal">
        <Paramsid="1"type="String">
          <paramName>FilePath</paramName>
          </Params>
          <Return>
            <Var>$returnParam</Var>
            <DataType>Integer</DataType>
          </Return>
          </Function>
          <Functionname="FileTypeCheck">
            <Paramsid="1"type="String">
              <paramName>FileType</paramName>
              </Params>
              <Paramsid="2"type="String">
                <paramName>FileExtenstion</paramName>
                </Params>
                <Return>
                  <Var>$returnParam</Var>
                  <DataType>Boolean</DataType>
                </Return>
                </Function>
                </Level>
                <LeveId="2">
                  <Functionname="DocSize">
                    <Paramsid="1"type="String">
                      <paramName>FileName</paramName>
                      </Params>
          <Return>
           <Var>$returnParam</Var>
           <DataType>Integer</DataType>
          </Return>
          </Function>
          <Functionname="IsDocument">
           <Paramsid="1"type="String">
            <paramName>FileName</paramName>
            </Params>
            <Paramsid="2"type="String">
             <paramName>SourceLanguage</paramName>
             </Params>
             <Return>
              <Var>$returnParam</Var>
              <DataType>Boolean</DataType>
             </Return>
             </Function>
             </Leve>
            </Functions>
<Body>

 <ProcessName="gic:Extraction">
  <State.Transitions>
   <TransitionDisplayName="T-LG-EX"

    </Transition>
   </State>
   </State.Transitions>

  <EventON="FileUpload">
   <Rulename="Level-1:ValidUpload">
    <VarId="1">
     <Name>Accept File Extenstion</Name>
     <DataType>String</DataType>
     </Var>
     <VarId="2">
      <Name>Max File Size(MB)</Name>
      <DataType>Integer</DataType>
      </Var>
      <VarId="3">
       <Name>Upload Directory</Name>
       <DataType>String</DataType>
       </Var>
       <Condition>
        <Rel>UploadCheck</Rel>
        <Param>
         <VarId="1">
          .pdf,.txt,.html</Var>
          <VarId="2">
           10</Var>
           <VarId="3">
            http://dsrl.nlplabs.org/DSRRepository/Upload</Var>
           </Param>
        <IF>
         <Check>FileTypeCheck==True</Check>
         <Action>
          <Do>ValidDocument()</Do>
         </Action>
        </IF>
```

```
                                        <Else>
                                          <Message>File type is not valid!</Message>
                                        </Else>
                                      </Condition>
                                      </Rule>
                                      <Rulename="Level-2:ValidDocument">
                                        <VarId="1">
                                          <Name>File Path</Name>
                                          <DataType>String</DataType>
                                          </Var>
                                          <VarId="2">
                                            <Name>Type of File</Name>
                                            <DataType>string</DataType>
                                            </Var>
                                            <VarId="3">
                                              <Name>File Extenstion</Name>
                                              <DataType>string</DataType>
                                              </Var>
                                              <VarId="4">
                                                <Name>File Name</Name>
                                                <DataType>String</DataType>
                                                </Var>
                                                <VarId="5">
                                                  <Name>Source Language</Name>
                                                  <DataType>String</DataType>
                                                  </Var>
                                                  <Condition>
                                                    <Rel>ValidDocument</Rel>
                                                    <Param>
                                                      <VarId="1">
                                                        C:\SourceFile\</Var>
                                                        <VarId="2">
                                                          Text File</Var>
                                                          <VarId="3">
                                                            .txt</Var>
                                                            <VarId="4">
                                                              English.txt</Var>
                                                              <VarId="5">
                                                                English</Var>
                                                              </Param>
                                                    <IF>
                                                      <Check>ValidFileSize==True</Check>
                                                      <Action>
                                                        <Do>Next</Do>
                                                      </Action>
                                                    </IF>
                                                    <Else>
                                                      <Message>File name is not valid!</Message>
                                                    </Else>
                                                  </Condition>
                                                  </Rule>
                                                  </Event>
                                                  </Process>
                                                </Body>
                        </DSR>
```

APPENDIX C

USER EXPERIENCE EVALUATION

C 1. Tasks Assigned on Dashboard



http://dsrl.nlplabs.org/Defa

dsrl.nlplabs.org/Default.aspx

Welcome Neel Mani | Sign in as a Different User | Sign out |

Domain-specific Rule Generation

Welcome to Domain-specific Rule Generation and Configuration Portal

Home Feature Selection SUS Help

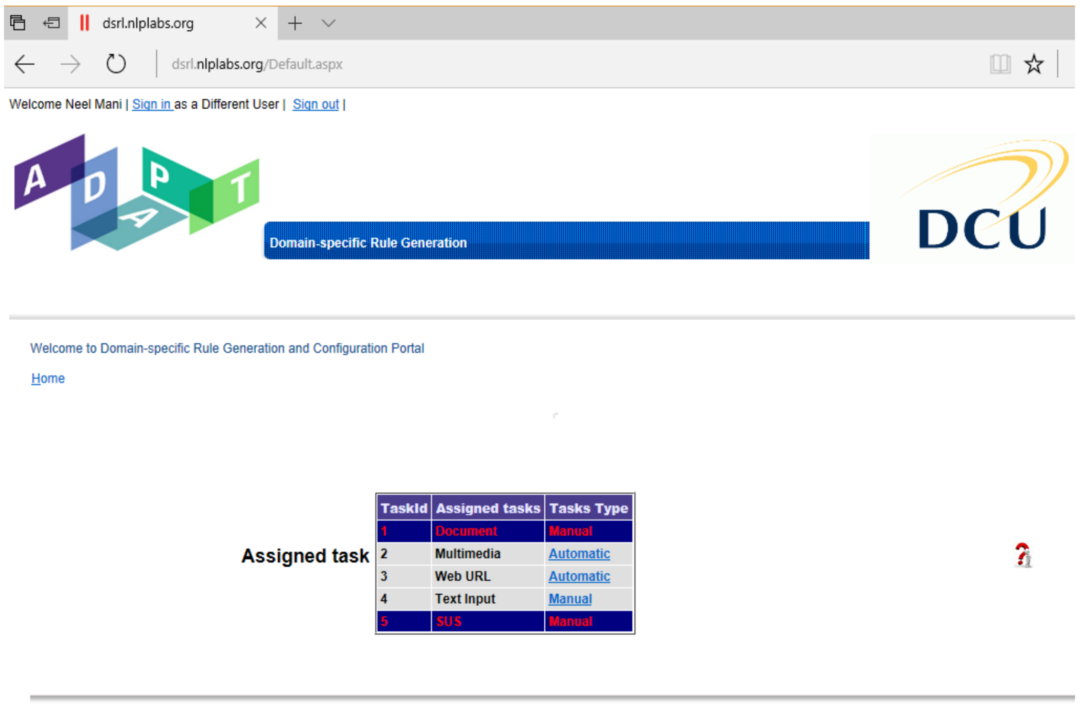
Assigned task

TaskId	Assigned tasks	Task Type
1	Document	Manual
2	Multimedia	Manual
3	WebURL	Automatic
4	Text Input	Automatic
5	SUS	Manual

Figure C.1: The Dashboard of the Participant

C 2. Tasks Finished on Dashboard

The finished tasks are illustrated with red color and mouse click is disabled on the dashboard



dsrl.nlplabs.org

dsrl.nlplabs.org/Default.aspx

Welcome Neel Mani | Sign in as a Different User | Sign out |

Domain-specific Rule Generation

Welcome to Domain-specific Rule Generation and Configuration Portal

Home

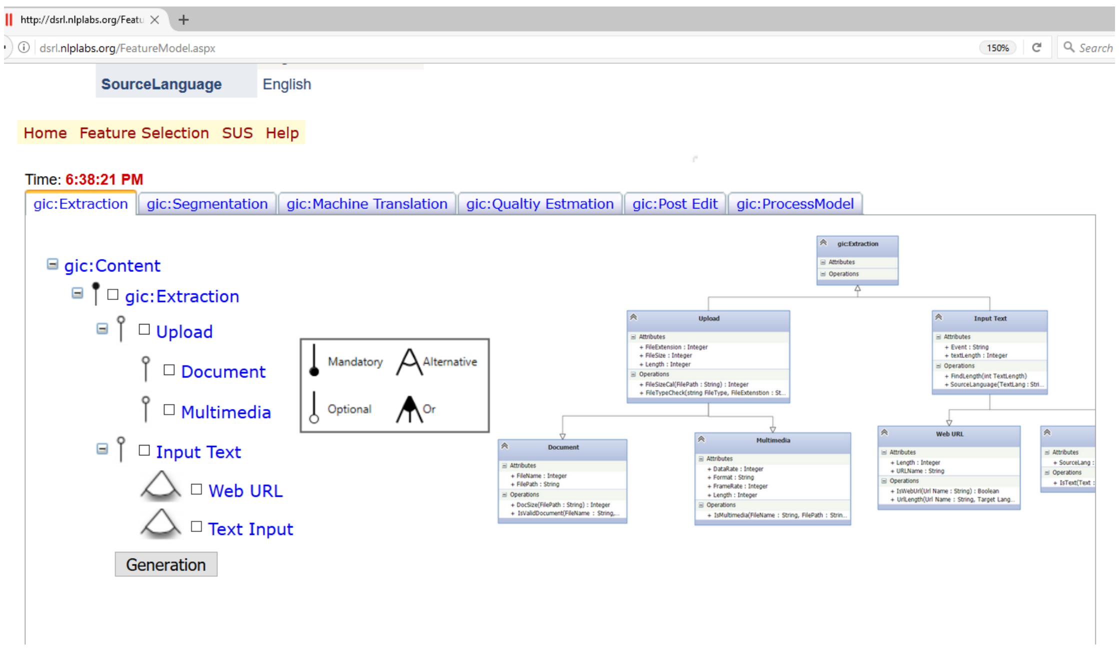
Assigned task

TaskId	Assigned tasks	Tasks Type
1	Document	Manual
2	Multimedia	Automatic
3	Web URL	Automatic
4	Text Input	Manual
5	SUS	Manual

Figure C.2: The Dashboard of the Participant after Task Finished

C 3. Feature Model Selection Interface

Where end-users select their desire tasks and requirement for DSR generation.



http://dsrl.nlplabs.org/Featu

dsrl.nlplabs.org/FeatureModel.aspx

150%

Search

SourceLanguage English

Home Feature Selection SUS Help

Time: 6:38:21 PM

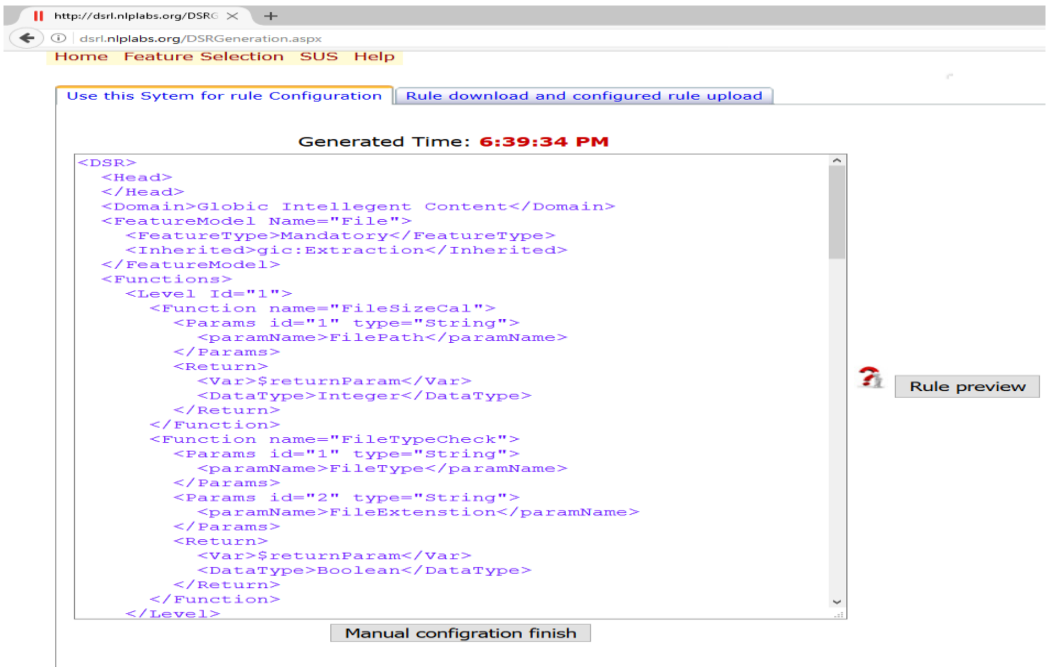
gic:Extraction | gic:Segmentation | gic:Machine Translation | gic:Qualtiy Estmation | gic:Post Edit | gic:ProcessModel

- gic:Content
  - ☐ gic:Extraction
    - ☐ Upload
      - ☐ Document
      - ☐ Multimedia
    - ☐ Input Text
      - ☐ Web URL
      - ☐ Text Input

Generation

Figure C.3: Feature Selection from Feature Model Interface

C 4. Interface of Generated Rule for Manual Configuration

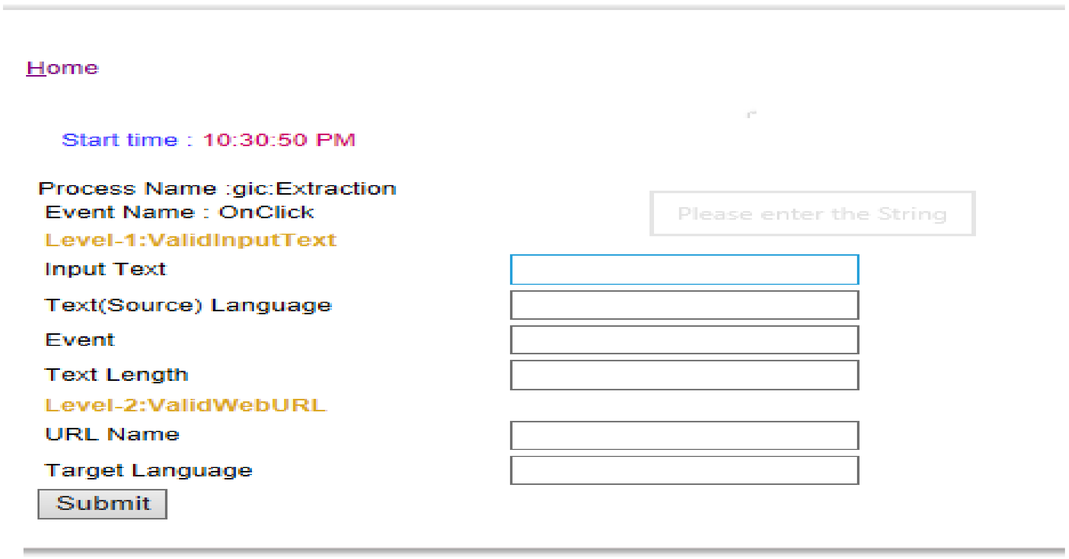

http://dsrl.nlplabs.org/DSRG
dsrl.nlplabs.org/DSRGeneration.aspx
Home Feature Selection SUS Help
Use this Sytem for rule Configuration
Rule download and configured rule upload
Rule preview
Manual configration finish

Generated Time: 6:39:34 PM

```
<DSR>
  <Head>
  </Head>
  <Domain>Globic Intellegent Content</Domain>
  <FeatureModel Name="File">
    <FeatureType>Mandatory</FeatureType>
    <Inherited>gic:Extraction</Inherited>
  </FeatureModel>
  <Functions>
    <Level Id="1">
      <Function name="FileSizeCal">
        <Params id="1" type="String">
          <paramName>FilePath</paramName>
        </Params>
        <Return>
          <Var>$returnParam</Var>
          <DataType>Integer</DataType>
        </Return>
      </Function>
      <Function name="FileTypeCheck">
        <Params id="1" type="String">
          <paramName>FileType</paramName>
        </Params>
        <Params id="2" type="String">
          <paramName>FileExtenstion</paramName>
        </Params>
        <Return>
          <Var>$returnParam</Var>
          <DataType>Boolean</DataType>
        </Return>
      </Function>
    </Level>
```

Figure C.4: Manual Configuration of Interface

C 5. Interface of End-user for Semi-Automatic Configuration



Start time : 10:30:50 PM

Process Name :gic:Extraction
Event Name : OnClick

Level-1:ValidInputText

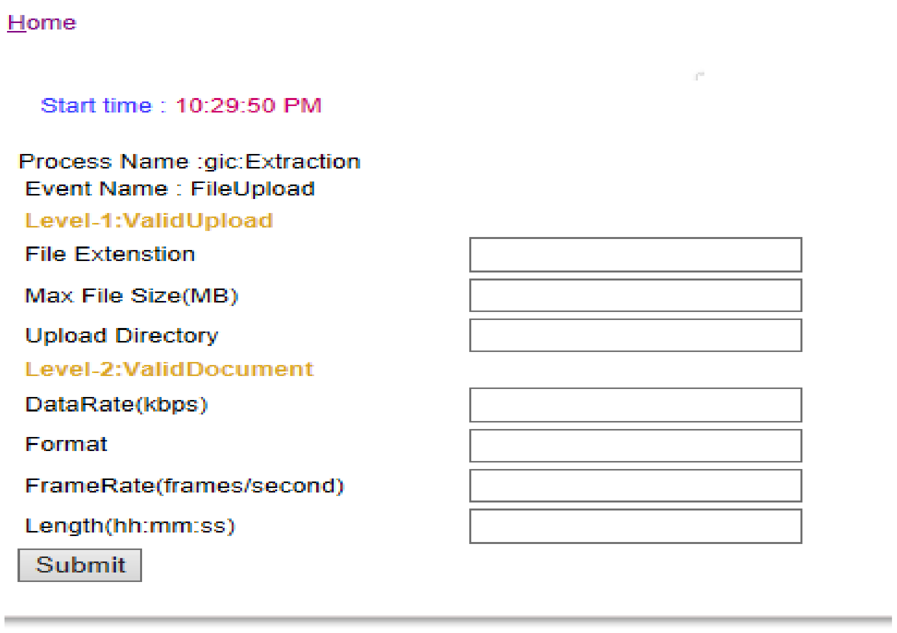
Input Text	
Text(Source) Language	
Event	
Text Length	

Level-2:ValidWebURL

URL Name	
Target Language	

Figure C.5: Semi-Automatic Configuration of Web URL

C 6. Semi-automatic Configuration of Multimedia


Home

Start time : 10:29:50 PM

Process Name :gic:Extraction
Event Name : FileUpload

Level-1:ValidUpload

File Extenstion	
Max File Size(MB)	
Upload Directory	

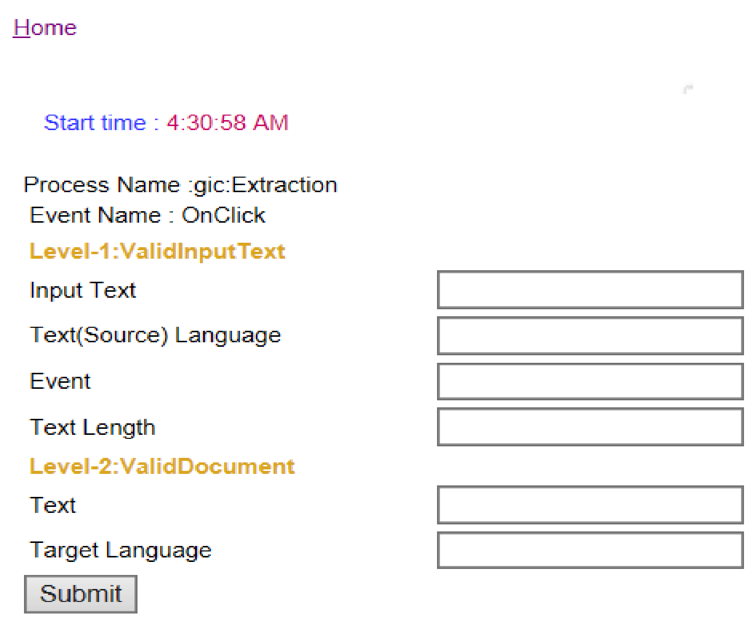
Level-2:ValidDocument

DataRate(kbps)	
Format	
FrameRate(frames/second)	
Length(hh:mm:ss)	

Figure C.6: Semi-Automatic Configuration of Multimedia

In Figure C.6 illustrated, the process model gic:`Extraction` as *Process Name* and *FileUpload* is the *Event Name* for Multimedia process activities or sub-process. This event validated on two level: *ValidaUpload* and ValidDocument. The *ValidaUpload* contains 3 domain constraints to configure and ValidDocument contains 4 domain constraints. Each user has to configure the constraints value according to their requirement or need.

C 7. Semi-automatic Configuration of Text Input



Home

Start time : 4:30:58 AM

Process Name :gic:Extraction
Event Name : OnClick

Level-1:ValidInputText

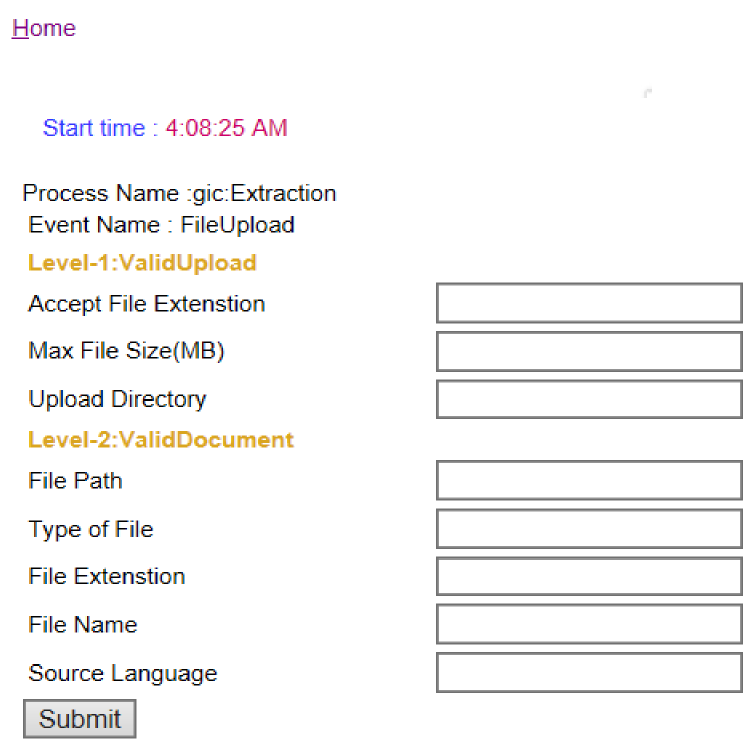
Input Text	
Text(Source) Language	
Event	
Text Length	

Level-2:ValidDocument

Text	
Target Language	

Figure C.7: Semi-Automatic Configuration of Text Input

C 8. Semi-automatic Configuration of Document



Home

Start time : 4:08:25 AM

Process Name :gic:Extraction
Event Name : FileUpload

Level-1:ValidUpload

Accept File Extenstion	
Max File Size(MB)	
Upload Directory	

Level-2:ValidDocument

File Path	
Type of File	
File Extenstion	
File Name	
Source Language	

Figure C.8: Semi-Automatic Configuration of Document Input

C 9. Tasks Distribution Matrix

Table C1: Tasks Distributions Matrix

1	1	1	1	1	1	2	2	2	2	2	2
2	2	3	3	4	4	1	1	3	3	4	4
3	4	2	4	3	2	3	4	1	4	3	1
4	3	4	2	2	3	4	3	4	1	1	3
3	3	3	3	3	3	4	4	4	4	4	4
1	1	2	2	4	4	3	3	2	2	1	1
2	4	1	4	2	1	2	1	3	1	2	3
4	2	4	1	1	2	1	2	1	3	3	2

APPENDIX D

SYSTEM USABLITY SCORE

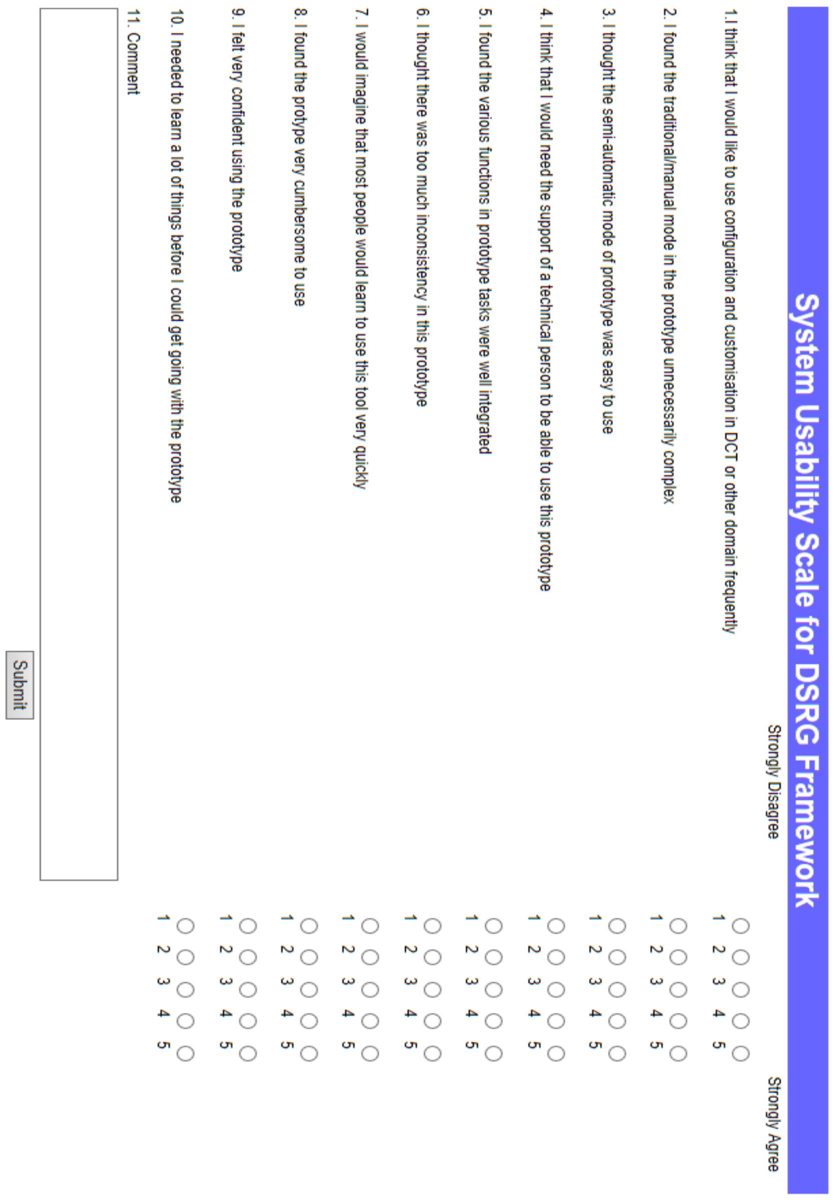
D 1. SUS Form

System Usability Scale Survey

1. I think that I would like to use configuration and customization in DCT or another domain frequently	1	2	3	4	5
2. I found the prototype tool unnecessarily complex	1	2	3	4	5
3. I thought the semi-automatic mode of prototype was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this prototype	1	2	3	4	5
5. I found the various functions in prototype tasks were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this prototype	1	2	3	4	5
7. I would imagine that most people would learn to use this tool very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with the prototype tool	1	2	3	4	5

Please provide your comments about the prototype and a short description about your choice for each of the SUS statements why and what make you take your decision?

D 2. SUS Web User Interface of SUS Form



System Usability Scale for DSRG Framework

	Strongly Disagree					Strongly Agree
1. I think that I would like to use configuration and customisation in DCT or other domain frequently		○ 1	○ 2	○ 3	○ 4	○ 5
2. I found the traditional/manual mode in the prototype unnecessarily complex		○ 1	○ 2	○ 3	○ 4	○ 5
3. I thought the semi-automatic mode of prototype was easy to use		○ 1	○ 2	○ 3	○ 4	○ 5
4. I think that I would need the support of a technical person to be able to use this prototype		○ 1	○ 2	○ 3	○ 4	○ 5
5. I found the various functions in prototype tasks were well integrated		○ 1	○ 2	○ 3	○ 4	○ 5
6. I thought there was too much inconsistency in this prototype		○ 1	○ 2	○ 3	○ 4	○ 5
7. I would imagine that most people would learn to use this tool very quickly		○ 1	○ 2	○ 3	○ 4	○ 5
8. I found the protype very cumbersome to use		○ 1	○ 2	○ 3	○ 4	○ 5
9. I felt very confident using the prototype		○ 1	○ 2	○ 3	○ 4	○ 5
10. I needed to learn a lot of things before I could get going with the prototype		○ 1	○ 2	○ 3	○ 4	○ 5

11. Comment

Figure D.1: SUS for DSRG Framework

D 3. SUS Form Analysis

Table D1: SUS Analysis

Question Number	Evaluation Criteria	Sentiment of Questions	System Usability Score
1.	Domain User	Positive	4
2.	UX	Negative	1
3.	UX	Positive	4
4.	Non-Technical User-UX	Negative	1
5.	Effectiveness	Positive	4
6.	Effectiveness	Negative	1
7.	Learnability	Positive	4
8.	Efficiency	Negative	1
9.	Learnability/User Friendly	Positive	4
10.	Learnability	Negative	1