



Pós-Graduação em Ciência da Computação

“ToolDAY – A Tool for Domain Analysis”

By

Liana Barachisio Lisboa

M.Sc. DISSERTATION



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

RECIFE, DEZEMBRO/2008



UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LIANA BARACHISIO LISBOA

"ToolDAy – A Tool for Domain Analysis"

*ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO
EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA
DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO
REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE
EM CIÊNCIA DA COMPUTAÇÃO.*

*THIS DISSERTATION WAS PRESENTED TO THE FEDERAL
UNIVERSITY OF PERNAMBUCO AS PARTIAL REQUIREMENT
FOR THE MASTER DEGREE IN COMPUTER SCIENCE.*

ADVISER: Silvio Romero de Lemos Meira

CO-ADVISER: Eduardo Santana de Almeida

RECIFE, DEZEMBRO/2008

Lisboa, Liana Barachisio
ToolDay – A Tool for domain analysis / Liana
Barachisio Lisboa - Recife : O autor, 2008.
xiii, 120 folhas : il., fig., tab. graf.

Dissertação (mestrado) - Universidade Federal
de Pernambuco. CIN - Ciência da Computação,
2008.

Inclui apêndice.

1. Engenharia de software. 2. Reutilização de software.
I. Título.

005.1

CDD (22.ed.)

MEI 2009-002

For my parents, Eduardo and Virginia, and my sister, Paula

Acknowledgements

There were a lot of people that directly or indirectly participated with me along my Master path. However, it will not be possible to remember of all of them now. My excuses for the ones I forgot, it does not mean you were not important.

I would like to thank Recife Center for Advanced Studies and Systems (C.E.S.A.R) that provided the infrastructure and environment necessary for this work.

Living away from home is not easy, for that a lot of people have helped me to feel a little bit like home, here in Recife, thanks to all of you. I would like to thank all of the RiSE members for their help and comments on this work, in special to Vinicius Garcia for their revisions, thoughts and comments. Eduardo Almeida, my co-advisor and friend, who said he would bring me to Recife during a carnival in Salvador, what definitely I did not believe at that time. Thanks for making it come true. Alexandre Martins for his patience, discussions, comments and support, which were very important during these almost three years living here; and to my advisor, Silvio Meira, for accepting me as his student.

Last, but not least, I would like to thank my family - my parents, Eduardo and Virginia, and my sister, Paula - without all of you this would have never been possible! Thanks for your support, care, homesickness and understanding, especially in the moments I could not be present due to the distance.

*“What is a scientist after all?
It is a curious man looking through a keyhole, the keyhole of nature,
trying to know what’s going on.”*

—JACQUES YVES COUSTEAU (1910-1997)

Resumo

A reutilização de software - o processo de criar sistemas através de artefatos existentes, ao invés de desenvolvê-los do zero - é um aspecto chave para melhorias em qualidade e produtividade no desenvolvimento de software. Contudo, os ganhos da reutilização são mais efetivos quando o reuso é planejado e gerenciado de forma sistemática no contexto de um domínio específico, onde famílias de aplicações compartilham algumas funcionalidades.

Neste contexto, uma das formas de se obter um processo de reuso mais sistemático é através do processo de análise de domínio - o processo de identificação de características comuns e variáveis de sistemas em um domínio específico. Este processo é composto por diversas atividades, como definição do escopo, modelagem e documentação do domínio, identificação das restrições, entre outros; e o seu sucesso é muito dependente de quão bem o mesmo é executado. Desta forma, torna-se essencial ter uma ferramenta de suporte para auxiliar a sua execução.

Atualmente, existem diversas ferramentas que provêem suporte a análise de domínio, todavia, as mesmas apresentam limitações, como não prover suporte ao processo completo. Assim, este trabalho apresenta os requisitos, a arquitetura e a implementação de uma ferramenta que provê suporte a análise de domínio e que foi focada em resolver as limitações identificadas nas ferramentas existentes. Além disso, esta dissertação descreve o processo e os resultados encontrados nas diversas avaliações que foram executadas em diferentes ambientes com a ferramenta proposta.

Palavras-chave: Reuso de Software, Análise de Domínio, Ferramenta

Abstract

Software reuse - the process of creating software systems from existing software rather than building them from scratch - is a key aspect for improving quality and productivity in the software development. However, reuse gains are more effective when it is systematically planned and managed in the context of a specific domain, where application families share some functionality.

In this context, one of the ways to achieve a more systematic reuse process is through the domain analysis process - the process of identifying common and variable characteristics of systems in a specific domain. This process is composed of several activities, which include the scope definition, the domain modeling and documentation, the restriction identification. Its success is largely dependent upon how well the process is carried out. Due to this, it is essential to have a tool support for aiding the process execution.

Currently, there are several existing tools for supporting the domain analysis. However, they present gaps, such as not supporting the complete process. Thus, this work presents the requirements, design and implementation of a domain analysis tool that was focused on solving the identified gaps in the current tools. In addition, it describes the process and results of some evaluations performed in different environments with the proposed tool.

Keywords: Software Reuse, Domain Analysis, Tool

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Problem Statement | 3 |
| 1.3 | Overview of the Proposed Solution | 3 |
| 1.3.1 | Context | 3 |
| 1.3.2 | Outline of the Proposal | 5 |
| 1.4 | Out of the Scope | 5 |
| 1.5 | Statement of the Contribution | 6 |
| 1.6 | Organization of the Dissertation | 7 |
| 2 | Software Tools and Environments | 9 |
| 2.1 | Why to use CASE | 10 |
| 2.2 | CASE Classification | 12 |
| 2.2.1 | MetaCASE | 15 |
| 2.2.2 | Integrated Environments | 16 |
| 2.3 | Software Reuse Environments | 18 |
| 2.4 | Chapter Summary | 19 |
| 3 | A Systematic Review on Domain Analysis Tools | 20 |
| 3.1 | Previous Studies | 21 |
| 3.2 | Systematic Review Process | 22 |
| 3.3 | Planning the Review | 22 |
| 3.3.1 | Research Question | 23 |
| 3.3.2 | Question Structure | 23 |
| 3.4 | Conducting the Review | 24 |
| 3.4.1 | Search Strategy | 24 |
| | Search Results | 26 |
| 3.4.2 | Studies Selection | 27 |
| | Inclusion Criteria | 27 |
| | Exclusion Criteria | 28 |
| 3.4.3 | Data Analysis | 28 |
| 3.4.4 | Data Extraction | 29 |
| 3.5 | Data Synthesis | 29 |
| 3.5.1 | Tools Selection | 30 |

| | | |
|----------|--|-----------|
| 3.5.2 | Research Question Result | 33 |
| | Domain Analysis Support | 33 |
| | Main Functionalities | 34 |
| | Tools Development and Usage | 40 |
| | Research Question Result Summary | 41 |
| 3.6 | Functionalities Priority | 43 |
| 3.6.1 | Essential Priority | 44 |
| 3.6.2 | Important Priority | 44 |
| 3.6.3 | Low Priority | 44 |
| 3.6.4 | Priority Support by Tools | 45 |
| 3.7 | Threats to validity | 45 |
| 3.8 | Chapter Summary | 46 |
| 4 | ToolDAy | 47 |
| 4.1 | Requirements | 47 |
| 4.2 | Tool Architecture | 48 |
| 4.3 | Technologies | 50 |
| 4.4 | ToolDAy | 52 |
| 4.4.1 | ToolDAy's Steps | 52 |
| | Domain Planning | 52 |
| | Domain Modeling | 54 |
| | Domain Validation | 57 |
| | Product Derivation | 61 |
| 4.4.2 | Usability Requirements | 62 |
| 4.5 | ToolDAy in Action | 65 |
| 4.6 | Chapter Summary | 71 |
| 5 | ToolDAy Evaluations | 72 |
| 5.1 | ToolDAy Experiments | 72 |
| 5.1.1 | Experiment Process | 73 |
| 5.1.2 | Definition | 74 |
| 5.1.3 | First Experiment | 75 |
| | ToolDAy's Evaluated Version | 76 |
| | Planning | 76 |
| | The Experimental Study Project | 78 |
| | Operation | 79 |

| | |
|---|------------|
| Analysis and Interpretation | 79 |
| Lessons Learned | 83 |
| 5.1.4 Second Experiment | 84 |
| Planning | 84 |
| The Experimental Study Project | 85 |
| Operation | 86 |
| Analysis and Interpretation | 86 |
| 5.1.5 Experiments Summary | 88 |
| 5.2 ToolDay's Industrial Case | 88 |
| 5.3 Chapter Summary | 90 |
| 6 Conclusion Remarks and Future Work | 92 |
| 6.1 Research Contribution | 92 |
| 6.2 Related Work | 93 |
| 6.3 Future Work | 94 |
| 6.4 Concluding Remarks | 97 |
| A Journals and Conferences | 109 |
| B Evaluation Feedback Form | 110 |
| B.1 First Feedback Form | 110 |
| B.2 Second Feedback Form | 114 |

List of Figures

| | | |
|------|---|----|
| 1.1 | RiSE Labs Influences | 4 |
| 1.2 | RiSE Labs Projects | 5 |
| 2.1 | Architectural model for the integration framework | 18 |
| 3.1 | Selected tools timeline | 33 |
| 3.2 | Domain analysis processes support | 34 |
| 3.3 | Specific Process Subdivision | 34 |
| 3.4 | Where the tools were developed | 40 |
| 3.5 | Number of tools per functionalities | 43 |
| 4.1 | ToolDAY's architecture | 48 |
| 4.2 | ToolDAY's technologies | 51 |
| 4.3 | ToolDAY's environment | 51 |
| 4.4 | ToolDAY's execution flow | 52 |
| 4.5 | Features types' representation | 55 |
| 4.6 | Domain documentation form | 58 |
| 4.7 | ToolDAY's spell check | 63 |
| 4.8 | Domain feature model metrics | 63 |
| 4.9 | ToolDAY's preferences | 64 |
| 4.10 | Atari game product map | 66 |
| 4.11 | Import feature with relationship | 67 |
| 4.12 | Atari game feature model | 68 |
| 4.13 | ToolDAY's consistency checker | 68 |
| 4.14 | Inconsistency report | 68 |
| 4.15 | Feature's severity marker and tool tip | 68 |
| 4.16 | ToolDAY's quick fix window | 69 |
| 4.17 | Atari's domain traceability model | 69 |
| 4.18 | ToolDAY's product selection | 70 |
| 4.19 | Initialize/Update the product model | 71 |
| 4.20 | Demon attack product model | 71 |
| 5.1 | Overview of the experiment process | 74 |
| 5.2 | Number of Subjects per Difficulty | 82 |
| 5.3 | Part of the social networks feature model | 89 |

List of Tables

| | | |
|-----|---|----|
| 3.1 | Search documentation process | 26 |
| 3.2 | Number of tools selected in each category | 30 |
| 3.3 | The analyzed information of each tool | 32 |
| 3.4 | Functionalities each tool supports | 36 |
| 3.5 | Where tools were used | 41 |
| 3.6 | Functionalities each tool support per priority | 45 |
| 4.1 | Summary of requirements | 49 |
| 5.1 | Subjects' profile in the first experiment | 80 |
| 5.2 | Subjects' Profile in the second experiment | 86 |
| 6.1 | ToolDAy's functionalities compared to other tools | 94 |

List of Acronyms

| | |
|------------|---|
| B.A.R.T | Basic Asset Retrieval Tool |
| BTT | Bug Triage Tool |
| CASE | Computer-Aided Software Engineering |
| DARE | Domain Analysis and Reuse Environment |
| DREAM | Domain REquirement Asset Manager in product lines |
| DoD | US Department of Defense |
| DSL | Domain Specific Languages |
| ETH | Swiss Federal Institute of Technology |
| FODA | Feature Oriented Domain Analysis |
| GMF | Graphical Modeling Framework |
| GQM | Goal-Question-Metric |
| GUI | Graphical User Interface |
| I-CASE | Integrated CASE |
| IDE | Integrated Development Environment |
| JDT | Java Development Tools |
| JEP | Java Math Expression Parser |
| LIFT | Legacy Information Retrieval Tool |
| PuLSE-BEAT | Product Line Software Engineering - Basic Eco Assistance Tool |
| PSE | Programming Support Environments |
| PSEE | Process-Centered Software Engineering Environments |
| RCP | Rich Client Platform |
| RiPLE | RiSE Process for Product Line Engineering |
| RiSE | Reuse in Software Engineering |
| RWTH | Research Group Software Construction |
| SE | Software Engineering |
| SEE | Software Engineering Environments |
| SOPLE | Service Oriented Product Line Engineering |
| SPL | Software Product Lines |
| SQ | Sub-Question |
| TMS | Tools management services |
| ToolDAy | Tool for Domain Analysis |

“Toda caminhada começa com o primeiro passo”

Acioli Neto (Musician)



Introduction

Software reuse is an important aspect for companies interested in improving software development quality and productivity (Krueger, 1992). This interest is usually leveraged by the way software systems are becoming bigger and more complex (Frakes and Kang, 2005), and the application of ad-hoc reuse, which is a reuse without any defined process and usually restricted to source code, has risks that can compromise future initiatives in this direction (Almeida, 2007).

One of the approaches that leads towards the software reuse benefits is a systematic reuse approach, which is domain focused, based on a repeatable process, and concerned with reuse of higher level life cycle artifacts (Frakes and Isoda, 1994). The focus of this dissertation is in providing a **tool support for domain analysis**, which is one of the ways to achieve systematic reuse.

This chapter contextualizes the focus of this dissertation and starts by presenting its motivation in section 1.1 and a clear definition of the problem in section 1.2. A brief overview of the proposed solution is presented in section 1.3, while section 1.4 describes some related aspects that are not directly addressed by this work. Section 1.5 presents the main contributions and, finally, section 1.6 outlines the structure of the remainder of this dissertation.

1.1 Motivation

Software reuse - the process of using existing software artifacts rather than building them from scratch (Krueger, 1992) - is generally regarded as the most important mechanism for performing software development more efficiently (McIlroy, 1968). There are several studies describing the effects of reuse in software development in terms of quality, time-to-market and costs over the years (Lim, 1994; Basili *et al.*, 1996; Frakes and Succi,

2001) and they show that it can be improved by reusing all kinds of proven experience, including products, processes and quality and productivity models.

There are a variety of reusable assets, such as requirements, use cases, architecture and frameworks, however, the most common is *source code*. On the other hand, reuse focused only on source code libraries is insufficient and the **key to successful reuse lies in understanding and defining the application domain for a collection of assets** (Biggerstaff, 1992), which can be achieved through a systematic reuse approach.

A way to accomplish this, is through a **domain engineering** process, which is the activity of collecting, organizing and storing past experience in building systems or parts of systems in a particular domain in a form of reusable assets (Czarnecki and Eisenecker, 2000). The results of the domain engineering will be reused in the application engineering, which is the process of producing systems with the reusable assets developed during the domain engineering. Furthermore, both processes - domain and application engineering - are divided in analysis, design and implementation.

This work focus only on the domain analysis phase of the process, since it is the initial part for executing this process and because it involves the management and analysis of a large quantity of information, a tool support is necessary.

Since its definition, several processes, such as (Frakes *et al.*, 1998; Weiss and Lai, 1999; Bayer *et al.*, 1999b; Kim *et al.*, 2003; Mei *et al.*, 2003; Moon *et al.*, 2005; Almeida *et al.*, 2006; Lucrédio *et al.*, 2008a), have been created providing different guidelines of how to identify common and variable characteristics of the domain and to represent them to the analysts. Therefore, independently from the process guideline being followed, it is composed of several interdependent activities, such as the analysis of the existing products from the domain and identification of the domain scope (the activities are described in details in chapter 3). Hence, this involves the management of complex and interrelated information from various sources.

Thus, tool support becomes essential in order to aid the domain analyst - the person who is responsible for conducting the process - during the management and representation of the domain's characteristics (Bass *et al.*, 1997; Succi *et al.*, 2000a; Moon *et al.*, 2005). Conversely, there is a lack of integrated tools to aid it, which frequently forces the use of several independent tools to perform the activities.

This scenario increases the risk of problems - such as delays, traceability management, degraded productivity and interoperability - during the process execution, because the information traceability among the tools, as well as its consistency and portability, must be performed manually by the domain analysts.

1.2 Problem Statement

Motivated by the scenario presented in the previous section, the goal of the work described in this dissertation can be stated as:

This work defines the requirements, design and implementation of a domain analysis tool, aiming at making the domain analysis process semi-automatic and at aiding the domain analyst to achieve systematic reuse in an effective way.

Consequently, it will reduce the risks associated to a project performed without automation or performed with several tools, and it will increase the quality and productivity of the artifacts developed for the domain.

1.3 Overview of the Proposed Solution

In order to accomplish the goal of this dissertation, ToolDAy (Tool for Domain Analysis) is proposed. This section presents the context where it is regarded and outlines the proposed solution.

1.3.1 Context

This dissertation is part of the RiSE Labs¹ (Almeida *et al.*, 2004), formerly called RiSE Project, whose goal is to develop a robust framework for software reuse in order to enable the adoption of a reuse program. However, it is influenced by a series of areas, such as software measurement, architecture, quality, environments and tools, and so on, in order to achieve its goal. The influence areas are depicted in Figure 1.1.

Based on these areas, the RiSE Labs is divided in several projects, as shown in Figure 1.2. As it can be seen in the Figure, this framework embraces several different projects related to software reuse. They are:

- **RiSE Framework:** Involves reuse processes (Almeida *et al.*, 2005; Nascimento, 2008), component certification (Alvaro *et al.*, 2006) and reuse adoption process (Garcia *et al.*, 2008a,b).
- **RiSE Tools:** Research focused on software reuse tools, such as the Admire Environment (Mascena, 2006), the Basic Asset Retrieval Tool (B.A.R.T) (Santos *et al.*, 2006), which was enhanced with folksonomy mechanisms (Vanderlei *et al.*, 2007),

¹<http://www.rise.com.br/research>

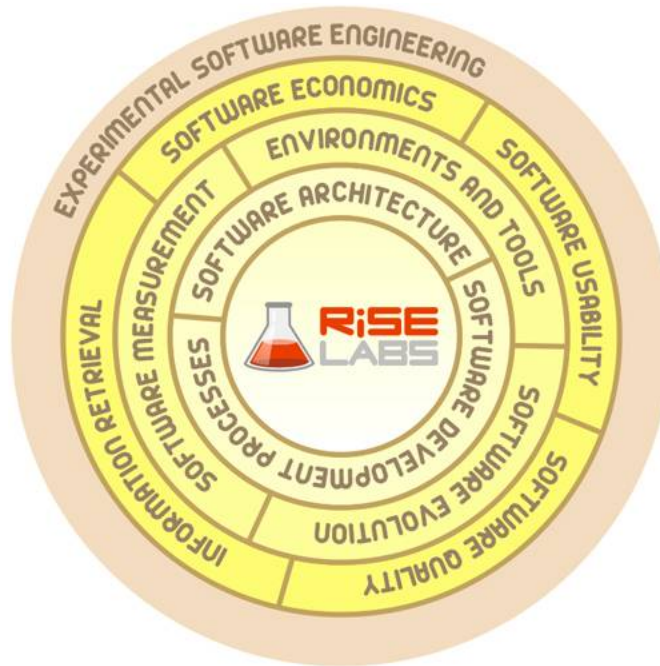


Figure 1.1 RiSE Labs Influences

semantic layer (Durão, 2008), facets (Mendes, 2008) and data mining (Martins *et al.*, 2008), and the Legacy InFormation retrieval Tool (LIFT) (Brito, 2007).

- **RiPLE:** Development of a methodology for Software Product Lines, like (Filho *et al.*, 2008).
- **SOPLE:** Development of a methodology for Software Product Lines based on services.
- **MATRIX:** Investigates the area of measurement in reuse and its impact in quality and productivity.
- **BTT:** Research focused on tools for detection of duplicated change requests, such as (Cavalcanti *et al.*, 2008).
- **Exploratory Research:** Investigates new research directions in software engineering and its impact on reuse.
- **CX-Ray:** Focused on understanding the C.E.S.A.R², its processes and practices in software development.



Figure 1.2 RiSE Labs Projects

This dissertation is part of the **RiSE Tools** project and its goal is to support a systematic reuse environment, providing mechanisms to aid the domain analysis process.

1.3.2 Outline of the Proposal

Aware of the problems stated before, this dissertation presents a domain analysis tool that aims at making the process semi-automatic and at aiding the domain analyst to achieve the systematic reuse in an effective way. This idea agrees in the same manner with the work of (Bass *et al.*, 1997; Succi *et al.*, 2000a; Moon *et al.*, 2005), which describes that tool support is essential for the domain analysis execution.

The proposed solution consists of 4 components - planning, modeling, validation and product derivation - that are integrated to a graphical interface and a persistency layer. The goal is not only to support the whole domain analysis process and the product derivation analysis, but also to permit the execution of activities or steps apart from the complete process.

1.4 Out of the Scope

As the proposed tool is part of a broader context, a set of related aspects were left out of its scope. Even though the provided functionalities are based on well-founded aspects of quality and performance, they do not discard future enhancements to answer more

²Recife Center for Advanced Studies and Systems - <http://www.cesar.org.br>

efficiently its purpose. Meanwhile, the aspects not directly addressed by this work are listed in the following:

- **Full Domain Engineering Support.** Domain engineering is usually divided in three phases - domain analysis, design and implementation (Czarnecki and Eise-necker, 2000). Although all of them are important for the process fulfillment, since each one focus on a specific part of the software life cycle, the support for the phases of design and implementation were not included as part of this work.
- **Full Application Engineering Support.** The application engineering is also di-vided in the three phases - analysis, design and implementation (Czarnecki and Eisenecker, 2000). However, this work includes only the support for the analysis phase of the application engineering.
- **Domain Analysis Process.** Software process is a set of activities that leads to the production of a software (Sommerville, 2007), and both academy and industry agree that processes are fundamental to software engineering. However, the tool presented in this work do not intend to support a specific domain analysis process, for this reason it was based on different tools and processes activities - as described in chapter 3. Therefore, no process definition is addressed in this work.
- **Domain Specific Languages (DSL).** Domain-specific languages are languages tailored to a specific application domain and their primary contribution is to enable reuse of software assets (Mernik *et al.*, 2005). In spite of being related to the domain analysis, it is not addressed in this work because the focus was to provide support to the domain analysis, and the DSL can be generated after its finalization.

1.5 Statement of the Contribution

As a result of the work presented in this dissertation, a list of contributions may be enumerated:

- A study of the classification types for Computer-Aided Software Engineering (CASE) tools and environments as a way to clarify the terminology used and to assess the capabilities and features that should be present in a product development, besides describing the benefits a CASE tool adoption can bring to a company.

- An extensive systematic review with nineteen domain analysis tools to identify and analyze how these tools are offering the support to the process. This analysis offered the base to define a set of requirements that should be present in domain analysis tools, and to map new tools development and usage.
- The requirements, design and implementation definition for the domain analysis tool.
- Three different evaluations - two of them as experiments in an academic environment and the other in an industrial project - for verifying the tool helpfulness to the process.

Besides the final contributions listed so far, some intermediate results of this work has been reported in the literature, as shown in the following:

- Lisboa, L. B., Garcia, V. C., Almeida, E. S., and Meira, S. L. (2007). **ToolDAy - a process-centered domain analysis tool**. In Brazilian Symposium on Software Engineering (SBES) - Tools Session, pages 54-60, João Pessoa, Paraíba, Brazil.
- Lisboa, L. B., Nascimento, L. M., Almeida, E. S., and Meira, S. R. L. (2008). **A case study in software product lines: An educational experience**. In 21st IEEE Conference on Software Engineering Education and Training (CSEET), Charleston, South Carolina.
- Lisboa, L. B., Lucrédio, D., Garcia, V. C., and Almeida, E. S. (2008). **A systematic review on domain analysis tools**. Technical Report 20080121-0101, C.E.S.A.R./RiSE.

1.6 Organization of the Dissertation

The remainder of this dissertation is organized as follows:

- Chapter 2 contains a comprehensive revision of the software tools and environments field with the goal of presenting the reasons to adopted CASE tools and the main classifications in this area, clarifying the terminology used and assessing the capabilities and features that should be present in a product development;
- Chapter 3 presents a systematic review on the available domain analysis tools with the objective of mapping how this area is currently supported by the tools;

- Chapter 4 describes the proposed domain analysis tool in details. The requirements, used technologies, design and the implementation aspects are discussed;
- Chapter 5 reports the entire environment used in the execution of all the evaluations. In addition, this chapter presents the expected goals, the methodology adopted, and the findings; and
- Chapter 6 concludes this dissertation by summarizing the findings and proposing future enhancements to the solution, along with some concluding remarks.

“Todo dia eu acordo buscando saber quem sou e de onde vem o mundo.”

Maíra Viana (Writer)

2

Software Tools and Environments

According to the Cambridge Dictionary¹, tools are “*something that helps you to do a particular activity*”, and they have been present in computer science since its first days. In Software Engineering (SE), tools and environments have existed for a long time, since compilers and editors became standard offerings with operating systems (Harrison *et al.*, 2000). However, these tools and environments have also significantly evolved over the years, moving from simple editors to more modern tools, such as requirements analysis tools.

The first significant efforts in producing tightly integrated development environments were those in the area of Programming Support Environments (PSEs), which were focused on coding activities (Harrison *et al.*, 2000). They involved programs such as compilers, language-sensitive editors (such as syntax-directed editors), and debuggers, and sometimes other tools as well (e.g., testing or documentation utilities).

Even though PSEs solved several problems from earlier and loosely integrated environments, they also presented limitations such as the support for only one software engineering activity. This lack of support among the software engineering activities - requirements, analysis, specification and testing - impacts in the type of traceability offered by PSEs, since the output of one phase may affect another, and changes to one artifact may necessitate changes to other related artifacts to ensure that the artifacts remain mutually consistent (Harrison *et al.*, 2000). Thus, this need for an integrated support to all the activities in software engineering in the software development lifecycle led to the birth of the Software Engineering Environments (SEE), also called CASE (Computer-Aided Software Engineering).

CASE is the term for software tool support for SE processes (Sommerville, 2007), and have the goal of aiding the software developers through the automation of different

¹<http://dictionary.cambridge.org/>

parts of the development process, allowing them to develop high quality systems, easy maintained and reliable (Albizuri-Romero, 2000). Furthermore, the current complexity in SE processes makes them unaccomplishable without reasonable tool support.

Nowadays, there is a variety of CASE tools for different purposes, ranging from traditional tools - like editors, compilers and debuggers to tools that aid in requirements analysis, design, building GUIs (Graphical User Interface), architecting systems, connecting components, testing, version control, configuration management, administering databases, reengineering, analysis, program visualization, and metrics gathering - to full-scale, process-centered software engineering environments that cover the entire lifecycle, or at least significant portions of it (Harrison *et al.*, 2000). According to Albizuri-Romero (2000), the ideal situation would be for CASE tools to cover the total life cycle in an integral way.

2.1 Why to use CASE

Companies are currently seeking for ways of improving their product's quality and reduce their time-to-market, and since CASE tools provide an effort reduction due to activities automation and new ways to improve the insight of engineers doing a work (Pressman, 2001), they can aid in this task.

Oakes *et al.* (1992) present a report for a strategic adoption of CASE tools in a company. They highlight some of the benefits of adopting a CASE tool:

- **Improved communications:** The communications seems to be enhanced both between project engineers and customer and among engineers working on a project; and,
- **Improved documentation:** The documentation improvement relates to a greater consistency and accuracy in specifying the system and in the direct generation of portions of the documentation by the CASE tools.

In the report, Oakes *et al.* (1992) underline other commonly cited benefits as *variable productivity gains*, *modest quality gains* and *enforcement of project methodology and standards*. Furthermore, according to a review literature performed by them, many industry analysts document productivity gains ranging from 10% to 30% resulting from CASE analysis and design tool usage, with similar gains in software quality and documentation.

Cronholm (1995) presents the results of a study conducted in 1994 in Sweden, in which he analyzed the motives a company has to invest in CASE, he acknowledged

purpose similar to Oakes *et al.* (1992) report's. In the study, Cronholm identified that the main intention for companies to invest in CASE tools is to increase their competitiveness in the market. To the companies, it can be reached in two ways:

- **Faster development;** and,
- **Improved product quality.**

In order to achieve the faster development, the companies should invest in making the working procedures easier and managing document handling more effective. Thus, they should:

- **Facilitate update and modification of diagrams:** The lack of tool support executing in this task makes it time consuming and difficult, and can, eventually, lead to problems in the documentation produced.
- **Assist method institutionalizing:** Companies aim at achieving a more standardized way of performing software development. For that, they need a better process structure through a wider use of methods that can be easily followed. Thus, tools focused on processes (Process-Centered Software Engineering Environments - PSEE) can secure the methods rules, which can guide to an easier maintenance process.
- **Achieve a faster dialogue with customers/end-users:** The conversation with the customers/end-user about the product requirements, usually occur through documents and diagrams. With the tools, the manipulation of these documents/diagrams becomes faster and easier, increasing the dialogues with the customers/end-user.

Regarding the improved product quality, the companies identified the following items for investing in CASE tools:

- **Increase the work flexibility:** The companies' motivation to use CASE was the possibility of easily modifying the development methodology according to the specific situation of the product to be developed.
- **Improve documentation:** Through the improvement in documentation, companies achieve not only an easier product to maintain, but it also improves product's comprehension during customer/end-users conversations. Another motive given by the companies is that they want to improve the development reports.

- **Reduce working effort:** The tools can be useful for saving working time in routine work products, leaving more time to focus in improving the product's quality.

Companies associate the product's quality with the method followed to develop it, and one of the ways they consider to improve the method is through the adoption of the CASE tools, which will ensure the method follow-up.

Lucrédio *et al.* (2008b) detail a survey conducted with 57 Brazilian small, medium and large software organizations. Through this survey, they identified some key factors in adopting an organization-wide software reuse program.

One of the factors, they investigated in the survey, was if the use of *CASE tools contribute to the success of software reuse in the companies*. In their study, Lucrédio *et al.* (2008b) conclude that CASE has a strong influence on reuse success, but they highlight that the kinds of CASEs used were not analyzed, nor which ones were the responsables for the success in reuse.

Furthermore, these different types of CASE tools have specific goals for different purpose, which are detailed in the next section.

2.2 CASE Classification

CASE classification helps in understanding the types of existing tools available and their roles in software engineering support. Some authors have proposed different forms of classifying CASE, which are described in this section. Even though the classifications provide useful information in order to understand the tool, the ones described here do not intend to provide a full classification, because it is not always easy to position a product using a classification (Sommerville, 2007).

There are several tools focusing on specific activities of the software engineering process, while other, such as text editing, document preparation and configuration management, can be used throughout the process. Thus, the different types of CASE provide specific characteristics in order to support the process.

This breadth of support is commonly used as a possible classification. Fuggetta (1993) proposes a classification in three categories:

- **Tools:** Support only specific tasks in the process, which are further classified in:
 - *Editing Tools:* Simple editors that can be divided in *textual* and *graphical*.
 - *Programming Tools:* Support coding and code restructuring.

- *Verification and Validation Tools*: Ensure the product’s functionalities according to what was asked (validation) and guarantee that the requirements are being met (verification).
 - *Configuration-Management Tools*: Coordinate and control the construction of a system.
 - *Metrics and Measurement Tools*: Collect data or programs and program execution.
 - *Project-Management Tools*: Involve tools for software-production costs, project planning and communication and coordination among project teams.
 - **Workbenches**: Support only one or few activities. Thus they can achieve the presentation, control, data and process integration and are usually built as a collection of tools. It is finer-grained in:
 - *Business Planning and Modeling Workbenches*: Include products to support the identification and description of complex business.
 - *Analysis and Design Workbenches*: Include tools to create, modify, analyze, simulate and transform specifications. They are also called “*Upper CASE*”.
 - *User-Interface-Development Workbenches*: Involve tools to easily create, test and integrate user-interface components. This class is not directly associated to software-process activities, but to activities that are important for the client’s deliverable.
 - *Programming Workbenches*: Aggregate tools for programming, and integrated support to compilers and debuggers.
 - *Verification and Validation Workbenches*: Integrate tools from metrics and measurement class and validation and verification class.
 - *Maintenance and Reverse-Engineering Workbenches*: Usually composed of the same tools and workbenches used in the development, it should also include tools for reverse-engineering, such as code restructurer, flowcharter, cross-reference generator and so on.
 - *Configuration-Management Workbenches*: Incorporate tools for supporting version control, configuration building and change control.
 - *Project-Management Workbenches*: Aggregate project-management functionalities such as project’s accomplishment, dependency among activities, meeting scheduler, besides project-planning and task-assignment tools.
-

- **Environments:** Support (a large part of) the software process through a collection of tools and/or workbenches. They are further classified in:
 - *Toolkit:* Loosely integrated collection of products easily extended by aggregating different tools and workbenches.
 - *Language-Centered Environments:* Provide automatic tool invocation and switching among them and are usually focused on edit-compile-debug, with little support to process and data integration.
 - *Integrated Environments:* These environments achieve presentation, data and control integration. However they do not explicitly tackle process integration.
 - *Fourth-Generation Environments:* Set of tools and workbenches supporting the development of a specific class of program: electronic data processing and business-oriented applications. They are often integrated in consistent interface, data are stored in a proprietary repository, but they provide a low-degree of process integration.
 - *Process-Centered Environments:* It is based on a formal definition of the software process. Thus, they guide the development process by automating process fragments, automatically invoking tools and workbenches, enforcing specific policies and assisting programmers, analysts and project managers in their work (Lehman, 1987).

Pressman (2001) gives a different classification based on the tools functionalities. Some of these classifications are equivalents to the ones defined by Fuggetta (1993), such as project planning and management, metrics, configuration management, analysis and design, interface design and development, programming and reengineering tools. Pressman's classification still includes:

- **Business Process Engineering Tools:** Aid in the business processes models creation through the representation of business data objects, their relationships, and how these data objects flow between different business areas within a company.
- **Process Modeling and Management Tools:** Used to represent the key elements of a process so that it can be better understood.
- **Risk Analysis Tools:** Enable a project manager to build a risk table by providing detailed guidance in the identification and analysis of risks.

- **Requirements Tracing Tools:** Provide a systematic approach to the isolation of requirements, beginning with the customer request for proposal or specification.
- **Documentation Tools:** Aid in creating and maintaining the documentation in an efficiency way and in improving the productivity.
- **System Software Tools:** Provide high-quality network system software, object managements services, electronic mail and other communications capabilities.
- **Database Management Tools:** Help the establishment of repositories (CASE database) in the project.
- **PRO/SIM Tools:** Allow the prediction of system behavior before it is completed through mock-up tests.
- **Prototyping Tools:** Allow test and validation of ideas through the creation of interface or functionalities prototypes.
- **Web Development Tools:** Aid in the generation of text, graphics, forms and other elements of a web page.
- **Integration and Testing Tools:** Help the data collection, during tests execution and analysis.

2.2.1 MetaCASE

The development of the CASE tools can be very complex, because they often involve several functionalities, as described in the previous section. Thus, the tool development itself becomes a difficult and complete process. However, it has been noticed that in spite of the different methods, data manipulation and the functionalities provided by each CASE tool, they have some standard characteristics, such as graphical interface and repositories (Gray *et al.*, 2000).

As a way to help its development, a new class of technology has been created and it is called *MetaCASE*, whose goal is to provide a way for building quality CASE tools quicker and easier (Gray *et al.*, 2000).

The metaCASE tries to automate the standard characteristics of the tools, therefore provide mechanisms to configure the graphical interface, the data repository and code generators (Gray *et al.*, 2000), they also include a description of the method components,

its rules and so on. These information are parsed and analyzed in order to generate the new CASE tool (Sommerville, 2007).

There are two main approaches to metaCASE products, they are (Gray *et al.*, 2000):

- **Frameworks.** In this approach, the MetaCASE tools provide the framework for the CASE tools allowing the customization of data structure, user interfaces and specific notations, besides specifying some other functionalities such as code generation.
- **Components.** In this approach, the user can use or extend the set of components provided by the MetaCASE tool to generate a tool, which is easier than building it from scratch.

According to Gray *et al.* (2000), it is not established which of these approaches is the better one, because both of them have advantages and disadvantages. The first approach (Frameworks) provides higher abstraction level allowing users to choose its programming language to be used; whereas the components approach supports the reuse of ready to use components, permitting a faster and easier tool development.

The MetaCASE usage provides the benefits of power, flexibility and rapid tool development time for new tools. This reduces the need for a costly and a long selection and adaptation process usually taken for new CASE tools acquisition.

2.2.2 Integrated Environments

The benefits of the CASE tools are best achieved when a set of tools are integrated, instead of using them one at a time. This set of CASE is known as CASE environments, Integrated CASE (I-CASE) or Software Engineering Environments (SEE).

Some of the I-CASE benefits are (Pressman, 2001):

- Smooth transfer of information from one tool to another and one software engineering step to the next inside the same environment.
- A reduction in the effort required to perform activities present in the whole software life cycle, such as software configuration management, quality assurance, and document production.
- An increase in project control that is achieved through better planning, monitoring, and communication.

- Improved coordination among staff members.

However, this integration involves different factors and challengers such as shared repository, standard inputs and outputs, event based integration, unified interface among the tools, homogeneous communication between the tools and the software engineers and so on (Harrison *et al.*, 2000; Pressman, 2001), which increases its complexity.

Pressman (2001) establishes a set of requirements for an I-CASE environment:

- Provides a mechanism for sharing software engineering information among all tools contained in the environment.
- Enables a change to one item of information to be tracked to other related information items.
- Provides version control and overall configuration management for all software engineering information.
- Allows direct, nonsequential access to any tool contained in the environment.
- Establishes automated support for the software process model that has been chosen.
- Collects both management and technical metrics that can be used to improve the process and the product.

In order to achieve these requirements, the I-CASEs should have a complete architecture for integration. The following architectural components must be present, as shown in Figure 2.1:

- **User Interface Layer.** Provides a common presentation interface to all the environments tools, aiding in the user adaptation to the environment interface.
- **Tools Layer.** Composed of a set of tools management services with the CASE tools. Tools management services (TMS) control the behavior of tools within the environment, providing a common platform among the tools.
- **Object Management Layer.** Provides the mechanism for tools integration with the repository data. In addition, it should provide the services for configuration management.
- **Shared Repository Layer.** Composed of the CASE database and the access control functions that enable the object management layer to interact with the database.

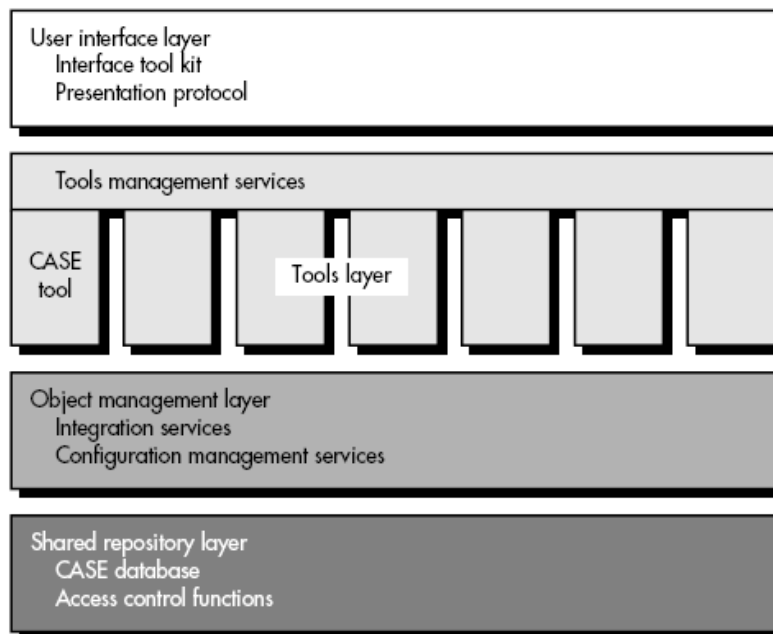


Figure 2.1 Architectural model for the integration framework (Pressman, 2001)

2.3 Software Reuse Environments

One way to facilitate the inclusion of software reuse in a company is to embed reuse practices and activities in the software environment. Through tools aiding in different aspects of reuse, software engineers can easily perform reuse-related activities.

In this sense, many specialized technologies have been produced to promote particular aspects of reuse. They have three trends: **(i)** reusable assets search engines, which are the most basic approach to promote reuse; **(ii)** repository systems, which attempt to centralize and manage all reusable information; and **(iii)** reuse environments, which attempt to cover a wider range of activities of a reuse process (Garcia *et al.*, 2006).

This dissertation focus on the last trend - reuse environments - since domain analysis tools focus on supporting several activities for different purpose in the process. With this in mind, it is possible to categorize them according to the classification presented before.

Since domain analysis tools, normally, offer support to different tasks, such as scope definition, visual representation, requirements definition and metrics (which will be detailed in the next chapter) they should be considered as a *workbench*, according to Fuggetta (1993) definition. Within the workbench, it can be further classified into *analysis and design*, because it involves functionalities for the analysis and modeling of the domain. Furthermore, it can also be classified as a *documentation tool*, due to the documentation

of the domain artifacts.

2.4 Chapter Summary

CASE tools span every activity in software process and in those that run throughout the process, such as configuration management. In the beginning, the CASE tools were focused only in increasing the productivity through the automation of repeatable activities. Nowadays, they are used in all software development life cycle, aiding their users to improve the products' quality.

Even though the CASE tools are being largely adopted over the years, they are not able to overcome the Software Engineers job, since the tools still need the human decision-making. However, the tools facilitate their job by automating activities such as creation and generation of models, planning estimative and project management.

“Sei que caminho com a história nas mãos”

Lau Siqueira (Poet)

3

A Systematic Review on Domain Analysis Tools

The key to a successful reuse lies in understanding and defining the application domain for a collection of assets (Biggerstaff, 1992), which can be achieved through a systematic reuse approach. One of these approaches is **Domain Analysis**, which is the process of identifying, capturing, and organizing the objects and operations of similar systems in a particular problem domain (Neighbors, 1981).

As said before (chapter 1), it is necessary to have a tool support to aid the organization's domain analyst during the process execution (Bass *et al.*, 1997; Succi *et al.*, 2000a; Moon *et al.*, 2005). However, it is not clear if all the available tools for domain analysis offer a good support to the process. Therefore, it is necessary to evaluate them in order to verify the degree of support the process currently has and the most common functionalities supported by the tools. Furthermore, this chapter can also provide an insight for new researchers in the software reuse area, since this information can be useful for researchers aiming at developing new tools, or working in steps after the domain analysis that need its work products, such as domain design, implementation and application engineering.

This chapter is organized as follows: section 3.1 discusses the previous studies. The systematic review process is described in section 3.2 and its phases' description start in section 3.3 describing the research question. Section 3.4 details how the review was conducted, and analyzed. The results obtained are presented in section 3.5 and the identified priorities for the functionalities are presented in section 3.6. The threats to validate the review are described in section 3.7 and, finally, section 3.8 summarizes the findings.

3.1 Previous Studies

Previous studies have suggested a set of requirements for the domain analysis and have analyzed them on existing tools.

Succi *et al.* (2000a) define some requirements necessary for a domain analysis tool, focusing on functionalities a tool should have in order to have a consistent environment, such as traceability, consistency check and tool integration. Furthermore, they discuss how four existing domain analysis tools satisfy them, afterwards they propose a new tool that fulfills all the requirements.

Gomaa and Shin (2004) propose a set of requirements for an effective variability management in product lines; these requirements are based on views that can be compared to requirements defined by them, such as feature model, metamodel, consistency checking and product derivation. They also propose a multiple product line view and a product line repository. Their requirements are not only for the domain analysis phase, but for the whole SPL process. On the other hand, these requirements are not analyzed on other tools, just with their prototype.

Furthermore, other studies have looked at restricted aspects of the process, such as variability modeling. These studies are described next.

Capilla *et al.* (2007) describe a set of concepts for variability modeling and management, such as binding time, features types, traceability, dependency model and product derivation; after they do an analysis of several existing tools according to these concepts. However, they analyzed it, not only for domain analysis, but for the complete software product line and domain engineering processes. As result, they defined the current limits of tool support for the processes and what should be expected for the next generation of tools.

Sinnema and Deelstra (2007) define criteria for comparing variability modeling notations and techniques and other criteria, such as model choices, constraints, configuration and specification, for the analyzed tools, which were selected based on the process they supported. However, the results focus on the techniques and not on the tool support.

Therefore, this study is the first to present a comprehensive and systematic analysis of domain analysis tools and their capabilities. Next section starts describing the methodology used in this review.

3.2 Systematic Review Process

Systematic review refers to a specific research method, developed as a way to enable the evaluation and interpretation of existing evidence to a research question, subject matter, or event of interest (Kitchenham, 2004; Biolchini *et al.*, 2005). Some of the reasons for performing a systematic review are (Kitchenham, 2004; Travassos and Biolchini, 2007):

- To review existing evidences about a treatment or a technology;
- To identify gaps in current research;
- To provide a framework/background for new research activities; and,
- To support the generation of new hypotheses.

The reasons for this systematic review were in identifying the current gaps in the available domain analysis tools and in providing a background for new researches activities.

This systematic review follows Kitchenham's (Kitchenham, 2004) guidelines, which are divided in three main phases:

- *Planning the review.* Has the goal of developing a protocol that specifies the plan that the systematic review will follow to identify, assess, and collate evidence.
- *Conducting the review.* Responsible for executing the protocol planned in the previous phase.
- *Reporting the review.* Responsible for relating the review steps to the community and it is fulfilled with this review chapter.

Each of these phases contains a sequence of stages, but the execution of the overall process involves iteration, feedback, and refinement of the defined process (Kitchenham, 2004). Next section starts describing the systematic review performed in this chapter.

3.3 Planning the Review

This section details the research question that guides this review and its structure.

3.3.1 Research Question

The objective of this review is to find out **how the available tools for domain analysis are offering the support to the process**. Through this question, it will be possible to identify if the existing tools support the same functionalities and how extensive is this support. However, this question is too generic for a complete evaluation. Thus, it was further divided into Sub-Questions (SQ), focusing in specific aspects of the evaluation.

SQ₁. **Do the tools support a specific or a generic process?** The idea of not being focused in a specific domain analysis process increases the chances of embracing this tool, since the organization does not need to adopt a new process just to use the tool.

SQ₂. **What are the main functionalities of the tools?** The second SQ regards how the tool supports the process guidelines, identifying its characteristics. The analysis did not focus on the strengths nor on the weakness of the tools, because the goal was to identify what the tools do and compare them. Thus, it is possible to map how the process is supported.

SQ₃. **Where the tools were developed and used?** The last SQ aims at identifying where the tools were developed and how they have been validated. Through these descriptions, it is possible to map the current adoption of the tools.

Finally, this research must offer a novel contribution to the software reuse, and therefore it should not expend effort in a direction that is already being explored by other researchers.

3.3.2 Question Structure

Kitchenham (2004) details the question structure in three aspects:

- **Population.** people, projects and application types affected by the intervention. They should be directly related to the question;
- **Intervention.** software technology, tool or procedures, which generates the outcomes; and,
- **Outcomes.** Technology impact in relevant information terms for practical professionals.

Therefore, the structure for the research question is:

Population: this question refers to the support of domain analysis tools to the process, thus the population is composed by domain analysts and/or domain experts seeking a way to have a more automated process support, and by researchers in domain engineering/software product line areas aiming at new tools.

Intervention: this review must search for indications that the domain analysis process can be fully supported.

Outcomes: the objective of this study is to map how tools are supporting a domain analysis process. If the process is not fully supported, i.e., it has many gaps existing in the tools or if there is a necessity of using several tools in the whole process; the necessity for a new domain analysis tool increases.

Next, it is described how the review was conducted.

3.4 Conducting the Review

The steps referred to how the review was conducted are detailed in this section. It involves the search strategy, the studies selection and the data analysis, extraction and synthesis.

3.4.1 Search Strategy

From the question structure and the research question, it was extracted some keywords used to search the primary study sources. The initial set of keywords were: *domain analysis*, *tool* and *domain engineering*. However, after a preliminary search, which is aimed at finding available systematic reviews and assessing the volume of potentially relevant studies (Kitchenham, 2004), other keywords were added. Some of them were derived from concepts of the domain analysis, such as *feature modeling* and *variability modeling*, while others were identified after the analysis of the preliminary search, like *software product line* and *requirements*.

Furthermore, sophisticated search strings could then be constructed using boolean AND's and OR's. Once the review aims at identifying the tools available, the keyword *tool* was included in every search string, because in the preliminary searches several results returned referred to only processes and not tools. The main search strings were "*domain analysis AND tool*" and "*software product line AND tool*", and also other keywords, such as "*feature modeling OR variability modeling AND tool*", "*software product lines AND requirements AND tool*", "*software product lines OR domain analysis AND tool*", "*domain engineering AND tool*" and "*feature modeling AND variability AND tool*".

The search for primary studies was performed in the digital libraries of the most famous publishers and organizations in software engineering. The list of sources, in alphabetical order, is the following: ACM Digital Library¹, IEEE Computer Society Digital Library², Science@Direct³ and Springer Link⁴. These searches had as target some journals and conferences, which are detailed in Appendix A. These libraries were chosen because they some of the most relevant sources in software engineering (Brereton *et al.*, 2007; Travassos and Biolchini, 2007).

Furthermore, the described search engines are focused on academic results; given the fact that the goal was to find the largest number of tools as possible, and these engines would not find commercial tools (as long as they do not have any paper or journal published), these keywords were also used in search in web search engines, such as Google. In the web engines, the target was for tools information and their grey literature (i.e. technical reports, whitepapers, manuals and works in progress).

Websites of researchers and research groups active in this area e.g. Alexandria⁵, Fraunhofer IESE⁶, RWTH Aachen⁷ and SEI⁸ were also reviewed. Besides the search in digital libraries and web engines, papers referenced, usually in the related work section, by the authors of the papers found during the first search were analyzed too.

Since the goal was to find tools that support the domain analysis process, the availability of tools executables was also investigated, such as prototypes and/or demos. Thus, a wider analysis of the tools could be undertaken. These executables, sometimes, had their download link available in research papers, others were found through a search in web engines and some were found through research groups' websites or with the direct contact with the tool's owners.

Based on these sources, there are two types of objects in this review. The first concerns the executables, through which the reviewers could test the tool's functionalities; and the second involves the written documentation found, for instance, conference papers, manuals, whitepapers and journals.

These searches were performed by a M.Sc. and a Ph.D. students, the achieved results were compared and then validated. All the search process results are documented

¹<http://portal.acm.org/>

²<http://ieeexplore.ieee.org/>

³<http://www.sciencedirect.com/>

⁴<http://springerlink.metapress.com/home/main.mpx>

⁵<http://www.theinf.tu-ilmenau.de/riebisch/pld/index.html>

⁶<http://www.iese.fraunhofer.de/fhg/iese/index.jsp>

⁷<http://www-lufgi3.informatik.rwth-aachen.de>

⁸<http://www.sei.cmu.edu/sei-home.html>

| Data Source | Documentation |
|-----------------|---------------------------------------|
| Digital Library | Library name Search string Date |
| Research Group | Group name Url Date |

Table 3.1 Search documentation process

at <http://www.cin.ufpe.br/~lbl2/systematicreview>. Therefore, it is clear for others how deep the search was, and how they can find the same documents.

The adopted documentation process for the results are shown in Table 3.1.

Search Results

During the searches (with all the defined search strings), several results not relevant for the review were found and discarded. The majority of unused results referred to domain analysis/software product line processes, domain design and experience reports that did not include any tool evaluation. In order to filter the results that were relevant or not, the result's title, abstract and keywords (when available) were superficially read. Furthermore, some of the processes description results had as future work a tool implementation or referred to existing tools. The tool's name was used as a search string.

From the direct search using the defined keywords, **thirty-two** relevant studies among conference papers, journals and technical reports plus **four** tool websites were identified.

Through the encountered papers references and the researchers in the field contacted, it was possible to identify new papers describing some of the tools already found and other new ones. Moreover, four more tool websites were located.

After obtaining documentation about the tools (papers and website), another search was performed on web engines with the particular information of every tool in order to find more documentation and/or executables. When the searches were not enough, the tool's owner (or author of the paper) was contacted. Thus, at the end of the search, there was a total of **fifty-nine** written reports (including white papers, manuals, conference papers, journals and thesis), **sixteen** websites and **thirteen** executables. Finally, **thirty-one** potentially relevant tools were selected, of which **four** only had the information available on the websites and their executables, i.e. had no written reports.

These searches were executed in December 2007. However, a previous survey was performed with few papers and tools in the beginning of 2007. It aimed at understanding

the domain analysis tool support, but it was not enough, therefore a systematically analysis was necessary. The papers and tools previously found were also found in this new search.

3.4.2 Studies Selection

Once the potentially relevant primary studies have been obtained, they need to be assessed for their actual relevance. To achieve that, the criteria for inclusion and exclusion of the objects in this review were defined.

The main goal of the review was to map the tool support for a domain analysis process. Therefore, all the available information, achieved in the search - i.e. executables, papers, grey literature and journals - according to the tools they refer to were grouped. Due to this, it was possible for a tool to be discussed in more than one paper. For the sake of simplicity, whenever the word *tool* is used in the rest of the review, it means the written documentation and the executable. The identified criteria are detailed next.

Inclusion Criteria

These criteria were aimed at establishing the inclusion reasons for each tool encountered during the search. The criteria are:

- a) **Tools that offer support to the domain analysis phase:** the encountered tools must support some functionality in the domain analysis process⁹. If the tool supports functionalities that do not belong to the domain analysis phase, they will not be considered in the evaluation.
- b) **Tools with available executables:** with the tool's prototype, demo or finalized product, it is possible to test how the tool fulfills the process.
- c) **Tools with documentation describing their functionalities:** if the explanation of the written documentations were clear and useful about the tool's functionalities, they were considered too.

Not all of these criteria must be present for every tool. However, at least two of them, the domain analysis support (a) and some other, are necessary, because not every tool have the information from (b) and (c) available. If all criteria were mandatory, the number of selected tools would decrease significantly.

⁹The only exception for this criterion is described in the second exclusion criterion, because the existence of tools without supporting the commonalities identification goes against the domain analysis definition.

Exclusion Criteria

The exclusion criteria are:

- a) **Tools supporting functionalities that are from steps after the domain analysis conclusion:** tools supporting just activities that are not part of the domain analysis process, but that need its results - such as product configuration and domain design - were not considered in the review.
- b) **Tools supporting only variability management:** tools whose focus is only on managing variabilities and not on supporting commonalities were not included, since the definition of domain analysis involves the identification of what is common and what is variable in the systems of a specific domain (Neighbors, 1981).
- c) **Tools with written documentation with no usable information:** tools with written documentation that did not have clear description about its functionalities were discarded.
- d) **Tools describing only few functionalities:** tools with only written documentation that did not focus on explaining the process support but only a singular functionality, were also discarded.

Every decision to include/exclude a tool was made after the full reading of the written documentation (except for those where the title and the abstract clearly indicated its content) and conducting some tests with the executables.

In cases when there was not an agreement between the researchers about a specific tool, there was a discussion in which the researchers related his/her reasons to include or not the tool. If even after this discussion an agreement was not achieved, a third researcher (the Ph.D.) analyzed the available information about the tool in order to achieve a consensus.

3.4.3 Data Analysis

The first and third sub-questions of the review had the goal of identifying the external characteristics of the tools, in order to obtain an overview of their environment. While the second sub-question focused on how the tools help the process execution.

First, all of this information were sought in the available documentation and executables, however, if it was not possible, further information about this would be researched within the authors research groups' past work and in the tool's website (if they exist).

One problem faced during the review was that, sometimes, there were contradictions among the available information of the tools. This happened when it had different sources with some years between them. Thus, the eldest sources from the review analysis were unconsidered.

3.4.4 Data Extraction

The objective of this stage is to design data extraction forms to accurately record the information obtained by the researchers from the primary studies. The form for data extraction provides some standard information, such as:

- Tool's name;
- Date of data extraction;
- Title, authors, journal, publication details (if available);
- Prototype information (if available);
- Website (if available); and,
- A list of each conclusion and statement encountered for each sub-question.

The form's template can be found at: <http://www.cin.ufpe.br/~lb12/systematicreview/selected/template.doc>.

Moreover, the form contains a specific space for each of the sub-question, from SQ1 to SQ3. For SQ1, the expected answer was the name of the process supported by the tool, or if it supported a general one. SQ2 is answered with a list of the functionalities identified, while SQ3 was further divided with the information regarding the name and type of the institution that developed the tool, and the where and how it was used.

In addition, extra information could also be included, these information usually referred to the tools functionalities that were not related to the domain analysis support.

3.5 Data Synthesis

Based on the search results and on the inclusion and exclusion criteria, a set of tools were selected, which is detailed next.

3.5.1 Tools Selection

The tool selection process was performed by a M.Sc. and two Ph.D. students and a Ph.D. in conjunction with weekly discussions and seminars with the Reuse in Software Engineering (RiSE)¹⁰ group. This group has over 30 members among Ph.D. and M.Sc. students and more than 4 years of experience in state-of-the-art/practice in software reuse. These discussions and seminars intended to improve the tools questionnaires completion and analysis.

After the analysis, **nineteen** tools were selected. Within these tools, **thirty-six** written reports, **nine** executables and **eleven** websites were kept for further analysis. The number of selected tools with each type (written documentation, executables and websites) of available information is described in Table 3.2.

| Category | Number of Tools Selected |
|---|--------------------------|
| Written documentation + executable + websites | 5 |
| Written documentation + websites | 3 |
| Written documentation + executables | 1 |
| Written documentation | 7 |
| Executable + websites | 3 |

Table 3.2 Number of tools selected in each category

A brief description about the selected tools is presented below in alphabetical order and the available information about each tools is depicted in Table 3.3.

001. The 001 tool support was developed at Hamilton Technologies, Inc in 1986. However, the Software Engineering Institute (SEI) developed its integration to domain analysis in 1993.

Ami Eddi. It was developed at the University of Applied Sciences Kaiserslautern, Germany, in 2000.

ASADAL. ASADAL (A System Analysis and Design Aid Tool) was developed at the Pohang University of Science and Technology, Korea in 2003.

CaptainFeature. CaptainFeature was developed at the Fachhochschule Kaiserslautern in Germany, 2002.

DARE. The Domain Analysis and Reuse Environment (DARE) was developed in 1998 by the companies Reuse Inc. and Software Engineering Guild, USA. The tool is currently in its third prototype.

Decimal. Decimal was developed at the Iowa State University, US in 2002.

¹⁰<http://www.rise.com.br/research>

Domain. In 1995 Loral federal Systems, US developed Domain as part of a project sponsored by the US Department of Defense (DoD) and the US air force.

DOORS extension. It is an extension to the tool Telelogic DOORS¹¹. The extension was developed in 2005 at the Institute for Computer Science and Business Information Systems (ICB), University of Duisburg-Essen, Germany.

DREAM. The Domain REquirement Asset Manager in product lines (DREAM) was developed by the Pusan National University, Korea in 2004.

FeaturePlugin. It was implemented at the University of Waterloo, Canada in 2004.

FeatureIDE. Otto-von-Guericke-University Magdeburg, Germany developed the FeatureIDE in 2005.

GEARS. GEARS was developed by BigLever, US, in 2001.

Holmes. Holmes was implemented by the University of Calgary, Canada in 1999.

Odyssey. It is a software development environment whose goal is to construct a reuse infrastructure based on domain models and component based development. It was developed by the Federal University of Rio de Janeiro, Brazil, in 2002.

Pluss toolkit. It is a set of extensions for the Telelogic DOORS and the IBM-Rational Rose developed in 2005 by the Umeå University and Land Systems Hägglunds, both from Sweden.

PuLSE-BEAT. Developed by the Institut Experimentelles Software Engineering (IESE), Germany in 2000, the Product Line Software Engineering - Basic Eco Assistance Tool (PuLSE-BEAT) is part of the Product Line Software Engineering - PuLSE methodology, which is an approach to product line engineering.

Pure::Variants. Implemented by Pure-Systems' company in Germany in 2005, Pure::Variants has three kinds of license, a free one and two paid ones: a professional and an enterprise. The analysis was done with its free license.

RequiLine. The Research Group Software Construction (RWTH), Germany, developed it in 2005.

XFeature. It was developed in 2005 by an association of P&P Software Company with Swiss Federal Institute of Technology (ETH).

Figure 3.1 details the selected tools in the chronological order. Through this Figure, it is possible to verify that the number of tools offering support to domain analysis is increasing, with almost half of them released in the last four years.

¹¹<http://www.telelogic.com/products/doors/index.cfm>

| Name | Documentation | Executable | Website |
|-----------------|--|------------|---|
| 001 | (Hamilton, 1991; Krut Jr, 1993) | No | http://world.std.com/hti/Product/Product.htm |
| Ami Eddi | No | Yes | http://www.generative-programming.org |
| ASADAL | (Postech, 2003; Kim <i>et al.</i> , 2006) | Yes | No |
| CaptainFeature | No | Yes | https://sourceforge.net/projects/captainfeature |
| DARE | (Prieto-Díaz <i>et al.</i> , 1992; Frakes <i>et al.</i> , 1997; Frakes <i>et al.</i> , 1998) | No | No |
| DECIMAL | (Padmanabhan and Lutz, 2005; Dehlinger <i>et al.</i> , 2007) | No | No |
| Domain | (Tracz and Coglianese, 1995) | No | No |
| Doors Extension | (Buhne <i>et al.</i> , 2005) | No | No |
| DREAM | (Park <i>et al.</i> , 2004; Moon <i>et al.</i> , 2005) | No | No |
| Feature Plugin | (Antkiewicz and Czarnecki, 2004; Czarnecki <i>et al.</i> , 2005) | Yes | http://gsd.uwaterloo.ca/projects/fmp-plugin/ |
| FeatureIDE | (Leich <i>et al.</i> , 2005) | Yes | http://www.witi.cs.uni-magdeburg.de/iti_db/research/featureide/ |
| GEARS | (BigLever, 2005; Krueger, 2005, 2007) | No | http://www.biglever.com |
| Holmes | (Succi <i>et al.</i> , 1999, 2000b, 2001b,a) | No | http://world.std.com/hti/Product/Product.htm |
| Odyssey | (Braga <i>et al.</i> , 1999; Braga, 2000) | Yes | http://reuse.cos.ufrj.br/site/pt/index.php |
| Pluss Toolkit | (Eriksson <i>et al.</i> , 2005a,b, 2006) | No | No |
| PuLSE-BEAT | (Schmid and Schank, 2000) | No | No |
| Pure::Variants | (Beuche and Spinczyk, 2003; Pure-systems, 2004; Spinczyk and Beuche, 2004) | Yes | http://www.pure-systems.com |
| RequilLine | (Massen and Lichter, 2003; RWTH-Aachen, 2005) | Yes | http://www-lufg13.informatik.rwth-aachen.de/TOOLS/requiline/ |
| XFeature | No | Yes | http://www.pnp-software.com/XFeature/ |

Table 3.3 The analyzed information of each tool

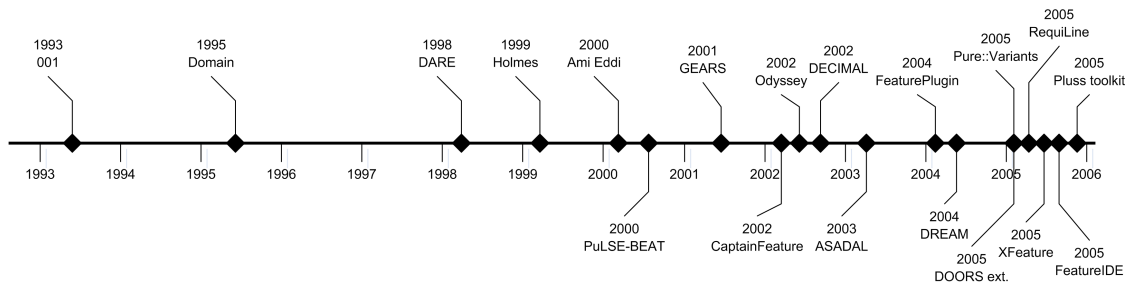


Figure 3.1 Selected tools timeline

3.5.2 Research Question Result

After the tools selection and their data extraction, each tool analysis was confronted with the research question - and its sub questions. Similar conclusions about the data were grouped and in cases where they were not much similar, the tools were ranked, so a better identification of the current scenario could be achieved.

Whenever doubts or uncertainty over an extracted data occurred, they were discussed with the research group. If no conclusion was reached, the analysis was repeated or further research was undertaken.

For the rest of this section, the results of the systematic review conducted with the objective of mapping **how the available tools for domain analysis are offering the support to the process** are presented.

Since the research question was further divided in three sub-questions, each one of them has its conclusion presented first. After the results of the SQs, a summary relating to the main question is presented.

Domain Analysis Support

Based on the nineteen tools analyzed, it was possible to identify that the tools are usually developed to support a specific process, under the justification that there are no tools supporting all functionalities of a specific process. These tools correspond to more than 78% of the total (fifteen tools, as shows Figure 3.2).

However, as shown in Figure 3.3, among the tools supporting Specific Processes, five of them supported the same process - the Feature Oriented Domain Analysis (FODA) process (Kang *et al.*, 1990). This process defines several activities for domain analysis, such as context model, feature model, entity-relationship, functional models and architectural models. Conversely, the majority of the tools (3) support only the feature model

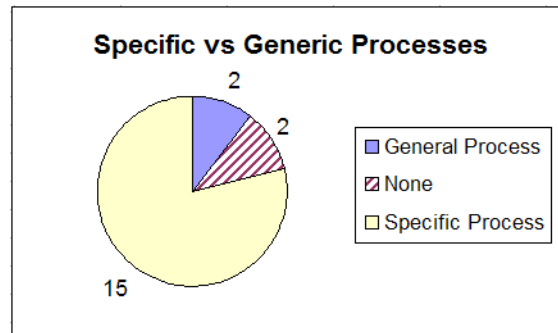


Figure 3.2 Domain analysis processes support

activity, while the other two tools support the whole FODA process (one of them supports an extension of the FODA process - the Feature Oriented Reuse Method (Kang *et al.*, 1998)). Thus, it is possible to consider that at least this activity - **feature modeling** - is generic.

Two of the remaining tools in Figure 3.2 support generic processes. They are based on a mix of several processes; however, one tool describes that the supported process has to be feature-oriented. Finally, the last two tools do not specify which process they support, but they do offer support to an activity similar to feature modeling. Thus, they can also be considered as tools supporting generic processes.

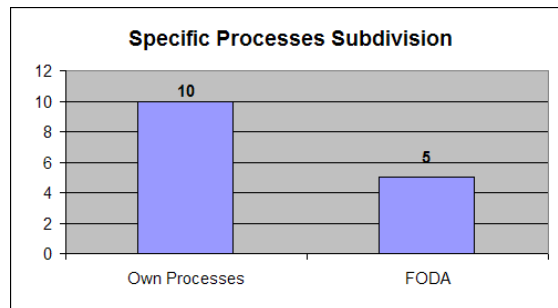


Figure 3.3 Specific Process Subdivision

Main Functionalities

Even though the tools' developers justify the construction of a new tool because the available ones do not support a specific domain analysis process, this review identified that the majority of the analyzed tools have similar functionalities.

Moreover, the extracted functionalities have analogous goals, so it was possible to

group them. This classification matched the domain analysis process subdivision that some processes have (Bayer *et al.*, 1999b; Almeida *et al.*, 2006). In this review, the same names used in (Almeida *et al.*, 2006) were adopted. The groups are:

- **Planning:** It is responsible to collect the data needed to define the domain scope. These data refer to information already available from the domain being analyzed, such as legacy systems, specialists and customer objectives. The collection of this information aids in identifying the domain's characteristics towards the definition of its scope.
- **Modeling:** It represents the domain scope in another way, which can be through diagrams, tables and so on. It presents domain's commonalities and variabilities, and it is responsible for creating the constraints between domain's characteristics.
- **Validation:** This group refers to functionalities responsible to validate the domain. These functionalities include documentation and reports.

Each group's functionalities are detailed next. Table 3.4 details which functionalities each tool offers support. The roman numbers refer to the functionalities described next and the grey columns separate each group (Planning, Modeling, Validation and Extras, respectively) of functionalities. This table facilitates the identification of the gaps in the selected tools, and in addition it can help discovering which tool best satisfies a company's needs.

Planning Functionalities

The identified functionalities and their explanation are:

- i) Pre Analysis Documentation:** stores and retrieves the information in order to help the identification of what characteristics should be part of the domain. This information can be stakeholders and market analysis, constraints, objectives definition and data collection.
- ii) Domain's Matrix:** represents the relationship between the domain's characteristics, also called features, and the applications included in the domain. Its representation is done through rows and columns, where the former represents the features and the latter the applications, so it is usually represented with tables. This matrix aids in the identification of which features are common or variable in the domain.

| Functionalities / Tools | i | ii | iii | iv | v | vi | vii | viii | ix | x | xi | xii | xiii | xiv | xv | xvi | xvii | xviii | xix | xx |
|-------------------------|---|----|-----|----|---|----|-----|------|----|---|----|-----|------|-----|----|-----|------|-------|-----|----|
| 001 | | | | | • | • | • | • | | | | • | | | | • | • | • | • | |
| Ami Eddi | | | | | • | • | • | | | | | | | | | | | | • | |
| ASADAL | | | | | • | • | • | • | • | • | | | | | | | | • | • | |
| Captain Feature | | | | | • | • | • | • | | | | | | | | | | • | • | |
| DARE | • | • | | | • | • | • | | | | | • | | | | • | | | | |
| DECIMAL | | | | | • | • | • | • | | | | | | | | | | • | • | |
| Domain | | | | | • | | | | | | | • | | • | | • | | | | |
| Doors Extension | | | | | • | • | • | • | | | | | | • | • | | | • | | |
| DREAM | | • | | • | • | • | • | | | • | | | | | • | | • | | | |
| Feature Plugin | | | | | • | • | • | • | | | • | | • | | | | | • | • | |
| FeatureIDE | | | | | • | • | • | • | | | | | | | | | | • | • | |
| GEARS | | | | | • | • | • | • | | | | | | | | | • | • | • | |
| Holmes | • | • | | | • | • | • | • | | | | • | | | | • | | • | • | |
| Odyssey | | | | | • | • | • | | | | | • | | • | • | | | | | • |
| Pluss Toolkit | | | | | • | • | • | • | | | | | | | • | | • | • | • | |
| PuLSE-BEAT | | • | • | • | • | • | • | | | | | • | • | | | | • | | | |
| Pure::Variants | | | | | • | • | • | • | | | | • | • | | | | • | • | • | |
| RequiLine | | | | | • | • | • | • | | • | • | • | • | • | • | | • | • | • | • |
| XFeature | | | | | • | • | • | • | | | • | • | • | | | | • | • | • | • |

Table 3.4 Functionalities each tool supports

iii) **Evaluation functions:** is responsible for the process metrics. It is divided in characterization and benefits functions. The former evaluates characteristics for the applications, while the latter uses the former's results to determine the best characteristics and best products for the domain analysis to cover (Bayer *et al.*, 1999b).

iv) **Scope definition:** identifies the features that should be part of the domain's reuse infrastructure.

In accordance with the processes, this group consists of functionalities that should be firstly executed in a domain analysis.

Modeling Functionalities

The modeling group represents the domain scope defined in the planning phase. Its functionalities are:

v) **Domain representation:** represents the defined scope. This representation can be through tables, models, tree and so on. This model attempts to formalize the commonalities and variations in the domain.

vi) **Variability:** represents the variabilities a feature can have. The possible types are *optional* - it can or not be present in the product; *alternative* - from a group of

features only one feature will be in the product; and *or* - from a group of features at least one feature will be in the product.

- vii) **Mandatory Features:** represents the features that will always be in the products.
- viii) **Compositions rules:** creates restrictions in the domain for representing and relating the features. They can be mutual exclusion and dependency, regular expressions, or artificial intelligence, among others.
- ix) **Feature group identification:** classifies the features according to the type of information they represent. They can be *capability*, *domain technology*, *implementation techniques* and *operation environment* (Lee *et al.*, 2000).
- x) **Relationship types:** provides different types of relationships between the features. They can be *composition*, *generalization/specification* and *implementation*.
- xi) **Feature's attribute:** permits the inclusion of specific information for each feature. It can also represent a variation point, if the number of variants for it is too big, providing a more concise feature model.

As it is visible in Table 3.4, almost every tool supports the first four functionalities in this group; because of that, they were further classified as “**Common Functionalities**”. This mean that every domain analysis tool should always have them.

Validation Functionalities

The functionalities of the validation group are not dependent of the previous groups, but they provide a greater understanding of the domain. They are:

- xii) **Domain documentation:** provides documentation to the domain, consisting of: description, context, dependencies between systems in the domain, among others. Each tool presents a different set of fields for documentation.
- xiii) **Feature documentation:** provides documentation for every feature of the domain, with information such as: description; rationale; priorities, etc. Each tool presents a different set of fields for documentation.
- xiv) **Requirements management:** provides support to include the requirements and/or use cases in the tool.
- xv) **Relationship between features and requirements:** relates the existing features of a domain to the requirements (functional or non-functional) and/or use cases

defined. Through this relationship it is possible to keep the traceability between the artifacts produced in the domain.

- xvi) Dictionary:** identifies the words commonly used in the domain and mostly found during pre-analysis. In addition, their meaning is also specified.
- xvii) Reports:** generates reports about the information available in the domain. The reports can represent, for example, *the number of possible combinations; the frequency in which features appear in the product; the documentation of the artifacts.*
- xviii) Consistency check:** verifies if the generated domain follows the composition rules created.

Since the *consistency check* is supported by most of the tools (see Table 3.4), it was also classified as a **common functionality**.

Extra Functionalities

Besides the functionalities described in the previous groups, which are intrinsic to the domain analysis, two other functionalities that refer to a step after the domain, i.e. the product derivation, were included. They are:

- xix) Product derivation.** identifies the features that belong to a product according to the features defined for the domain, i.e. it generates a feature model instance based on the domain feature model.
- xx) Product documentation.** presents documentation to every product with information, such as *product description* and *domain version*.

Even though they are not part of the domain analysis process, the product derivation function is supported by the majority of the analyzed tools.

Usability Requirements

Through the analysis of the selected tools - considering only the ones that had the executables available - some usability requirements were identified. These requirements aim at providing an easier and more intuitive environment for tool's users. However, they are not presented in Table 3.4, because not all tools had executables and some of them were identified due to its non-existence in the executables, making it harder to use the tool.

The identified requirements are detailed next:

- **Tutorial:** the goal of the tutorial is to describe the first execution steps in the tool in order to support the domain analysis process.
- **Graphical User Interface (GUI):** the existence of a graphical interface makes the environment more intuitive to the users.
- **Help/Manual:** the help/manual existence, usually, provides a detailed explanation of the tool's functionalities.
- **Example solutions:** example solutions of domains already analyzed in the tool are useful to help in the process of identifying the expected input and output, besides showing a real case to be consulted.
- **Import/Export from/to other applications:** imports and/or exports the generated documentation to/from others types of files, such as XML, XMI, PDF, XLS, JPEG and GIF. Furthermore, it allows the visualization of domain data in other tools.

The requirements *Tutorial* and *Example solutions* are mainly important for users that do not have any kind of previous training before the tool usage, because these requirements show how the user can start the domain analysis on it. And the lack of training was the case of this review, therefore the tools with these requirements were easily analyzed.

Functionalities Summary

Table 3.4, it is possible to notice that, apart from the **common functionalities**, the other identified functionalities are still too spread among the tools.

In the planning group, the identified functionalities were extracted only from **four** tools, and the maximum number of functionalities support by one tool is **three**. Therefore, it shows that there is still much room for improvements.

Although we identified that the common functionalities are supported by almost every tool, the way these functionalities are supported vary significantly from tool to tool. For example, the domain representation can vary from a descriptive visualization - through a table view - to a diagram view - using the feature model defined by (Kang *et al.*, 1990). The functionalities description details some of the variabilities encountered. Besides the common functionalities, the remaining ones for the modeling group are also seldom supported, especially the *feature group identification* that only one feature supports it.

Regarding the validation group, only one tool (*Ami Eddi*) does not support any functionality, in spite of this, not all functionalities are achieved by a unique tool. *RequiLine*

is the one that gets closer to it, but it misses the *dictionary*, besides this one, only other **two** tools support more than half of requirements for this group.

Tools Development and Usage

From the selected tools, most of them were developed in the academic environment - twelve - while **four** were developed exclusively in the industry and the remaining three tools were developed in both environments, academic and industrial, as shown in Figure 3.4.

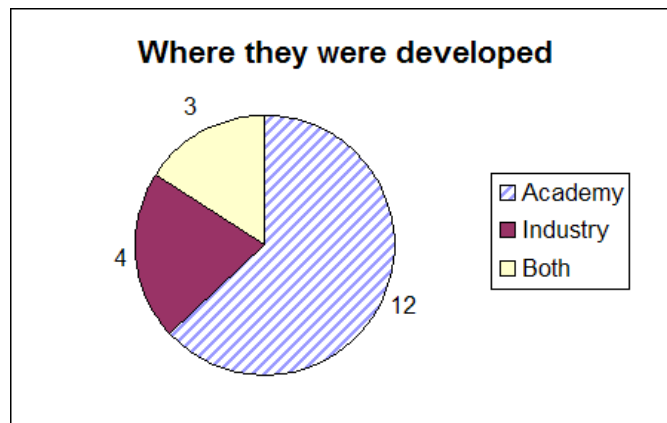


Figure 3.4 Where the tools were developed

However, for the second part of this sub-question - *where the tools were used?* - there were some troubles in finding this information for some tools. For several tools, especially the ones from the academy, these data were not found.

Since there was the information in which environment the tool was developed, it was assumed that the tool was used at least once in this environment, even without a report detailing it. Table 3.5 details in which environments the tools were used, and it includes the reports that describes their usage. Whenever a dash appears in the report column, it means that no report from the analyzed written documentation described any use of the tool.

The possible results for the *used* column (Table 3.5) are: Academic (A), Industry (I) and Both (B).

Even though the majority of the tools were developed in the academy, several of them were also used in the industrial environment - five. Furthermore, one of the tools that were developed just in the industry environment was also used in the academy. The rest were used in the same environment they were developed.

| Tool | Used | Report |
|-----------------|------|---|
| 001 | A | - |
| Ami Eddi | A | - |
| ASADAL | A | - |
| Captain Feature | A | - |
| DARE | B | (Prieto-Díaz <i>et al.</i> , 1992; Frakes <i>et al.</i> , 1998) |
| DECIMAL | B | (Padmanabhan, 2002) |
| Domain | I | - |
| Doors Extension | B | (Buhne <i>et al.</i> , 2005) |
| DREAM | B | (Moon <i>et al.</i> , 2005) |
| Feature Plugin | A | - |
| FeatureIDE | A | - |
| GEARS | I | (Krueger, 2002) |
| Holmes | A | - |
| Odyssey | B | (Braga, 2000) |
| Pluss toolkit | B | (Eriksson <i>et al.</i> , 2006) |
| PuLSE-BEAT | B | - |
| Pure::Variants | I | http://www.pure-systems.com |
| RequiLine | B | (Massen and Lichter, 2003) |
| XFeature | B | - |

Table 3.5 Where tools were used

Research Question Result Summary

In this section, the results of the review are summarized according to the ones extracted from the sub-questions. As described in **SQ1**, the development of a tool usually comes from the necessity of supporting a specific process and not a generic one. However, this usually obligates the company that wishes to use the tool to modify or adapt the process it already adopts. Thus, the learning curve and the impact it will cause to the company's development life cycle is bigger.

Considering the functionalities in **SQ2**, the tools are mainly focused on the modeling phase due to the common functionalities (see Table 3.4). However, if only the functionalities not considered as common are taken into account, just three tools offer some kind of support to them. That is similar to the support rate of the planning group functionalities, in which only the *domain's matrix* is supported by all the tools with functionalities within this group.

Regarding the common functionalities, from the nineteen tools supporting the *domain representation* in the modeling phase, seven of them do it through feature models. It reinforces the assumption that the feature modeling activity can be considered fundamental.

Analyzing Table 3.4, for the validation group, it is possible to observe that the tools do not provide much support to the documentation activities, even though domain analysis is a process whose results depend on the analysis and management of information from various sources (Bayer *et al.*, 1999a).

Comparing the black dots, in the validation group, with the year the tools were developed (detailed in section 3.5.1), it becomes clear that documentation functionalities are being more explored by recent tools. In addition, even though most of the tools (thirteen) permit the *product derivation* - among the extra functionalities - for the domain, only a small part of them (three) offer support to the *product documentation*.

Furthermore, based on Table 3.4 it is visible that there is still a lot of functionalities being supported by only a few tools. Thus, there are several improvements to be done in tools support for domain analysis process, particularly because there is not a tool that consistently supports all the functionalities in each group, since RequiLine offers no support to the planning group functionalities.

This lack of a consistent support also implicates in the degree of tools adoption by the companies, because it can force them to use several tools at the same time, one for each group's functionalities, to do the aimed domain analysis. In addition, this is exactly the scenario that must be avoided, since it probably forces the domain analyst to manually perform the data traceability, increasing the chances of errors in the analysis. There are reports considering that the existence of different tools, without a standardization on data format, during the process increases the work and the data traceability, therefore delaying the project execution (Steger *et al.*, 2004; Almeida *et al.*, 2007).

Moreover, it is expected that the domain analysis tools market continue to increase, as shown in Figure 3.1, with the release of new tools, or modification/extension of already available ones to fulfill some of gaps identified here.

Another outcome for this review, based on the functionalities identified, is the definition of priorities for these functionalities, once not all of them are essential for a domain analysis process execution. The priorities definition is detailed in the next section.

Through the results obtained for **SQ3**, it is clear that, even with the increasing adoption of software product lines and/or domain engineering by organizations (Bass *et al.*, 1997), the majority of tools are still being developed and used in the academic environment. Although there are other industrial tools that were excluded due to the defined criteria or because they are developed and used only by their companies (Succi *et al.*, 1999; Jaring and Bosch, 2002), the tools are still too focused on academic issues, and this is not desirable for a wider dissemination of their usage.

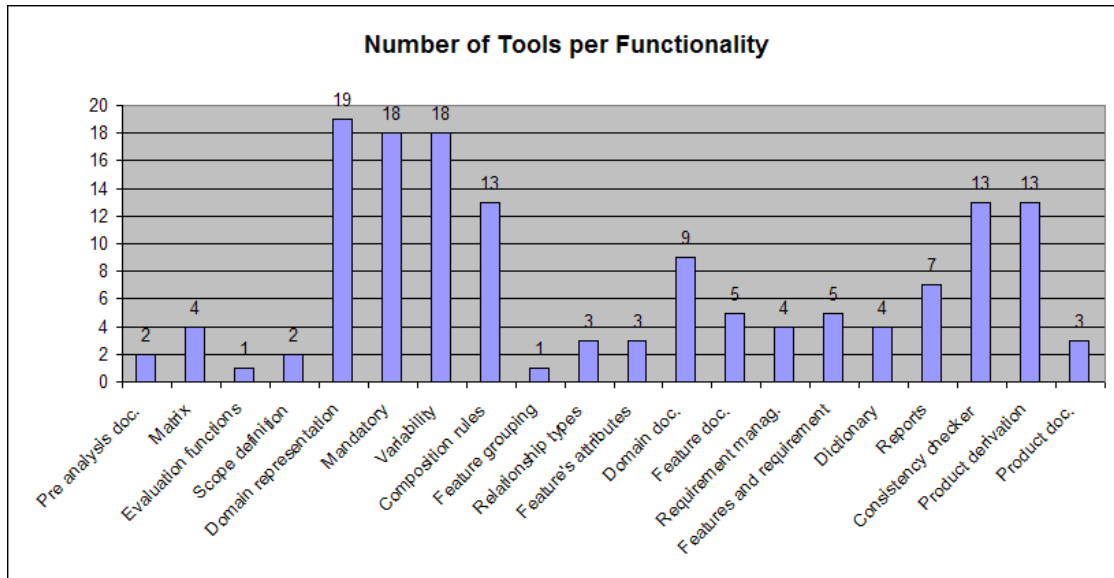


Figure 3.5 Number of tools per functionalities

3.6 Functionalities Priority

After the identification of the functionalities that are supported by the tools, not all of them must be fulfilled when executing a domain analysis project. Therefore, some levels of priorities were defined in order to categorize them. They are:

- **Essential.** It represents the indispensable and high-priority requirements that must be carried out. The lack of them turns the application useless.
- **Important.** It represents the medium-priority requirements that are strongly advisable for better usage of the tool.
- **Low.** It represents the low-priority requirements that are required for particular situations or enhancements for current development.

This categorization was based on the experience of the participants involved in the review and on the number of tools supporting each functionality, which is depicted in Figure 3.5. Therefore, the largest number of tools support should indicate that it is mandatory in every domain analysis project.

In Figure 3.5, the functionality “*relationship between feature and requirements*” had to be renamed to “*feature and requirements*” due to the lack of space in the graph.

Based on this Figure and the categorization results described next, researchers aiming at developing new domain analysis tools can have a defined set of requirements to imple-

ment, or others focusing on tools for the next phases - domain design and implementation - already have a set of artifacts to expect from the domain analysis. Furthermore, companies can use these results for identifying which tools best fit their needs or as requirements to develop their own tool.

3.6.1 Essential Priority

The first functionalities to be defined as *essential* are the ones defined as **common**, because the majority of tools already support them and they are also present in many domain analysis processes (Arango, 1994; Frakes *et al.*, 1997; Weiss, 1998; Almeida, 2007). Besides, other functionalities, from planning and validation, were set as essential, because they are key functions in the domain analysis process.

The complete set of essential functionalities is: *Domain's Matrix*, *Domain Representation*, *Variability*, *Mandatory*, *Composition Rules*, *Consistency Check* and *Domain Documentation*.

Even though the *Product Derivation* functionality is supported by 13 out of 19 tools; this activity is not a part of the domain analysis process, therefore, it was not defined as *essential*, but as a *important priority*.

3.6.2 Important Priority

In addition to the *Product Derivation*, the other important functionalities are: *Scope Definition*, *Feature Documentation*, *Requirements Management*, *Dictionary*, *Reports* and *Product Documentation*.

Moreover, the *Tutorial* and *GUI* functionalities from the usability requirements were also considered as important priorities, because they are directly related to the tool usage, and CASE tools should provided an easy-to-learn interface (Fuggetta, 1993).

3.6.3 Low Priority

The remaining functionalities had their priority defined as *Low*. It does not mean that these functionalities do not aggregate value to the domain analysis process, however their results do not provide a great influence the domain analysis final artifacts.

The complete set of essential functionalities is: *Pre Analysis Documentation*, *Evaluation Functions*, *Feature Group Identification*, *Relationship Types*, *Feature's Attribute*, *Relationship Between Requirements and Features*, and the usability functionalities *Example Solutions*, *Import/Export* and *Help/Manual*.

3.6.4 Priority Support by Tools

After the definition of which tools support the functionalities and their priority, it was possible to compare the proportion of priority functionalities each tool delivers. This proportion is depicted in Table 3.6. However, it does not include the usability functionalities, i.e. it has only the ones depicted in Table 3.4.

| Tool | Essential | Important | Low |
|-----------------|-----------|-----------|-----|
| 001 | 6 | 3 | 0 |
| Ami Eddi | 3 | 1 | 0 |
| ASADAL | 5 | 1 | 2 |
| Captain Feature | 5 | 1 | 0 |
| DARE | 5 | 1 | 1 |
| DECIMAL | 5 | 1 | 0 |
| Domain | 2 | 2 | 0 |
| Doors Extension | 5 | 1 | 1 |
| DREAM | 4 | 2 | 2 |
| Feature Plugin | 5 | 2 | 1 |
| FeatureIDE | 5 | 1 | 0 |
| GEARS | 5 | 2 | 0 |
| Holmes | 7 | 2 | 1 |
| Odyssey | 4 | 1 | 1 |
| Pluss toolkit | 5 | 2 | 1 |
| PuLSE-BEAT | 5 | 3 | 1 |
| Pure::Variants | 6 | 3 | 0 |
| RequiLine | 6 | 5 | 3 |
| XFeature | 6 | 3 | 1 |

Table 3.6 Functionalities each tool support per priority

The rows emphasized refer to the tools that fulfill more requirements - *RequiLine* and *Holmes*. Since this review did not focused on comparing how the functionalities are implemented, this emphasis is just according to the number of functionalities they support.

3.7 Threats to validity

The main threats to validity identified in the review are described next:

Tools selection. A possible threat in such review is to exclude some relevant tool. In order to reduce this possibility, the selection of tools was based on the identification of the key research portals in computer science and wide range web search engines, besides

aiming at target journals and conferences (Appendix A). The defined criteria intended to select tools supporting some functionalities of the domain analysis process and not just supporting specifics requirements.

Data extraction. In order to ensure the validity, multiple sources of data were analyzed, i.e. papers, prototypes, technical reports, white-papers and manuals, in addition to the tools' executables.

Functionality Fulfillment. Even though the support of functionalities is the main goal of this review, it did not verify how they are fulfilled by the tools. In spite of that, the functionalities description includes some of the ways that they vary among the tools, such as the *domain representation* that can be implemented through a table, tree or model. Consequently, the functionalities fulfillment can be verified within a more detailed review, and also as a way to identify the best solution among the options.

3.8 Chapter Summary

This chapter presented a systematic review on domain analysis tools, whose goal was to identify how the available domain analysis tools are supporting the process. Through the review, it was possible to identify which current functionalities are being supported by the tools, and which ones should be present in every domain analysis tool based on their priorities.

Several of the functionalities identified in this review were also related by the related work, however they first defined the functionalities to be analyzed in each tool to, latter, verify if the tools supported them. This type of approach restricts a complete mapping of the tool support scenery, such as the one reported here.

By publishing the set of characteristics described in this review, it is expected that it helps classifying some basic tool functionalities and also influencing future tools development, both in industry and in academia. And this review's output was the initial start for the tool development, as described in the next chapter.

*“Qual é a parte da tua estrada
No meu caminho”*

Zeca Baleiro and Alice Ruiz (Musicians)

4

ToolDay - Tool for Domain Analysis

The results achieved from the systematic review, described in the previous chapter, showed that there is a wide variety of available tools with different focuses and processes, what makes even harder the decision of which tool to use. Furthermore, the necessity of using several tools during the same domain analysis process increases the “manual” work that has to be performed by the domain analysts, since he/she will be the one to keep the traceability and consistency among the tools in the project.

Due to this and in accordance to Sommerville (2007), who says that process standardization is important in order to improve its quality and support, it was developed a tool called ToolDay - which stands for Tool for Domain Analysis - whose development included the requirements previously defined.

ToolDay’s goal is to facilitate the domain analysis process, making it more automated and aiding the domain analyst to achieve a systematic reuse in an effective way.

This chapter is organized as follows: section 4.1 lists the requirements proposed for the domain analysis tool and section 4.2 presents the tool architecture. Section 4.3 describes the set of frameworks and technologies utilized during the implementation. Section 4.4 details ToolDay’s implementation, while section 4.5 simulates a domain analysis process using ToolDay and, finally, section 4.6 summarizes the content presented in this chapter.

4.1 Requirements

Based on the study previously described and on RiSE’s group discussions and experience, the functionalities, with their levels of importance, identified in chapter 3 were the considered requirements for ToolDay’s development. Plus, it was added the **Spell Checker** and **Metrics** requirements.

Since ToolDAY provides a great set of documentation fields, it is important to offer a *spell checker* in order to facilitate the identification and correction of misspelled words. In addition, the *metrics* provide information about the quantity of items in the project.

Table 4.1 shows the expected requirements against their priority criterium. The requirements were divided according to the phases identified in chapter 3.

4.2 Tool Architecture

ToolDAY's architecture was developed as a three-layer view, as shown in Figure 4.1. It is composed of a *Graphical User Interface* layer (GUI) whose goal is to provide a friendly environment for the user, a *Business layer* that has each ToolDAY phases components, i.e. Planning, Domain Modeling, Domain Validation and Product Derivation; and the last layer regards the *Data Persistency* in ToolDAY.

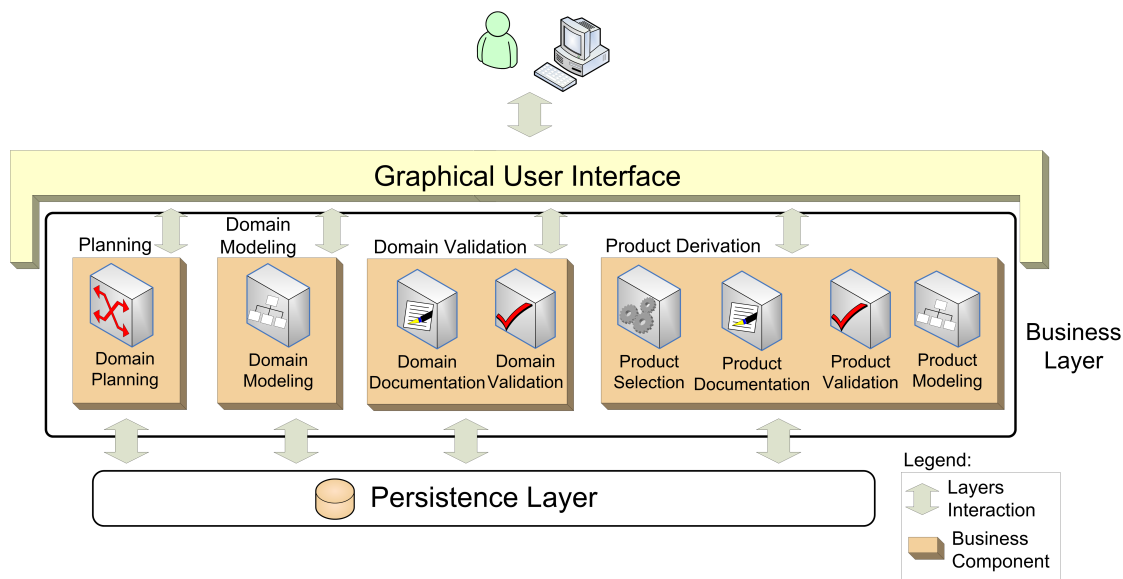


Figure 4.1 ToolDAY's architecture

Each one of these components is composed by some modules, and contributes in a proper way to satisfy the requirements. Figure 4.1 also details the interactions (arrows) among the layers in which every business layer component communicates with the GUI and the persistency, and all of its components are independent from each other.

| Requirement | Priority |
|--|-----------|
| Domain Planning | |
| Pre Analysis Documentation | Important |
| Domain's Matrix | Essential |
| Evaluation Functions | Important |
| Scope definition | Important |
| Domain Modeling | |
| Domain representation | Essential |
| Variability | Essential |
| Mandatory | Essential |
| Compositions rules | Essential |
| Feature group identification | Aimed |
| Relationship types | Aimed |
| Feature's Attribute | Aimed |
| Domain Validation | |
| Domain documentation | Essential |
| Feature documentation | Essential |
| Requirements management | Important |
| Relationship between features and requirements | Aimed |
| Dictionary | Important |
| Reports | Important |
| Consistency Check | Essential |
| Product Derivation | |
| Product modeling | Important |
| Product documentation | Important |
| Usability Requirements | |
| Tutorial | Important |
| Graphical User Interface (GUI) | Important |
| Example solutions | Aimed |
| Import/Export from/to other applications | Important |
| Spell Checker | Aimed |
| Metrics | Aimed |

Table 4.1 Summary of requirements

4.3 Technologies

Eclipse's platform has become the IDE choice for Java developers (Goth, 2005; Zetie, 2005), some of the reasons for that is because it provides a rich Java Development Tools (JDT) support and a plug-in architecture that allows tight integration of third-party functionality. Then, the tool's architecture could benefit from extensible plug-ins.

Thus, it was decided to develop ToolDay as a *stand-alone* tool based on Eclipse's Rich Client Platform (RCP) that is a platform for building and deploying rich client applications providing common applications services, such as native look and feel, windows management and help systems. These characteristics made RCP to be the chosen platform instead of Swing and applets applications.

Taking advantage of Eclipse's environment, several of its plug-ins were used to provide a friendly tool. Some of them were forms editors - for artifacts documentation; views; and its help technology of cheat sheets, which are special views that aid the user through a series of tasks to achieve an overall goal.

Moreover, for the graphical interface, it was used Eclipse's Graphical Modeling Framework (GMF)¹ that provides a generative component and runtime infrastructure for developing graphical editors, thus the graphical interface and some of the business behind it were abstracted.

Besides Eclipse's technologies, other technologies and frameworks were:

- **Java Math Expression Parser (JEP) library**², which is a Java library for parsing and evaluating mathematical expressions. It was used for the evaluation functions.
- **iText**³ is a library that allows you to enhance applications with dynamic PDF document generation and/or manipulation. It was used for the PDF reports creation.
- **Apache POI**⁴, it is a project that consists of APIs for manipulating various file formats based upon Microsoft's OLE 2 Compound Document format using pure Java. It was used for the excel report creation.

These libraries and frameworks were chosen because they best attended the requirements, and because they are free. The JEP version being used has a GPL license which is no longer available.

¹<http://www.eclipse.org/gmf/>

²<http://www.singularsys.com/jep/>

³<http://www.lowagie.com/iText/>

⁴<http://poi.apache.org/>

Based on these technologies, Figure 4.2 presents the technologies each architecture layer has. The GMF is part of the *GUI* and *business* layers because it abstracts the graphical interface and it has the business logic for the models representation in ToolDay.

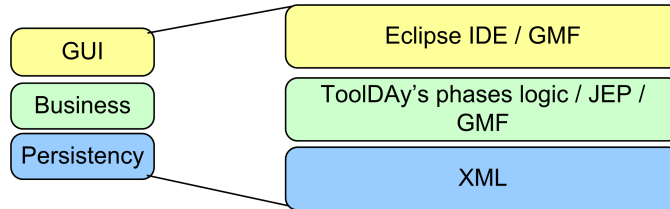


Figure 4.2 ToolDay's technologies

Using these technologies, ToolDay environment is depicted in Figure 4.3. The environment is similar to Eclipse, due to the RCP, with the files on the left side (a) and, when opened, they are visualized on the center (b). Also, different views can be added to the environment, like Metrics and Properties (c), other type of view is the tutorial (d) - on the right side - that explains the basic steps to do a domain analysis with ToolDay.

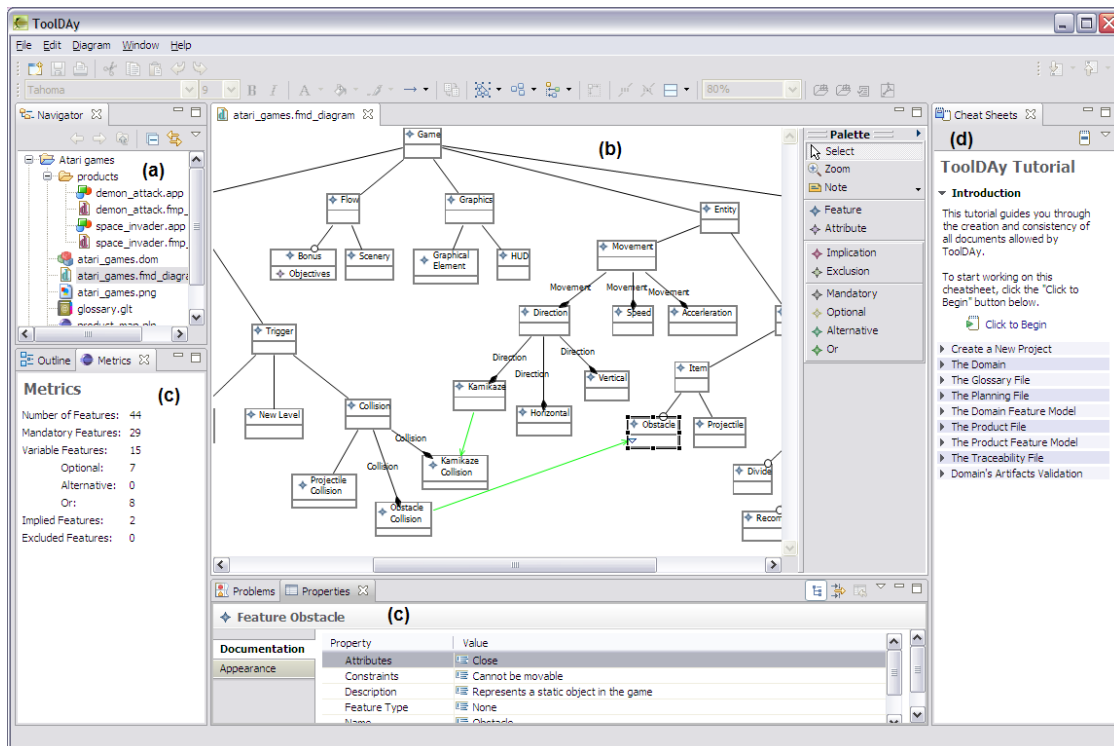


Figure 4.3 ToolDay's environment

4.4 ToolDay

A general definition of the architecture's proposed solution was presented in the last section. This section details how the requirements were implemented.

4.4.1 ToolDay's Steps

Based on the requirements identified before, it was possible to define the domain analysis process supported by ToolDay. Its flowchart is depicted in Figure 4.4 and details the process execution flow, however, the first step - *define the domain scope* - may or not be executed in the tool, i.e. the domain analyst can start the domain analysis process through modeling the domain.

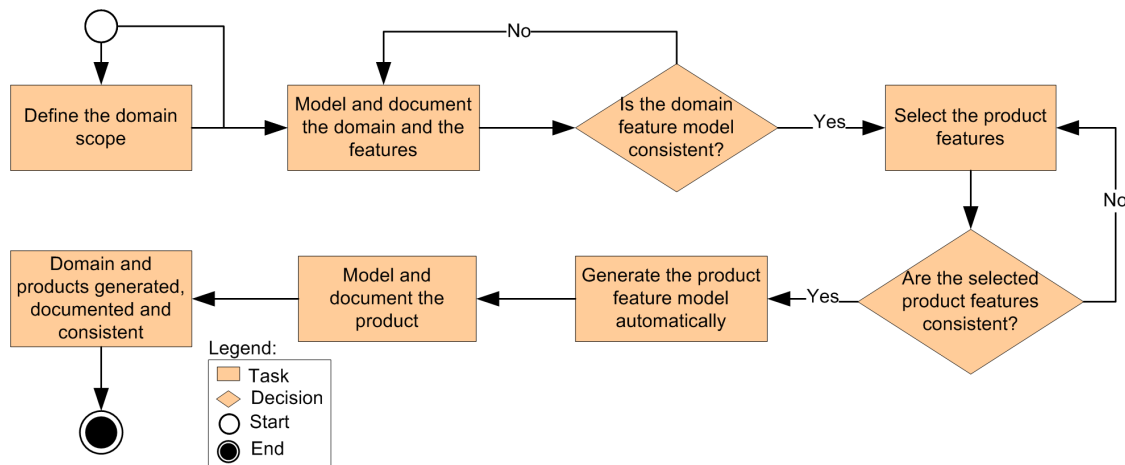


Figure 4.4 ToolDay's execution flow

This process is meant to be executed by the domain analyst, who is the principal user of this tool, however, the domain expert may also fulfill these steps. Whenever the word *user* appears in text it refers to the domain analyst role.

Domain Planning

This phase refers to the *define the domain scope* step, which is the first step in the domain analysis process in Figure 4.4, and details how the requirements identified for this phase were implemented.

ToolDay provides a set of documentation fields for the pre analysis documentation (Almeida, 2007):

- **Stakeholder analysis.** Identification of the stakeholders and their roles within the process.
- **Objectives definition.** Definition of the stakeholders' objectives for the domain.
- **Constraint definition.** Definition of the constraints imposed by the organization and/or market conditions.
- **Market analysis.** Identification of the external factors that determine the success of the domain in the marketplace.
- **Data collection.** Analysis of all available documentation (project plans, user manuals, modeling, and data dictionary), existing applications and knowledge from domain experts.

Even though ToolDay defines these fields, none of them are mandatory to be fulfilled by default.

The main artifact of this phase is the fulfillment of the **product map** - which corresponds to the *matrix* requirement - that allows the user to identify, among all possible features extracted during the pre analysis documentation, which ones should be part of the domain reuse infrastructure.

The identification of which features should or not be in the domain is achieved through the *evaluation functions* determined for the product map. These evaluation functions can be configurable according to the company's own metrics, i.e., the user can define the number of functions with their own formulas to be calculated during the product map fulfillment.

Despite permitting the functions customization, ToolDay provides a set of default functions, in cases when the company does not have its functions defined. The default functions are (Schmid and Schank, 2000):

- **Characterization Functions.** These functions are related to every application included in the product map, i.e. if there are 3 applications in the product map, each characterization function will appear 3 times. The functions are:
 - *Required.* Means if the feature is required or not in the product, and it is called *Req* in the tool.
 - *Similarity.* How similar the feature is according to the standard adopted, and it is represented in the tool as *Sim*.

- **Benefit Functions.** These functions totalize the values of the characterization functions for each feature, i.e. they will only appear once in the product map, independently of the number of applications. The functions are:
 - *Proximity.* Represents the proximity of the characteristic in the products in relation with the standard, named $P(c)$ in ToolDAY.
 - *Distance.* Represents the distance of the characteristic in the products in relation with the standard, is called $D(c)$.

There is no restriction over the values each of these functions can be given, these values can range as much as the domain analyst wants.

In addition, the tool proposes possible scopes to the feature according to the informed values of the evaluation functions and the limit values defined (they can be: *maximum*, defining from which value the feature should be considered mandatory, and *minimum*, that characterize until which value the feature should not belong to the domain). This information is represented in the *scope column* in the map and can range from:

- **Mandatory.** The feature will have to be present in every product.
- **Variable.** It includes *optional*, *alternative* and *or* types - and they may or not be present in the products. These types are further described in the next section.
- **Out of Scope.** It means that the feature should not be a part of the domain.

In spite of the fact that ToolDAY proposes the features' scope, it does not obligate the user to accept it, since he/she can change the proposed scope as needed. Sometimes this change is necessary, because even with the feature not existing in any products (making ToolDAY to propose it as *out of the scope*), it can be considered, by the user, a key feature for the products to come in this domain.

With all the features scope defined, the domain planning is completed.

Domain Modeling

After the planning completed, the next step for the user to fulfill is the domain modeling, which is part of the second step depicted in Figure 4.4, the *model and document the domain and features*. As shown in the Figure, it is possible for the user to start the domain analysis process with ToolDAY from the domain modeling. This does not mean that the first phase (*planning*) should not be done, but it can be performed in a different tool or environment than ToolDAY with the achieve results being modeled in it.

The domain representation is done with the **feature model** (Kang *et al.*, 1990), which is a hierarchical way of organizing commonalities and variabilities into levels of increasing detail, i.e. the root feature is the most abstract feature in the model. In ToolDay, the feature models are modeled with the feature as diagrams and their types as relationships. There are four possible types for the features, they are derived from the scope options detailed in the previous phase - *domain planning*:

- **Mandatory.** The feature will always be in the present in the product.
- **Optional.** The feature may or not be present in the product.
- **Alternative.** From a set of features, one and only one of them will be present in the product.
- **Or.** From a set of features, at least one feature of them will be present in the product.

Figure 4.5 shows the representation of the possible feature's type within ToolDay. The *GroupId* tags in the *alternative* and *or* relationships identifies the features for a particular group. This notation was extended from the Feature Oriented Domain Analysis (FODA) notation (Kang *et al.*, 1990).

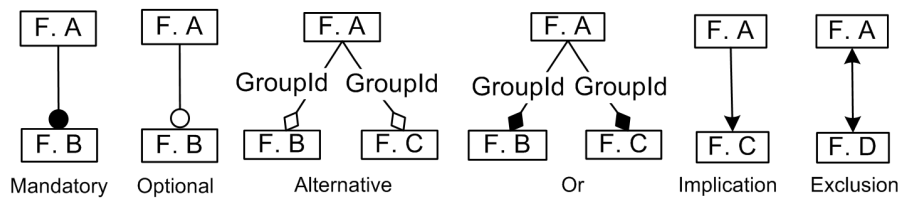


Figure 4.5 Features types' representation

Additionally, ToolDay also supports the inclusion of dependencies between the features. These dependencies are represented in Figure 4.5, and they can be:

- **Implication.** Obligates the inclusion of destination feature in the product, whenever the first is selected.
- **Exclusion.** Means that it is not possible to have both of the features in the relationship in the same product

Also regarding the relationship between the features, they can be represented in the diagram with three different types, besides the default relationship, which is not classified. The other types are:

- **Composition.** It is used if there is a whole-part relationship between a feature and its sub-features.
- **Generalization/Specialization.** In cases where features are generalization of sub-features.
- **Implementation.** It is used when a feature is necessary to implement the other feature.

Besides the relationship's classification, ToolDay also permits the identification of which group the feature belongs to. These groups provide different concerns, depending on the interest one might have in system development (Lee *et al.*, 2000). This classification followed (Lee *et al.*, 2000) definitions:

- **Capability.** A capability feature characterizes a distinct service, operation or functionality a product may have. This is usually the user's type.
- **Operating Environment.** An operating environment feature represents an attribute of the environment in which the product is used. This is usually the system analyst's and designers' type.
- **Domain Technology.** A domain technology feature corresponds to a domain specific implementation details. This is also the system analyst's and designers' type.
- **Implementation Technique.** An implementation technique feature embodies implementation details that can be reuse cross domains. This is usually the developers' type.

Whenever a feature and/or relationship is given a classification differently from the default, its visualization changes. For the features, theirs diagram border lines switches colors, while the relationships change from a solid line to a dot and/or dash one.

Furthermore, there are variabilities that have a large quantity of possible variations, however the representation of all of these variations in the feature model decrease the understandability of the model. Thus, ToolDay's solution to this problem is to permit the

inclusion of attributes in each feature (Czarnecki and Eisenecker, 2000), this allows a more concise representation of the feature model, and the attributes can have a description field that can be used to define their value range.

In order to facilitate the model creation, ToolDay permits to import the features defined in the product map to the model. There are two import types:

- **Import Feature.** Imports only the features classified as *mandatory* and *variable*.
- **Import Feature with Relationship.** Imports the feature from the product map as a child feature of the one selected from the model with the type of relationship specified.

Once the domain model has been finalized, it is necessary to validate its consistency and to document its artifacts.

Domain Validation

The domain validation involves the documentation and consistency of all artifacts generated until now in the domain - that refers to part of the *model and document the domain and features* and to the *domain feature model consistency* steps in Figure 4.4. This phase is not mandatory, but it is one of the advantages for providing an easier understanding of the domain by new members of the team and for increasing the documentation, which is one of the reasons companies adopt CASE tools, as described in chapter 2.

ToolDay provides a large set of documentation for the domain and features. The domain documentation is presented to the users as a form editor, as shown in Figure 4.6, and it consists of the following information (Almeida *et al.*, 2006):

- **Description.** Defines the domain responsibilities.
- **Defining rules.** Includes decision criteria about the domain limits.
- **Systems examples.** Description of which systems are similar to this domain.
- **Available documentation.** Existing documentation from the example systems.
- **Domain context.** Defines the relations among this and other domains.
- **Genealogy.** Identify information about the evolution and dependencies among systems within a domain.
- **Release notes.** Describes the difference of current version from previous one.

The feature documentation is available to the user in the same environment of the domain modeling. The documentation consist of the following fields (Czarnecki and Eisenecker, 2000):

- **Description.** Describes the feature semantics.
- **Rationale.** Explains why the feature is included in the model.
- **Constraints.** Represents hard dependencies between variable features.
- **Attributes.** Marks the variation point as open if new direct variable subfeatures are expected. On the other hand, marking a variation point as closed indicates that no other direct variable subfeature is expected.
- **Priority.** Describes their relevance in the domain.

Domain

| | |
|--|--|
| General Information This section describes general information about this domain Name: <input type="text" value="example"/> Version: <input type="text"/> Domain Examples Information about examples of the domain applicability and its evolution Systems Examples: <input type="text"/> Available Documentation: <input type="text"/> Genealogy: <input type="text"/> | Addition Information Addition information for a better understanding of the domain Description: <input type="text"/> Domain Context: <input type="text"/> Defining Rules: <input type="text"/> Release Notes: <input type="text"/> |
|--|--|

Figure 4.6 Domain documentation form

Another part of the domain documentation, is the description of the requirements and use cases. Differently from the features inclusion, that is expected for every new domain analysis performed with ToolDay, the requirements and use cases are considered

optional artifacts in project execution. It is even possible to have just one of them, or the requirements or the use cases.

The requirements have the following fields:

- **Type.** It can be *functional* or *non-functional*.
- **Priority.** The priority the requirement has in the domain.
- **Description.** A short description about the requirement's goal.

And the use cases have:

- **Pre Condition.** It is the state in which the system must be in order to carry out the use case.
- **Post Condition.** It is a list of possible states in which the system could be immediately after carrying out the use case.
- **Main Flow.** It describes the main events flow that occurs during the execution of the use case.
- **Alternative Flow.** It describes the events of an alternative flow that can occur during the main flow.
- **Exception Flow.** It describes the events of an exception flow that can occur during the main flow.

After informing the requirements, uses cases and features, it is possible to map how the traceability among these artifacts is, which facilitates the identification of the impacts in modifying one of these artifacts. Due to the optionality of the requirements and use cases in ToolDay, it is possible to relate both, the requirements and the use cases, to a feature, and vice-versa. Therefore, the relationship cardinality adopted by the tool among these artifacts is $n \times m$, because it causes less impact on companies' methodologies.

This traceability is done through a graphical editor, in which the artifacts already created in different editors are imported, and later the relationships among them are created. Moreover, ToolDay also supports the inclusion of a dictionary for the domain. Each word in this dictionary is composed of a word and its meaning.

There is no advantage in proving a large set of documentation for each artifact in the tool, if these documentations were only available within the tool environment. Thus, in

order to provide the visualization of the information in other environment not related to ToolDay, it permits the creation of several reports.

For the written documentation, such as the domain, features, requirements, use cases, glossary and products (that will be described in the next Subsection), ToolDay generates a PDF report. For the every model generated, it is possible to export it as a JPEG, GIF and/or PNG figure, in addition, the traceability information can also be exported as an excel sheet.

This phase also includes the consistency verification of the information inserted in the previous phases. This corresponds to the first decision step - *domain feature model consistency* - in Figure 4.4.

ToolDay supports several consistency rules within the consistency check requirement. The feature model rules are divided in three types (Massen and Lichter, 2004):

- **Redundancy.** It happens when the same semantic information is represented in more than one way. It can affect the model maintainability; in contrast, it can increase its readability. Because of that, it is treated as an informative alert. They are:
 - An implication relationship having as end a *mandatory* feature, independent of which feature type is the source.
 - An implication relationship between features that already have a relationship.
 - An exclusion relationship between features of the same *alternative* group.
- **Anomalies.** It occurs when features derivation are being lost, thus the domain cannot be completely configurable as it should be. Because of that, they are represented as warning sign. They are:
 - An *option* feature implied or excluded by a *mandatory*.
 - A *mandatory* feature implying in or excluding an *alternative* feature.
 - An *or* feature excluded by a *mandatory* feature.
 - An *alternative* feature implied by a *mandatory* feature.
- **Inconsistency.** Existence of information that contradicts with some other information already in the model puts it inconsistent. That way, is almost impossible to derive a consistence product, therefore it is represented as errors. The inconsistencies are:

- Features in the model not connected to any other feature, called as *orphans*.
- An exclusion relationship between *mandatory* features.
- An implication between *alternatives* features.
- Exclusion relationships between features that already have a relationship.
- *Alternative* or *or* features with only one feature in the group.

Whenever a problem is found during the validation, ToolDAY generates a report (presented through Eclipse's *problem view*) with what was found. For some of these problems, the tool affords possible automatic solutions to them through Eclipse's quick fix, which is a popup window that describes possible solutions and automatically executes the selected one. This was included as a way to facilitate the problem identification, and its solution. Furthermore, ToolDAY also considers the lack of documentation of the artifacts as a warning. Guaranteed that the domain is consistent and documented, the steps regarding the domain analysis are completed.

Product Derivation

The product derivation phase includes the steps from *select the product features* to *model and document product* in Figure 4.4. For the selection of the product features, ToolDAY provides the tree view of the domain modeled with checkboxes beside the features names. This way, the user visualizes the complete domain hierarchy, facilitating the product selection. It is also necessary to validate if the product selection is correct according to the consistency rules. Besides the composition rules created by the user, the product consistency rules also include:

- Selection of more than one *alternative* feature in a group.
- Not selecting at least one *or* feature in a group.
- Not selecting a *mandatory* feature in product derivation.
- Not selecting implied features.
- Selection of mutual exclusion features.

All of these rules are considered inconsistencies, thus they are represented as errors.

Once the product selection is without errors problems, it is possible to generate its feature model, which is similar to the domain feature model. However the main

difference between the domain and the product feature models is that the latter only has the mandatory relationship type, since one goal of product derivation is to resolve domain's variation points.

In the product feature model, the user can include new features, which are usually the features that were considered as *out of scope* in the product map. Furthermore, all the features documentation included and their attributes in the domain feature model are kept in the product model, however, the user can change it as he/she wants. In case new features are added, their lack of documentation validation will follow the same restrictions defined for the domain feature model.

Moreover, each new product has a similar form to the domain to fulfill with its documentation, however, the product documentation consists only of:

- **Domain Version.** Identifies the domain version the product is based on. It is necessary because the domain may evolve while the product does not.
- **Description.** Defines in which part of the domain the product is focused.

After the fulfillment of all these steps, the user has a fully documented and consistent domain and products.

4.4.2 Usability Requirements

Regarding the usability requirements defined in Table 4.1, ToolDay provides an interactive tutorial explaining the first steps to be executed, their order and what their goals are. The *tutorial* was developed with Eclipse's help technology of *cheat sheets*.

ToolDay also provides two example solutions demonstrating a full domain analysis with it. The examples are for an atari game and for basic automobile domains, and the step-by-step performed for the latter is detailed in the next section.

The spell checker was implemented due to the large quantity of documentation available in the tool, and the dictionary used as base to correct the misspelled words can be modified to correspond to the user's language. The misspelled words are marked as red and if the user clicks above it, a menu appears with the possible corrections for them or the option to include it into the dictionary, as depicted in Figure 4.7.

The *import/export to/from other applications* requirement is fulfilled through the reports presented in the *domain validation phase*.

As well, ToolDay has a *metric component* that provides simple quantity metrics about the artifacts being populated in a domain analysis project.

Some of the metrics per artifact are:

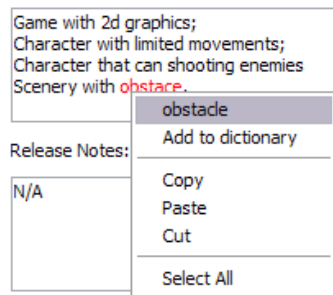


Figure 4.7 ToolDay's spell check

- *Domain*. Number of requirements and use cases;
- *Glossary*. Number of words;
- *Product Map*. Number of products, evaluation functions (divided in characterization and benefits) and features (divided in mandatory, variable, out of scope and undefined);
- *Domain Feature Model*. Number of features - divided in mandatory and variable, which is further divided in *alternative*, *or* and *optional* - and also the number of implied and excluded features. These metrics are shown in Figure 4.8; and
- *Product Feature Model*. Number of features.

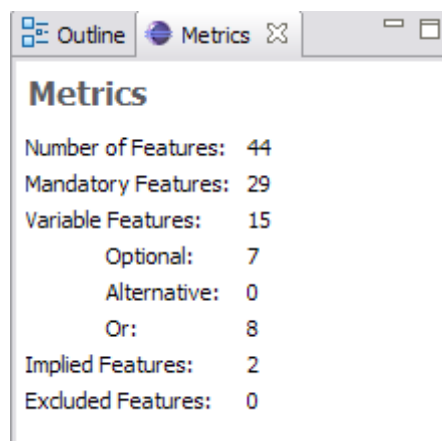


Figure 4.8 Domain feature model metrics

Apart from these requirements, ToolDay was developed to be less intrusive as possible to the company. Because of that, it permits several customizations within its

preference window depicted in Figure 4.9, which is accessed through the menu *Windows -> Preferences... -> ToolDay Preferences*.

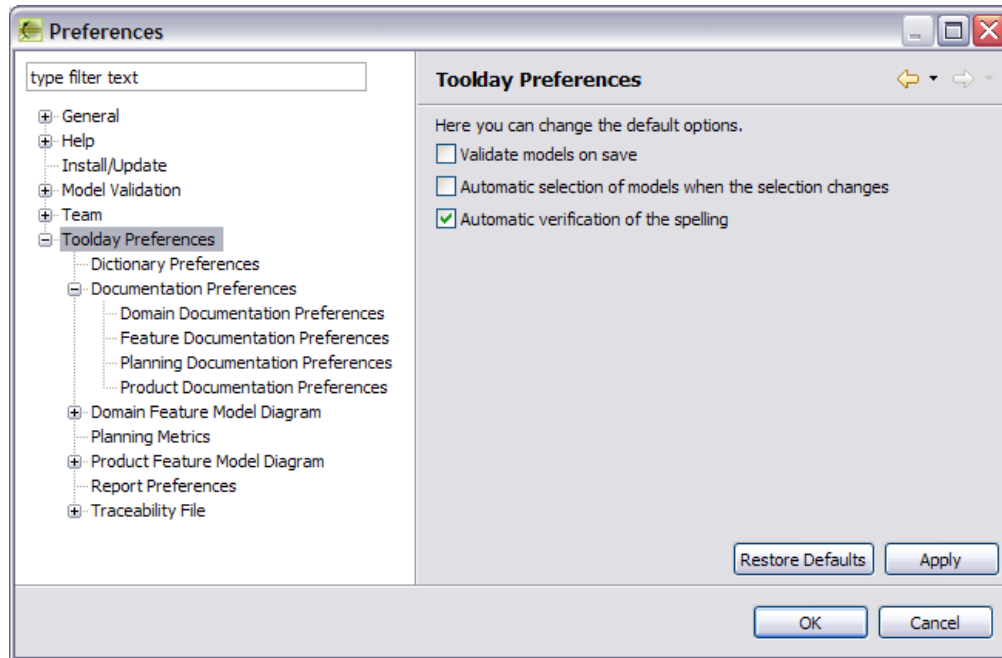


Figure 4.9 ToolDay’s preferences

The customizations are:

- **Dictionary Preferences.** The user can switch the dictionary file to be used during the spell check. Therefore, it is possible to change its language (the default dictionary is English) or to change it to a dictionary with more specific terms of the domain.
- **Automatic Selection of Features.** Automatic selection of implied and mandatory features in the product selection.
- **Automatic Validation.** Automatic validates the editors according to the consistency rules whenever they are saved.
- **Automatic Spell Verification.** Automatic verifies the text for misspelled words while the user is inserting the documentation.
- **Documentation Fields.** Permits the selection of which documentation fields should be verified for the artifacts during the validation.

- **Report Preference.** Allows the user to inform an image (such as the company's logo) and a text (company's name) that will be included in every report created by ToolDay.
- **Evaluation Functions.** Permits the inclusion and selection of which evaluation functions (characterization and benefit) will be included in the product map.

Furthermore, each model editor has its own configuration that includes printing options, appearance and so on.

4.5 ToolDay in Action

In order to facilitate the understanding of the proposed solution, this section simulates ToolDay's use in a **fixed shooter atari game domain**⁵. In this kind of game, the user has limited control over its character, which is usually a spaceship, and has the objective of killing all its enemies. This analysis occurred during a RiSE course⁶, in which this domain was the course project with its results detailed in (Lisboa *et al.*, 2008a).

Furthermore, this description does not focus on the detailing the domain information included during the process execution in the tool, instead it explains **how** and **what** the user has to do in order to complete the domain analysis process using ToolDay.

The initial screen presented to the user when the tool is opened for the first time is depicted in Figure 4.3, without any editor open. The tutorial, on the right side (d), presents the basic steps to be executed within ToolDay for completing the domain analysis, and it also provides a shortcut to create all files mentioned here.

The first action the user has to perform is to create a new simple project for the domain analysis. In the example the new project's name is **Atari Game**. Afterwards, the user starts the domain planning by defining the domain scope.

For that, he/she selected three applications to work as legacy information in order to map what should be the domain reusable artifacts. The selected applications were: demon attack⁷, phoenix⁸ and space invaders⁹. Latter, the evaluation functions to be used should be informed. For the fixed shooter domain the default evaluation functions (`Req` and `Sim` for the characterization functions and `D (c)` and `P (c)` for the benefit functions)

⁵http://en.wikipedia.org/wiki/Fixed_shooter#Fixed_shooter

⁶<http://www.rise.com.br/research/courses/2006/reuse>

⁷http://en.wikipedia.org/wiki/Demon_Attack

⁸[http://en.wikipedia.org/wiki/Phoenix_\(arcade_game\)](http://en.wikipedia.org/wiki/Phoenix_(arcade_game))

⁹http://en.wikipedia.org/wiki/Space_Inviders

were maintained by the user, along with the maximum and minimum limit values for scope definition, i.e. 70% and 20%, respectively. In the last, but not least, part of the domain planning, the user includes the possible features and their evaluation functions values in the product map. Figure 4.10 depicts the product map fulfilled with the possible features, their characterization and benefit functions - which were kept the default - along with their values for each feature and application and their scope defined for the example domain.

| Features | Demon Attack | | Phoenix | | Space Invader | | Total | | Scope | |
|---------------------|--------------|-----|---------|-----|---------------|-----|-------|------|-----------|---|
| | Req | Sim | Req | Sim | Req | Sim | D(c) | P(c) | | |
| Accerlation | 1 | 0.5 | 1 | 0.5 | 1 | 0.5 | 0.50 | 0.50 | Variable | ▼ |
| Bonus | 1 | 0.5 | 1 | 0.5 | 1 | 0.5 | 0.50 | 0.50 | Variable | ▼ |
| Continue | 0 | 0 | 0 | 0 | 0 | 0 | 0.00 | 0.00 | Variable | ▼ |
| Create | 1 | 1 | 1 | 1 | 1 | 1 | 0.00 | 1.00 | Mandatory | ▼ |
| Destroy | 1 | 1 | 1 | 1 | 1 | 1 | 0.00 | 1.00 | Mandatory | ▼ |
| Divide | 1 | 1 | 0 | 0 | 0 | 0 | 0.00 | 0.33 | Variable | ▼ |
| Enemy | 1 | 1 | 1 | 1 | 1 | 1 | 0.00 | 1.00 | Mandatory | ▼ |
| Game Initialization | 1 | 1 | 1 | 1 | 1 | 1 | 0.00 | 1.00 | Mandatory | ▼ |
| Game over | 1 | 1 | 1 | 1 | 1 | 1 | 0.00 | 1.00 | Mandatory | ▼ |
| Graphical Element | 1 | 0.5 | 1 | 0.5 | 1 | 0.5 | 0.50 | 0.50 | Variable | ▼ |
| High Score | 0 | 0 | 0 | 0 | 0 | 0 | 0.00 | 0.00 | Variable | ▼ |
| Horizontal | 1 | 1 | 1 | 1 | 1 | 1 | 0.00 | 1.00 | Mandatory | ▼ |
| HI ID | 1 | 0.5 | 1 | 0.5 | 1 | 0.5 | 0.50 | 0.50 | Variable | ▼ |

Figure 4.10 Atari game product map

Regarding the scope values proposed by ToolDay (see Figure 4.10), the features `Continue` and `High Score` should not be part of the domain. However, the user altered their scope to `variable`, because the analyzed games are old (from the 1980s) and these features were not usually implemented at that time. For this domain, the user considered that all the features proposed as `out of scope` (there was still background music) should be included as `variable` in the domain. This decision was taken as a way to include new features in the old domain game.

With the scope defined, the user can start the domain modeling. ToolDay permits the synchronization between the product and the modeling editors in two ways: import all the features - except the ones defined as *out of scope* - and import a feature with a relationship. Figure 4.11 shows the window to import features with a relationship from the product map. In this example `Vertical` feature (a) is being imported as child of `Direction` feature (b) with the `Or` relationship and group id `Direction` (c).

After the user imports all the features from the product map, he/she begins to edit the domain model. This can be achieved through the palette available in the model editor (on

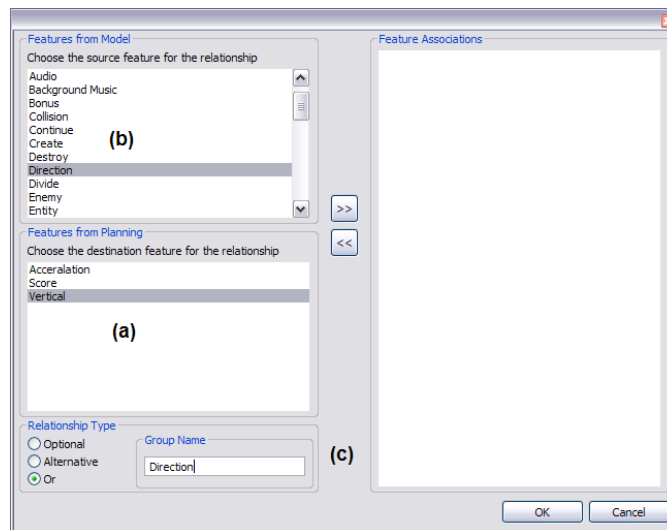


Figure 4.11 Import feature with relationship

the right side in Figure 4.12), through it, the user can insert the features, their attribute, relationships and constraints.

Figure 4.12 shows part of the fixed shooter feature model with the implication between the `Obstacle Collision` feature and the `Obstacle` one (a). Furthermore, the user included an attribute, named `How many shoots?` on the `Destroy` feature (b) to represent the number of shoots necessary to destroy the enemies, which can vary from game to game, and even inside the same game. The feature documentation (c) was set in the same environment as a way to facilitate for the user to document it.

With the domain model finalized, the user does the model consistency verification through the `Consistency` button on the toolbar (4.13 (a)) or through the mouse right button over the file's name (4.13 (b)). If the domain feature model is modeled incorrectly - based on the consistency rules defined in the previous section - ToolDay shows for the user the collection of inconsistencies classified by their severity (*errors*, *warnings* or *information*), as depicted in Figure 4.14.

Furthermore, in the feature model, the problem features are marked with their highest severity, and the tool tip details the problem, as depict in Figure 4.15. This reduces the user's effort to find the inconsistent features.

For some problems encountered in the model, ToolDay provides a few automatic solutions to it, as described before in section 4.4.1. For the `Orphan Feature` error, ToolDay offers two possible solutions, depicted in Figure 4.16:

- **FIX #1.** Connect this feature to another one through a hierarchy relationship

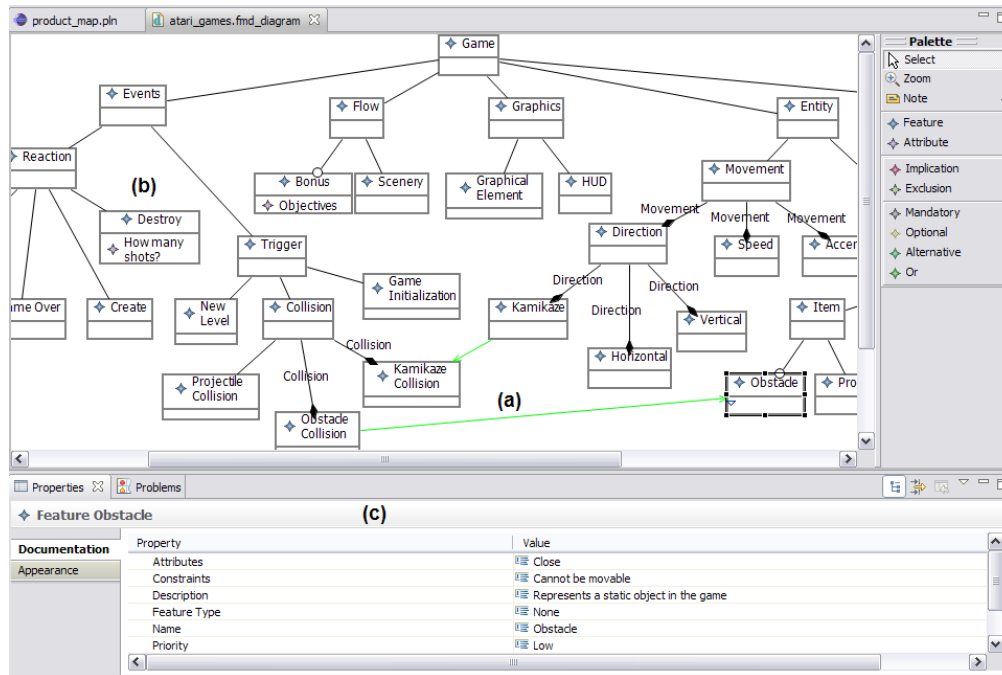


Figure 4.12 Atari game feature model

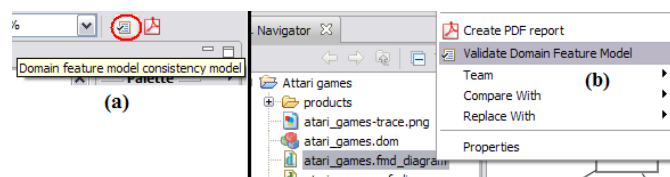


Figure 4.13 ToolDay's consistency checker

| Properties Problems | | | |
|--|-------------------------|-------------|----------------------------|
| 1 error, 3 warnings, 1 info | | | |
| Description | Resource | Path | Location |
| Errors (1 item) | | | |
| Feature High Score is orphan | atari_games.fmd_diagram | Atari games | Feature: High Score |
| Warnings (3 items) | | | |
| The field Description in Feature Vertical is not documented | atari_games.fmd_diagram | Atari games | Feature: Vertical |
| The field Rationale in Feature Vertical is not documented | atari_games.fmd_diagram | Atari games | Feature: Vertical |
| There is more than one root feature, there should be only one | atari_games.fmd_diagram | Atari games | Feature: Game |
| Infos (1 item) | | | |
| The implication between Graphical Element and Collision is unnecessary, since the destination is already a mandatory feature | atari_games.fmd_diagram | Atari games | Feature: Graphical Element |

Figure 4.14 Inconsistency report

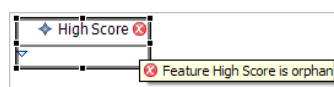


Figure 4.15 Feature's severity marker and tool tip

(Mandatory, Optional, Alternative or Or).

- **FIX #2.** Remove this feature from the model.

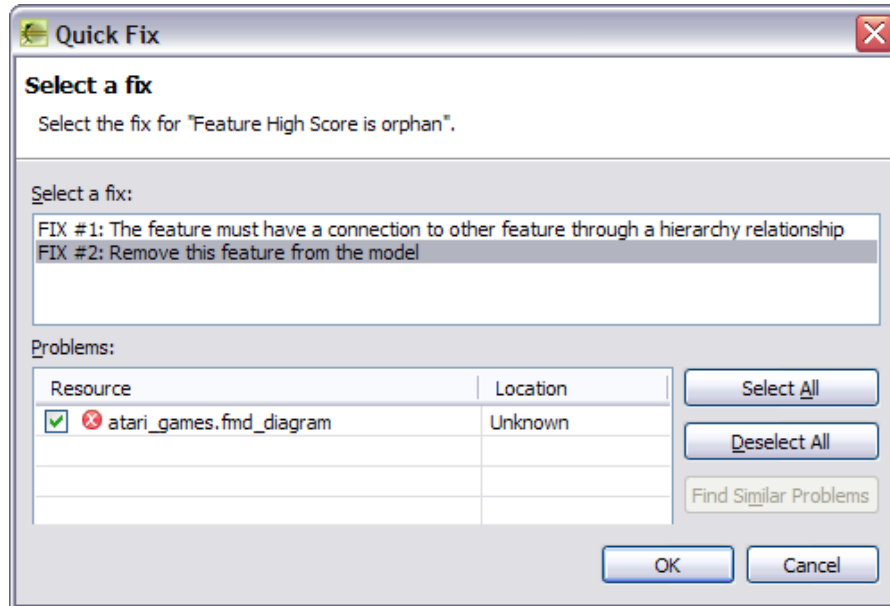


Figure 4.16 ToolDay's quick fix window

After the inclusion of the requirements and use cases, during the domain documentation, the user generated the traceability model, as shown in Figure 4.17. This model may have different abstractions levels, this happens because the user can choose which features should be in the traceability model and the feature model reduces the abstraction as the level increases (Kang *et al.*, 1990).

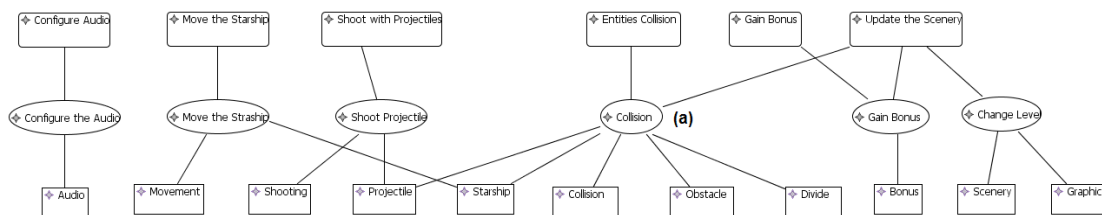


Figure 4.17 Atari's domain traceability model

In Figure 4.17, the first line (rounded rectangle) represents the domain requirements, the second line (elipses) are the domain use cases, while the last line (rectangle) are the domain features.

In this example, the user selected a *parent feature*, Collision (a) in Figure 4.17, instead of its “leaves features” (which would be Projectile, Kamikaze and Obstacle Collisions - as shown in Figure 4.12). Thus, the traceability model is more abstract but its understandability is also better, since the number of elements in the model is reduced.

For the product selection, the user visualizes the tree view of the complete domain. If the automatic selection of features customization is selected, the user can check the domain root, and ToolDay automatically selects all the mandatories. After that, the user has to select which features of the alternative and or groups the product should have. Figure 4.18 illustrates the features the user selected for the demon attack game.

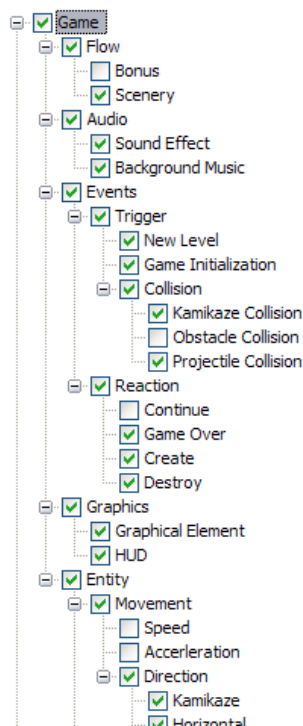


Figure 4.18 ToolDay’s product selection

Guaranteed that the product selection is correct through the product consistency checker, the user initializes the product’s model with the Initialize/Update the product model button in the toolbar, as shown in Figure 4.19.

The generated product model (Figure 4.20) keeps all the documentation from the features and attributes added in the domain model. Moreover, the user can include new features and/or attributes, but in this case it was not necessary because all of the features

that could be considered as out of scope by ToolDay, were included in the domain as variable.

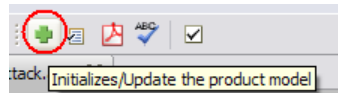


Figure 4.19 Initialize/Update the product model

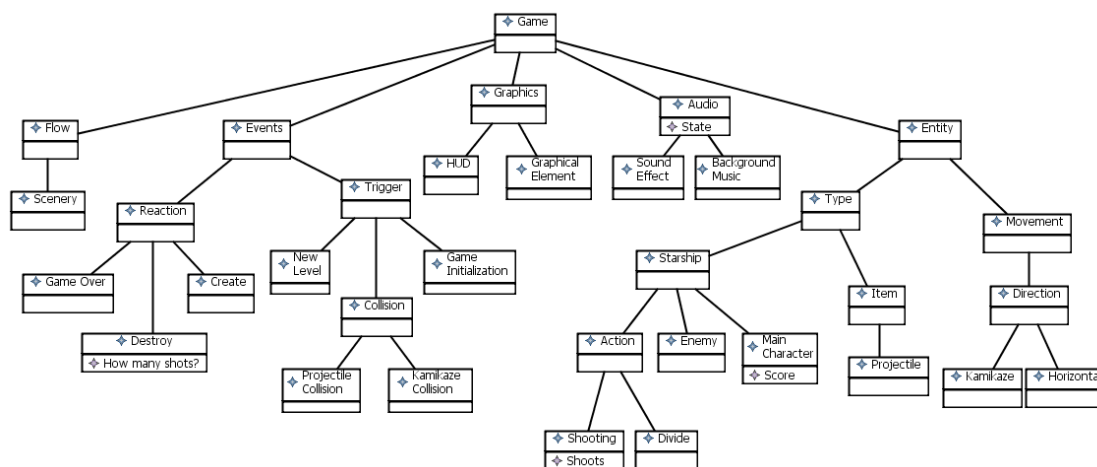


Figure 4.20 Demon attack product model

The user repeats the products steps for as much product he/she wants to create for this domain. After that, the domain analysis and product derivation with ToolDay is complete.

This section presented a real scenario for the domain analysis process, demonstrating the advantages of using the tool.

4.6 Chapter Summary

This chapter presented the main aspects of ToolDay. The requirements, architecture and the set of technologies employed during its construction were discussed. Furthermore, it presented how a step-by-step process can be applied within the tool. Next chapter presents the evaluations performed to verify the tool's helpfulness to the process.

“Com as estatísticas pode-se chegar a qualquer conclusão...”

Silvio Ribeiro de Castro (Writer)

5

ToolDay Evaluations

Once ToolDay was described and its initial implementation detailed, two experiments and a case study were performed in order to evaluate the benefits of the proposed solution. The experiments aim at validating the tool acceptance during the process execution and the intuitiveness for ToolDay’s usage, while the case study describes the ToolDay usage in an industrial project for a social network.

This chapter is organized as follows: section 5.1 describes the steps executed for each experiment, such as their definition, design, instrumentation, threats, operation and analysis, besides the projects performed for the tool evaluation. Latter, the industrial case executed with ToolDay is described in section 5.2. Finally, section 5.3 draws the chapter summary.

5.1 ToolDay Experiments

Both of the experiments were conducted with the goal of analyzing if ToolDay aids the domain analyst in the project execution and if it provides intuitive and easy to use environment in order **not** to inhibit its usage.

The secondary goal of the experiment - intuitiveness for ToolDay’s usage - was included because, as stated by Fuggetta (1993) and described in chapter 2, CASE tools need “*sophisticated user-interface-management systems to design and develop graphical, easy-to-learn user interfaces*”, however the focus of the experiments are on project execution aid.

5.1.1 Experiment Process

Experiments should be used when a situation has to be controlled over and has its behavior manipulated directly, precisely and systematically (Wohlin *et al.*, 2000). Therefore, in order to experiment it is necessary to define some variables. Wohlin *et al.* (2000) defines two types of variables: *independent* and *dependent*.

Variables are the objects of the study. The *dependent*, are the ones studied to see the effect of the changes in independent variables, while all the variables in a process that are manipulated and controlled are called *independent* variables.

An experiment studies the effect of changing one or more independent variables. Those variables are called *factors*. The other independent variables are controlled at a fixed level during the experiment, or else it would not be possible to determine if the factor or another variable causes the effect. A *treatment* is one particular value of a factor.

The treatments are being applied to the combination of *objects* and *subjects*. An object can, for example, be a document that will be reviewed with different inspection techniques. The people that apply the treatment are called subjects. Both the objects and the subjects can be independent variables in the experiment. Furthermore, an experiment consists of a set of *tests* where each test is a combination of treatment, subject and object.

In order to perform an experiment, several steps have to be done and in a certain order. The experiments in the chapter followed the process defined by Wohlin *et al.* (2000), which can be divided into the main activities: **Definition**, where the experiment is defined in terms of problem, objective and goals. **Planning**, where the design of the experiment is determined, the instrumentation is considered and the threats to the experiment are evaluated. **Operation**, in which measurements are collected, analyzed and evaluated in the **analysis and interpretation**. Finally, the results are presented and packaged in the **presentation and package**. They are depicted in Figure 5.1.

Even though the process' steps should be executed in a certain order, it does not mean that the process should be a waterfall model (Wohlin *et al.*, 2000). The order, in Figure 5.1, primarily indicates the starting order of the activities, but it may be necessary to go back and refine a previous activity - except when the operation has already started (Wohlin *et al.*, 2000). Therefore, the process can be considered iterative.

Next section presents the definition step for both of the experiments.

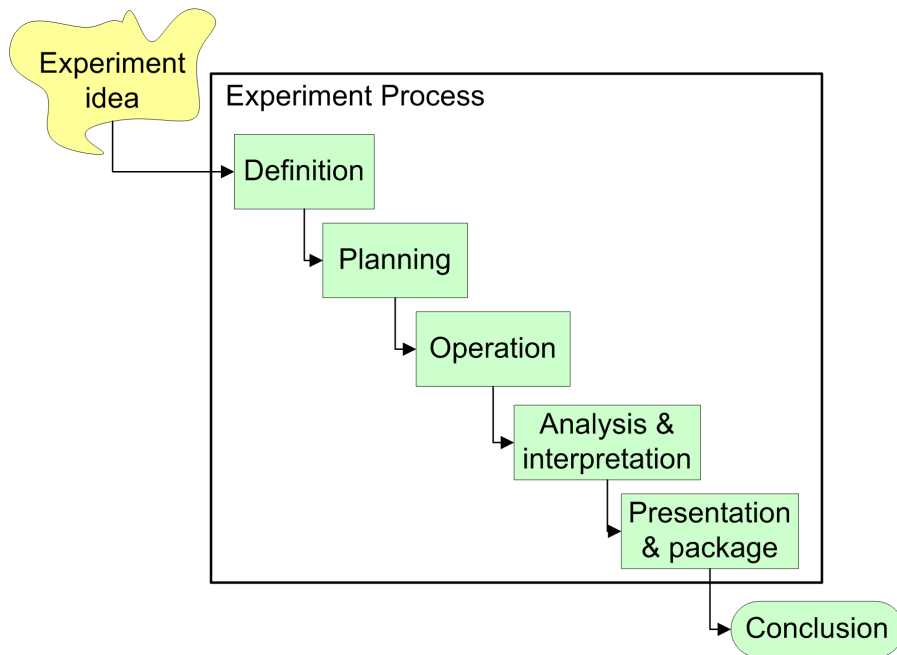


Figure 5.1 Overview of the experiment process (Wohlin *et al.*, 2000)

5.1.2 Definition

In the definition step, the objectives and goals have to be defined. To achieve it, Wohlin *et al.* (2000) follow the GQM (Goal-Question-Metric) template (Basili *et al.*, 1994). The GQM is based upon the assumption that for an organization to measure in a purposeful way, it must first specify the goals for itself and its projects, then it must trace those goals to the data that are intended to define those goals operationally, and finally provides a framework for interpreting the data with respect to the stated goals.

Goal. The goal of both experiments is to analyze the *domain analysis tool* for evaluating it with respect to *its aid in the process execution* and *its intuitiveness and easy to use environment* from the point of view of *domain analysts* in the context of *domain analysis project*.

Questions. To achieve this goal, several questions were defined:

Q_1 . Do the subjects consider that ToolDay aids in the process execution?

Q_2 . Do the subjects have difficulties to execute the process with the tool?

- Q*₃. Do the subjects consider the consistency checker messages helpful in the inconsistency identification and solution?
- Q*₄. Do the subjects consider the tool's tutorial sufficient for learning how to use it for a domain analysis process?
- Q*₅. Do the subjects believe that ToolDay customizations are useful?

Metrics. Even though several metrics were defined for the questions, none of them were never used before. Therefore there are no well-known values for them, and also the literature does not present useful techniques or metrics to measure it. The metrics are presented as follow:

- *Helpful in the Process Execution.* % of subjects that agreed the tool aids the process execution.
- *Difficulty to Execute the Process.* % of subjects that had difficulty during the process execution with the tool.
- *Consistency Checker Messages.* % of subjects that agreed the consistency checker messages aid the inconsistency identification and solution.
- *Sufficient Tutorial.* % of subjects that considered the tutorial adequate to learn the tool's basic concepts to perform the domain analysis.
- *Customization Usefulness.* % of subjects that believed the customizations are useful for a less intrusive adoption of the tool.

Both of the experiments had the same goals, question and metrics. Later on, each experiment step is described.

5.1.3 First Experiment

The first experiment was executed from November, 2007 to January, 2008. At that time, ToolDay implementation was not completed, therefore the version used during the experiment is different than the description presented in chapter 4. Before presenting the experimentation steps, the differences between the evaluated and the described version in chapter 4 are detailed.

ToolDay's Evaluated Version

This section details how the functionalities were implemented in the tool version used during the experiment. Great part of these functionalities were modified based on the feedback given by the experiment's subjects.

- **Consistency Checker.** In the evaluated version, the consistency report did not inform the exactly place where the problem occurred, it only showed the editor file. Furthermore, the features in the feature model were not marked with the highest problem they had (as depicted in Figure 4.15 in chapter 4).
- **Requirements and Use Cases.** The requirements and use cases in the evaluated version had only a name and description field.
- **Traceability.** The requirements, use cases and features traceability was restricted to the traceability model - with no excel report.
- **Zoom.** The editor model did not supported the *ctrl + mouse wheel* shortcut to modify the zoom.
- **Product map.** It was not possible to import the features included in the feature model to the product map, increasing the complexity to keep these editors synchronized.

The other functionalities were the same as described in the previous chapter.

Planning

The experiment was conducted in a graduate course¹ at the university lab, hence the experiment ran off-line (not industrial software development) and it was conducted by the students that performed a domain engineering project based on a real-world case. Although the students performed a complete domain engineering process, ToolDay was only used during the domain analysis step. Thus, this evaluation only considers the results of this step.

The instrument used to validate the experiment was a feedback form composed of questions about the subjects' professional experience, their knowledge regarding domain analysis processes, questions concerning to the tool and the subjects' opinions about its usage. The complete feedback form can be seen in Appendix B.

¹<http://www.rise.com.br/research/courses/2007/reuse/>

Through the questionnaires answers, it was possible to achieve an evaluation about the tool's helpfulness and usability based on qualitative data. This study evaluation was based on the following hypotheses:

Null Hypothesis. It is the hypothesis that the experimenter wants to reject with as high significance as possible. As stated before, there is no well-known value for these metrics, therefore, arbitrary values were chosen, based on practical experience and common sense. The hypotheses and the values defined were:

H_{0a} : μ helpfulness in the process $< 70\%$.

H_{0b} : μ process difficulty $\geq 40\%$.

H_{0c} : μ consistency checker messages $< 80\%$.

H_{0d} : μ sufficient tutorial $< 80\%$.

H_{0e} : μ customization usefulness $< 70\%$.

Alternative Hypothesis. This is the hypothesis in favor of which the null hypothesis is rejected. In this study, the alternative hypothesis determines that the use of the tool produces benefits that justify its use. They were:

H_{1a} : μ helpfulness in the process $\geq 70\%$.

H_{1b} : μ process difficulty $< 40\%$.

H_{1c} : μ consistency checker messages $\geq 80\%$.

H_{1d} : μ sufficient tutorial $\geq 80\%$.

H_{1e} : μ customization usefulness $\geq 70\%$.

The study was conducted as a *single object study*, in which studies are performed on a single team and a single project. The subjects were selected based on convenience - that are the nearest and more convenient persons - and they were the M.Sc. and Ph.D. students in Software Engineering from the Federal University of Pernambuco, Brazil.

Furthermore, the subjects did not have a training about the tool usage, it occurred as a way to validate if the tutorial was sufficient for the users to learn how use the tool. However, the subjects, in case of doubt, could ask questions about the tool functionalities.

To provide a set of valid results, Wohlin *et al.* (2000) defend the establishment of four types of threats to the validity of the experiment. They are:

- *Internal validity.* It is defined as the capacity of a new study to repeat the behavior of the current study, with the same subjects and objects with which it was executed. The internal validity of the study is dependent of the number of subjects.
- *External validity.* It measures capability of the study to be affected by the generalization, i.e., the capability to repeat the same study in other research groups.

- *Conclusion validity*. It is concerned with the relationship between the treatment and the outcome, and determines the capability of the study to generate conclusions.
- *Construct validity*. It refers to the relation between the theory that is to be proved and the instruments and subjects of the study.

Based on these definitions, the threats to validity identified for this experiment were: *Internal validity* depended upon the number of students in the course; it was assumed that the quantity and quality of the subjects chosen provided a good internal validity. A risk related to *external validity* was the subjects' motivation, since some subjects could perform the study without responsibility or without a real interest in performing the project with a good quality as it could happen in an industrial project. Conversely, since the results and artifacts achieved by the subjects with ToolDay's use, were necessary for the other phases (domain design and implementation), the external validity was considered sufficient.

The *conclusion validity*, no threats were found because the conclusions were drawn through descriptive statistic. In addition, the *construction validity* included three threats; (i) the first referred to the familiarity the subject had in Eclipse platform, and the others were (ii) the subjects' knowledge regarding domain analysis processes and (iii) their experience in domain analysis projects. However, the subjects had a training about the process used in the project before it started, and the other two threats were included in the questionnaire for correlating them with possible difficulty encountered by the subjects to use the tool, however the goal of the experiment is not to validate if the generated artifacts represent a real domain analysis, but to validate if the tool aided during the project execution.

The Experimental Study Project

The project used in the experimental study aimed at performing the domain engineering of the reuse tools domain using the RiDE - RiSE Process for Domain Engineering (Almeida, 2007). All the students were presented to this process in class, in which they could interrupt to ask issues related to the process understanding. The project domain consisted of four reuse tools in this domain which were presented to the students. The tools domain provides solutions to increase the organization's productivity through reuse according to its maturity level. The tools focus on search and retrieval of assets, components repositories, knowledge extraction and domain analysis.

Even though the execution of the complete domain engineering process involved

thirteen students, only six of them used the tool during the domain analysis step. Consequently, the subjects of the study are restricted to them.

The subjects had access to requirements, design specification, source code and prototypes for each of the tools. Furthermore, they could also consult experts about the tools.

Operation

The experimental study was conducted during part of a graduate course in Software Reuse, from November, 2007 to February, 2008, at Federal University of Pernambuco. The domain analysis step ran from November, 2007 to January, 2008. and it generated a feature model with 64 features, 14 requirements and 38 use cases.

As described before, the first two parts of the questionnaire applied to the subjects asked about their professional experiences and knowledge in domain analysis processes. Their answers are detailed below.

All subjects in this study have already participated in projects using the same development platform used by ToolDay, which is the Eclipse. This suggests that the learning curve to adapt to the tool's environment is smaller. However, one of them rated the knowledge in Eclipse as being *basic*, another rated it as *good*, while the others were *very good*. Furthermore, two subjects know four domain analysis processes, another knows three and the rest know just the one used during the project.

In addition, the subjects did not have much experience in domain analysis projects, since all of them have only participated in one to two academic projects and all of them participated in these projects as System Engineers, but some also had others roles. Table 5.1 shows a summary of the subjects' profile.

Analysis and Interpretation

This section describes the analysis and interpretation of the experimental study.

Helpfulness in the Process. After the analysis of the subjects answers regarding the tool helpfulness during the process execution, four subjects considered that the tool aided in the process execution, another one judged that the tool did not help a lot during the process execution, and the other subject did not see the gain in using the tool. This represents approximately 67% of approval, which **confirms** the null hypothesis (H_{0a}). However, it is necessary to highlight that this value for the null hypothesis (H_{0a}) was defined without any previous data, since it was the first time that this aspect was analyzed.

| ID | Projects with Eclipse | Eclipse's Knowledge | Domain Analysis Processes | Domain Analysis Projects | Projects's Roles |
|----|-----------------------|---------------------|---------------------------|--------------------------|------------------------------------|
| 1 | 3 Projects | Basic | 1 Process | 1-2 Academic Projects | System Engineer and Domain Analyst |
| 2 | 5 Projects | Good | 4 Processes | 1-2 Academic Projects | System Engineer and Domain Analyst |
| 3 | 7 Projects | Very Good | 4 Processes | 1-2 Academic Projects | System Engineer and Domain Analyst |
| 4 | 25 Projects | Very Good | 1 Process | 1-2 Academic Projects | System Engineer |
| 5 | 6 Projects | Very Good | 3 Processes | 1-2 Academic Projects | System Engineer and Architect |
| 6 | 3 Projects | Very Good | 1 Processes | 1-2 Academic Projects | System Engineer |

Table 5.1 Subjects' profile in the first experiment

The reasons given by the two subjects for not considering the tool helpful were, for one subject (ID 4) great part of the process can be done manually (without tool support) and the lack of integration with the next steps of domain engineering (design and implementation). The other (ID 5) related that the tool is not completely integrated with the domain engineering process used. However, this response may be influenced by the lack of experience in domain analysis projects, in addition the Subject 4 only knows one domain analysis process. Furthermore, the tool focuses is on the support of the domain analysis process, some steps of the product derivation, and on generating the artifacts that will be later used in the remaining domain engineering phases, it does not intend to integrate the complete process.

The remaining subjects were also asked to explain why they considered the tool adoption helpful. The reasons were, helps the execution of the process' steps (1 subject - ID 1), aided in the scope definition (1 subject - ID 2) and in the domain modeling (3 subjects - ID 2, 3, 5).

Regarding this question, the subjects were asked to describe the strengths and weakness of using the tool. Besides the aspects given as reasons for not considering the tool helpful, other weakness generally described by them were, traceability among requirements, use cases and features is too simple; lack of requirement management (by the time of the evaluation, the requirements and the use cases only had the description field); changes in the domain feature model are not reflected to the product map - included in the current version; the requirements cannot be classified in functional and non-functional nor have prioritization, a model with too many feature becomes too polluted. Furthermore,

aspects related to usability, like common shortcuts and design, such as the symbols used to represent the relationships and feature, were also commented. Conversely, several of these reasons were considered valid and are now implemented within the tool, as described in section 5.1.3.

The strengths described by the subjects, apart from reason described previously, were the consistency checker, generation of reports and domain model visual representation.

Process Difficulty. Through the answers provided by the subjects about the difficulties found during the process execution in the tool, only one subject (ID 1) did not have difficulty in any steps with the tool. The subjects were given three options to choose from: *hard navigation among the tool's steps*; *lack or insufficient explanation for using the tool*; and *lack of knowledge in the DA process*.

Regarding the others subjects: one subject (ID 3) selected the hard navigation among the tool steps (this subject had a very good familiarity with eclipse and also knew four domain analysis processes). The other (ID 5) selected the lack or insufficient explanation for using the tool and the hard navigation (this subject, like the previous subject, had a very good familiarity with eclipse and knew three domain analysis processes). Moreover, one subject (ID 6) selected the lack of knowledge in the domain analysis process and the lack or insufficient explanation for using the tool; this subject had a great knowledge in Eclipse, but knew only one domain analysis - the one used in this project. Therefore, this case was probably an aggravating for the project execution.

The other two subjects did not choose any of the options; one (ID 4) informed that the difficulty occurred because of the symbols used to represent the relationships - this subject had a very good familiarity with Eclipse. While the other (ID 2) informed the difficulty in the traceability between the domain feature model and the product map - this subject is acquainted with several domain analysis processes, and has a good knowledge in Eclipse. Figure 5.2 details the selected difficulties with the number of subjects.

Based on these results, this **confirms** the null hypothesis (H_{0b}), since approximately 83% of the subjects had difficulties in some step of the tool execution. However, in the same way as in the process helpfulness, this value for the null hypothesis (H_{0b}) was defined without any previous data. Even though the null hypothesis was confirmed with a high rate, some of these aspects were related to GUI, or to specific aspects of traceability, not to the main functionalities of the tool, which is the main goal of this experiment. Despite the fact that the subjects were asked to select the type of difficulty they had, it is not possible to analyze in which phase of the process this difficulty occurred. Thus, this situation indicates a failure in the questionnaire.

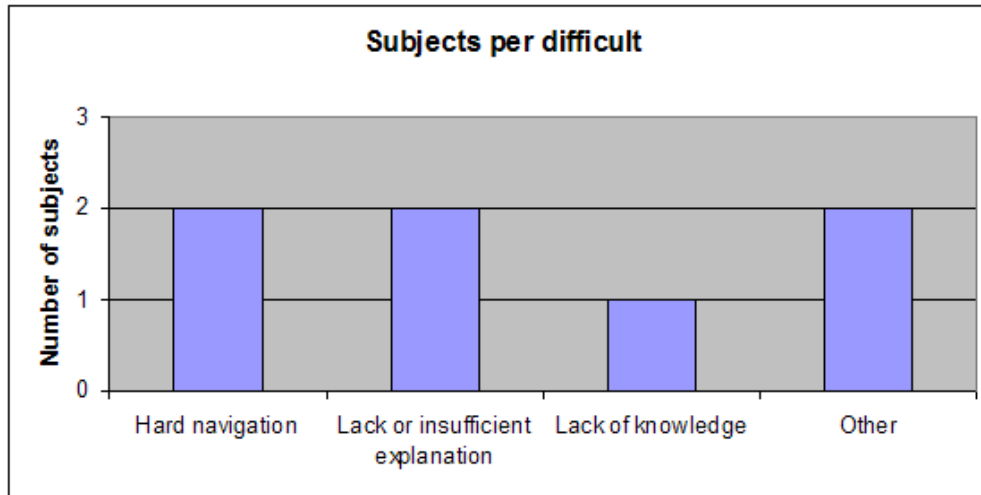


Figure 5.2 Number of Subjects per Difficulty

Consistency Checker Solution. After the analysis of the answers about the consistency checker messages, only one subject (ID 5) informed that the consistency checker did not fully aid the problem identification. The tool restriction, according to him, refers to the lack of an easier identification where the problem is occurring, i.e. the tool should select the exact local where the problem is occurring when the problem message is visualized - aspect already present in the current version of the tool. The remaining subjects, approximately 83% of the total, considered the consistency checker message sufficient for identifying and resolving the problems. Thus, this **rejects** the null hypothesis (H_{0c}).

Sufficient Tutorial. Analyzing the subjects' answers regarding the tutorial, there were two subjects (ID 1, ID 6) that did not use the tutorial during the tool usage, thus, these data were excluded from this question analysis. The other four subjects used the tutorial and considered its information sufficient for learning how to use the tool, representing 100% of the total. Therefore, this **rejects** the null hypothesis (H_{0d}).

In addition, one of the subjects (ID 6) that did not use the tutorial was the same one who declared that had some difficulty due to lack or insufficient explanation for using the tool. Thus, this indicates that the tutorial may overcome this issue. Additionally, the other subject (ID 1 - who has only a basic knowledge in Eclipse) was the same one that did not have any difficulty with the tool during the process. Thus, this shows that even though ToolDay is based on Eclipse's platform, it is not necessary to have a great knowledge in it to use the tool.

Customization Usefulness. It was not possible to validate this question, because no

subjects used the customization options. Therefore, the null hypothesis (H_{0e}) can neither be rejected nor the alternative hypothesis can be confirmed (H_{1e}) since all data were excluded. The lack of use of the customizations options can indicate that they were not necessary because the default option is already sufficient, or the subjects may not knew about them. However, both cases indicate a failure in the questionnaire.

Analysis and Interpretation Conclusions. Even with the analysis not being conclusive, the experimental study indicates that the tool has some strength for the domain analysis support, mainly with the consistency checker and the domain model representation. On the other hand, aspects related to understanding (difficulties during the process execution) need to be taken into consideration for a new development cycle. Conversely, aspects such as the usefulness of the provided customization and which steps of the tool support were more difficulty to be executed, could not be conclusive.

Even though the requirements management has been changed since this evaluation, it is not the main functionality in a domain analysis process, for this reason it may still be improved, specially on how to treat the variabilities. RiSE labs has some works focused on this theme, requirements engineering and scoping for software product lines, which can lead to the identification of new requirements to be implemented for ToolDay.

Nevertheless, some of the problems described by the subjects (such as, expectation about the requirement managements and the available customizations) can be resolved if a proper training, before the tool starts to be used, is performed. Conversely, with the results identified in the experiment, the values can be calibrated in a more accurate way.

Lessons Learned

After concluding the first experimental study, some aspects should be considered to repeat the experiment, as they were seen as limitations of the first execution.

Training. Instead of letting the subjects to learn how to use the tool with only the tutorial, a basic training can be applied. Through the training it is possible to emphasize on aspects not much used by the subjects of this experiment and on the complains related in this experiment.

Questionnaires. The questionnaires should be reviewed to collect more precise data related to feedback and the helpfulness of the tool. One option is to include questions regarding each process phase - i.e. Planning, Domain Modeling, Domain Validation, Product Modeling and Product Validation. Thus, it may be easier to map the exactly problems the users had during the tool usage.

Pilot. A project pilot should be performed before the experiment. Through it, it is

possible to identify feasible problems in the experiment planning, especially with the questionnaires.

5.1.4 Second Experiment

The second experiment replicated the first one, therefore, its phases were very similar. Later on this section, it is described the differences between the experiments and the results achieved this time.

Planning

The experiment was conducted in a professional master's degree at the university lab, hence, like the previous one, the experiment ran off-line and it was conducted by the students that performed a domain analysis step in a small real-world case.

As the first experiment, the instrument used to validate it was a feedback form. However, the *questionnaire was modified according to some of the lessons learned from the first experiment*. Thus, its questions regarding the tool usage and the subject's opinion about it were further divided in general use of the tool and on specific questions about each step supported by ToolDAy, while the other questions were kept the same. The complete feedback form can be seen in Appendix B.

The hypotheses evaluated in this experiment were the same, but the values defined for them were calibrated based on the previous results.

Null Hypothesis. The values for the null hypotheses were:

H_{0a} : μ helpfulness in the process $< 60\%$.

H_{0b} : μ process difficulty $\geq 60\%$.

H_{0c} : μ consistency checker messages $< 80\%$.

H_{0d} : μ sufficient tutorial $< 80\%$.

H_{0e} : μ customization usefulness $< 70\%$.

Alternative Hypothesis. The alternative hypothesis values were:

H_{1a} : μ helpfulness in the process $\geq 60\%$.

H_{1b} : μ process difficulty $< 60\%$.

H_{1c} : μ consistency checker messages $\geq 80\%$.

H_{1d} : μ sufficient tutorial $\geq 80\%$.

H_{1e} : μ customization usefulness $\geq 70\%$.

The study was conducted as a *single object study* and the subjects were also selected based on convenience, being all of them M.Sc. students from the Professional Master at

CESAR.EDU², Pernambuco, Brazil.

Before the subjects started to use the tool, they had *training* about ToolDay's flowchart, detailing its optional steps and the expected artifacts as result, as opposed to the first experiments, in which the subjects had no training. And they could also ask question about the tool functionalities. Furthermore, the threats to validity for this experiment were the same of the previous one, but it was added a new *internal validity* that refers to the restrict duration of the experiment.

It happens because in the professional master the classes for a specific topic - software reuse in this case - last only one week. However, the experiment goal was not to analyze the quality of the produced artifacts, instead it evaluated how the tool aided during the artifacts production. Thus, this internal validity was considered sufficient.

The Experimental Study Project

The project used in the experimental study was to perform the domain engineering of the fixed shooter game domain - the same domain as the example described in chapter 4. In order to do it, three games in this domain were presented to the subjects, which they had just the executables without any documentation (requirements and design specification, source code, etc) about the games. It is important to highlight that the subjects could analyze other games, consult experts, and so on, as part of the process.

Moreover, the subjects were presented, in class, to three different domain analysis processes, and they could choose anyone of them to do the project.

The total number of subjects was ten and they were divided in two groups of five people. Even though all subjects participated in domain analysis phase, only two of them (one for each group) actually used the tool, which provided the results taken into account in the experiment analysis. The number of subjects were reduced due to the rapid duration of the course.

This aspect was not expected (as it was considered an internal threat of the study), because the number of students in the class were enough to obtain a valid feedback and all of them were supposed to use the tool during the project. Since only these two subjects actually participated in the experiment, the hypotheses limits were discarded because the possible percentage results to analyses were just 0%, 50% and 100%. Therefore, it was not possible to provide a valid quantitative analysis about the experiment.

²<http://www.cesar.edu.br/>

Operation

The experimental study was conducted during a course of the professional master in software engineering at CESAR.EDU in June 2008. The course lasted only one week, which was the time the subjects had to conduct the project.

The subject's knowledge in the Eclipse platform in this experiment varied a lot. One of them considered his knowledge as *good*, while the other rated it as *low*. In addition, both subjects had no experience with domain analysis processes, and they had never participated in domain analysis projects other than the one being described. Table 5.2 presents the summary of the subjects' profile.

| ID | Eclipse's Knowledge | Domain Analysis Processes | Domain Analysis Projects | Projects's Roles |
|----|---------------------|---------------------------|--------------------------|------------------|
| 1 | Low | - | 1 Project | Manager |
| 2 | Good | - | 1 Project | System Engineer |

Table 5.2 Subjects' Profile in the second experiment

Analysis and Interpretation

This section describes the feedback and comments made by the two subjects about the tool. However, their answers are not quantitatively interpreted due to the reduced number of subjects.

Helpfulness in the Process. Through the answers provided by the subjects about the aid the tool usage provides to the project, all of them considered that the tool use brought benefits to the project execution. Some of the benefits described by them were: facilitated the identification and validation of the domain scope and the variability points within it, provides a good graphical representation of the domain features and the consistency checker quickly aids in the inconsistency identification.

Moreover, one subject included a following comment regarding the tool usage: *“Due to the lack of previous experience in domain analysis, and on how to do it, the tool usage was essential for a good understanding of the project and on how to fulfill the goals of the master course.”*

The subjects were also asked to include improvements about the tool. Some of them were: improve the help system with examples, better explanation about the documentation fields, reduce the quantity of documentation fields and about the interface design.

Process difficulty. According to the questionnaire answers, just one subject (ID 2) had difficulty during the project execution. Since the questionnaire prepared for this

experiment asked the difficulty for each step of the process, it was possible to identify that his difficulty was in the planning step - in which he claimed the lack of knowledge in the domain analysis process - and in the validation one - in which there was not enough information regarding the expected data in the documentation fields.

Even though some difficulties were included, both the subjects considered that every step executed in ToolDay - the product model step was not used - brought benefits to the process. Some of the benefits were the domain scope definition through the product map; the representation and identification of the domain features in the feature model, and the project documentation, which helped in mapping the domain.

Consistency checker messages. After the analysis of the answers about the consistency checker messages, both subjects agreed that the consistency checker messages aid in the problem identification and solution.

Sufficient tutorial. Regarding the tutorial's content for explaining the tool steps, one subject (ID 1) considered the tutorial enough to understand the tool usage and the other (ID 2) answered that the tutorial's sequence of steps was not clear, because it lacks real examples of how and what to fulfill in each step. Hence, it indicates an improvement in the tutorial details.

Customization usefulness. Once again, no subject used the customizations, however one of them informed that the customizations were not used because the default configuration is good. Conversely, the other subject described that he did not know about what was possible to personalize in the tool, this indicates that even though there was a training before the project start that mentioned the tool customizations, it should be more highlighted.

Analysis and Interpretation Conclusions. Even with the reduced number of subjects, it was possible to identify that this questionnaire resolved some of the issues highlighted in the previous experiment, such as the difficulty and benefits each step had with/brought for the tool usage. Furthermore, this experiment proposed some improvements to the tutorial - aspect not related in the first experiment - besides minor improvements to the tool itself, as described before. But, once again, the majority of improvements and difficulties described do not concern the tool functionalities supporting the domain analysis process, they focused more on usability items, like necessity of better explanation and design in some steps of the tool.

5.1.5 Experiments Summary

The section presented the definition, planning, operation, analysis and interpretation of two experimental studies that evaluated the helpfulness provided by ToolDay's usage in a domain analysis project.

Even though the second experiment could not have a valid data analysis, due to the restrict participation, its indications along with the results from the first experiment, suggests that the tool provides benefits when used in a domain analysis project. These benefits were especially described to the graphical representation and the consistency checker, but the tool also guides the subjects during the project.

Next section describes ToolDay's use in an industrial project.

5.2 ToolDay's Industrial Case

Apart from the two controlled experiments executed for validating ToolDay's helpfulness in domain analysis projects, it was also applied in an industrial context.

The industrial case was developed at C.E.S.A.R.³, a Brazilian Innovation Institute, with 600 employees and CMMI level 3. And it is part of the company's software reuse effort.

C.E.S.A.R's reuse program was conducted according to RiSE-TCM guidelines and process Garcia *et al.* (2008a). This process is divided in five phases:

- *Adoption/Improvement Reuse Strategy - Goals Definition*: identifies the business goals of the organization for which software reuse may be helpful, and to build a Reuse Adoption/Improvement Strategy that addresses the attainment of those goals.
- *Reuse Maturity Level Evaluation*: understands the organization's situation with respect to those business goals identified in the previous phase Garcia *et al.* (2008b).
- *Reuse Adoption/Improvement Planning*: develops all the necessary plans to start a structured use of the software reuse technology in the organization.
- *Implementation of the Reuse Adoption/Improvement Pilot*: implements and monitors several trials or pilot projects in which software reuse is used in those areas of the system development identified as improvement candidates in the previous phase.

³Recife Center for Advanced Studies and Systems - <http://www.cesar.org.br>

- *Reuse Deployment*: deploys the Reuse Adoption/Improvement solutions to new and ongoing projects.

The business goals defined for the reuse project were to increase the productivity, to reduce maintenance costs and development efforts. Before introducing the program in the whole company, a pilot project was selected. The chosen one was a *web/social network* with a team of nine people and one product with seven different releases.

Due to these characteristics, the goal of the pilot was to adopt a software product line with the aimed benefits of: **(i)** better understandability of the project business by the team; **(ii)** identification of new market opportunities for the product; **(iii)** identification of new functionalities for the product; and, **(iv)** decrease the maintenance cost for the releases.

Before starting the domain analysis execution, the whole team was trained to use ToolDay, however, only one person of the team took the domain analysis role, being the only one that used the tool. Even though only one person used the tool, all the artifacts were validated by the complete team.

The team's goal for the complete project was to identify the domain modules to transform into components, and for the domain analysis phase was to identify existing features from other tools that were not present in their tool.

They performed the domain analysis of the social network domain, documenting the domain, defining its scope, building a glossary, the product map of the analyzed applications (composed of 7 concurrent products) and the features model of domain with 74 features. Figure 5.3 shows part of the feature model for the social network domain.

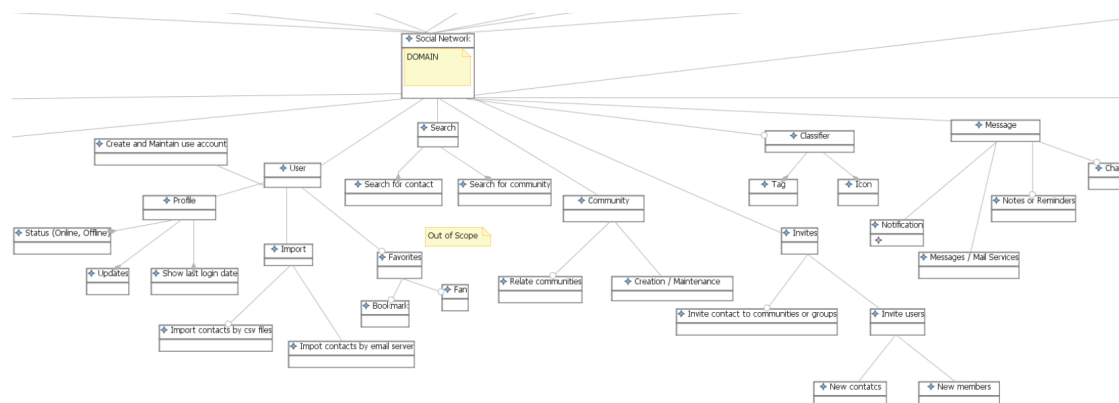


Figure 5.3 Part of the social networks feature model

The project is still under execution building the reference architecture of the domain, however the part involving ToolDay's usage is completed.

At the end of ToolDay's usage, the domain analyst was asked to fulfill a questionnaire about the domain analysis phase. He, after the team validation of the artifacts, considered that the project goal was achieved along with a better understanding of the domain being developed, since several new features were identified, and are now being planned to be developed.

Thus, through the domain analysis it was possible to map the list of common and variable features of the domain, identifying them within the releases the project already has (it is currently 7) and comparing the project current stage to other products from web/social network.

Furthermore, it took approximately 3 weeks for the finalization of the domain analysis step. However, in this period, the domain analyst was not fully allocated to it and no type of research, regarding the concurrent products, was done before the initialization of the project. At the end of the analysis, it was generated a feature model for the web/social network with seventy-four features.

The user was asked about the domain analysis process it used and the difficulty encountered for the tool. However, no formal process was followed during the project, they only followed ToolDay's steps in order to achieve the complete process. Moreover, the user had no difficulty using the tool.

The analyst also related some improvements for the tool. They are the possibility to export the product map as table or document and to provide some other ways to visualize and print the feature model, since when the number of features get to high, it becomes hard to see the complete domain.

Even though the goal was achieved and the process result brought benefits to the team, for the domain analysts, there is no real data that using ToolDay during the domain analysis process aided it. However, it is necessary to highlight that since the user did not follow a specific process, the tool contributed in the process execution because it provided a guideline to define the domain scope, its feature model and to document them. Furthermore, this industrial case worked as a proof of concept for the tool.

5.3 Chapter Summary

In this chapter was presented all the evaluations performed for ToolDay. Two of these evaluations occurred in an academic environment, in which the first experiment brought valid consideration for the tool improvement, while the second one could not be fulfilled because one of its threats (number of subjects using the tool) could not be resolved. The

last evaluation described ToolDay's benefits when used in an industrial project.

Next chapter concludes this dissertation by summarizing the analysis performed on this chapter, reviewing some related works, pointing directions for future enhancements to the environment and presenting some final considerations.

*“De tudo ficam três coisas:
A certeza de que estamos sempre começando...
A certeza de que precisamos continuar...
A certeza de que seremos interrompidos antes de terminar...”*

Fernando Pessoa (Poet)



Conclusion Remarks and Future Work

The implications to reduce development time and improve product quality make the software reuse approach very attractive for software organizations. Nowadays, some organizations are adopting domain analysis processes as a way to achieve these benefits.

Although the benefits of software reuse are promising, it is a complex task to put it in practice. For instance, the domain analysis process involves a large quantity of interrelated activities and different sources of information to be analyzed, increasing its complexity, therefore a tool support becomes essential (Bass *et al.*, 1997; Succi *et al.*, 2000a; Moon *et al.*, 2005).

In this sense, in order to facilitate the process usage and the to aid the domain analyst to achieve systematic reuse in an effective way, this dissertation presented ToolDAY - Tool for Domain Analysis. The tool is based on an extensive systematic review of current available tools, and on the RiSE's group expertise.

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

6.1 Research Contribution

The main contributions of this work can be split into the following aspects: **i.** the realization of a brief survey on CASE tools and Environment classification; **ii.** an extensive systematic review on existing domain analysis tools; **iii.** the definition of requirements, architecture, and implementation of a domain analysis tool; **iv.** two experiments and an industrial case study that evaluated the tool helpfulness in three domain analysis projects; and **v.** a commercial product.

- **Survey on CASE tools and environment classification.** The objective was to map different types of classification that exist for CASE tools and environment as a way to identify and understand characteristics that should be present in the tool development.
- **Systematic review on domain analysis tools.** Through this study, nineteen tools were identified and analyzed according to aspects related to the systematic review's question - *how the available tools for domain analysis are offering the support to the process*. This analysis offered the base to define a set of requirements that should be present in domain analysis tools, and to map new tools development and usage. Furthermore, these results can be used to guide the development of new tools, both for domain analysis and for domain design and implementation - since they can consider the work products defined in systematic review as input for the next phases. In addition, the companies can use them as a way to identify the best tool according to their needs.
- **ToolDay.** After the systematic review, its results were the initial step for the definition of ToolDay's requirements. Afterwards, its architecture was defined and the tool implemented.
- **The evaluation.** ToolDay was used in three different evaluations, two of them as experiments in an academic environment and the other in an industrial project. The studies analyzed the tool helpfulness in the process, and their findings suggest that the tool usage can aid in the project execution.
- **Commercial Product.** The final version of ToolDay is now being commercialized as one of RiSE's portfolio product to aid in productivity gains for companies.

6.2 Related Work

Several domain analysis tools were identified during this research, as described in chapter 3, and their analyses served as input for ToolDay's development. Therefore, ToolDay focused on the gaps identified, in which the main refers to the lack of tool providing full support to the three phases of the process - planning, modeling and validation; aspect fulfilled in ToolDay implementation, as described in chapter 4.

Table 6.1 compares ToolDay's functionalities (depicted in the last row) against the other tools. The roman numbers follow the same order as presented in chapter 3. Through

this Table, it is possible to see that ToolDay fulfills all the defined requirements, even the ones rated as “low”, therefore it provides a wider support to the domain analysis process when compared to the related works.

| Functionalities / Tools | i | ii | iii | iv | v | vi | vii | viii | ix | x | xi | xii | xiii | xiv | xv | xvi | xvii | xviii | xix | xx |
|-------------------------|---|----|-----|----|---|----|-----|------|----|---|----|-----|------|-----|----|-----|------|-------|-----|----|
| 001 | | | | | • | • | • | • | | | | • | | | | • | • | • | • | |
| Ami Eddi | | | | | • | • | • | | | | | | | | | | | | • | |
| ASADAL | | | | | • | • | • | • | • | • | | | | | | | | • | • | |
| Captain Feature | | | | | • | • | • | • | | | | | | | | | | • | • | |
| DARE | • | • | | | • | • | • | | | | | • | | | | • | | | | |
| DECIMAL | | | | | • | • | • | • | | | | | | | | | | • | • | |
| Domain | | | | | • | | | | | | | • | | • | | • | | | | |
| Doors Extension | | | | | • | • | • | • | | | | | | • | • | | | • | | |
| DREAM | | • | | • | • | • | • | | | • | | | | | • | | • | | | |
| Feature Plugin | | | | | • | • | • | • | | | • | | • | | | | | • | • | |
| FeatureIDE | | | | | • | • | • | • | | | | | | | | | | • | • | |
| GEARS | | | | | • | • | • | • | | | | | | | | | • | • | • | |
| Holmes | • | • | | | • | • | • | • | | | | • | | | | • | | • | • | |
| Odyssey | | | | | • | • | • | | | | | • | | • | • | | | | | • |
| Pluss Toolkit | | | | | • | • | • | • | | | | | | | • | | • | • | • | |
| PuLSE-BEAT | | • | • | • | • | • | • | | | | | • | • | | | | • | | | |
| Pure::Variants | | | | | • | • | • | • | | | | • | • | | | | • | • | • | |
| RequiLine | | | | | • | • | • | • | | • | • | • | • | • | • | | • | • | • | • |
| XFeature | | | | | • | • | • | • | | | • | • | • | • | | | • | • | • | • |
| ToolDay | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |

Table 6.1 ToolDay’s functionalities compared to other tools

Furthermore, ToolDay also supports several usability requirements as a way to facilitate its usage, especially with the inclusion of two example projects already fulfilled using ToolDay. Therefore, as described in the previous chapters, ToolDay implements all the requirements identified in Table 4.1 presented in chapter 4.

6.3 Future Work

In spite of the commitment to develop the tool, some improvements are visualized. They were not included in the initial tool development because some of them were considered out of the scope of the master degree, but are important to a full industrial tool, and others were comments/feedbacks from users. In this fashion, some important aspects that were left out of this version are enumerated:

- **Metrics.** The metrics currently provided by ToolDay inform the quantity of items per type (features, requirements, use cases, and so on). However, this type of

information does not aggregate much value for the domain analyst. Therefore, the inclusion of more elaborated metrics, such as the number of products to be derived from the domain, or the number of product the feature belongs to, can provide a better insight for the domain analyst. Furthermore, other metrics can provide a qualitative analysis on the domain, like the optimal percentage of variable features in a domain, however, there is no real metric in the literature defining it.

- **Later binding times.** Variabilities are becoming more and more important in software engineering because of the costs associated to modifying existing software systems and, recently, there have been an increasing demands for the postponement of variability decisions to later stages, i.e. at design, implementation and run-time (Gurp *et al.*, 2001; Lee and Muthig, 2006). However, ToolDAy's current version obligates the user to resolve all the variation options during the product derivation.

One of the ways to treat this type of problem is through the use of *Feature Binding Units* (FBU) (Lee and Kang, 2004a), which is defined as a set of features that are related to each other by the relationships in a feature model. Therefore, with FBUs the features (common and variables) grouped together focus on a common service and must exist together for a correct operation of the services (Lee and Muthig, 2006). This grouping intends to reduce the complexity of managing the variabilities and to permits the variabilities within an unit to be resolved later on the project.

- **Different visualizations view (Alonso and Frakes, 2000).** As related in chapter 3, there are several ways to represent the domain, such as feature models, tables, tree views and so on. Each of these representation types have its own benefits and disadvantages. For example, one of the main problems with feature models is that when the domain grows a lot, it becomes harder to visualize it in an understandable way, this is actually one on the main challengers in domain analysis (Frakes and Kang, 2005).

This variety of representation can aid the domain analyst since it allows he/she to choose his/her preferable visualization. Besides, different views can also represent new information, which currently is not available. One of this information can be the FBU described before, they can be represented through a new editor derived from the domain feature model.

- **Complete support to domain engineering.** A part from the domain analysis step, the process includes domain design and implementation. Similar to analysis, these

steps involve a several activities that must be fulfilled for an effective result. However, differently from the domain analysis, these steps depend on the results from the previous one and they can influence the modification of previous artifacts, since it is an iterative process (Prieto-Díaz, 1990). Therefore, a continues traceability of the artifacts among them is the ideal situation.

- **Complex consistency rules.** Consistency rules are an extremely important functionality because they delimited the possible configurations, not allowing inconsistency products. In spite of the available rules within ToolDAy, there are more complex rules that involve the combination of two or more features to implicate or exclude another feature and these rules can also involve the values defined for the feature's attributes. In addition, it also involves the treatment of feature interactions, since a feature variation may cause changes to many components implementing other features (Lee and Kang, 2004b).
- **Product requirements.** Even though ToolDAy permits the selection of which features belong to the product for later generate its feature model, it does not permit the identification of requirements and use cases for the product. Some possibilities to resolve it are automatically doing it based on the correlations existing in the traceability editor or allowing the user to select them - similar as available to the features.
- **Export to semantic models.** The information collected and described during the domain analysis project can be considered as a knowledge base for the domain, therefore this information can be used by web applications needing to consume the domain knowledge. As a way to achieve it, ToolDAy should export its domain analysis information to semantic models, represented in Mark-up Ontology Web Languages, such as Resource Description Framework (RDF) and Web Ontology Language (OWL), which is a formal data model that supports the semantics of the entities as well as allows applications to process and manipulate its content (Berners-Lee, 1996).
- **More evaluations.** Besides being evaluation in three situations, new studies in different contexts, including more subjects and others domains are still necessary in order to the tool helpfulness and more improvements. Other evaluation is to compare ToolDAy usage and results with another domain analysis tool, specially the industrial ones, like Pure::Variants and Gears.

- **Integration with third part tools.** There are other tools that can complement ToolDAy functionalities, such as IBM Telelogic Doors¹, which is family of products to define and manage requirements to meet business goals; IBM Rational², for proving other modeling options, and so on. For each of these third part tools, it is necessary to create plug-ins and/or extensions.

Despite the work, there are also some usability enhancement to perform in the tool, such as the inclusion of more meaningful icons for the actions representation and different symbols for representing the features, e.g. variation point features can have a different symbol than “leaf” features.

6.4 Concluding Remarks

This work presented a tool for providing a semi-automated support to domain analysis projects. This tool was based on an extensive systematic review, whose results can be used, besides for new tools development, for a valid set of work products to be expected in the next phases - domain design and implementation - and for companies to identify possible tools to use. Additionally, the evaluations findings indicates that the tool is well suited for aiding the domain analysts in project execution.

In this chapter, the findings of the studies were presented, along with some related works that in some way have influenced in this work and finally introduced the future directions for the tool, such as metrics support, more consistency rules, different binding times and visualizations options.

¹<http://www.telelogic.com/products>

²<http://www-01.ibm.com/software/rational/sw-bycategory/>

Bibliography

- Albizuri-Romero, M. B. (2000). A retrospective view of case tools adoption. *SIGSOFT Software Engineering Notes*, **25**(2), 46–50. 2
- Almeida, E. S. (2007). *RiDE: The RiSE Process for Domain Engineering*. Ph.D. thesis, Federal University of Pernambuco, Brazil. 1, 3.6.1, 4.4.1, 5.1.3
- Almeida, E. S., Alvaro, A., Lucrédio, D., Garcia, V. C., and Meira, S. R. L. (2004). Rise project: Towards a robust framework for software reuse. In *IEEE International Conference on Information Reuse and Integration (IRI)*, pages 48–53, Las Vegas, NV, USA. 1.3.1
- Almeida, E. S., Alvaro, A., Lucrédio, D., Garcia, V. C., and Meira, S. R. L. (2005). A survey on software reuse processes. In *IEEE International Conference on Information Reuse and Integration (IRI)*, pages 66–71, Las Vegas, Nevada, US. 1.3.1
- Almeida, E. S., Mascena, J. C. C. P., Cavalcanti, A. P. C., Alvaro, A., Garcia, V. C., Meira, S. R. L., and Lucrédio, D. (2006). The domain analysis concept revisited: A practical approach. In M. Morisio, editor, *International Conference on Software Reuse (ICSR)*, pages 43–57, Turin, Italy. 1.1, 3.5.2, 4.4.1
- Almeida, E. S., Alvaro, A., Garcia, V. C., Lucrédio, D., Fortes, R. P. M., and Meira, S. R. d. L. (2007). An experimental study in domain engineering. In *IEEE EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Component-Based Software Engineering Track*, Lubeck, Germany. 3.5.2
- Alonso, O. and Frakes, W. B. (2000). Visualization of reusable software assets. In W. B. Frakes, editor, *ICSR*, volume 1844 of *Lecture Notes in Computer Science*, pages 251–265. Springer. 6.3
- Alvaro, A., Almeida, E. S., and Meira, S. R. L. (2006). A software component quality model: A preliminary evaluation. In *32nd IEEE EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Component-Based Software Engineering Track*, pages 28–35, Cavtat/Dubrovnik, Croatia. 1.3.1
- Antkiewicz, M. and Czarnecki, K. (2004). Featureplugin: feature modeling plug-in for eclipse. In *OOPSLA workshop on eclipse technology eXchange*, pages 67–72, New York, NY, USA. ACM Press. 3.5.1

- Arango, G. (1994). Domain analysis methods. In E. Horwood, editor, *Software Reusability*, pages 17–49, Chichester, England. 3.6.1
- Basili, V. R., Caldiera, G., and Rombach, H. D. (1994). The goal question metric approach. In *Encyclopedia of Software Engineering*, pages 528–532. John Wiley & Sons, Inc. 5.1.2
- Basili, V. R., Briand, L. C., and Melo, W. L. (1996). How reuse influences productivity in object-oriented systems. *Communications of the ACM*, **39**(10), 104–116. 1.1
- Bass, L., Clements, P., Cohen, S. G., Northrop, L., and Withey, J. (1997). Product line practice workshop report. Technical report, Technical Report CMU/SEI-97-TR-003, Software Engineering Institute, Carnegie Mellon University, Pittsburgh. 1.1, 1.3.2, 3, 3.5.2, 6
- Bayer, J., Muthig, D., and Widen, T. (1999a). Customizable domain analysis. In *International Symposium on Generative and Component-Based Software Engineering*, pages 178–194. Springer-Verlag. 3.5.2
- Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schimd, K., Widen, T., and DeBaud, J.-M. (1999b). Pulse: a methodology to develop software product lines. In *Symposium on Software reusability*, pages 122–131, Los Angeles, California, United States. ACM Press. 1.1, 3.5.2, 3
- Berners-Lee, T. (1996). Wwv:past, present and future. *IEEE Computer*, **29**(10), 69–77. 6.3
- Beuche, D. and Spinczyk, O. (2003). Variant management for embedded software product lines with pure::consul and aspectc++. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 108–109, Anaheim, CA, USA. ACM. 3.5.1
- Biggerstaff, T. J. (1992). An assessment and analysis of software reuse. *Advances in Computers*, **34**, 1–57. 1.1, 3
- BigLever (2005). Gears user’s guide. Technical report, BigLever. 3.5.1
- Biolchini, J., Mian, P. G., Natali, A. C. C., and Travassos, G. H. (2005). Systematic review in software engineering. Technical Report ES 679/05, System Engineering and Computer Science Department - COPPE UFRJ, Rio de Janeiro. 3.2

- Braga, R. M. M. (2000). *Components search and retrieval in software reusable environments (in Portuguese)*. Ph.D. thesis, UFRJ - Federal University of Rio de Janeiro, Brazil. 3.5.1, 3.5.2
- Braga, R. M. M., Werner, C., and Mattoso, M. (1999). Odyssey: A reuse environment based on domain models. In *IEEE Symposium on Application-Specific Systems and Software Engineering & Technology (ASSET)*, pages 49–57, Richardson, Texas. 3.5.1
- Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., and Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, **80**, 571–583. 3.4.1
- Brito, K. S. (2007). *LIFT: A Legacy InFormation retrieval Tool*. Master’s thesis, Federal University of Pernambuco, Recife, Pernambuco, Brazil. 1.3.1
- Buhne, S., Lauenroth, k., and Klaus, P. (2005). Modelling requirements variability across product lines. In *IEEE International Conference on Requirements Engineering*, pages 41–52. 3.5.1, 3.5.2
- Capilla, R., Sánchez, A., and Dueñas, J. C. (2007). An analysis of variability modeling and management tools for product line development. In *Software and Service Variability Management Workshop - Concepts, Models, and Tools*, pages 32–47, Helsinki, Finland. 3.1
- Cavalcanti, Y. C., Martins, A. C., Almeida, E. S., and Meira, S. R. L. (2008). Avoiding duplicate cr reports in open source software projects. In *In The 9th International Free Software Forum (IFSF), Free Software Workshop*, Porto Alegre, Brazi. 1.3.1
- Cronholm, S. (1995). Why case tools in information systems development? - an empirical study concerning motives for investing in case tools. In *18th Information Systems research In Scandinavia (IRIS 18)*, Gjern, Denmark. 2.1
- Czarnecki, K. and Eisenecker, U. (2000). *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley. 1.1, 1.4, 4.4.1, 4.4.1
- Czarnecki, K., Antkiewicz, M., Kim, C. H. P., Lau, S., and Pietroszek, K. (2005). fmp and fmp2rsm: eclipse plug-ins for modeling features using model templates. In *Object-oriented programming, systems, languages, and applications (OOPSLA)*, pages 200–201, San Diego, CA, USA. ACM. 3.5.1

- Dehlinger, J., Humphrey, M., Suvorov, L., Padmanabhan, P., and Lutz, R. (2007). Decimal and plfaultcat: From product-line requirements to product-line member software fault trees. In *Companion to the proceedings of the 29th International Conference on Software Engineering*, pages 49–50. IEEE Computer Society. 3.5.1
- Durão, F. A. (2008). *Semantic Layer Applied to a Source Code Search Engine*. Master’s thesis, Federal University of Pernambuco, Recife, Pernambuco, Brazil. 1.3.1
- Eriksson, M., Börstler, J., and Borg, K. (2005a). The pluss approach - domain modeling with features, use cases and use case realizations. In *Software Product Lines Conference*, volume 3714 of *Lecture Notes in Computer Science*, pages 33–44, Rennes, France. 3.5.1
- Eriksson, M., Morast, H., Borstler, J., and Borg, K. (2005b). The pluss toolkit - extending telelogic doors and ibm-rational rose to support product line use case modeling. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 300–304, Long Beach, CA, USA. ACM. 3.5.1
- Eriksson, M., Morast, H., Borstler, J., and Borg, K. (2006). An empirical evaluation of the pluss toolkit. Technical Report UMINF-06.31, Department of Computing Science, Umea University. 3.5.1, 3.5.2
- Filho, E. D. S., Cavalcanti, R. O., Neiva, D. F. S., Oliveira, T. H. B., Lisboa, L. B., Almeida, E. S., and Meira, S. R. L. (2008). Evaluating domain design approaches using systematic review,. In *2nd European Conference on Software Architecture (ECSA)*, Cyprus. Lecture Notes in Computer Science (LNCS), Springer Verlag. 1.3.1
- Frakes, W. and Succi, G. (2001). An industrial study of reuse, quality, and productivity. *Journal of Systems and Software*, **57**(2), 99–106. 1.1
- Frakes, W. B. (1997). Automating domain engineering. In *Workshop on Institutionalizing Software Reuse (WIRS)*. 3.5.1
- Frakes, W. B. and Isoda, S. (1994). Success factors of systematic reuse. *IEEE Software*, **11**(5), 14–19. 1
- Frakes, W. B. and Kang, K. C. (2005). Software reuse research: Status and future. *IEEE Transactions on Software Engineering*, **31**(7), 529–536. 1, 6.3

- Frakes, W. B., Prieto-Diaz, R., and Fox, C. (1997). Dare-cots. a domain analysis support tool. In *Proceedings of the 17th International Conference of the Chilean Computer Science Society (SCCC '97)*, page 73. IEEE Computer Society. 3.5.1, 3.6.1
- Frakes, W. B., Prieto-Diaz, R., and Fox, C. J. (1998). Dare: Domain analysis and reuse environment. *Annals of Software Engineering*, **5**, 125–141. 1.1, 3.5.1, 3.5.2
- Fuggetta, A. (1993). A classification of case technology. *Computer*, **26**(12), 25–38. 2.2, 2.3, 3.6.2, 5.1
- Garcia, V. C., Lucrédio, D., Lisboa, L. B., Martins, A. C., Almeida, E. S., Fortes, R. P. M., and Meira, S. R. L. (2006). Toward a code search engine based on the-state-of-art and practice. In *13th IEEE Asia Pacific Software Engineering Conference (APSEC), Component-Based Software Development Track*, Bangalore, India. 2.3
- Garcia, V. C., Lisboa, L. B., Durão, F. A., Almeida, E. S., and Meira, S. R. L. (2008a). A lightweight technology change management approach to facilitating reuse adoption. In *2nd Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS 2008)*, Porto Alegre, Brazil. 1.3.1, 5.2
- Garcia, V. C., Lisboa, L. B., Meira, S. R. L., Almeida, E. S., Lucrédio, D., and Fortes, R. P. M. (2008b). Towards an assessment method for software reuse capability. In *8th International Conference on Quality Software (QSIC 2008)*, Oxford, UK. 1.3.1, 5.2
- Gomaa, H. and Shin, M. E. (2004). Tool support for software variability management and product derivation in software product lines. In *Workshop on Software Variability Management for Product Derivation, Software Product Line Conference (SPLC)*, Boston, USA. 3.1
- Goth, G. (2005). Beware the march of this ide: Eclipse is overshadowing other tool technologies. *IEEE Software*, **22**(4), 108–111. 4.3
- Gray, J. P., Liu, A., and Scott, L. (2000). Issues in software engineering tool construction. *Information and Software Technology*, **42**, 73–77. 2.2.1
- Gurp, J. V., Bosch, J., and Svahnberg, M. (2001). On the notion of variability in software product lines. In *Working IEEE/IFIP Conference on Software Architecture (WICSA '01)*, page 45, Washington, DC, USA. IEEE Computer Society. 6.3
-

- Hamilton, M. H. (1991). 001: A full life cycle systems engineering and software development environment development before the fact in action. http://world.std.com/~hti/Articles/Full_Life_Cycle.htm. 3.5.1
- Harrison, W., Ossher, H., and Tarr, P. (2000). Software engineering tools and environments: a roadmap. In *Conference on The Future of Software Engineering*, pages 261–277, Limerick, Ireland. ACM Press. 2, 2.2.2
- Jaring, M. and Bosch, J. (2002). Representing variability in software product lines: A case study. In *Software Product Line Conference (SPLC)*, pages 15–36, San Diego CA. 3.5.2
- Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990). Feature-oriented domain analysis (foda) feasibility study. Technical Report Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University. 3.5.2, 3.5.2, 4.4.1, 4.5
- Kang, K. C., Kim, S., Lee, J., Kim, K., Shin, E., and Huh, M. (1998). Form: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Engineering*, **5**, 143–168. 3.5.2
- Kim, K., Kim, H., Ahn, M., Seo, M., Chang, Y., and Kang, K. C. (2006). Asadal: a tool system for co-development of software and test environment based on product line engineering. In *International Conference on Software Engineering (ICSR)*, pages 783–786, New York, NY, USA. ACM Press. 3.5.1
- Kim, M., Yang, H., and Park, S. (2003). A domain analysis method for software product lines based on scenarios, goals and features. In *Asia-Pacific Software Engineering Conference (APSEC)*, pages 126–136, Thailand. 1.1
- Kitchenham, B. (2004). Procedures for performing systematic reviews. Technical report, Joint Technical Report, Keele University TR/SE-0401 and NICTA 0400011T.1. 3.2, 3.3.2, 3.4.1
- Krueger, C. W. (1992). Software reuse. *ACM Computing Surveys*, **24**(2), 131–183. 1, 1.1
- Krueger, C. W. (2002). Data from salion’s software product line initiative. Technical Report 2002-07-08-1, Biglever, US. 3.5.2
- Krueger, C. W. (2005). *Software Mass Customization*. Biglever, technical white paper edition. 3.5.1
-

- Krueger, C. W. (2007). Biglever software gears and the 3-tiered spl methodology. In *Object oriented programming systems and applications (OOPSLA)*, pages 844–845, Montreal, Quebec, Canada. ACM. 3.5.1
- Krut Jr, R. W. (1993). Integrating 001 tool support in the feature-oriented domain analysis methodology. Technical Report CMU/SEI-93-TR-11, ESC-TR-93-188, SEI. 3.5.1
- Lee, J. and Kang, K. C. (2004a). *Software Product-Family Engineering*, volume 3014 of *Lecture Notes in Computer Science (LNCS)*, chapter Feature Binding Analysis for Product Line Component Development, pages 266–276. Springer Berlin / Heidelberg. 6.3
- Lee, J. and Muthig, D. (2006). Feature-oriented variability management in product line engineering. *Communications of the ACM*, **49**(12), 55–59. 6.3
- Lee, K. and Kang, K. C. (2004b). Feature dependency analysis for product line component design. In J. Bosch and C. Krueger, editors, *8th International Conference on Software Reuse (ICSR)*, pages 69–85, Madrid, Spain. 6.3
- Lee, K., Kang, K. C., Chae, W., and Choi, B. W. (2000). Feature-based approach to object-oriented engineering of applications for reuse. *Software Practice and Experience*, **30**(9), 1025–1046. 9, 4.4.1
- Lehman, M. M. M. (1987). Process models, process programs, programming support. In *ICSE '87: Proceedings of the 9th international conference on Software Engineering*, pages 14–16, Monterey, California, United States. IEEE Computer Society Press. 2.2
- Leich, T., Apel, S., Marnitz, L., and Saake, G. (2005). Tool support for feature-oriented software development: featureide: an eclipse-based approach. In *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*, pages 55–59, San Diego, California. ACM. 3.5.1
- Lim, W. (1994). Effects of reuse on quality, productivity and economics. *IEEE Software*, **11**(5), 23–30. 1.1
- Lisboa, L. B., Garcia, V. C., Almeida, E. S., and Meira, S. L. (2007). Toolday - a process-centered domain analysis tool. In *Brazilian Symposium on Software Engineering (SBES) - Tools Session*, pages 54–60, João Pessoa, Paraíba, Brazil.

- Lisboa, L. B., Nascimento, L. M., Almeida, E. S., and Meira, S. R. L. (2008a). A case study in software product lines: An educational experience. In *21st IEEE Conference on Software Engineering Education and Training (CSEET)*, pages 155–162, Charleston, South Carolina. 4.5
- Lisboa, L. B., Lucrédio, D., Garcia, V. C., and Almeida, E. S. (2008b). A systematic review on domain analysis tools. Technical Report 20080121-0101, C.E.S.A.R./RiSE.
- Lucrédio, D., Fortes, R. P. M., Almeida, E. S., and Meira, S. R. L. (2008a). Performing domain analysis for model-driven software reuse. In Springer-Verlag, editor, *10th International Conference on Software Reuse (ICSR), Lecture Notes in Computer Science*, Beijing, China. 1.1
- Lucrédio, D., Brito, K. S., Alvaro, A., Garcia, V. C., Almeida, E. S., Fortes, R. P. M., and Meira, S. R. L. (2008b). Software reuse: The brazilian industry scenario. *Journal of Systems and Software (JSS)*, **81**, 996–1013. 2.1
- Martins, A. C., Garcia, V. C., Almeida, E. S., and Meira, S. R. L. (2008). Enhancing components search in a reuse environment using discovered knowledge techniques. In *2nd Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS)*, Porto Alegre, Brazil. 1.3.1
- Mascena, J. C. C. P. (2006). *ADMIRE: Asset Development Metric-based Integrated Reuse Environment*. Master’s thesis, Federal University of Pernambuco, Recife, Pernambuco, Brazil. 1.3.1
- Massen, T. v. d. and Lichter, H. (2003). Requiline: A requirements engineering tool for software product lines. In *International Workshop on Product Family Engineering (PFE)*, Siena, Italy. Springer Verlag. 3.5.1, 3.5.2
- Massen, T. v. d. and Lichter, H. (2004). Deficiencies in feature models. In *Workshop on Software Variability Management for Product Derivation - SPLC’04*, Boston, EUA. 4.4.1
- McIlroy, M. D. (1968). Software engineering: Report on a conference sponsored by the nato science committee. In N. S. A. Division, editor, *NATO Software Engineering Conference*, pages 138–155. 1.1
- Mei, H., Zhang, W., and Gu, F. (2003). A feature oriented approach to modeling and reusing requirements of software product lines. In *International Conference*
-

- on Computer Software and Applications (COMPSAC)*, pages 250–256, USA. IEEE Computer Society. 1.1
- Mendes, R. C. (2008). *Search and Retrieval of Reusable Source Code using Faceted Classification Approach*. Msc. dissertation, Federal University of Pernambuco, Recife, Pernambuco, Brazil. 1.3.1
- Mernik, M., Heering, J., and Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys*, **37**(4), 316–344. 1.4
- Moon, M., Yeom, K., and Chae, H. S. (2005). An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. *IEEE Transactions on Software Engineering*, **31**(7), 551–569. 1.1, 1.3.2, 3, 3.5.1, 3.5.2, 6
- Nascimento, L. M. (2008). *Core Assets Development in SPL - Towards a Practical Approach for the Mobile Game Domain*. Master's thesis, Federal University of Pernambuco, Recife, Pernambuco, Brazil. 1.3.1
- Neighbors, J. (1981). *Software Construction Using Components*. Ph.D. thesis, University of California, US. 3, 2
- Oakes, K. S., Smith, D., and Morris, E. (1992). Guide to case adoption. Technical Report CMU/SEI-92-TR-015, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213. 2.1
- Padmanabhan, P. (2002). *DECIMAL: A Requirements Engineering Tool for Product Families*. Ph.D. thesis, Iowa State University, US. 3.5.1, 3.5.2
- Padmanabhan, P. and Lutz, R. R. (2005). Tool-supported verification of product line requirements. *Automated Software Engineering*, **12**(4), 447–465. 3.5.1
- Park, J., Moon, M., and Yeom, K. (2004). Dream: Domain requirement asset manager in product lines. In *International Symposium on Future Software Technology (ISFST)*, Xian, China. 3.5.1
- Postech (2003). *ASADAL / FORM Tool. User guide*. Software Engineering Laboratory. 3.5.1
- Pressman, R. S. (2001). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 5th edition. 2.1, 2.2, 2.2.2, 2.1
-

- Prieto-Díaz, R. (1990). Domain analysis: an introduction. *ACM SIGSOFT Software Engineering Notes*, **15**(2), 47–54. 6.3
- Prieto-Díaz, R., Frakes, W. B., and Gogia, B. (1992). Dare - a domain analysis and reuse environment. Technical report, Reuse, Inc. 3.5.1, 3.5.2
- Pure-systems (2004). *Variant Management with pure::variants*. Pure-systems. 3.5.1
- RWTH-Aachen (2005). *The RequiLine User Manual*. Germany. 3.5.1
- Santos, E. C. R., Durão, F. A., Martins, A. C., Mendes, R., Melo, C., Garcia, V. C., Almeida, E. S., and Meira, S. R. L. (2006). Towards an effective context-aware proactive asset search and retrieval tool. In *6th Workshop on Component-Based Development (WDBC)*, pages 105–112, Recife, Brazil. 1.3.1
- Schmid, K. and Schank, M. (2000). Pulse-beat – a decision support tool for scoping product lines. In *Software Architectures for Product Families, International Workshop IW-SAPF-3*, pages 65–75, Las Palmas de Gran Canaria, Spain. 3.5.1, 4.4.1
- Sinnema, M. and Deelstra, S. (2007). Classifying variability modeling techniques. *Information and Software Technology*, **49**, 717–739. 3.1
- Sommerville, I. (2007). *Software Engineering*. Addison Wesley, Harlow, 8 edition. 1.4, 2, 2.2, 2.2.1, 4
- Spinczyk, O. and Beuche, D. (2004). Modeling and building software product lines with eclipse. In *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, pages 18–19, Vancouver, BC, CANADA. ACM. 3.5.1
- Steger, M., Tischer, C., Boss, B., Müller, A., Pertler, O., Stolz, W., and Ferber, S. (2004). Introducing pla at bosch gasoline systems: Experiences and practices. In *Software Product Lines (SPLC)*, pages 34–50, Boston, MA, USA. 3.5.2
- Succi, G., Eberlein, A., Yip, J., Luc, K., Nguy, M., and Tan, Y. (1999). The design of holmes: a tool for domain analysis and engineering. In *IEEE Pacific Rim Conference*, pages 365–368, Victoria, BC, Canada. 3.5.1, 3.5.2
- Succi, G., Yip, J., and Liu, E. (2000a). Analysis of the essential requirements for a domain analysis tool. In *ICSE 2000 Workshop on Software Product Lines: Economics, Architectures and Implications*. 1.1, 1.3.2, 3, 3.1, 6
-

- Succi, G., Yip, J., Liu, E., and Pedrycz, W. (2000b). Holmes: a system to support software product lines. In *International Conference on Software Engineering (ICSE)*, page 786, Limerick, Ireland. ACM. 3.5.1
- Succi, G., Yip, J., and Pedrycz, W. (2001a). Holmes: An intelligent system to support software product line development. In *International Conference on Software Engineering (ICSE)*, pages 829–830, Toronto, Ontario, Canada. 3.5.1
- Succi, G., Pedrycz, W., yip, J., and Kaytazov, I. (2001b). Intelligent design of product lines in holmes. In *Electrical and Computer Engineering*, volume 1, pages 75–80, Canada. 3.5.1
- Tracz, W. and Coglianese, L. (1995). A dssa domain analysis tool. Technical Report ADAGE-LOR-94-13, Loral Federal System, Owego, US. 3.5.1
- Travassos, G. H. and Biolchini, J. (2007). Systematic review applied to software engineering. In *Brazilian Symposium on Software Engineering (SBES) - Tutorials*, page 436, João Pessoa, Brazil. 3.2, 3.4.1
- Vanderlei, T. A., Durão, F. A., Martins, A. C., Garcia, V. C., Almeida, E. S., and Meira, S. R. L. (2007). A cooperative classification mechanism for search and retrieval software components. In *22nd Annual ACM Symposium on Applied Computing, Information Access and Retrieval (IAR) Track*, pages 866–871., Seoul, Korea. ACM Press. 1.3.1
- Weiss, D. and Lai, C. T. R. (1999). *Software Product-Line Engineering: A Family-Based Software Development Process*. Addison-Wesley. 1.1
- Weiss, D. M. (1998). Commonality analysis: A systematic process for defining families. In *Proceedings of the Second International ESPRIT ARES Workshop on Development and Evolution of Software Architectures for Product Families*, pages 214–222, London, UK. Springer-Verlag. 3.6.1
- Wohlin, C., Runeson, P., Host, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers. 5.1.1, 5.1.1, 5.1, 5.1.2, 5.1.3
- Zetie, C. (2005). Eclipse has won: What's next for eclipse? www.forrester.com/Research/Document/Excerpt/0,7211,36165,00.html. 4.3
-



Journals and Conferences

The target journals were:

- ACM Computing Survey
- Annals of Software Engineering
- Automated Software Engineering
- IEEE Software
- IEEE Transactions on Software Engineering
- Journal of Systems and Software
- Software Process: Improvement and Practice, and
- Software Practice and Experience

And the target conferences were:

- Computer Software and Applications Conference (COMPSAC)
- Generative Programming and Component Engineering (GPCE)
- International Conference on Software Engineering (ICSE)
- International Conference on Software Reuse (ICSR)
- Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)
- Software Product-Family Engineering (PFE)
- Software Product Line Conference (SPLC)
- Variability Modeling of Software-intensive Systems (VaMoS)

B

Evaluation Feedback Form

This appendix presents the questionnaires used in the experimental studies.

B.1 First Feedback Form

This questionnaire was used in the first experiment.

A - Personal Experience

1 - How do you consider your knowledge in Eclipse platform?

- a) ☐ Very good
- b) ☐ Good
- c) ☐ Basic
- d) ☐ Low

2 - In how many projects, with Eclipse platform, have you participated at?

B - Domain Analysis Knowledge

3 - Do you know any domain analysis process?

- a) ☐ Yes
- b) ☐ No - Jump to question 5

In case of answering “Yes”, which processes do you know?

- a) ☐ Feature-Oriented Domain Analysis (FODA)

- b) ☐ Product Line Software Engineering (PuLSE)
- c) ☐ Domain Analysis Reuse Environment (DARE)
- d) ☐ RiSE Domain Engineering Process (RiDE)
- e) ☐ FAST
- f) Other: _____

4 - How many projects have you participated in?

- a) ☐ 1-2 commercial projects
- b) ☐ More than 2 commercial projects
- c) ☐ More than 2 academic projects
- d) ☐ Self-Learning
- e) ☐ Not applied

5 - In case you have already participated in domain analysis project, what was your role?

- a) ☐ Manager
- b) ☐ Domain Specialist
- c) ☐ Domain Analyst
- d) ☐ System Engineer
- e) ☐ Architect
- f) Other: _____

C - Tool Usage

6 - Was there any difficult in installing the tool?

- a) ☐ Yes
- b) ☐ No

In case of answering “Yes”, which ones:

7 - Have you used the tutorial?

- a) ☐ Yes
- b) ☐ No - Jump to question 9

8 - Was the tutorial enough for you to learn how to use the tool?

a) ☐ Yes

b) ☐ No

In case of answering “No”, describe the difficult:

a) ☐ Sequence of steps is not clear

b) ☐ Incomplete information

c) ☐ Not all functions are described

d) Other: _____

Describe how and what can be improved:

9 - In the artifacts validation (domain, models and products), when a problem is found, the tool provides some messages errors. Do these messages aid in identifying the problem?

a) ☐ Yes

b) ☐ No

Describe examples when they did not help:

10 - The tool offers several customizations through the *Preferences in Menu Windows*. Were the customizations used?

a) ☐ Yes

b) ☐ No - Jump to question 12

11 - Were the customization helpful to customize the tool according to the project's need?

a) ☐ Yes

b) ☐ No

In case of answering “No”, describe which customizations are necessary:

D - Modules Usage

12 - Were there any difficult during the execution of some activity in the tool?

- a) ☐ Yes
b) ☐ No - Jump to question 14

13 - Describe which type(s) of difficult was(were) found:

- a) ☐ Hard navigation
b) ☐ No knowledge about the process
c) ☐ Lack or inefficiency explanation to use the tool
d) Other: _____

Detail the difficult found:

14 - Which was the major difficult concerning the execution of the domain analysis process with the tool?

E - Opinion

15 - Do you consider that tool aided the domain analysis process?

- a) ☐ Yes
b) ☐ No

Why (For both answers):

16 - What are the positive points of using the tool?

17 - What are the negative points of using the tool?

B.2 Second Feedback Form

This following questionnaire was used in the second experiment.

A - Personal Experience

1 - How do you consider your knowledge in Eclipse platform?

- a) ☐ Very good
- b) ☐ Good
- c) ☐ Basic
- d) ☐ Low

B - Domain Analysis Knowledge

2 - Do you know any domain analysis process?

- a) ☐ Yes
- b) ☐ No - Jump to question 5

In case of answering “Yes”, which processes do you know?

- a) ☐ Feature-Oriented Domain Analysis (FODA)
- b) ☐ Product Line Software Engineering (PuLSE)
- c) ☐ Domain Analysis Reuse Environment (DARE)
- d) ☐ RiSE Domain Engineering Process (RiDE)
- e) ☐ FAST
- f) Other: _____

3 - How many project have you participated in?

- a) ☐ 1 project
- b) ☐ 2 projects
- c) ☐ 3-4 projects
- d) ☐ 5 or more projects
- e) ☐ None

4 - In case you have already participated in domain analysis project, what was your role?

- a) ☐ Manager
- b) ☐ Domain Specialist
- c) ☐ Domain Analyst
- d) ☐ System Engineer
- e) ☐ Architect
- f) Other: _____

C - Tool Usage

5 - The existing tutorial was sufficient to learn how to use the tool?

- a) ☐ Yes
- b) ☐ No
- c) ☐ The tutorial was not used

In case of answering “No”, describe the difficult:

- a) ☐ Sequence of steps is not clear
- b) ☐ Incomplete information
- c) ☐ Not all functions are described
- d) Other: _____

Describe how and what can be improved:

6 - In the artifacts validation (domain, models and products), when a problem is found, the

tool provides some messages errors. Do these messages aid in identifying the problem?

- a) ☐ Yes
- b) ☐ No
- c) ☐ Sometimes

Give examples when they did not help:

7 - The tool offers several customizations through the *Preferences in Menu Windows*. Were the customization helpful to customize the tool according to the project's and/or user's need?

- a) ☐ Yes
- b) ☐ No
- c) ☐ The customizations were not used

In case of answering "the customizations were not used", select the reasons for it.

- a) ☐ Not necessary, the standard was sufficient
- b) ☐ Could not apply the customizations
- c) ☐ The existing customizations were not the expected ones
- d) Others: _____

Describe which customizations are necessary:

D - Modules Functionalities

Planning Module

8 - Was there any difficult during the planning execution?

- a) ☐ Yes
- b) ☐ No - Jump to question 11
- c) ☐ The module was not used

B.2. SECOND FEEDBACK FORM

In case your answer was “The module was not used”, describe why it happened:

9 - Describe which type(s) of difficult was(were) found:

- a) ☐ Hard navigation
- b) ☐ No knowledge about the process
- c) ☐ Lack or inefficiency explanation to use the tool
- d) Other: _____

Detail the difficult found:

10 - This module fulfillment brought any benefit to the project?

- a) ☐ Yes
- b) ☐ No

In case you answered “Yes”, detail the benefits:

Modeling Module

11 - Was there any difficult during this module execution?

- a) ☐ Yes
- b) ☐ No - Jump to question 14
- c) ☐ The module was not used

In case your answer was “The module was not used”, describe why it happened:

12 - Describe which type(s) of difficult was(were) found:

- a) ☐ Hard navigation
- b) ☐ No knowledge about the process
- c) ☐ Lack or inefficiency explanation to use the tool
- d) Other: _____

Detail the difficult found:

13 - This module fulfillment brought any benefit to the project?

- a) ☐ Yes
- b) ☐ No

In case you answered “Yes”, detail the benefits:

Validation Module

14 - Was there any difficult during this module execution?

- a) ☐ Yes
- b) ☐ No - Jump to question 17
- c) ☐ The module was not used

In case your answer was “The module was not used”, describe why it happened:

15 - Describe which type(s) of difficult was(were) found:

- a) ☐ Hard navigation
- b) ☐ No knowledge about the process

c) () Lack or inefficiency explanation to use the tool

d) Other: _____

Detail the difficult found:

16 - This module fulfillment brought any benefit to the project?

a) () Yes

b) () No

In case you answered “Yes”, detail the benefits:

Product Derivation Module

17 - Was there any difficult during this module execution?

a) () Yes

b) () No - Jump to question 20

c) () The module was not used

In case your answer was “The module was not used”, describe why it happened:

18 - Describe which type(s) of difficult was(were) found:

a) () Hard navigation

b) () No knowledge about the process

c) () Lack or inefficiency explanation to use the tool

d) Other: _____

B.2. SECOND FEEDBACK FORM

Detail the difficult found:

19 - This module fulfillment brought any benefit to the project?

a) ☐ Yes

b) ☐ No

In case you answered “Yes”, detail the benefits:

E - Opinion

20 - Do you consider that tool aided the domain analysis process?

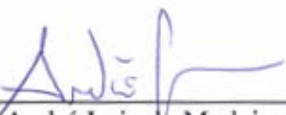
a) ☐ Yes

b) ☐ No

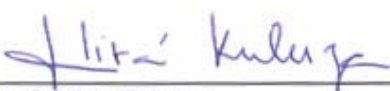
Why (For both answers):

21 - Improvements suggestions:

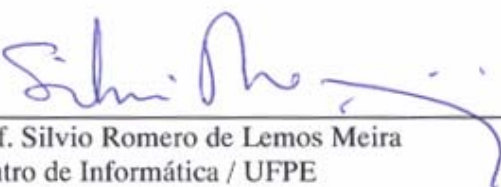
Dissertação de Mestrado apresentada por **Liana Barachisio Lisboa** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**ToolDay – A Tool for Domain Analysis**", orientada pelo **Prof. Silvio Romero de Lemos Meira** e aprovada pela Banca Examinadora formada pelos professores:



Prof. André Luis de Medeiros Santos
Centro de Informática / UFPE

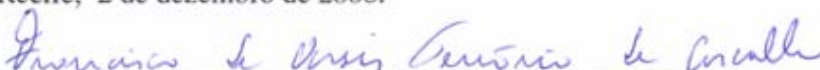


Prof. Uira Kulesza
Departamento de Informática e Matemática Aplicada / UFRN



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

Visto e permitida a impressão.
Recife, 2 de dezembro de 2008.



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