“Software Reuse Process Adaptation”

By

Ana Paula Carvalho Cavalcanti

Master's Dissertation

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, SEPTEMBER/2007
ANA PAULA CARVALHO CAVALCANTI

“SOFTWARE REUSE PROCESS ADAPTATION”

THIS WORK WAS PRESENTED TO THE POST GRADUATE IN COMPUTER SCIENCE PROGRAM FROM INFORMATICS CENTER OF FEDERAL UNIVERSITY OF PERNAMBUCO, AS A REQUIREMENT TO OBTAIN THE TITLE OF MASTER IN COMPUTER SCIENCE.

ADVISOR: PROFESSOR SILVIO ROMERO DE LEMOS MEIRA

RECIFE SEPTEMBER, 2007
This work is dedicated to my father, Carlos, my mother, Vera, my loving husband, Felipe and my little son, Pedro.
Acknowledgments

The execution and completion of this work represents an effort that could only be accomplished with the support of very important people in my life that always remarked a presence during special moments of my professional and personal life.

First, I would like to thank my family that believed in me and, even far way from Recife, was always present on thoughts and wishing words of success and braveness. My fondness thanks to all my brothers, Carlos Junior, Ana Carla Cavalcanti, Denio Cavalcanti and João Paulo Cavalcanti.

To my brave father, Carlos de Lima Cavalcanti, that has always believed and encouraged me, who always tried to bring up all his children with the best he could provide. Even during difficult times, he has shown us that there is always a way out and solutions to the obstacles faced by life.

To my careful mother, Vera Maria Cavalcanti, an example of a strong woman who has done her best to raise her children and always helps us raise ours. Her presence as a mother, a friend and educator is essential for my life.

For my loving husband, Felipe Furtado, that showed me a new way of seeing life, always with patience, fondness and love. Without his great support, I would not be able to finish this work. Thank you for always standing by me.

To my four months old child, Pedro Cavalcanti Furtado who, during the hours of breastfeeding always looked softly and smiled at me, making my days have a different value. My endless loving son, I want to be a person for you to be proud of.

I would like to thank all the teachers that supported me during the tracking of my path here at Federal University of Pernambuco. Specially to Jones Albuquerque, who has given me the opportunity to work together and learn with all his didactics and experience in being a software engineer.
professor and a great personal friend. *Silvio Meira*, the greatest visionary man, who has supported me since my arrival in Recife, being a professor of the future and example of success, always coming up with great ideas of projects to our careers.

To all my friends, old and new ones, that have shared great academic and personal moments of friendship, specially to *Erick Lopes, Silvia Pereira, Angelo Ribeiro* and *Rogerio Nibon*. Thank you for being present and supporting me to finish this work.

To *RiSE group*, that is always showing an growing presence on software engineering around the world and have been perseverant in supporting my work during the past 2 years since my arrival in Recife. This work would not be completed without their support.

Last, but not least, thanks to God, the greatest provider of this opportunity and strength in my life.
Abstract

Today’s context of software development requires a certain amount of organization to build products and Process Definitions appear as an attempt to formalize and structure the set of steps, followed by a team, with the objective to deliver a final product with quality. Within this context, Software Reuse emerges to increase the company’s profit and make possible the reuse of components produced in previous projects. The benefits provided by the introduction of reuse discipline is to increase a team’s productivity and the proper development of reuse based products also requires a determined activities guide, also contemplated by a Reuse Process definition to help teams realize the development. Nevertheless, the introduction of reuse processes, into organizations that already have established functional practices, is even a more complex activity because it is important to preserve the institutionalized habits and not to drastically change the practices properly executed by the team. In this scenario, an Adaptation, to introduce reuse driven development process in an organization that already has a non-reuse development process, is necessary so that to minimize barriers of a new process deployment. Therefore, this work has the objective to present an Adaptation Process that links characteristics from a reuse and non-reuse development process in order to generate a common description that collects characteristics and specification from both process scenario.

Keywords: Software development processes, Software Reuse and Process Adaptation.
Resumo

O contexto atual de desenvolvimento de software requer certo grau de organização para desenvolver produtos e as Definições de Processos aparecem como uma tentativa de formalizar e estruturar o conjunto de passos, seguidos pela equipe, com o objetivo de entregar o produto final com a qualidade desejada pelo cliente. Nesse contexto, Reuso de Software emerge para aumentar o lucro das empresas e possibilitar o reuso de componentes produzidos em projetos anteriores. Os benefícios provenientes através introdução da disciplina de reuso para aumentar a produtividade das equipes e o desenvolvimento apropriado de produtos através de reuso requerem um determinado guia de atividades, contemplados pela definição de um Processo de Reuso para auxiliar as equipes executarem o desenvolvimento. No entanto, a introdução de processos de reuso, em organizações que já possuem práticas funcionais bem estabelecidas, é uma atividade mais complexa porque é importante preservar os hábitos institucionalizados e não mudar drasticamente práticas executadas propriamente pelas equipes. Nesse cenário, uma Adaptação, para introduzir um processo orientado a reuso em uma organização que já possui um processo de desenvolvimento, é necessária para minimizar as barreiras da implantação de um processo. Dessa forma, este trabalho tem o objetivo de apresentar um Processo de Adaptação, que coleta especificações de um processo orientado a reuso e um processo tradicional, para gerar uma descrição comum contemplando características e especificações dos cenários dos dois processos.

Palavras-chaves: Processos de desenvolvimento de software, Reuso de Software e Processo de Adaptação.
# Content

<table>
<thead>
<tr>
<th>ACKNOWLEDGMENTS</th>
<th>ABSTRACT</th>
<th>RESUMO</th>
<th>LIST OF ACRONYMS</th>
<th>LIST OF FIGURES</th>
<th>LIST OF TABLES</th>
<th>CHAPTER 1</th>
<th>CHAPTER 2</th>
<th>CHAPTER 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>II</td>
<td>IV</td>
<td>V</td>
<td>IX</td>
<td>X</td>
<td>XI</td>
<td>1</td>
<td>10</td>
<td>18</td>
</tr>
</tbody>
</table>

## CHAPTER 1

1. INTRODUCTION ............................................................................................................... 1
   1.1 Motivation .................................................................................................................. 1
       1.1.1 Reuse and Non-Reuse Processes ................................................................. 2
       1.1.2 Process Adaptation .......................................................................................... 3
   1.2 Problem Statement ................................................................................................. 4
   1.3 Overview of the Proposed Solution .................................................................. 6
       1.3.1 Context .............................................................................................................. 6
   1.4 What is Left out of Scope ................................................................................. 7
   1.5 Statement of the Contributions ....................................................................... 8
   1.6 Organization of the Dissertation ................................................................... 8

## CHAPTER 2

2. SOFTWARE DEVELOPMENT PROCESS ............................................................................ 10
   2.1 Introduction ........................................................................................................ 10
   2.2 Software Process Overview .......................................................................... 11
   2.3 Software Reuse Process .................................................................................. 14
       2.3.1 Domain Engineering ...................................................................................... 16
       2.3.2 Product Lines .................................................................................................. 16
   2.4 Summary ............................................................................................................ 17

## CHAPTER 3

3. PROCESS ADAPTATION: AN OVERVIEW .......................................................................... 18
   3.1 Introduction ........................................................................................................ 18
   3.2 Adaptation Overview ......................................................................................... 19
       3.2.1 Rational Unified Process - RUP .................................................................... 20
       3.2.2 Reuse Oriented Software Evolution - ROSE .............................................. 21
       3.2.3 Organizational Domain Model - ODM ....................................................... 22
       3.2.4 Feature Oriented Domain Analysis - FODA ............................................. 23
       3.2.5 Reuse Driven Software Engineering Business – RSEB and FeatureRSEB ...... 24
       3.2.6 Product Lines Software Engineering - PuLSE ......................................... 25
3.2.7 Component Based Product Lines Development - KobrA ............................................ 27
3.2.8 Component-Oriented Platform Architecting Method – CoPAM ............................ 29
3.2.9 Pervasive Computer Systems – PECOS ................................................................. 30
3.2.10 Feature Oriented Reuse Method – FORM ........................................................... 31
3.2.11 Asset Based Development – ABD ......................................................................... 31
3.3 Summary ............................................................................................................. 32

CHAPTER 4 ............................................................................................................................ 35
4 ADAPTATION PROCESS .................................................................................................. 35
4.1 Introduction ............................................................................................................... 35
4.2 Process Overview ..................................................................................................... 36
  4.2.1 Objectives............................................................................................................. 37
  4.2.2 Terminology ......................................................................................................... 38
  4.2.3 Foundations.......................................................................................................... 39
    4.2.3.1 RiSE Approach for Domain Engineering – RiDE ........................................... 39
    4.2.3.2 Rational Unified Process – RUP ...................................................................... 42
    4.2.3.3 Product Line Software Engineering – PuLSE ................................................. 44
4.3 Process Activity Description .................................................................................. 45
  4.3.1 Understand Context.............................................................................................. 46
  4.3.2 Define Process ...................................................................................................... 47
    4.3.2.1 Define Process Phases..................................................................................... 47
    4.3.2.2 Define Process Activities .............................................................................. 48
    4.3.2.3 Define Process Vocabulary............................................................................ 49
  4.3.3 Consolidate Process.............................................................................................. 50
  4.3.4 Maintain Process.................................................................................................. 51
4.4 Summary .................................................................................................................. 51

CHAPTER 5 ............................................................................................................................ 54
5 CASE STUDIES................................................................................................................. 54
5.1 Introduction ............................................................................................................... 54
5.2 Case Study 1 ............................................................................................................ 55
  5.2.1 Understand Context............................................................................................. 55
  5.2.2 Process Definition ............................................................................................... 56
    5.2.2.1 Define Phases.................................................................................................. 56
    5.2.2.2 Define Activities ............................................................................................ 57
    5.2.2.3 Define Process Vocabulary............................................................................ 58
  5.3 Case Study 2 ............................................................................................................ 61
  5.3.1 Understand Context............................................................................................. 61
  5.3.2 Process Definition ............................................................................................... 63
    5.3.2.1 Define Phases.................................................................................................. 63
    5.3.2.2 Define Activities ............................................................................................ 63
    5.3.2.3 Define Process Vocabulary............................................................................ 64
5.4 General Findings........................................................................................................ 67
5.5 Summary .................................................................................................................. 68
List of Acronyms

ABD – Asset Based Development
CBD – Component-Based Development.
CBSE – Component-Based Software Engineering
CFRP – Conceptual Framework for Reuse Processes
CMMI – Capability Maturity Model Integration
COPAM – Component-Oriented Platform Architecting Method
FODA – Feature-Oriented Domain Analysis
FORM – Feature-Oriented Reuse Method
KobrA – Komponentenbasierte Anwendungsentwicklung
ODM – Organization Domain Modeling
OSGi – Open Service Gateway Interface
PECOS – Pervasive Component Systems
PML – Process Modeling Languages
PULSE – Product Line Software Engineering
RAS – Reusable Asset Specification
RFL – Reuse Library Framework
RiSE – Reuse in Software Engineering
RiDE – The Rise Process for Domain Engineering
ROSE – Reuse-Oriented Software Evolution
RSEB – Reuse-Driven Software Engineering Business
RUP – Rational Unified Process
SPEM – Software Process Engineering Metamodel
SPL – Software Product Lines
STARS – Software Technology for Adaptable Reliable Systems
List of Figures

Figure 1: Representation of two scenarios of process adaptation. .........................4
Figure 2: RiSE Framework for Software Reuse ......................................................7
Figure 3: PuLSE Initialization (Bayer et al., 1999 ) ..............................................27
Figure 4: Adaptation Process Workflow ...............................................................37
Figure 5: RUP Overview ....................................................................................43
Figure 6: Meta-model applied in the first case study .........................................60
Figure 7: Meta-Model Applied on the Second Case Study. .............................65
List of Tables

Table 1: Summary of adaptation techniques found on development processes........34
Table 2: Foundations for the Adaptation Process .......................................................53
Table 3: Selected Customization Factors Summary ..................................................56
Table 4: Process Phases Map .................................................................................57
Table 5: Process Activity Map ................................................................................58
Table 6: Process Vocabulary ..................................................................................59
Table 7: Adapted Process Artifacts ......................................................................61
Table 8: Selected Customization Factors of Case Study 2 ...................................62
Table 9: Process Phases Map of Case Study 2 ......................................................63
Table 10: Process Activities Map of Case Study 2 ...............................................64
Table 11: Process Vocabulary Map Applied on the Second Case Study ...............66
Chapter 1

1 Introduction

Reuse is considered an important factor to the survival of software development companies on today’s market. Competitive values are growing and the return on investments around the community must increase in order to maintain sustainable organizations that deliver products of quality.

Software development is an activity executed together with a development process, and, to successfully achieve desirable levels of reuse, it is necessary to understand, assess and apply processes that best fit your organization reality. Therefore, there is a need to apply reuse centric development processes to systematically introduce reuse in a planned and organized standard.

This Chapter presents a survey on considerable aspects of software reuse and the main problems faced, thus to comprehend the necessity to propose an Adaptation Process that gathers characteristics from reuse and non-reuse centric development process in order to adapt to a unique process. Therefore, it is organized as it follows: Section 1.1 presents the motivation of this dissertation followed by the problem definition on Section 1.2. Section 1.3 shows an overview of the proposed solution, Section 1.4 clearly mentions the points not covered by the work, Section 1.5 outlines the contributions and Section 1.6 depicts the overall organization of the dissertation.

1.1 Motivation

Nowadays, market’s reality is gradually forcing enterprises to think about economical benefits through the return on investment made on technological investments. Specifically on how to spend less and increase the company’s profit in order to have competitive advantage and survive in software market. In this context, software reuse is seen as “the use of any information which a developer may need in the software creation process”, in a definition that dates back to 1987 (Freeman, 1987).
In a more recent view, Krueger states that reuse is “is the process of creating software systems from existing software rather than building them from scratch” (Krueger, 1992). Both consider the fact of planned reuse to higher productivity levels and time to market of the companies. During the 90’s, “the state of reuse practices was informal, either based on individual basis (reuse of its own components) or on commercial products (data base systems, user interface systems)” (Ribot et al., 1994). There was a need to formalize and enhance reuse practices and such changes led to a fundamental organization conversion, in both business model and organizational culture.

Yet, it is important to provide and understand the relationship that involves technical and non-technical aspects needed for the success of a reuse implementation plan. Desirable results, thought, can be achieved through the verification of efforts made on some challenges (Basili & Rombach, 1988), such as:

- “Verifying return on investment (ROI) either in reduced time-to-market, increased productivity or in improved quality;
- Identifying the preconditions to start a reuse program; and
- Developing processes for software reuse, roles, steps and adapting existing processes.”

Based on this context, market’s concern is an important fact that is seen around companies, such as IBM (Bauer, 1993), Industrial Software Labs (Endres, 1993), Hewlett-Packward – HP (Griss, 1994), Motorola (Joos, 1994). Implementation techniques, reuse approach, environment and other technical and non-technical aspects are presented in these four different company’s context, thus emphasizing the necessity for further investigation on how to smooth the introduction of reuse practices into business reality.

1.1.1 Reuse and Non-Reuse Processes

Another relevant aspect considered by the market is software product quality, which is seen as “the degree in which the system, component or process satisfies its specific requirements and clients’ expectation” (IEEE,
The survey of Rine and Sonneman (Rine & Sonnemann, 1998), on over 99 projects in 83 organizations, comments that software reuse capability positively influences the overall quality and productivity.

Within reuse context, quality issues are correlated to the analysis of reuse usage evolution along the years. There must be a systematic approach so that organizations are able to evaluate benefits and gain, not only in process performance, but also on the quality delivered by the product. In this direction, reuse centric development processes focuses on detailed activities and steps to contemplate the development of products within a reuse scenario.

On the other hand, conventional software development processes can be understood as a set of steps and guidelines in which a system can be methodologically developed. There are several approaches to apply a development process into an organization and it is necessary to comprehend the characteristics and necessities of each place in order to deploy a process that best fits its needs. The implantation and execution of such processes happen together with the evaluation of metrics related to both organization and process, focusing on the identification of deviations on the instantiated process, and on the product delivered. In this context, quality issues are also related to the quality of the final product.

1.1.2 Process Adaptation

Techniques and methods to adjust defined processes to specific realities, such as companies’ development process or specific projects’ demands, are treated as adaptation, instantiation or tailoring methods. It is seen that both conventional and reuse processes, that have general guidance on how to apply their procedures, require a certain amount of instantiation when they are applied to specific scope, thus, the importance to consider adaptation procedures when defining processes.

Adaptation, from a process framework to a specific project, must consider the characteristics of the application, scenario that is being built,
scenario to be deployed and quality issues required by both developer and costumer.

Adaptation between processes requires a comprehension between both process vocabulary, in terms of what is produced and required by each step of the processes, and make a correlation between them so that a uniform process is generated and can be applied by its own. Figure 1 depicts both scenarios of adaptation mentioned before.

Figure 1: Representation of two scenarios of process adaptation.

1.2 Problem Statement

Based on today’s reality of software development, it is seen that organizations are moving forward to the establishment of software reuse practices, and according to a recent survey mapping the current state of Brazilian industry, (Brito et. al, 2006), 53% of software reuse factories obtained success in development projects through software reuse. Accordingly, the adoption of a defined reuse process is one of the key factors that influence that success, and the numbers showed that 73% of the cases of success is due to a definition of a reuse process.

Although “formal reuse processes can be easily integrated into the classical development life-cycle” (Ribot et al., 1994), there is very little on practical guidance to inform the steps of the method to modify or omit existing organizational standards.
In one of few empirical research about software reuse published, Rine (Rine, 1997) states that “presently there is no set of success factors which are common across organizations and have some predictable relationship to software reuse”. The survey conducted to investigate the success factors of software reuse showed some common characteristics present on the investigated organizations, which mainly focus on:

- Product Line approach to support definition and execution of software development within a determined scope of applicability;
- Domain Engineering to define a development process that considers the comprehension and development of a determined scope domain; and
- Reuse Process, describing a systematic way to implement planned reuse into a project.

Following, another approach, a survey in industrial projects that attempts to introduce reuse, Morisio (Morisio et al., 2002) analyzes the results and states that “not modifying non-reuse processes was the immediate cause of failure”, thus presenting the need to adapt reuse-specific processes before their introduction in organizations that already have a process institutionalized.

The integration of reuse practices in the overall development process is considered by (Rothenberger et al., 2003), where Rothenberger mentions that “the reuse practices applied must be understood in order to best comprehend the strategies and implications by the adopters”.

Ultimately, reuse context defines the necessities of what a such context must have, but it doesn’t deal with how to apply to:

- Generalized model of software development;
- Lifecycle specific definitions;
- Specific organizational processes; and
- Instantiated Process to a chosen domain within a software product line.
Based on the scenario presented above, the objective of the work described in this dissertation can be stated as:

This work defines an adaptation process between reuse centric processes, in the scope of domain engineering activities, and a conventional development process through a definition of a set of activities.

1.3 Overview of the Proposed Solution

In order to accomplish the defined objective, stated in the previous Section, an Adaptation Process is proposed. This section describes the context in which this work is inserted and the architecture of the proposed solution.

1.3.1 Context

This work was developed together with the research of RiSE\(^1\) group, whose goal is to investigate and develop the state-of-the-art and practice related to software reuse. Figure 2 presents an overall structure in which reuse problem was mapped and the framework constitutes a solid basis for organizations that are moving towards an effective reuse program. Such framework is composed of best practices in software reuse (education, trainings, reuse incentives and program adoption) organized into the non-technical aspects, and the technical aspects of software reuse, represented by processes, environments and tools (Almeida 2007).

The designed software reuse environment has the objective to map the existing problems faced on the adoption of the reuse culture in companies. The framework scope is composed of processes, procedures, guidelines and tools to enhance the application of reuse in a practical environment. Based on

\(^{1}\) RiSE – Reuse in Software Engineering Group
http://www.rise.com.br
that, software reuse processes are the focus of this study to the comprehension of their behavior and outline the proposed solution.

![RiSE Framework](image)

**Figure 2: RiSE Framework for Software Reuse**

The RiSE Process for Domain Engineering (RiDE) defined by (Almeida, 2007) will be used to guide the adaptation process based on the activities of the phases Domain Analysis, Domain Design and Domain Implementation.

### 1.4 What is Left out of Scope

The aspects not directly addressed by this work are listed:

- **Empirical Study**: although the execution of an empirical study to formally measure the results achieved by an adaptation between two process could bring more reliable results to validate the process, this work does not contemplate such execution. The rationale is that the effective measurement of reuse introduction into organization is not dependent only in the adaptation process. Based on the framework explained on Figure 2, the whole reuse problem can be solved together with other scope of studies;

- **Role Definition on Process Adaptation**: the definition of roles is out of the scope due to the fact that we consider the existing ones, in the processes being adapted. It is assumed that Domain Engineering and Non-reuse processes already contain a description of the roles to execute their activities and the adaptation condensates a group of previously pre-defined activities and roles.
1.5 Statement of the Contributions

As a result of the work presented in this dissertation, the following contributions may be listed.

- A bibliographical review on how software reuse processes are treating adaptation matters to introduce their reuse processes into an organization;
- A adaptation process that collects information from reuse and non-reuse processes to transform into; and
- The Adaptation Process execution through the execution of two case studies to collect a set of observed results in order to improve the process.

Besides the contributions listed above, the following works have been published in the literature as consequences of this work:

- **The Domain Analysis Concept Revisited: A Practical Approach**, In The 9th International Conference on Software Reuse (ICSR), Lecture Notes in Computer Science, Springer-Verlag, Torino, Italy, 2006;
- **Reuse Process Adaptation Strategies**, In the 32nd IEEE Euromicro Conference on Software Engineering and Advanced Applications, Work in Progress Session, Cavtat/Dubrovnik, Croatia, 2006; and

1.6 Organization of the Dissertation

The reminder of the dissertations is organized as follows:

- **Chapter 2** describes an overview on software development processes, in both reuse and non-reuse centric approaches, together with the most relevant definitions of these contexts;
- **Chapter 3** presents a bibliographical revision about how adaptation is being treated on software development processes, in both reuse and non-reuse processes;
• **Chapter 4** presents the adaptation process proposed by this work, together with the process overview and activities description;

• **Chapter 5** reports the results of the adaptation process in two case studies in order to best understand the meta-model and assess the results achieved;

• **Chapter 6** concludes the work by explaining the general findings and analysis of the work and comparing with related work. Future research related to this work will also be discussed.

• **References** lists the references used in this dissertation;

• **Appendix A** contains the Domain Requirement Template adapted for the case study;

• **Appendix B** contains the Architecture Document Template used on the case study; and

• **Appendix C** contains the Component Specification Template used on the case study.
Chapter 2

2 Software Development Process

The introduction of well defined software reuse processes into organizations is the motivation of this work. Therefore, a proper revision on software development processes and reuse general concepts is necessary to understand the basic notion of the Adaptation Process.

This Chapter presents an overview on software development and reuse basic concepts and it is organized as follows: Section 2.1 presents an introduction to the topics mentioned, Section 2.2 shows an overview of the software process background and Section 2.3 remarks general considerations on reuse centric development processes. Finally, Section 2.4 summarizes the main findings of the overview.

2.1 Introduction

“Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software” (IEEE, 1990) and this term has been popularized since 1968 on NATO Software Engineering Conference². Such discipline, as it has been approached by the computer science community, “encompasses knowledge, tools, and methods for defining software requirements, and performing software design, software construction, software testing, and software maintenance tasks” (SWEBOK, 2004).

Software Engineering discipline has an evolving history and Software Development Processes appear in the community between the 60’s and 70’s. According to the free encyclopedia that anyone can edit, a definition that represents the popular culture and condenses a global view on the subject, Wikipedia states that “processes relate to the sequence of operations and involved events, taking up time, space, expertise or other resources, which

² NATO Software Engineering Conference was held in Garmisch, Germany, 7-11 October, 1968.
lead/(should lead) to the production of some outcome” (Wikipedia, 2007). There is a consideration about the aspects of systematic execution of activities, which can be applied not only in software industry. In a more conservative point of view, a “process is a partially ordered group of activities (or steps) to achieve a goal” (Humphrey, 1989). Both definitions concern the regular and continuous actions to execute a certain activity within a determined project scope, and an import step that software engineering has achieved in history by process definitions.

Another trend of the constant evolution is the introduction of Software Reuse, appearing in 1968 as one of the directions to be followed when Douglas McIlroy (McIlroy, 1968) proposed basing the software industry on reusable components. In this scope, software reuse processes are established to systematically introduce reuse practices into the industry.

All definitions presented are relevant factors to support the study of software development processes, considering both non-reuse and reuse aspects. Therefore, it is important to understand their basis and evolution along with software engineering history.

2.2 Software Process Overview

Software Development Processes were introduced in software engineering community as an attempt to define a set of activities and expected outputs to guide the delivery of a specific product. The ad-hoc manner to describe the client’s necessity and build products has shown that such way of developing software conducts to product’s delivery without quality, which tends to undesired results by the end of a project.

According to Osterweil (Osterweil, 1987) “humans have an innate facility for indirect problem solving through process specification”, which is the indirect way of solving troubles through process definition. Processes without any specification are hard to be understood, because they are abstract, while processes definition are more tangible, becoming easier to be understood. Based on that, the difference between the process and its description is that the process is the vehicle for doing the job and its description is the
specification detailing the content. This idea is applied in software development field, where process description is a broad guide to be instantiated to each project context. Independent on the approach given, processes are necessary instances in a manner of a tool to facilitate software engineers’ work and spread a uniform comprehension on the job to be done, in order to deliver the final product.

In another point of view, Sommervile highlights (Sommerville, 2006), “software processes are complex and, like all intellectual processes, are reliant on human judgment”, leading us to the concern that an abstract and subjective way that they are presented conducts to miscomprehension among the process’s users.

Based on the timeline, presented by the paper Software process: a roadmap, Fuggetta (Fuggetta, 2000) shows an overall view of software development processes appearance in the community, along with the movements made to establish the research society in the 80’s. The initial contributions have been the increasing awareness that developing software is a complex process. During the 60’s and 70’s, software engineering work was focused on:

- Structured programming languages;
- Design methods and principals; and
- Software lifecycle definition.

Lifecycle was defined not to describe a precise course of actions, but to build the foundation and philosophy of software process concept and (Fuggetta, 2000) defines it “as the coherent set of policies, organizational structures, technologies, procedures and artifacts that are needed to conceive, develop, deploy, and maintain a software product”. Software lifecycle could be understood as a starting point to definitions of processes and they were basically the following:

- **Spiral model**: it was proposed as an attempt to integrate evolutionary and requirement management with the characteristic that development spirals outward from a specification. The objective
is to mitigate and assess risks in each round of the spiral (Boehm, 1988);

- **Incremental models**: they were defined to supply the necessity for specification change, where the system is partitioned into increments that are delivered while others go through the phases in parallel. The Cleanroom (Linger, 1994) approach considers increments formally specified, an approach that has led to a very low number of defects in delivered systems;

- **Waterfall**: it consists of four phases: specification, design and implementation, integration and testing, operation and maintenance of which the lack of feedback from one phase to another states a remarkable problem; and

- **Evolutionary development models**: in this model, the stages are based on: (1) formulation of the system requirement; (2) development, as fast as possible and (3) evaluation with user and modification until functionality meets the user’s need. This leads to a process suited to the end-user’s need requirements, and the relevant problems are: (i) focus on end-user’s, so critical organizational requirements may not be given sufficient priority; (ii) the constant change to software degrades its structure and the final result is often difficult and expensive to change; and (iii) it does not have a high visibility which is difficult to manage.

Software development process models depend on the organization, type of software, and characteristics of the team, and so, hybrid models can also be applied according to the project’s scope.

Due to the necessity to represent processes in an accurate and clear manner, Process Modeling Languages – PMLs – have been created by researchers to formalize process description, in terms of activities, roles, artifacts and tools that support them. According to (Ambriola, 1997), “a PML can be used to improve process comprehension and documentation through formal modeling, analysis, and simulation".
Another trend, regarding software processes, is related to quality models and process improvement, that appeared based on the observation of past experiences to assess good practices of software process in a range of projects. Quality models, such as CMMI (Chrissis et al., 2003), ISO 12207 which is a standard for software lifecycle processes (ISO, 1995) and ISO 15504 that is a framework for the assessment of processes (ISO, 2004), can be understood as a group of practices or guidelines that were applied before and gathered as best practices that software processes can follow to achieve the desirable quality level.

In addition, software metrics, with Software measurement: a necessary scientific basis (Fenton, 1994), and empirical studies (methods and results), with Experimental Software Engineering: a report on the state of the art (Votta & Porter, 1995), are significant contributions to the process. Numbers, indicators and experimental approaches are means to be applied in order to evaluate and understand the situation of a process. The growing interest in such topics is the result of the desire to increase the comprehension on principals and nature of software development.

Such issues are summed up to the fact that Software Processes Are Processes Too (Osterweil, 1987), playing an import role in the future of software engineering. Software development is a critical activity of our society and it is important to comprehend how people consider and evaluate the effectiveness of a process within a software development project. The concept and evolution of software process are necessary to form the basis for the comprehension on software reuse processes, which will be discussed on the next section.

2.3 Software Reuse Process

Douglas McIlroy (McIlroy, 1968) first introduced the idea of reuse as the planned development and widespread use of software components. According to another researcher (Morisio et al., 2002), “Reuse is an umbrella concept, encompassing a variety of approaches and situations” and the benefits should be quantified and empirically assessed.
Developing for reuse has its price, which is the reason for analyzing the success of reuse programs to improve the chances of succeeding. A Survey on Software Reuse (Almeida et al., 2005) presents a detailed research on software reuse processes, where there is a remarks that such processes “must describe two essential activities: the development for reuse and the development with reuse”. According do Ribot et al. (Ribot et al., 1994), there are two distinct aspects in the overall reuse development process: the development of new inherently reusable components and the development of systems using existing reusable components.

Typically, knowledge acquired on the development of asset, within a determined domain, must be considered to achieve desirable reuse levels, and process definitions are an important matter to help analyze and compare the expected reuse metrics. All considerations made to process definition on conventional development process, such as analysis, modeling, implementation and testing, are considered in the range of reuse processes too, with additional points including reuse metrics, component qualification, domain modeling and others.

Along with that, reuse process was discussed on a working group session held at the SIGSOFT Foundations of Software Engineering in 1996 SIGSOFT FSE\(^3\), where such group intended to assess and discuss software reuse process and reusability, “specifically how to pragmatically and systematically standardize and replicate project-specific processes in an industrial software environment”.

Software Process Reuse in an Industrial Setting (Hollenbach & Frakes, 1996) was used as a starting point, and conclusions claim that little was found on reuse process and there is still a need for further validation study on the method.

\(^3\)SIGSOFT FSE - Symposium on Foundations of Software Engineering http://www.informatik.uni-trier.de/~ley/db/conf/sigsoft/fse96.html
2.3.1 Domain Engineering

Domain Engineering can be understood as “the activity of collecting, organizing, and storing past experience in building systems or parts of systems in a particular domain in the form of reusable assets, as well as providing an adequate means for reusing these assets when building new systems” (Czarnecki & Eisenecker, 2000).

It is focused on the creation of a competence for building a family of similar systems through the use of core assets. It is necessary to understand similarities and differences among features that are used to build software applications addressing the systematic creation of domain models and architecture. The main motivation is to achieve significant increase in software development productivity within a specific problem domain.

Initial incentives shows that in the 80’s, the first domain engineering approach appeared with Neighbors (Neighbors, 1980), followed by (Simos et al., 1996), (Jacobson et al., 1997), (Griss et al., 1998) and (Kang et al., 1998).

2.3.2 Product Lines

In the early 1999, software reuse development began to related product lines issues and Bayer (Bayer et al., 1999) presented the first approach to product lines processes, with the Product lines Software Engineering (PuLSE) methodology. According to (Clements & Northrop, 2001) it can be understood as “a set of software-intensive systems that share a common, managed set of features satisfying 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”.

In order to apply software product lines practices into an organization, it is necessary to clearly understand the benefits that can be brought and evaluate if such approach brings economical payback to the organizational context, understanding the real return on the investments made. Product lines development is focused on the comprehension of commonalities and variabilities from the experience and know-how that an organization gathers
from their product development along the years. In order to achieve this, one specific approach is to assess isolated projects and understand how they fit into the group of projects available into the organization.

The overall context of reuse and non-reuse processes are necessary to understand and extend the study related to the adaptation of these processes in organizations that have institutionalized processes.

2.4 Summary

Software Development Processes have been investigated for decades and a growing research area is being brought to software engineering community through many different kinds of approach, each one is applied in determined scopes of applicability.

Within this scenario, Software Development Processes appear with distinct description together with the evolution of research topics, such as lifecycle models, process modeling languages, quality models for process improvement, metrics and empirical studies.

Another trend regards Software Reuse, which was introduced by Douglas McIlroy, representing that products and knowledge produced before need not to be thrown away and can be reused in further contexts.

Therefore, this Chapter presented the concepts that are necessary to set up the background knowledge and basis for the study of process adaptation, on both reuse and non-reuse processes. Next Chapter will present the complementing part of the work review, a detailed study about reuse and non-reuse process adaptation in software community.
Chapter 3

3 Process Adaptation: An Overview

Adaptation techniques can be presented in different forms, depending on the approach shown by the process in question. Based on a research about how adaptation matters have been treated among software process development, presented on Chapter 2, a set of findings were achieved with the objective to comprehend different ways to realize an adaptation.

This Chapter presents an overview of software development processes adaptation. It is organized as it follows: Section 3.1 presents the introduction of the concepts about process adaptation followed by an overview on the available adaptation techniques on Section 3.2 and Section 3.3 outlines the summary of the Chapter.

3.1 Introduction

The boundaries of software process adaptation are now being widely explored in a way that processes are adjusted to the actual and specific needs of a development context. The necessity, to introduce such approach, arose from the industry and the problem was brought to the academy.

According to RUP (RUP 2003), process adaptation or process tailoring has the purpose “to right size software development process according to the specific needs of the project”. The specific needs are based on factors, such as:

- Customer requirements;
- Project requirements;
- Commitments;
- Operational environment; and
- Business environment.
These projects’ requirements together with the institutionalized process guidelines are the variables analyzed to execute a process adaptation into a specific project. Criteria and guidelines are used to describe "procedures that must be followed in performing and documenting process tailoring" (Chrissis et al., 2003), in order to alter or adapt a process description for a particular end.

Besides the comprehension of process tailoring and guidelines, institutionalization is also an important concept, and ( Chrissis et al., 2003) states that "institutionalization implies that the process is ingrained in the way the work is performed and there is commitment and consistency to performing the process". In order to systematically execute the adaptation of a process, the institutionalization must happen beforehand because it is necessary to first understand the process in order to, consequently, adapt it.

All contexts, regarding process adaptation, lead to the activity of understanding the set of variables that are around development project contexts and set up the appropriate configuration to be executed in the best way expected. This, however, does not free engineers to have problems with the process, which needs to be assessed and improved during the course of the project.

3.2 Adaptation Overview

Tactics on adaptation, tailoring or instantiation refer to how a process framework can be configured to a determined project’s scope or specific organizational needs. Hence, different contexts were assessed in order to provide a background on the strategies available on the community and practiced by the industry.

Adaptation techniques were studied, in the scope of non-reuse and reuse processes, to understand and assess the patterns, techniques and activities applied and they are presented on a chronological order, beginning with the approach of RUP as a non-reuse process, followed by the approaches of reuse processes.
3.2.1 Rational Unified Process - RUP

In conventional software development process, Rational Unified Process, RUP (RUP, 2003), is a “software engineering process that provide tasks and responsibilities to assign software development activities with the goal to ensure the production of high quality software according to the customers needs”. RUP’s process is based on iterative and incremental lifecycle, composed of four phases: Inception, Elaboration, Construction and Transition. Each phase has its own specific goals and the activities to be performed are organized in terms of disciplines. RUP’s framework is divided into eight disciplines: Requirements, Analysis and Design, Implementation, Test, Deployment, Configuration and Change Management, Project Management and Environment; each one presents a set of associated activities, artifacts and roles to achieve a determined objective.

Adaptation on RUP is described in Environment discipline that focuses on “preparing the development environment for a means of turning the underlying development process into an enactable project-specific development process”. Such discipline describes activities, steps, artifacts and roles in order to comply the preparation of the framework to a project, based on activities necessary to configure the process for the project. The objective is to provide software process and tools to support development activities. RUP’s process tailoring, or adaptation, can happen in two levels:

- **Organizational level**, which is the adaptation of RUP to an organization context, according to issues such as the application domain, reuse practices and core technologies mastered by the company; and
- **Project level**, which is the adaptation of the organization process according to a project specific needs.

According to such considerations, it is seen that RUP provides a complex and extended structure to support process adaptation, based on guidelines, support tools and meta-model compliant with Object Management

### 3.2.2 Reuse Oriented Software Evolution - ROSE

On reuse specific processes, the approach presented by the Reuse Oriented Software Evolution, ROSE (STARS, 1993), is a process model that was developed by STARS Team as an evolution of the Conceptual Framework for Reuse Processes (CFRP). CFRP was considered too generic to be used as a basis for developing reuse centric software products, therefore ROSE was created with the attempt to be a more specific and detailed method than its antecessor.

ROSE Process Model is based on four sub-models: Organizational Management, Domain Engineering, Asset Management and Asset Engineering. According to the process description, “one key objective that influenced decisions about the level of detail and specialization in the ROSE model was that ROSE should remain quite generic and tailorable in its own right”, where a preoccupation on adaptation strategies is stated in the process itself. The description of the model adoption acquire concepts and principals to use them as process framework for reuse programs and projects, reconfiguring the sub-models in order to generate “ROSE-consistent” process models.

For reconfiguring activities, ROSE presents a series of key principles, detailing that (i) a process idiom can be applied at many organizational levels; (ii) evolution and maintenance phases are part of the Reuse management structure and (iii) reuse engineering define distinct set of activities to form the basis of individual projects. It is possible to identify, adapt, or define process and methods with the concern to ensure compatibleness across process categories, families and sub-models. Therefore, methods for the production of assets, domain engineering activities, must be compliant with the methods for their consumption, the application engineering.

There is also the adaptation towards an organization’s existing structure, where ROSE activities can be performed across a number of
management levels or tailored to a particular management style. Another option shown is to gather its use in order to evolve reuse practices within an organization. Thus, it is seen that a certain amount of application is feasible to be applied, but they are described in a too generic way, without specifying details on how to perform them.

3.2.3 Organizational Domain Model - ODM

Organizational Domain Model – ODM (Simos et al., 1996) is a guidebook with the objective to “systematize key aspects of the domain modeling process and provide an overall framework for a domain engineering life cycle”. It originated during a Unysis STARS project to develop a Reuse Library Framework (RFL), a tool for domain modeling, when the initial basis of ODM method was established. Subsequently, it was refined by the collaboration of Hewlett-Packard Company and from Lockheed Martin Corporation (formerly Loral Defense Systems-East and Unisys Government Systems Group).

The guidebook specifies strategies and transformations to apply phases and activities to a specific context of project. The process framework is organized hierarchically in terms of: (1) life cycle; (2) phases: Plan Domain, Model Domain and Engineer Asset Base; (3) sub-phases and (4) tasks.

Process tailoring is a straight relation between the guidebook and each domain engineering project, which can be taken in three forms:

- **Specialization**, where the framework is tailored to a process model, that is a framework itself, focused on a constricted organizational context. The relation is between ODM framework to another process framework;
- **Instantiation** between ODM and an specific project, where the framework is tailored according to the project's domain specific scope; and
- **Documentation** of project’s process history, to be used for process improvement within the organization, comparison of different projects, or for evolution, validation and refinement of the ODM method itself.
The guidebook also provides a number of specific transformations formalized in terms of supporting methods, such as deletion, addition, sequencing, renaming and restructuring processes or workproducts. On the other hand, Simos et al. (Simos et al., 1996) comment that a "complete guide to tailoring process is beyond the scope of the guidebook", leaving a gap on direct tailoring activities.

3.2.4 Feature Oriented Domain Analysis - FODA

FODA – Feature Oriented Domain Analysis (Kang et al., 1990) is a method for discovering and representing commonalities, through the means of feature identification, to support the development of reusable software resources. The method defines products and a process that gather information from multiple resources, such as source code, requirement and design documents, domain literature and domain experts, supporting reuse at both functional and architectural levels. The applicability scope encompasses a family of systems in a domain, producing a domain model with parameterization to accommodate the differences and standard architecture for developing software components. The method is based on several reports describing major efforts in domain analysis, which include methods descriptions, case studies and tools recommendations.

The method is organized in order to provide specific representations to document domain analysis activities results and is composed of three phases:

- **Context Analysis** to provide the domain context by representing primary inputs, outputs and other software interfaces of the domain. The main products are Structure and Context Diagram;
- **Domain Modeling** to present the features, standard vocabulary, documentation of the entities embodied in software and generic software requirement via control flow, data flow, and other specification techniques. The main products are entity relationship model, features model, functional model and domain terminology dictionary; and
• **Architecture Modeling** to establish the implementation structure in order to provide a set of architectural models for constructing applications and mapping from the domain model to architectures. Process interaction model and module structure chart are the main products of this phase.

The method discusses about adaptation in terms that an architectural model can be adapted to include future changes in the problem and technology adaptation achieved through architecture layering, considering the levels of abstraction. There is also the statement that process aspect considers how domain analysis method will affect an organization, but there are no detailed guidelines on the inclusion of domain analysis in other scope of development processes.

### 3.2.5 Reuse Driven Software Engineering Business – RSEB and FeatureRSEB

Reuse Driven Software Engineering Business (RSEB) (Jacobson et al., 1997) – is a systematic model-driven approach to software reuse engaged in building sets of related applications from reusable components. It captures domain information in several different models reflecting different point of view. Its domain analysis method derives from Feature Oriented Domain Analysis (FODA) (Kang et al., 1990). From Organization Domain Modeling (ODM) (Simos et al., 1996), RSEB is based on “how and why systems are chosen and analyzed to meet technical and stakeholders needs and feature analysis”, according to (Griss et al., 1998).

RSEB framework is based on Unified Modeling Language (UML) (OMG, 2005) to model and specify applications systems, components and architecture, in order to express variability of the systems. The process framework is divided into:

• **Domain Engineering** that is organized in two processes (1) Application Family Engineering and (2) Component System Engineering; and
• **Application Engineering** consisting in (1) Capture requirements, (2) Perform robustness analysis, (3) Design, Implement, (4) Test and (5) Package the application system.

Based on the limitations of RSEB, the evolution of this process was presented later in 1998, where Griss et al. (Griss et al., 1998) extended RSEB model to represent the relationship between use case modeling and feature modeling. Such extension was necessary because RSEB used features informally and did not provide explicit feature models or steps to build it.

Taking into consideration that there is a direct relationship between use cases and features, it is seen that there is traceability between a process that focus on reuse oriented development, such as features in FeatureRSEB, and use cases, that are part of ordinary software development processes. (Griss et al., 1998) also question the relationship between features and use case modeling. Accordingly, use case and feature modeling are two representations that have different objectives, the use cases are user oriented, and features, which are reuser oriented. Although questioned in the process, that is no clear definition on how use cases and features are related.

### 3.2.6 Product Lines Software Engineering - PuLSE

On the other hand, Bayer’s et al. (Bayer et al., 1999) approach, PuLSE – Product lines Software Engineering, provide a customizable process to exploit experiences and know-how accumulated by isolated projects that are viewed as a line of projects with some common purpose. The engineer is around commonalities and variabilities to create domain-specific software architecture from an application centric approach. The advantages manage the optimization of development processes, cost reduction, time to market minimization, explicit knowledge representation and quality improvement. PuLSE’s methodology is based on an attempt to transition domain engineering expertise, by the use of documented methods, such as Synthesis (SPC, 1993), Commonality Analysis (Ardis, 1997) and Feature-oriented Domain Analysis (Kang et al., 1990).
The framework that is built under three main elements: Deployment Phase, Technical Components, and Support Components where each of them is composed into a set of activities. Deployment Phase describes activities performed to set up and use the product lines, organized into Initialization, Infrastructure Construction, Infrastructure Usage and Infrastructure Evolution and Management. Technical components are composed of Customization (BC), Scoping (Eco), Modeling (CDA), Architecting (DSSA), Instantiating (I) and Evolution and Management (EM). Support Components are organized into Project Entry Points, Organizational Issues and Maturity Scale. These components are necessary to assess the current organization in order to customize the framework, define the application area based on an economical analysis, model the area in terms of concepts and relationship, transition the domain model into a fully reusable design and specify an application engineering process that makes use of the reference architecture and maintain it over time.

Inside this scope, the framework talks about customization under three perspectives: (1) Value based PL development, to ensure that the customization is focused on return on investment perspective; (2) Model-driven PL development, helps selection of tools, meta-models and techniques to implement and maintain a PL infrastructure and (3) Reengineering-driven PL development, to capture advantages of previous development assets and experiences to systematically reuse them.

According to the framework, the initialization activity focuses on detailed steps to tailor PuLSE – Product lines Software Engineering – methodology and introduce it to an enterprise that will apply it. A workflow of activities is defined, according to Figure 3, where there are processes and products, guided by control flows and data flow, organized into three parts: Baselining, Evaluation and Customization. The last one requires deriving a complete process, including the definition of the workproducts used, their relations, and representations, based on decisions made during evaluation.

Moreover, from the processes assessed so far, PuLSE is the one that best describes how the core process can be applied. However, Bayer et al.
(Bayer et al., 1999) say that “the technology is not understood well enough to make it flexible enough to adapt to all sorts of environment context”.

Figure 3: PuLSE Initialization (Bayer et al., 1999)

3.2.7 Component Based Product Lines Development - KobrA

The Component-Based Product lines Development, KobrA approach (Atkinson et al., 2000), is a component based development method that was developed to support the practical application of a product lines development. “It is a synthesis of several advanced software engineering technologies, including product lines development, component based software development, frameworks, architecture centric inspections, quality modeling and process modeling”, according to (Atkinson et al., 2000). The objective is to increase software reuse level in industrial settings providing systematic approach to the development of high-quality component based applications. It
is a product line process that focuses on the creation of a framework to embody all product variants that are part of the product family.

The method distinctively describes products to be generated independent of the process that create them. Such products can be understood as models, documents, code modules, test cases etc. KobrA originates on PuLSE approach, which is an object oriented customization that already contains a software development process.

The process overall structure is divided into:

- **Framework Engineering Activity** that focuses on the creation of the framework which support all products variations, analyzing commonalities and variabilities, of the product lines. Components are described in two abstract levels: specification and realization. A framework, then, can be understood as an arrangement of components specifications and realizations, contemplating a reuse infrastructure for creating systems within a determined domain; and

- **Application Engineering Activity** is the instantiation of such framework to enable the creations of product variants in the product family. It is center on framework models and decision models where framework models recursively resolves decisions until generic models are transformed into specific models for particular applications. This process is split into two steps: (1) context realization instantiation and (2) framework instantiation.

To the point of view of adaptation, the process description lacks any specific guidance to this topic. There is guidance on how to produce a product lines and how to instantiate it to a particular scope, but it does not relate any direct interaction on how engineering activities are mapped into an established process. According to (Atkinson et al., 2000), the process states that “there is also an erroneous perception that product lines development is incompatible with regular development of single systems” but there is no further details on how KobrA can be compatible with a non reuse development
process. The process description is more concerned on how engineering activities are realized but not on their applicability.

### 3.2.8 Component-Oriented Platform Architecting Method – CoPAM

In the family of methods described by America et al. (America et al., 2000), CoPAM’s process – Component-Oriented Platform Architecting Method Family for Product Lines Family Engineering, there is a method to support the development of product family architectures, component based and commonalities and differences. The reason for such approach is justified in an environment of larger population of products with enough similarities to be developed as a population, coordinating marketing, development and sharing knowledge about product families, even if they do not share assets among themselves.

The framework supports the structure of the following sub-processes: (1) a Family Engineering Process dealing with family development; (2) Platform Engineering Process consisting of a number of reusable components; and (3) Product Engineering Process to develop using and adding platform components when necessary. Consequently, the Family Engineering Method Family is the set of family engineering processes that can be carried out either in parallel or consecutively, depending on the organization context and on the business choice.

The Family Engineering Process is based on the approaches of (DeBaud, 1998), and defines several steps to the detailed component architecture for the family. Although mentioned that there are several approaches that influence CoPAM processes, there are only references to (DeBaud, 1998) and (Jacobson et al., 1997) and (Jacobson et al., 1998).

Due to the wide scope that CoPAM intends to reach, there is no clear process definition and the steps to be followed are uncertain. Based on the process structure, one of the steps of Family Engineering sub-process is Define the Process, whose expected result is a Detailed Process Definition. Theses steps are comparable to the framework of (Jacobson et al., 1997) however there are no further regards on how the process can be defined.
3.2.9 Pervasive Computer Systems – PECOS

Winter et al. (Winter et al., 2002) approach, with PECOS process, address management of non-functional requirements and constraints, such as limited CPU power, memory and hard real-time, and architectural issues enabling component-base development for embedded system, in the domain of field devices. The process emphasizes that tools are necessary if applied systematically, in parallel to the process description.

The process framework concerns Component Development and Application Development, where the trigger for the development of a component is during application composition. Iterative development model is applied and each iteration uses the waterfall model, following requirement specification, design, implementation, documentation and testing. The process describes task for component development, such as (1) Requirement Elicitation and Analysis, (2) Interface Design, (3) Component Implementation, (4) Testing, (5), Profiling and (6) Documentation and Release. For the application development, the activities are (1) Preamble, (2) Identify Components, (3) Query Components, (4) Select Components, (5) Compose Components, (6) Application Tests and (7) Application Documentation and Deployment.

It is identified that Application Development out of components and component development itself share an ordinary vocabulary to describe the process activities. Requirement Elicitation regards functional and non-functional requirements, Component Architecture describes how components should be specified, Component Implementation can be taken through C++ or Java language and Testing is done by unit testing or test case description. Therefore, we can conclude that PECOS is not described under any specific platform to product lines development because its framework and activities are based on software engineering common practices. It does not regard domain engineering or product lines development approach. Therefore, there are no details on process adaptation because the process already mentions a non-reuse approach of process.
3.2.10 Feature Oriented Reuse Method – FORM

Feature Oriented Reuse Method, FORM (Kang et al., 1998), focuses on domain feature analysis, modeling commonalities and variability in the applications of a given domain and the use of features to systematically derive reusable domain components. FORM extends FODA (Kang et al., 1990) in software design phase by "prescribing how the feature model is used to develop domain architectures and components for reuse." The process states the necessity that reuse must be planned early in software development cycle, thus the necessity to introduce feature models, to capture the services provided by applications in the selected domain, during the early phases of the development.

The method framework consists of two major engineering processes:

- **Domain Engineering** that focuses on analyzing systems within a determined domain in order to produce a referenced architecture to be reuse by all applications in the reference domain; and

- **Application Engineering** consists on activities for developing applications by the use of components generated in domain engineering phase.

The core process describes specific details on how to provide a product lines based on features description. There is a process composition framework and case studies but no regards on its adaptation on organizations that already have a process instantiated.

3.2.11 Asset Based Development – ABD

Asset Based Development, ABD (RUP, 2003), is a RUP plug-in to organize software development in order to structure previous investments to influence future ones, reusing all documented software artifacts or workproducts. The objective is to accelerate development and reduce cost and risk.

Although mentioned that two separate processes are necessary for an effective reuse program, one to manage reuse at the organizational level and
the other to produce and consume assets, the plug in covers the second matter, structured in two workflows: (1) Asset production develops and/or harvests the artifacts to be included in the asset; and (2) Asset consumption locates and evaluates assets to be applied to the project and report feedbacks on the applied asset.

RUP provides role and artifacts definition to support the project, and also guidance on how to apply into a project lifecycle. ABD permeates the whole of RUP and the process of producing/reusing asset can be applied to any activity and any RUP artifact can be packed as a reusable asset. The definitions available and the overall structure is a positive remark to what the ABD is intended to do, providing guidance on how to:

- Tailor process for asset consuming projects;
- Tailor process for asset producing project;
- Select asset based development tools;
- Describe an extensible organization to software artifacts through the use RAS⁴; and
- Prepare project-specific reuse guidelines, among others.

On the other hand, ABD usage is tied to organizations that follow a RUP based process, where there is no correlation between ABD and other processes that do not follow RUP lifecycle definitions and therefore, narrowing its applicability.

### 3.3 Summary

This Chapter presented a review about adaptation in software development processes, summarized in Table 1. Such review is, together with the comprehension of the basic concepts regarding software development processes (reuse and non-reuse centric), necessary to formalize an adaptation process based on the survey about the possible techniques in which processes can be adapted.

---

⁴ RAS- Reusable Asset Specification: a pattern used to represent assets.
As briefly described in this Chapter, there is very little on practical
guidance regarding the adaptation of a reuse process into organization that
already have established processes. The approach presented by RUP and
PuLSE are the ones that best describes adaptation because they present
specific and detailed steps on how to instantiate a process into a project.

Therefore, RUP and PuLSE are part of the foundation for the
adaptation process and next Chapter will present what was used from each
process together with the process activities description.
### Table 1: Summary of adaptation techniques found on development processes.

<table>
<thead>
<tr>
<th>Development Approach</th>
<th>Process</th>
<th>Adaptation</th>
</tr>
</thead>
</table>
| **Non-reuse**        | RUP     | Process Tailoring in two levels:  
                        - Organizational: RUP adaptation to an organization context; and  
                        - Project Level: RUP adaptation to a project level. |
|                      | ROSE    | ROSE Reconfiguration through key principals:  
                        - process idiom;  
                        - evolution and maintenance phase; and  
                        - set of activities to apply in individual projects.  
                        Generic description, without details on how to perform. |
|                      | ODM     | Adaptation by: (1) Specialization; (2) Instantiation; and (3) Documentation |
|                      | FODA    | Adaptation in terms that the architectural model can include changes and domain analysis method influences the organization. |
|                      | RSEB / FeatureRSEB | Direct relationship between use cases and features, which has two representation with different objectives: user oriented (use cases) and reuser oriented (features). |
|                      | FORM    | No regards on its adaptation on organizations that already have a process instantiated |
| **Domain Engineering** | PuLSE  | Definition of a workflow of activities with processes and products, organized into three phases:  
                        - Baselining;  
                        - Evaluation; and  
                        - Customization. |
|                      | KobrA   | The process description lacks any specific guidance on process adaptation. There is guidance on how to produce a product line and how to instantiate it to a particular scope, but it does not relate any direct interaction on how engineering activities are mapped into an established process. |
|                      | CoPAM   | Family Engineering Sub-process defines a sequence of steps and one of them is Define the Process, whose expected result is a Detailed Process Definition. However there are no further regards on how the process can be defined. |
|                      | PECOS   | It is not described under any specific platform to product lines or domain engineering development approach because its framework and activities are based on software engineering common practices. Therefore, there are no details on process adaptation because the process already mentions a non-reuse approach of process. |
|                      | ABD     | There is a guidance on how to apply into a project lifecycle:  
                        - Tailor process for asset consuming projects;  
                        - Tailor process for asset producing project;  
                        - Select asset based development tools;  
                        - Describe an extensible organization to software artifacts through the use RAS; and  
                        - Prepare project-specific reuse guidelines, among others. |
Chapter 4

4 Adaptation Process

Based on the results of the research on software development processes, specifically on how adaptation is being treated by them, presented in Chapters 2 and 3, a set of requirements has been defined for the proposal presented by the Adaptation Process.

Nevertheless, the research realized was also necessary to select which processes were used as the foundations for the Adaptation Process definition. Besides, a Domain Engineering, Product Line and Non-reuse development approach were chosen to set the relation among the Adaptation basis.

Therefore, this Chapter is organized as it follows: Section 4.1 introduces the adaptation process; Section 4.2 describes the process overview in terms of its objectives, terminology and foundation followed by the process description on Section 4.3 and Section 4.4 summarizes the Chapter.

4.1 Introduction

Software engineering processes are written to encompass technical and managerial activities and, according to IEEE (IEEE, 1990), "processes are written in the plural meaning that there are many processes involved (e.g. Configuration Management Process, Development Process, etc.)", which comprehends the idea that many distinct areas of concern are gathered to set the development approach planned for an organizational context.

In another point of view, according do SWEBOK (SWEBOK, 2004), a "process definition can be a procedure, a policy or a standard" and their descriptions are relevant not only in large organizations, but on the contrary, process related activities have been performed successfully by small organizations, teams and individuals. However, it is necessary to focus on the
type of process definition required in each context, which depends on the reasons and necessities for defining it.

Therefore, in order to design a process and deliver it to the people who use it, “process engineers and methods gurus have found that some level of formalism and modeling capabilities are useful” (Kruchten & Robillard, 2003). Such formalism can be used to describe a concrete software development process that becomes better to offer some visualization of the intended development approach.

Accordingly, the Adaptation Process is intended to support the introduction of reuse concepts into software development processes that do not treat Domain Engineering activities, and is written based on a set of pre-defined objectives, terminology and foundations, depicted in more details on the following sections.

4.2 Process Overview

The Adaptation Process provides a disciplined approach to the execution of an adaptation between a reuse and non reuse process. It delivers a software development practitioner with standard based process to set up reuse practices into an organization that already has an institutionalized process. The key to achieving the delicate balance between the establishment of reuse culture into the organization and not modifying the existing practices is to understand the essentials of both processes and follow an adaptation between them both.

The process is depicted by an UML Activity Diagram, representing a typical sequence of events when conducting the flow of the work. It is composed of 6 activities, presented on Figure 4. The left flow of the activity diagram comprehends the execution of the process during the start of the project and first iteration that it is executed. The right side represents the maintenances of the adaptation process and each process activity is described in terms of purpose, steps, inputs and outputs.
4.2.1 Objectives

The adaptation process was generated under the following objectives:

- To define activities necessary to configure a process based on a reuse and a non-reuse process;
- To facilitate the comprehension on how both processes relate;
- To represent traceability between both reuse and non-reuse process vocabulary; and
- To establish an abstraction level of the basic elements of the processes, through a metamodel of process vocabulary, which helps to manage and understand the complexity of the traceability and correctly use their entities.

![Diagram: Adaptation Process Workflow](image-url)
4.2.2 Terminology

The adaptation process was developed making use of a group of basic concepts that are set as the basic terminology and they were captured from the processes that are part of the foundation to develop the adaptation.

Part of the concepts is related to domain engineering processes and these definitions were taken from (Almeida, 2007):

- **Domain:** In the software reuse community, the term domain encompasses not only the real world knowledge in a given problem area, but also the knowledge about how to build software systems in that area, corresponding to the domain as a set of system views;

- **Applications:** Products will be considered as applications. Core asset and product development will be referred, respectively, as domain and application engineering, as widely used by the reuse community, especially in the context of domain engineering; and

- **Feature:** An enduser-visible characteristic of a system, the FODA definition (Kang et al., 1990).

The other definitions are related to ordinary software development processes, and they were taken from RUP (RUP, 2003):

- **Phases:** The time between two major project milestones, during which a well-defined set of objectives is met, artifacts are completed, and decisions are made to move or not move into the next phase; and

- **Activities:** A unit of work a role may be asked to perform.

The next definitions were taken from PuLSE regarding the adaptation that is mention on its such process (Bayer et al., 1999):

- **Customization Factors:** Characteristics of a product line situation (e.g. type of application or amount of resources) that have an impact on the process for developing the product line.
The following definition was adapted from RUP, taking as a reference what it mentions about artifact, and will be used on the adaptation process to guide the elaboration of the process meta-model.

- **Process Vocabulary**: It can be understood as a piece of information that can be produced, modified, or used by a process. It can be a model, a model element, or a document.

### 4.2.3 Foundations

The foundations represent the reference in which the process was based, and are used to guide the definition of the general aspects and directions used by the process. Three main pillars are used as the foundations for the process, such as:

Process Adaptation was written under the considerations of three main pillars:

- **RiDE** – The domain engineering process chosen to be the one to guide activities related to the production of software components;
- **RUP** – The non-reuse software development process which has a discipline, Environment, that details how such process can be adapted to different contexts of software development; and
- **PuLSE** – a product line process that has specific activities regarding the adaptation of the process into different contexts.

A set of aspects, from each of the pillar mentioned, where highlighted to form the foundations for the development of the adaptation process. Therefore, these relevant aspects will be presented in the next section.

### 4.2.3.1 RiSE Approach for Domain Engineering – RiDE

RiDE, Domain Engineering Process Overview, was described with the intuition to support the development of components, according to a defined group of steps to facilitate the creation of a reuse context in which components will be further used in a product line environment. It was developed based on the best practices available on reuse community and the
process can be applied in contexts that aim the introduction of a disciplined reuse practice. It was described in terms of the collection, organization and storing past experiences in building software in a specific domain.

The following concepts are the basis for the process definition and will also be used for the adaptation, according to what is presented below:

- **Domain**: corresponds to the aggregated knowledge in a given problem area, and in software reuse community domain also corresponds to the knowledge related to the construction of software systems in a determined area;

- **Feature**: this concept refers to the granularity in which requirements are treated in a specific domain, and the definition of Kang (Kang et al., 1900), that states “Features and Feature Models are widely used in domain analysis to capture the commonalities and variabilities of systems in a domain” is the one used by domain engineering process.

- **Applications**: corresponds to the product generated by the product line of a specific domain;

RiDE is based on some foundations presented below:

- **Process Model**: “corresponds to the abstract representation of a software process” (Sommerville, 2006) and is used to represent processes from a certain perspective, as for example Waterfall Model, Evolutionary Development, Component Based Software Engineering (CBSE), Spiral, among others. RiDE process is based on Spiral Model with some characteristics of CBSE model, due to the part related to the development of software from reusable assets.

- **Domain Driven**: the process is focused on a set of applications from a specific domain, identifying common and specific features from the application family.

- **Iterative and Incremental**: refers to how projects are conducted in terms of how the steps are followed and incremented according to
the growth of the product. Iterations are controlled to be carried out as planned in order to gain the benefits provided by the way it is produced.

- **Software Architecture**: concerning specific structural organizations by which systems are organized and interact between each other, separating the overall structure of a system in terms or components and their relations.

- **Component Based Development**: it arose in software engineering community as an attempt to fill in the gaps left by object oriented development that did not support the construction of modularized architecture based on components that could be easily changing without affecting other parts of the architecture.

- **Design Patterns**: are based on the idea that there are patterns for solving the same class of problems, describing a recurring design problem to be solved, the related solution and context of application.

The RiDE process is composed of three steps, which are divided into activities and sub-activities. They can be applied separately and each step has an approach for specific domain engineering lifecycle.

- **Domain Analysis**: it focuses on the analysis of systems collectively, rather than separately in order to identify common and variables characteristics of the domain. The work addresses issues on “how to perform”, introducing systematic way to develop domain analysis. The step consists of three activities: Plan Domain, Model Domain and Validate Domain.

- **Domain Design**: its general purpose is to produce Domain-Specific Software Architecture (DSSA), a common architecture for a set of applications, where the outputs of domain analysis are mapped to design specification. The objective is that a main architecture should be used by all applications in the domain, for the production of domain specific and reference architectures. Domain Design step consists of five activities: Decompose Module, Refine Module,
4.2.3.2 Rational Unified Process – RUP

The Rational Unified Process is a software engineering process focused on providing an approach to define tasks and responsibilities within software engineering context. The process is based on structured disciplines that map activities to be executed by defined roles into a software development process organization. It is organized according to Figure 5, where the horizontal axis illustrates dynamic aspects of the process, representing time and lifecycle aspects as it unfolds. The vertical axis represents the static characteristics of the process, presenting how it is described in terms of components, disciplines, activities, workflows, artifacts, and roles.

Within this scope, a discipline is a collection of related activities that are grouped to a major area of concern and they are structured in order "to describe a set of associated activities and artifacts based around a common skillset" (RUP, 2003). Each RUP discipline describes basic concepts, workflow with a sequence of events and guidelines to execute specific practices.
Within this scope, RUP’s disciplines Environment focuses on activities to configure an executable process, based on the core description of RUP. The process configuration relies on a process definition to support the development team taking into consideration the environmental characteristics. The activities that compose environment discipline are:

- **Prepare Environment for the Project**: it has the objectives to turn the overall process into a project specific process. It involves defining how the project is going to execute the process and describing the process deviations. Nevertheless, it is necessary to organize artifacts according to the time that they are going to be produced and prepare process guidelines to the project in order to generate a list of tools to be used during the course of the project. The preparation of the process environment should start in the first phase of the project and the plan is assessed and it evolutes throughout project development.

- **Prepare Environment for an Iteration**: it has the goal to ensure that the project process is ready for the upcoming iteration. The execution of this activity is focused to complete development case for the iteration, prepare and customize tools for project execution,
provide templates and guidelines to support project execution and communicate project team on process changes.

- **Support Environment During an Iteration**: it has the objective to support developers and project team on the use of both process and tools of the project. It is an ongoing activity so that the team can execute their job properly. The project environment needs continuous attention due to the fact that improvement opportunities can appear at any moment.

### 4.2.3.3 Product Line Software Engineering – PuLSE

According to the review on domain engineering and software product lines processes, PuLSE is the one that best presents steps detailing adaptation from its framework into a specific scope of project. Therefore, its activities and steps were used as a basis for the definition of the adaptation process.

Within this scope, Deployment Phase “*describes activities performed to set up and use a product line*” (Bayer et al., 1999). The first phase defined by the process is Initialization Phase and focuses on baseline the enterprise, and customize PuLSE approach where the information required for tailoring is defined by customization factors. They can be understood as “*characteristics of the product line that have impact on the process for developing the product line (e.g. type of application or amount of resources)*” (Bayer et al., 1999).

The initialization phase is split into three parts:

- **Baselining**: that has the objective to gather information for tailoring PuLSE. First, there is the elicitation of customization factors, which are not independent and one can influence the other. Then, baselining strategies are used on the selected customization factors through the guidelines provided to group necessary information and record the values determined for the enterprise;

- **Evaluation**: it captures the dependencies among customization factors to determine the effects on the components of PuLSE and evaluates impact on process parts to generate raw instantiation profile which contains decisions made for customization (e.g. type
of workproducts). If the current profile matched the existing one in the organization, no such customization will be necessary and the existing one can be reused; and

- **Customization:** it defines the complete process (workproducts, their relationship and their representation) based on decision made on evaluation part, including the definition of the process, products and attributes. By the end, a consolidation of the process definition is made.

### 4.3 Process Activity Description

The software process adaptation is composed of six activities: **Understand Context, Define the Process, Consolidate the Process and Maintain the Process.** The activity Define the Process is decomposed into Define Process Phases, Define Process Activities and Define Process Vocabulary. The description of each activity is contemplated in this section where each of them is described in terms of Purpose, Steps, Inputs and Outputs.
### 4.3.1 Understand Context

<table>
<thead>
<tr>
<th><strong>Purpose:</strong></th>
<th>Gather project information to support process adaptation.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Steps:</strong></td>
<td>Elicit customization factors regarding product line characteristics; such as:</td>
</tr>
<tr>
<td></td>
<td>- Product Line Domain;</td>
</tr>
<tr>
<td></td>
<td>- Non-reuse Development Process Approach;</td>
</tr>
<tr>
<td></td>
<td>- Domain Engineering Approach;</td>
</tr>
<tr>
<td></td>
<td>- Product Line Technology;</td>
</tr>
<tr>
<td></td>
<td>- Project Location;</td>
</tr>
<tr>
<td></td>
<td>- Project Team; and</td>
</tr>
<tr>
<td></td>
<td>- Project Objectives.</td>
</tr>
<tr>
<td></td>
<td>Select relevant customization factors that will be used to guide process adaptation,</td>
</tr>
<tr>
<td></td>
<td>Determine whether there are any dependency among customization factors;</td>
</tr>
<tr>
<td></td>
<td>Save the current profile for the company containing the relevant customization factors for that organization;</td>
</tr>
<tr>
<td></td>
<td>Describe any necessary deviations of the process;</td>
</tr>
<tr>
<td></td>
<td>Understand and define the scope of the process, contemplating which areas of application they will reach; and</td>
</tr>
<tr>
<td></td>
<td>Analyze project information that has an impact on the process for developing a product line.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Input:</strong></th>
<th>Reuse Process</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-reuse Process</td>
</tr>
<tr>
<td></td>
<td>Product Line Characteristics</td>
</tr>
</tbody>
</table>

| **Output:**  | Project Customization Factors. |
### 4.3.2 Define Process

#### 4.3.2.1 Define Process Phases

<table>
<thead>
<tr>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td>• Collect phases characteristics from processes in order to set up the adapted phases to be followed</td>
</tr>
</tbody>
</table>

| Steps: |
| • Understand processes phases description; |
| • Understand processes phases objectives; |
| • Asses and understand domain engineering process phases description; |
| • Asses and understand non-reuse process phases description; |
| • Adapt non-reuse process to introduce domain engineering processes objectives and description. |
| • Generate process phases map describing the relation between both processes. |

| Input: |
| • Domain Engineering Phases |
| • Non-Reuse Process Development Phases |

| Output: |
| • Non-Reuse Process Development Phases Adapted |
| • Process Phases Map |
### 4.3.2.2 Define Process Activities

<table>
<thead>
<tr>
<th>Activity</th>
<th>Purpose:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Collect activities characteristics from both processes in order to set up the adapted activities to be followed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Steps:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Understand processes activities description;</td>
</tr>
<tr>
<td>• Understand processes activities objectives;</td>
</tr>
<tr>
<td>• Asses activities inputs;</td>
</tr>
<tr>
<td>• Asses activities outputs;</td>
</tr>
<tr>
<td>• Adapt non-reuse process to introduce domain engineering processes activities;</td>
</tr>
<tr>
<td>• Set up domain engineering activities as specific guidelines on non-reuse process.</td>
</tr>
<tr>
<td>• Generate process activities map.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Domain Engineering Process activities;</td>
</tr>
<tr>
<td>• Non-reuse process activities;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Non-reuse process activities adapted;</td>
</tr>
<tr>
<td>• Process Activities Map</td>
</tr>
</tbody>
</table>
4.3.2.3 Define Process Vocabulary

**Activity**

**Purpose:**
- Identify the terminology and basic concepts of each process in order to correlate them.

**Steps:**
- Understand processes vocabulary description;
- Understand processes vocabulary objectives;
- Map process vocabulary of terms (terminology);
- Generate a process metamodel of terms that represents the traceability between process vocabulary;
- Adapt non-reuse process to introduce domain engineering processes vocabulary;
- Generate process vocabulary map as guidelines on how to use the process;

**Input:**
- Domain Engineering Process Terminology
- Conventional Software Development Terminology.

**Output:**
- Terminology vocabulary of terms traceability metamodel
- Artifacts adapted
- Vocabulary of Terms
### 4.3.3 Consolidate Process

<table>
<thead>
<tr>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td>• Turn the underlining process into an enactable process contemplating both reuse and non-reuse matters; and</td>
</tr>
<tr>
<td>• Consolidate information acquired in order to provide a complete process description.</td>
</tr>
<tr>
<td><strong>Steps:</strong></td>
</tr>
<tr>
<td>• Derive a complete process by joining the workproducts, their relations and representation based on the decisions of previous activities.</td>
</tr>
<tr>
<td>• Adapt process templates for the project;</td>
</tr>
<tr>
<td>• Prepare guidelines for the project;</td>
</tr>
<tr>
<td><strong>Input:</strong></td>
</tr>
<tr>
<td>• All information acquired on previous activities.</td>
</tr>
<tr>
<td><strong>Output:</strong></td>
</tr>
<tr>
<td>• Adapted Process Description; and</td>
</tr>
<tr>
<td>• Adapted Process Templates.</td>
</tr>
</tbody>
</table>
### 4.3.4 Maintain Process

<table>
<thead>
<tr>
<th><strong>Activity</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td>• Maintain the process;</td>
</tr>
<tr>
<td>• Support the development with the process.</td>
</tr>
<tr>
<td><strong>Steps:</strong></td>
</tr>
<tr>
<td>• Evaluate the process to assess project changes;</td>
</tr>
<tr>
<td>• Evaluate the process templates to assess necessary changes;</td>
</tr>
<tr>
<td>• Capture lessons learned from the process to improve its description;</td>
</tr>
<tr>
<td><strong>Input:</strong></td>
</tr>
<tr>
<td>• Process description</td>
</tr>
<tr>
<td>• Process artifacts</td>
</tr>
<tr>
<td>• Process vocabulary</td>
</tr>
<tr>
<td><strong>Output:</strong></td>
</tr>
<tr>
<td>• Process description adapted</td>
</tr>
<tr>
<td>• Process artifacts adapted</td>
</tr>
<tr>
<td>• Process vocabulary adapted</td>
</tr>
</tbody>
</table>

### 4.4 Summary

This Chapter presented the **Software Process Adaptation** together with the objectives, approach used as terminology and the foundations of the process. The Adaptation Process suggests a workflow and a set of activities peculiar to the scope of adaptation between a reuse and non-reuse process.

The process was formatted taking as a basis on the fundamentals of a reuse process, RiDE, a non-reuse process, RUP, and a software product line process, PuLSE and Table 2 summarizes the contributions from each of them to the Adaptation Process. The reuse process used was RiDE, a domain engineering process focused on the production of reusable assets. RUP was
used as the basis for the non-reuse process, specifically on the Environment
discipline that describes how a process is instantiated into an specific project
context. Finally PuLSE, a product line process approach, was used in terms of
the description on how it can be adapted to a project context.

Consequently, the Adaptation process is described in terms of a UML
Activity diagram, contemplating activities that are executed throughout
software development lifecycle. The evolution of the process, regarding the
overview presented on Chapter 3, in addition to the foundations for the
process, is related in terms that:

- Condensates the adaptation techniques described in PuLSE and
  RUP;
- Extends the process description to include:
  - definition of customization factors;
  - definition of project’s phases based on domain engineering
    and RUP’s process phases;
  - definition of project’s activities that contemplate both scopes
    of process activities;
  - definition of a process vocabulary of terms, contemplating
    process item and description;
  - generation of a metamodel to contemplate the traceability
    describing the correlation between both reuse and non-reuse
    processes vocabulary;

Upon the process definition, the next Chapter will present an
experience report contemplating the use of process adaptation in two
contexts.
Table 2: Foundations for the Adaptation Process

<table>
<thead>
<tr>
<th>Process</th>
<th>Description of the contributions for the definition of the Adaptation Process</th>
</tr>
</thead>
</table>
| RiDE    | The **Process Definitions** were used as a basis for the Adaptation Process, specifically:  
- Domain:  
- Applications:; and  
- Features:  
The **Foundations** were aggregated to the foundations of process adaptation:  
- Spiral Process Model;  
- Domain Driven;  
- Iterative and Incremental lifecycle;  
- Software Architecture;  
- Component Based Development; and  
- Design Patterns: |
| RUP     | **Environment Discipline** describes activities and steps to configure an executable process:  
- Prepare Environment for the Project;  
- Prepare Environment for an Iteration; and  
- Support Environment During Iteration. |
| PuLSE   | **Deployment Phase describes** activities to set up and use a product line, with the objective to:  
- collect project information for tailoring through the use of customization factors;  
- capture dependencies among customization factors; and  
- define the process for the project. |
Chapter 5

5 Case Studies

Beyond the definition of the Adaptation Process represented by a group of activities to adapt a reuse based process into a non-reuse process, two case studies were conducted to evaluate the process. Each case study was carried out in a specific context of project that will be further described in this Chapter. The results analyzed represented a set of information that was used to improve and evaluate the viability of the adaptation process.

This Chapter is organized as it follows: Section 5.1 presents the introduction on the execution of the case study, followed by the first case study execution on Section 5.2 and second case study on Section 5.3. Section 5.4 comments the general findings of the work and Section 5.5 summarizes the case study.

5.1 Introduction

As stated in Chapter 4, one of the objectives of the proposed process is to facilitate the comprehension on how both reuse and non-reuse processes relate in order to assist the introduction of formal and planned reuse practices in organizations that already have institutionalized processes. Within this scope, to properly validate the proposed objective, two case studies were executed to assess and refine it through the collection of lessons learned.

The case studies were executed in two different contexts, by different teams and at a different time. It was necessary to define the objectives and scenario in which the project was executed, in order to comprehend the necessities of each client to adapt the process according to the domain characteristics.

Another part of the case study definition was to choose the Domain Engineering Process and Non-Reuse Process that the project would execute. The domain engineering process is responsible for the steps to produce
components that can be reused on further projects and the Non-Reuse Process is responsible for the definitions of the activities, roles and lifecycle of the project. This is important to compose the basis of the adaptation.

Besides that, the process execution was followed to validate whether the objectives of each activity of the adaptation matched the validation of the results, according to what will be presented in the following sections.

5.2 Case Study 1

5.2.1 Understand Context

The first case study was conducted to apply domain engineering process, under the coordination of RiSE\(^5\) in partnership with CESAR\(^5\) and CIn\(^6\). The project ran together with a post-graduate discipline, where the focus was the application of a reuse process into a non-reuse process. The Domain Engineering Process chosen was RiDE (Almeida, 2007), that was in progress as a doctoral thesis, and at that time, only Domain Analysis Process was defined, Domain Design and Domain Implementation were in the course of their definition. The Non-Reuse Process used for this case study was a RUP based software development process.

The domain scope was based on the necessity to specify and develop a group of tools to support software reuse and the solution would affect System Analysts, Software Architects, Developers and Assets Consumptions. A good solution to the problem would be the specification of a software product line for reuse repository systems.

The project started in September 2005 and lasted five months to build a product line. The team was composed of project manager (1), architect (1), team leader (1), analyst (2), system engineer (9), quality engineer (2), and configuration manager (1) and RiSE's staff supporting the project, totalizing 10 people that could assume more than one role, depending on the project phase.

---

\(^5\) CESAR – Recife Center for Advanced Studies and Systems: http://www.cesar.org.br  
\(^6\) CIn – Informatics Center of Federal University of Pernambuco: http://cin.ufpe.br
This information was collected during the activity Understand Context, and can be summed up as the customization factors of the product line, which is best presented by the Table 3.

Table 3: Selected Customization Factors Summary

<table>
<thead>
<tr>
<th>Product Line Domain</th>
<th>Reuse Repository System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good Project Solution</td>
<td>Specification of a software product line for reuse repository systems</td>
</tr>
<tr>
<td>Software Process Development</td>
<td>RUP Based Development Process</td>
</tr>
<tr>
<td>Domain Engineering Approach</td>
<td>RiDE</td>
</tr>
<tr>
<td>Technology</td>
<td>Java</td>
</tr>
<tr>
<td>Project Site</td>
<td>CESAR.</td>
</tr>
<tr>
<td>Project Team Size</td>
<td>10</td>
</tr>
</tbody>
</table>

The objectives of the first case study were:

- to apply the process to the domain engineering process;
- assess the results in order to provide the lessons learned and improvements on the process;
- investigate the state-of-the-art of domain engineering processes applied to an organizational culture, and
- apply mapping strategies (Cavalcanti et al., 2006) to an organizational process in order to produce a meta-model describing how non-reuse and reuse processes related to each other.

5.2.2 Process Definition

The process adaptation of the first case study followed a certain degree of difficulty due to the fact that Domain Engineering Process – RiDE – was a work in progress therefore, the traceability between RiDE and the non-reuse process contemplated mainly the entities of domain analysis process.

5.2.2.1 Define Phases

To the definition of the adapted process phases, it was necessary to understand the objectives of RiDE process phases. After an analysis, we
could generate Table 4 that contemplates the correlation among process phases. Accordingly, RiDE follows an iterative development cycle, and the matching to RUP based process phases was linear due to the fact that the objectives of domain engineering process could be introduced into the objectives achieved by the end of RUP’s phases.

<table>
<thead>
<tr>
<th>Development Process Phases</th>
<th>Domain Engineering Process Phases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>Domain Analysis</td>
</tr>
<tr>
<td>Elaboration</td>
<td>Domain Design</td>
</tr>
<tr>
<td>Construction</td>
<td>Domain Implementation</td>
</tr>
<tr>
<td>Transition</td>
<td>No related phase</td>
</tr>
</tbody>
</table>

5.2.2.2 Define Activities

To the definition of the process activities, a correlation between RiDE and RUP was also necessary. Domain engineering activities were assessed in order to understand in which activity of RUP it would best fit. Based on the analysis, we could observe that domain analysis activities best fit the activities from requirement discipline. The objectives and constraints of domain engineering activities were introduced in the related RUP activity. Table 5 represents the processes activities map, contemplating the relation among RUP and RiDE’s activities, focusing only on Domain Analysis due to the fact that the other phases were not going to be assessed by that time, according to what was defined on Section 5.2.1.
Table 5: Process Activity Map

<table>
<thead>
<tr>
<th>Development Process Activities</th>
<th>Domain Engineering Process Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>Domain Analysis</td>
</tr>
<tr>
<td>Requirement Discipline – Activities: (1) Analyze the problem, (2) Understand Stakeholders Needs, (3)</td>
<td>Plan Domain</td>
</tr>
<tr>
<td>Requirement Discipline – Activities: (1) Define the System</td>
<td>Model Domain</td>
</tr>
<tr>
<td>Requirement Discipline – Activities: (1) Define the System</td>
<td>Validate Domain</td>
</tr>
</tbody>
</table>

5.2.2.3 Define Process Vocabulary

The definition of the process vocabulary and the relationship between them represents the traceability between both reuse and non-reuse processes. Table 6 represents the collection of the vocabulary. After that, it is necessary to correlate them, in order to clearly understand their traceability. The process vocabulary traceability could be best presented by an UML class diagram, contemplating the detailed relationship between processes, according do Figure 6.

The generation of the traceability was based on the work of (Eriksson et al., 2005), with some considerations: (1) sub-feature entity was added in order to group a lower level of feature granularity and (2) use case realization and parameter are not treated in our scope, excluding from the referenced meta-model.
Table 6: Process Vocabulary

<table>
<thead>
<tr>
<th>Process Vocabulary</th>
<th>Definition:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain Engineering</strong></td>
<td></td>
</tr>
<tr>
<td>Domain Model</td>
<td><em>Domain:</em> In the software reuse community, the term domain encompasses not only the real world knowledge in a given problem area, but also the knowledge about how to build software systems in that area, corresponding to the domain as a set of systems view (Almeida, 2007).</td>
</tr>
<tr>
<td>Feature</td>
<td>An enduser-visible characteristic of a system, the FODA definition (Kang et al., 1990).</td>
</tr>
<tr>
<td>Sub-feature</td>
<td>The smallest unit of system characteristics that could be grouped to form a feature.</td>
</tr>
<tr>
<td>System</td>
<td>A collection of connected units that are organized to accomplish a specific purpose. A system can be described by one or more models, possibly from different viewpoints (RUP, 2003).</td>
</tr>
<tr>
<td>System Family</td>
<td>A group of related system within the same domain.</td>
</tr>
<tr>
<td><strong>Non-reuse</strong></td>
<td></td>
</tr>
<tr>
<td>Use Case</td>
<td>A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor. A use case contains all flows of events related to producing the &quot;observable result of value&quot;, including alternate and exception flows. More formally, a use case defines a set of use-case instances or scenarios (RUP, 2003).</td>
</tr>
<tr>
<td>Scenario</td>
<td>A specific sequence of actions that illustrates behaviors. A scenario may be used to illustrate an interaction or the execution of one or more use-case instances (RUP, 2003).</td>
</tr>
</tbody>
</table>

Due to lack of link between reuse and non-reuse processes, the main difficulty faced was to correlate features, requirements and use cases. Therefore, one of the questions of the meta-model adopted was the semantic difference between features and sub-features, which was not clear. In the domain analyzed, there were features and sub-features with the same...
granularity of representation and it was necessary to better define the taxonomy of the processes adaptation.

Besides the meta-model, and based on that, the process artifacts were adapted to contemplate the related vocabulary. Due to the fact that non-reuse process does not treat domain engineering process vocabulary, it was necessary to adapt the artifacts to include its specifications. For that some artifacts were adapted and others were created. Table 7 summarizes the modifications.

The project was executed in one iteration of the iterative lifecycle, according to what is defined on the non-reuse process, following the phases of Planning, Design, Construction and Closing. During domain analysis, nine repository systems were analyzed and over seventy features were generated.

Based on that, the results were grouped by phases, describing strong and week points of the process. In the project's first phase, domain analysis, the group started by the process instantiation to an organizational process, following iterative and incremental lifecycle, and mapping activities from one process to another.

In the second phase of the project, the domain design process was adapted and the meta-model refined to include the component entity. The objective of the process was to build a domain architecture based on components, generalized for effective use across the domain of repository systems and composed in a standardized structure, effective for building successful applications.
Table 7: Adapted Process Artifacts

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement Document</td>
<td>Artifact adapted in terms to contemplate domain requirements, not only a single project scope. The adaptation also included a view of the domain applications available.</td>
</tr>
<tr>
<td>Feature Matrix</td>
<td>Artifact used to define the relationship among features and to calculate their variability in the product line, which was introduced because it is related to RiDE process.</td>
</tr>
<tr>
<td>Feature Model</td>
<td>Artifact that graphically represents relationships and variability of the features, which was introduced because it is related to RiDE process.</td>
</tr>
<tr>
<td>Use case document</td>
<td>It was not adapted, the difference is that the use case described are related to the domain in question.</td>
</tr>
</tbody>
</table>

The result analysis demonstrated complex traceability between features and classes generated, where the addition of one class changed all measurements and models generated by the domain. Besides that, process automation and tools are necessary to manage a product line, which were not available at that time.

5.3 Case Study 2

5.3.1 Understand Context

The second case study was also based on the RiDE domain engineering process, along with RiSE and CESAR. But at this time, RiDE Process had all his phases completed, so Domain Design and Implementation could be best assessed and adapted. The team used the templates defined on the first case study to refine and reuse them on the second; therefore, the non-reuse
process used was the same as before, a RUP based software development process.

In this cycle, the domain scope was arcade games of fixed fires, three games were analyzed and over 43 features were generated.

The project lasted five months to build a software product line. Its team was composed of project manager (1), architect (2), domain analyst (2), system engineer (6), quality engineer (2), configuration manager (1) and test engineer (1), totalizing 7 people that could assume more than one role, depending on the project phase.

This information was collected during the activity **Understand Context** is best summarized by Table 8.

| Table 8: Selected Customization Factors of Case Study 2 |
|---------------------------------|---------------------------------|
| **Product Line Domain** | Arcade games of fixed fires |
| **Good Project Solution** | Specification and implementation of a software product line for arcade games of fixed fires |
| **Software Process Development Approach** | RUP Based Development Process |
| **Domain Engineering Approach** | RiDE |
| **Technology** | Java |
| **Project Site** | CESAR\(^7\). |
| **Project Team Size** | 7 |

The objectives, besides the ones applied in the first case study, were:

- Refine the traceability meta-model, describing the changes and consolidate its final version;
- Identify which steps are necessary for a domain analysis tool, which was the greatest difficulty faced by the first case study; and
- Verify if the use of OSGi\(^8\) for the implementation phase is valid to any SPL.

---

\(^7\) CESAR – Recife Center for Advanced Studies and Systems: http://www.cesar.org.br

\(^8\) OSGi – Open Services Gateway Initiative
5.3.2 Process Definition

5.3.2.1 Define Phases

The adapted process phases definition followed the same approach applied on the first case study and, again, the objectives and general considerations of RiDE phases objectives and milestones were mapped to the existent non-reuse development process. At this time, a better representation and mapping of the phases could be realized due to the fact that domain engineering process was in a more mature way. All the three phases had a better description and could be best mapped to the non-reuse phases. However, the mapping from the first case study did not change because neither domain engineering process nor non-reuse processes changed, they were the same applied in the first case study, according to what is seen on Table 9.

<table>
<thead>
<tr>
<th>Development Process Phases</th>
<th>Domain Engineering Process Phases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>Domain Analysis</td>
</tr>
<tr>
<td>Elaboration</td>
<td>Domain Design</td>
</tr>
<tr>
<td>Construction</td>
<td>Domain Implementation</td>
</tr>
<tr>
<td>Transition</td>
<td>No related phase</td>
</tr>
</tbody>
</table>

5.3.2.2 Define Activities

Accordingly, the relation between both processes activities could be best mapped because RiDE process was more mature in terms of the process definition and description. Domain design and implementation phases were finished and the complete relation to RUP’s activities could be finished, as seen on Table 10.

---

8 OSGi technology provides a service-oriented, component-based environment for developers and offers standardized ways to manage the software lifecycle. http://www.osgi.org/
Table 10: Process Activities Map of Case Study 2

<table>
<thead>
<tr>
<th>Development Process Activities</th>
<th>Domain Engineering Process Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inception</strong></td>
<td></td>
</tr>
<tr>
<td>Requirement Discipline – Activities: (1) Analyze the problem, (2) Understand Stakeholders Needs, (3)</td>
<td>Domain Analysis</td>
</tr>
<tr>
<td>Requirement Discipline – Activities: (1) Define the System</td>
<td>Plan Domain</td>
</tr>
<tr>
<td>Requirement Discipline – Activities: (1) Define the System</td>
<td>Model Domain</td>
</tr>
<tr>
<td></td>
<td>Validate Domain</td>
</tr>
<tr>
<td><strong>Elaboration</strong></td>
<td></td>
</tr>
<tr>
<td>Analysis and Design Discipline – Activity: Define Architecture Candidate</td>
<td>Domain Design</td>
</tr>
<tr>
<td>Analysis and Design Discipline – Activity: (1) Refine Architecture</td>
<td>Decompose Module</td>
</tr>
<tr>
<td>Analysis and Design Discipline – Activity: (1) Design Component</td>
<td>Refine Module</td>
</tr>
<tr>
<td></td>
<td>Represent Variability</td>
</tr>
<tr>
<td></td>
<td>Define Component</td>
</tr>
<tr>
<td></td>
<td>Represent Domain Architecture</td>
</tr>
<tr>
<td><strong>Construction</strong></td>
<td></td>
</tr>
<tr>
<td>Implementation Discipline – Activity: (1) Implement Components</td>
<td>Domain Implementation</td>
</tr>
<tr>
<td></td>
<td>Component Implementation</td>
</tr>
<tr>
<td></td>
<td>Component Documentation</td>
</tr>
<tr>
<td><strong>Transition</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No related phase</td>
</tr>
</tbody>
</table>

5.3.2.3 Define Process Vocabulary

For the second case study, the definition of the process vocabulary and metamodel to represent the relation among them had to go through some modifications. Domain engineering process was executed and the metamodel was refined to what is represented on Figure 7: (1) the requirement class was included between the feature and use cases association; (2) sub-feature was represented as a relation and not as class; and (3) the class representing the structure of the domain model was also included. Besides, this meta-model also presented the relation between use cases and components implementation.
Such modifications were done due to the necessity of:

- A better representation and documentation of the domain, that’s why requirements class was added;
- There was no semantic difference between feature and sub-feature entity of the first case study meta-model; and
- The introduction of domain entity.

They were done due to the necessity of a clearer representation of domain model, in a structured form, to identify requirements to build a domain analysis tool and improve estimative method for a product line.

The process vocabulary, that composes the domain traceability meta-model, is presented on Table 11.
Table 11: Process Vocabulary Map Applied on the Second Case Study

<table>
<thead>
<tr>
<th>Process Vocabulary</th>
<th>Definition:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain Engineering</strong></td>
<td></td>
</tr>
<tr>
<td>Domain Model</td>
<td>Domain: In the software reuse community, the term domain encompasses not only the real world knowledge in a given problem area, but also the knowledge about how to build software systems in that area, corresponding to the domain as a set of systems view (Almeida, 2007).</td>
</tr>
<tr>
<td>Feature</td>
<td>An enduser-visible characteristic of a system, the FODA definition (Kang et al., 1990).</td>
</tr>
<tr>
<td>System</td>
<td>A collection of connected units that are organized to accomplish a specific purpose. A system can be described by one or more models, possibly from different viewpoints (RUP, 2003).</td>
</tr>
<tr>
<td>Components</td>
<td>A modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces (OMG, 2005).</td>
</tr>
<tr>
<td><strong>Non-reuse</strong></td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td>A requirement describes a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document (RUP, 2003).</td>
</tr>
<tr>
<td>Use Case</td>
<td>A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor. A use case contains all flows of events related to producing the &quot;observable result of value&quot;, including alternate and exception flows. More formally, a use case defines a set of use-case instances or scenarios (RUP, 2003).</td>
</tr>
</tbody>
</table>

Domain design and implementation phases executed based on process techniques focused on domain engineering processes. Since the product line was not completed implemented, the OSGi objective had not been verified. However, once the OSGi technology provides a service-oriented architecture that enables components to dynamically discover each
other for collaboration, the use of this technology will probably facilitate the identification of variabilities in the software product line.

Another difficulty faced was the analysis of the chosen domain which was unknown to the majority of the team, outstanding the lack of experience for the identification of features.

5.4 General Findings

Upon the execution of the case study, some general aspects were found for the result analysis, according to the following:

- **Process Phases Definitions**: the execution of the projects help make a correlation between both process and the findings demonstrates that Domain Analysis is directly related to a phase that focuses on requirements elicitation and definition (Inception Phase on RUP). The same analogy can be made on Domain Design and an architecture definition phase (Elaboration Phase on RUP) and Domain Implementation to a phase that contemplates activities to build the product.

- **Process Activities Definitions**: it is observed that activities related to domain engineering can be introduced in a non-reuse process in terms of guidelines, describing specific details and techniques to realize a product line. In this scenario, the activities from the non-reuse process are incremented to include these guidelines because they need to be executed in order to generate the final product;

- **Process Vocabulary Metamodel**: it is a necessary tool to formalize the interaction between the entities of a reuse and non-reuse process. It is helpful to guide process adaptation, in terms of the process description, and the adaptation of the project artifacts; and

- **Domain Experts**: it is important to have experts in the project domain chosen so that the production of components can be efficiently executed. engineering process to execute the project.
5.5 Summary

This Chapter presents the results acquired from the experiments to apply a traceability model to assess the introduction of reuse process in organizations that have a institutionalized process.

Next Chapter concludes this work by presenting the findings of the work, reviewing some related works, listing the future works that are resulted from this work and concluding remarks.
Chapter 6

6 Conclusion and Future Work

Upon the definition and execution of the process on two cases studies, an evaluation about the findings of the research on software process adaptation, some conclusions and comparisons can be delineated in order to indicate future works.

This chapter is organized as follows: Section 6.1 summarizes the findings of the work and Section 6.2 presents a comparison with some related works. Section 6.3 points some directions for future enhancements and research opportunities unexplored by this work and, finally, Section 6.4 contains a concluding discussion on the topics covered in this dissertation.

6.1 Summary of the Findings

This Section summarized the main findings and contributions of the work, according to what is enumerated bellow:

- A definition of a process to adapt non-reuse and reuse based process is only one part of the work that needs to be done in order to help the introduction of reuse practices; and
- The introduction of reuse practices into complex organizations requires a complete evaluation of the existing restrictions on the context, not only procedurally talking, but also to understand business and general environmental constraints.

6.2 Related Work

In the literature researched, depicted on Chapter 3, some related work could be identified based on an overview realized in two main topics: Adaptation Techniques on Non-reuse process and Adaptation on Reuse processes. Eleven development processes were assessed, and a set of approaches were
analyzed in terms of what they consider about process adaptation, tailoring, or instantiation into a determined context.

Based on the review, Adaptation on Non-reuse Process Context, one related work could be found on RUP, where there is a disciple that describes steps to adapt in two contexts: (1) Organizational level; or (2) Project level.

Adaptation on Reuse Process Context was researched on ten others processes related to domain engineering or product line approaches and the relation found is that some of them provide specifics approach to realize and adaptation, that can be found in terms of process phases or steps (COPAM), PuLSE, ABD, ODM, ROSE. FODA, RSEB, FORM, Kobra and PECOS do not treat process adaptation techniques in terms of how the process platform can be instantiated. Table 1, presented on Chapter 3, best summarizes the findings in each process.

However, the existing solutions fail to describe adaptation in terms of how the process guide can systematically be instantiated to a determined project scope or even, how can this process be adapted taking into consideration the existing practices of an organization. The key difference of the proposed work from the other is the systematization of an adaptation process by the collection of the best practices from the community. The objective is to fill in the gap and lack of material considering an adaptation between two processes to generate a final one.

6.3 Future Works

The presented version of the Reuse Process Adaptation does not cover some important areas that can improve the results achieved during the process execution. Since the goal was to adapt process considering aspects from a non-reuse process applied in the organization and a reuse process to be introduced, some enhancements of the process can be introduced in order to improve the process description and the desirable results. Some important aspects that were left out of the scope are enumerated:
• **Introduction of Knowledge Management Discipline:** in order to adapt process taking into consideration not only the procedural context of organizations, but also the knowledge aggregated from previous projects execution. This approach can improve process definition and the execution of the adapted process can have better results on project teams. New process activities and procedures must be included in order to contemplate Knowledge Management into the proposed process;

• **Formalize the Process Description:** due to the necessity to accurately present a process so that its description can improve comprehension by all developer, a PML can be used to describe the adaptation process. An extension to the overall view can be made and the description can be compliant to a SPEM, for instance;

• **Experimental Reports:** the execution of the case study in a context that formally defines the variables, hypothesis for a quantitative study through an empirical investigation could bring up some more accurate results and provide improvement opportunities to the process. The execution of this work is dependent on the availability of a project context that gathers the specification of the proposed process; and

• **Definition of Metrics to Formally Measure Process Adaptation:** Numbers, indicators and experimental approaches can be a mean to be introduced in the Adaptation Process so that the objectives of the Adaptation can be formally measured by its effectiveness, expected results and process execution. Thus, a more detailed study on the topic is necessary to include a set of metrics on the process.

### 6.4 Concluding Remarks

This work has investigated a set of proposed solutions to find a systematical way to introduce reuse process in organizations, and, through the analysis of the results, it is possible to modify existing processes in order to climb one more step on the reuse problem faced today.
However, to achieve the desirable results, there is a greater amount of processes, tools, metrics and investigations to be made in order to build a more favorable environment to effectively apply reuse based software development and to fully extract all reuse benefits.

This reuse investigation is part of a broader context, RiSE project, which aims at defining a framework for software reuse and incrementally guiding organizations to achieve the objectives proposed by a reuse framework. The adaptation process, is then, one part of the overall structure that can be used to guide adaptation into organizations.
References


(Griess, 1994) M. L. Griss, Software Reuse Experience at Hewlett-Packard, 16th International Conference on Software Engineering (ICSE), Sorrento, Italy, May, 1994, pp. 270.


(Osterweil, 1987) L. Osterweil, Software processes are software too, 9th International Conference on Software Engineering (ICSE), Monterey, California, USA, March/April, 1987, pp. 02-13.


(Ribot et al., 1994) D. Ribot, B. Bongard, and C. Villermain, **Development Life-Cycle with Reuse,** ACM Symposium on Applied Computing, Phoenix, Arizona, USA, March, 1994, pp. 70-76.

(Rine, 1997) D. C. Rine, **Success Factors for software reuse that are applicable across Domains and businesses,** ACM Symposium on Applied Computing (SAC), San Jose, California, USA, March, 1997, pp. 182-186.


Appendix A – Domain Requirement Template

In this template, the domain analysis step is described in details, especially, the planning, the applications, the features and the domain.

Planning

This section describes the preparation process for domain analysis.

<table>
<thead>
<tr>
<th>Stakeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>definition</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>definition</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Market</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>collection</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Applications

This section describes the existing, future and potential applications in the domain.

<table>
<thead>
<tr>
<th>Existing Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Future Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Potential Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

Feature Model

This section shows the domain feature model.
# Domain Documentation

This section describes the domain documentation.

<table>
<thead>
<tr>
<th>Domain’s Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining rules</td>
<td></td>
</tr>
<tr>
<td>Exemplar system selection</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td>Domain Context</td>
<td></td>
</tr>
<tr>
<td>Domain Genealogy</td>
<td></td>
</tr>
</tbody>
</table>
Feature Documentation

This section describes each feature in details.

<table>
<thead>
<tr>
<th>Features’ Name</th>
<th>Semantic Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale</td>
<td></td>
</tr>
<tr>
<td>Stakeholders and clients programs</td>
<td></td>
</tr>
<tr>
<td>Existing Applications</td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td></td>
</tr>
<tr>
<td>Variation Points</td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B – Architecture Document Template

This template is responsible for documenting the domain architecture defined in Domain Design step presenting its goals, modules, quality attributes, views, and component diagrams.

Architecture Goals and Constraints

This section describes the goals and architectural constraints.

Modules

This section presents the name and description of each module in the architecture.

Architecture Views

This section describes the architectural views for the designed architecture.

Module View

This section presents the module view of the domain architecture.

[Module Name]

<table>
<thead>
<tr>
<th>Name:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td></td>
</tr>
<tr>
<td>Use Cases:</td>
<td></td>
</tr>
<tr>
<td>Features:</td>
<td></td>
</tr>
</tbody>
</table>

Quality Attributes

This section describes the quality attributes for each module.
**[Sub-Module Name]**

<table>
<thead>
<tr>
<th>Name:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td></td>
</tr>
<tr>
<td>Use Cases:</td>
<td></td>
</tr>
<tr>
<td>Features:</td>
<td></td>
</tr>
</tbody>
</table>

**Quality Attributes**

This section describes the quality attributes for each sub-module.

**Other Views**

This section, if applicable, shows other views for the domain architecture.

**Quality Attributes**

This section presents the quality attributes for the architecture.

**Component Diagrams**

This section shows the components, their interfaces and communication among them.
Appendix C – Component Specification
Template (CST)

This template is responsible for representing the component specification defined in Domain Design step. For each component, it is presents its description, use cases, features, workflow, class diagrams, interfaces, packaging, and quality attributes.

Components
[CP 01] <Component Name>

<table>
<thead>
<tr>
<th>Name:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td></td>
</tr>
<tr>
<td>Use Cases:</td>
<td></td>
</tr>
<tr>
<td>Features:</td>
<td></td>
</tr>
</tbody>
</table>

Workflow

This section describes the component workflow.

Classes

This section presents the component internal structure with its classes and relationships.

Business Interface

This section shows the component business interface.

Provided Interfaces

This section presents the component provided interfaces and its services offered.

Required Interfaces

This section presents the component required interfaces.
Component Packaging

This section describes the internal component packaging.

Quality Attributes

This section presents the component quality attributes.
Dissertação de Mestrado apresentada por Ana Paula Carvalho Cavalcanti à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "Adaptação de Processo de Reuso de Software" orientada pelo Prof. Silvio Romero de Lemos Meira e aprovada pela Banca Examinadora formada pelos professores:

Prof. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

Prof. Ana Cristina Rouiller
Departamento de Estatística e Informática / UFRPE

Prof. Silvio Romero de Lemos Meira
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 5 de setembro de 2007.

Prof. FRANCISCO DE ASSIS TENÓRIO DE CARVALHO
Coordenador da Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco.
Cavalcanti, Ana Paula Carvalho

xi, 85 folhas : il., fig., tab.


Inclui bibliografia e apêndices.

1. Engenharia de software. I. Título.

005.1 CDD (22.ed.) MEI2008-051