Information and Software Technology 52 (2010) 1–13

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A systematic review of domain analysis tools

Liana Barachisio Lisboa [a,*], Vinicius Cardoso Garcia [a,b,d], Daniel Lucrédio [a,c], Eduardo Santana de Almeida [a,e], Silvio Romero de Lemos Meira [a,b], Renata Pontin de Mattos Fortes [c]

[a] RiSE, Reuse in Software Engineering, Recife, PE, Brazil
[b] Informatics Center, Federal University of Pernambuco, Recife, PE, Brazil
[c] Institute of Mathematical and Computer Science - University of São Paulo (ICMC/USP), São Carlos, SP, Brazil
[d] Caruaru Faculty of Science and Technology (FACITEC), Pernambuco University, Caruaru, PE, Brazil
[e] Federal University of Bahia, Salvador, BA, Brazil

ARTICLE INFO

Article history:
Received 12 September 2008
Received in revised form 15 May 2009
Accepted 16 May 2009
Available online 31 May 2009

Keywords:
Systematic review
Domain analysis
Tools

ABSTRACT

The domain analysis process is used to identify and document common and variable characteristics of systems in a specific domain. In order to achieve an effective result, it is necessary to collect, organize and analyze several sources of information about different applications in this domain. Consequently, this process involves distinct phases and activities and also needs to identify which artifacts, arising from these activities, have to be traceable and consistent. In this context, performing a domain analysis process without tool support increases the risks of failure, but the used tool should support the complete process and not just a part of it. This article presents a systematic review of domain analysis tools that aims at finding out how the available tools offer support to the process. As a result, the review identified that these tools are usually focused on supporting only one process and there are still gaps in the complete process support. Furthermore, the results can provide insights for new research in the domain engineering area for investigating and defining new tools, and the study also aids in the identification of companies' needs for a domain analysis tool.

Contents

* Corresponding author. Tel.: +55 713 346 4855.
E-mail address: lblisboa@gmail.com (L.B. Lisboa).

0950-5849/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2009.05.001

1. Introduction

Nowadays, companies are seeking for ways to improve their competitiveness, which involves less time-to-market and high quality for the products. The adoption of software reuse [1] is an option to obtain these benefits. Although the benefits of software reuse are promising; it is a complex task to put it into practice. A way to maximize these possible benefits is through the adoption of a systematic reuse approach, which is domain focused, based on a repeatable process, and concerned with reuse of higher level life cycle artifacts [2].

Reuse can be obtained in all kinds of assets, such as requirements and use cases, architecture, frameworks. However, the most common is the 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 [3]. This can be obtained through a domain analysis (DA), which identifies common and variable characteristics of systems in a specific domain.

To achieve an effective result with this process, it is necessary to collect, organize and analyze several sources of information about different applications in the domain. This task involves distinct phases and activities. The analysis of the existing products from the domain and their correlation, to identify the domain scope, is one example. These activities involve the management of interrelated artifacts that have to be kept traceable and consistent. Due to this, using only human expertise in industrial projects without automation can contribute to risks in a project, such as incomplete information, lack of support for information management, project delays, or degraded productivity. Thus, it is necessary to have tool support to aid the organization's domain analyst during the execution of the process [4–6].

Furthermore, tool support should assist the complete process, and not just some functionalities, because it would lead to the need to use several tools and information traceability among them would, probably, have to be done manually by the domain analyst. Therefore, it is necessary to evaluate the current domain analysis tools in order to verify the degree to which they support the process. This article presents a systematic review of domain analysis tools conducted to address this goal.

This review also provides a starting point either for new research aimed at developing new DA tools, or for investigating the processes that follow domain analysis and make use of DA outputs (e.g. domain design, implementation and application engineering). In addition, the review should benefit companies interested in purchasing a DA tool.

2. Domain analysis concepts

Domain analysis is part of the 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 [7], and also encompasses domain design and domain implementation. The results of domain engineering will be reused in application engineering, which is the process of producing systems with the reusable assets developed during the domain engineering.

The term *Domain Analysis* was first defined by Neighbor's [8] as "*the activity of identifying objects and operations of a class of similar systems in a particular problem domain*", where he compares single system analysis with domain analysis. The main point is that single systems analysis is concerned with actions and functionalities of just one system while domain analysis is concerned with actions and functionalities of several systems within an application area [9].

Nowadays, several domain analysis processes have been created – such as [10–14,6,15,16], providing different guidelines for identifying common and variable characteristics of a domain. In 1994, Arango [17] performed a study of eight existing domain analysis processes. He presented, as his results, a set of activities that are common to the analyzed processes:

- **Domain scoping.** Feasibility analysis and planning of the domain.
- **Data collection.** Data collection from different sources, which can vary from experts to documents.
- **Data analysis.** Descriptions of reusable elements, identifying the similarities and differences between them.
- **Classification.** Refinement of the information, clustering similar descriptions, abstracting relevant common features from the descriptions in each cluster and a vocabulary construction.
- **Evaluation of the domain model.** Domain model evaluation, and correction of any defects found.

Even though this evaluation was performed 15 years ago, our review shows that the common activities are still relevant.

The next section describes some previous studies of domain analysis tools.

3. Related work

Analysis of existing domain analysis tools has been performed in previous studies, such as the ones detailed below.

Succi et al. [5] define some requirements for a domain analysis tool, focusing on the functionalities that a tool should possess in order to have a consistent environment, such as traceability, consistency checking and tool integration. Later, they discuss how existing domain analysis tools satisfy these requirements.

Gomaa and Shin [18] propose a set of requirements for effective variability management in product lines. These define some views that are responsible for specific functionalities, 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 domain and application engineering processes. On the other hand, these requirements are not tested on other tools, just on their prototype.

However, these studies need updating and they focus on analyzing only a pre-defined set of requirements. Furthermore, other studies have looked at restricted aspects of the process, such as variability modeling. These studies are described next.

Capilla et al. [19] describe a set of concepts for variability modeling and management, and analyze several existing tools based on these concepts. Their analysis focused on the complete software

product line and domain engineering processes. As a 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 [20] define criteria for comparing variability modeling notations and techniques and other criteria for analyzing the tools. These were selected based on the process they supported. However, the results focus on the techniques and not on the tool support.

Even though several tools were analyzed in these studies, they were not the studies' focus; in addition, these analyses were also performed according to a pre-defined set of requirements.

4. Motivation

The available analyses of domain analysis tools - described in Section 3 – did not intend to find out how the existing domain analysis tools support the domain analysis process. Instead, they define a set of functionalities and analyze the tools according to them.

Another characteristic of previous studies is that they do not explain how the requirements were defined, if these requirements are sufficient to fulfill a complete domain analysis process, nor if they are focused just on a few functionalities, nor how the tools were selected. In addition, the studies are usually described as a way of introducing a new tool, like [5,18].

Therefore, this review intends to identify how the available tools support the DA process in a systematic way, in order to discover if the available tools are fully supporting the process or if there are still gaps.

5. Method

There are several reasons to perform a systematic review, and the usual ones are [21,22]:

- 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.

Based on the motivation described above, the second reason fits the purpose of this review. Furthermore, the results achieved with the review can also provide a background for new researchers interested in domain analysis tools or work products to be expected in the following phases of the process (domain design and implementation).

The systematic review described here was based on Kitchenham and Charters' [21] guidelines, which is 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 report.

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 [21]. Furthermore, as it is described next, some of these stages were adapted from Kitchenham and Charters' definition for more complete results.

5.1. Research question

As described before, the objective of this review is to find out **how the available tools for domain analysis are providing support to the process**. However, this question is too generic for a complete evaluation. Thus, it was further divided into sub-questions (SQ), focusing on 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 adopting a tool, since an 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 is concerned with how the tool supports the process guidelines, identifying its characteristics. The analysis did not focus on the strengths nor on the weaknesses 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 used. Through these descriptions, it is possible to map the current adoption of the tools.

According to the systematic review process [21], the question structure is divided in four aspects (PICO):

- **Population**: people, projects and application types affected by the intervention. They should be directly related to the question;
- **Intervention**: software technology, tool or procedure, which generates the outcomes;
- **Comparison**: some other type of intervention – if applicable – and,
- **Outcomes**: technology impact in relevant information terms for practical professionals.

And, the structure for this research question is:
Population: this question refers to the support of the domain analysis tools to the process. 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 the development of 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 and if the process is not fully supported, i.e. there are many gaps in the existing tools, or if there is a necessity of using several tools in the whole process, in which cases the necessity for a new DA tool increases.

5.2. Search strategy

Based on the structure and the research question, some keywords were extracted and used to search the primary study sources. The initial set of keywords was: *domain analysis*, *tool* and *domain engineering*. However, after a preliminary search, which aimed at finding available systematic reviews and assessing the volume of potentially relevant studies [21], 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 results, like *software product line* and *requirements*.

Furthermore, sophisticated search strings could then be constructed using boolean AND and OR operators. Since the review aims at identifying tools, the keyword *tool* was included in every

search string, because in the preliminary searches several results referred only to 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 based on the following digital libraries: ACM Digital Library,[1] IEEE Computer Society Digital Library,[2] Science@Direct[3] and Springer Link.[4] These searches had as target some journals and conferences, which are detailed in Appendix A, but if relevant results from different journals or conferences were found, they were not discarded. These libraries were chosen because they are some of the most relevant sources in software engineering [23,22].

Each search string was used in all digital libraries, and the strings results in each digital library are detailed in the web material at http://www.rise.com.br/research/personal/liana/systematicreview.

Additionally, 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 web search engines, such as Google. In the web engines, the search was targeted at tools information and their grey literature (i.e. technical reports, white papers, manuals and works in progress).

Websites of researchers and research groups active in this area, such as Alexandria,[5] Fraunhofer IESE,[6] RWTH Aachen[7] and SEI,[8] were also reviewed. In addition to the search in digital libraries and web engines, the references of the primary studies were also read in order to identify other tools.

Even though the contact with research groups and the use of web search engines depart from a systematic process – thus making it harder to replicate the review – they were necessary in order to have a more complete set of information and references about the tools.

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, especially for identifying the functionalities. These executables, sometimes, had their download link available in research papers, while others were found through a search in web engines and some were found through research groups' websites or through the direct contact with the tool's developers.

After obtaining the documentation about all 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.

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

[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.theoinf.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.

Table 1
Search documentation process.

Data source	Documentation
Digital library	Library name Search string Date
Research group	Group name Url Date

At the end of the search, a total of **31** potentially relevant tools were selected, of which **four** only had the information available on the websites and their executables (i.e. no written reports).

These searches were performed in January, 2008 by a M.Sc. and a Ph.D. students; the achieved results were crossed and then validated. All the results of the search process are documented in the web material. Therefore, it is clear to others how thorough the search was, and how they can find the same documents.

The adopted documentation template for the results is shown in Table 1.

5.3. Study selection

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

Since the goal of the review was on the tools, all the available information achieved in the search – i.e. executables, papers, grey literature and journals – were grouped according to the tool they referred to. 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 inclusion criteria were:

(a) **Tools that offer support to the domain analysis phase**: the encountered tools must support some functionality in the DA process.[9] If the tool supports functionalities that are not from the DA process, these functionalities will not be evaluated, but the tool might still be selected.
(b) **Tools with available executables**: with its prototype, demo or finalized product, it is possible to test how the tool realizes the process.
(c) **Tools with documentation describing their functionalities**: if the written documentation of the tool's functionalities was clear and useful, they were considered too.

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

The exclusion criteria were:

(a) **Tools supporting functionalities that are from steps after domain analysis**: tools supporting only 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.

[9] The 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.

(b) **Tools supporting only variability management**: tools focusing 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 [8].
(c) **Tools with written documentation with no usable information**: tools with written documentation that did not have usable description about its functionalities were discarded.
(d) **Tools describing only a few functionalities**: tools with only written documentation that did not focus on explaining the supported process and the functionalities, but only a singular functionality, were also discarded.

Also, tools with only white papers could not be selected, because they usually contain just marketing material, therefore another data source should exist in other to validate their information.

5.4. 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 [21]. The form for data extraction provides some standard information:

- Name of the tool;
- 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 template of the form can be found in the web material.[10]

Moreover, the form contains a specific space for each 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 generic one. SQ2 is answered with a list of the identified functionalities, while SQ3 was further divided with the information regarding the name and type of the institution that developed the tool, and where it was used.

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

5.4.1. Data extraction procedure

Every decision about including/excluding 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.

One problem faced during the review was that, sometimes, there were contradictions among the available information of the tools. This happened when a tool had different sources with some years between them. In this case the oldest sources were discarded.

In cases when there was no 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 analyzed the available information about the tool and discussed with the other reviewers in order to achieve a consensus.

[10] http://www.rise.com.br/research/personal/liana/systematicreview.

Table 2
Number of tools selected in each category.

Category	Number of tools selected
Written documentation + executable + websites	5
Written documentation + websites	3
Written documentation + executables	1
Written documentation	7
Executable + websites	3

This review protocol was revised by the Reuse in Software Engineering (RiSE)[11] group in its weekly discussions and seminars. 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.

6. Results

Based on the search results and on the inclusion and exclusion criteria, a set of tools were selected, as described below.

6.1. Tools selection

The tool selection process was performed by a M.Sc., two Ph.D. students and a Ph.D. in conjunction with the RiSE group. It intended to improve the tools' questionnaires completion and analysis.

After the analyses of the inclusion and exclusion criteria, from the 31 potentially relevant tools, **19** were selected. Among these tools, **36** written reports, **9** executables and **11** 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 2.

A brief description about the selected tools is presented below in alphabetical order and the available information about each tool is depicted in Table 3.

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

Ami Eddi. Ami Eddi 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, US. The tool is currently in its third prototype.

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

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 Telelogic's DOORS.[12] 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.

[11] http://www.rise.com.br/research.
[12] http://www.telelogic.com/products/doors/index.cfm.

Table 3
The analyzed information of each tool.

Name	Documentation	Executable	Website
001	[24,25]	No	http://world.std.com/hti/Product/Product.htm
Ami Eddi	No	Yes	http://www.generative-programming.org
ASADAL	[26,27]	Yes	No
CaptainFeature	No	Yes	https://sourceforge.net/projects/captainfeature
DARE	[28–30,10]	No	No
DECIMAL	[31–33]	No	No
Domain	[34]	No	No
Doors Extension	[35]	No	No
DREAM	[36,6]	No	No
Feature Plugin	[37,38]	Yes	http://gsd.uwaterloo.ea/projects/fmp-plugin/
FeatureIDE	[39]	Yes	http://wwwiti.cs.uni-magdeburg.de/iti_ib/research/featureide/
GEARS	[40–42]	No	http://www.biglever.com
Holmes	[43–46]	No	http://world.std.com/hti/Product/Product.htm
Odyssey	[47,48]	Yes	http://reuse.cos.ufrj.br/site/pt/index.php
Pluss Toolkit	[49–51]	No	No
PuLSE-BEAT	[52]	No	No
Pure::Variants	[53–55]	Yes	http://www.pure-systems.com
RequiLine	[56,57]	Yes	http://www-lufgi3.informatik.rwth-aachen.de/TOOLS/requiline/
XFeature	No	Yes	http://www.pnp-software.com/XFeature/

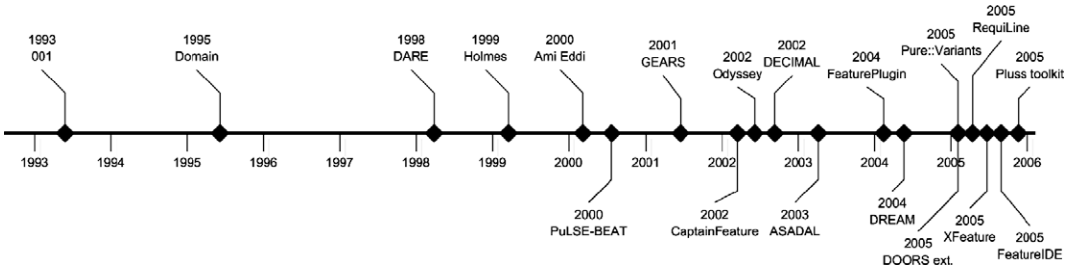

Fig. 1. Selected tools timeline.

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 Umes University and Land Systems HSgglunds, 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 2003, 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).

Fig. 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.

6.2. Research question result

After the selection and data extraction of the tools, each tool analysis was confronted with the research question and similar conclusions about the data were grouped.

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 subquestions, each one of them has its conclusion presented first. After the results of the SQs, a summary relating to the main question is presented.

6.2.1. Domain analysis support

Based on the 19 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 correspondent to more than 78% of the total (15 tools, as shown in Table 4).

Among the tools supporting specific processes, five support the same process – the Feature Oriented Domain Analysis (FODA) process [58]. This process defines several activities for domain analysis, such as context model, feature model, entity-relationship,

Table 4
The type and name of the processes each tool supports.

Tool	Process support	Process name
001	Specific	FODA [58]
Ami Eddi	Specific	FODA [58]
ASADAL	Specific	FODA Extension - FORM [59]
CaptainFeature	Specific	FODA [58]
DARE	Specific	DARE [10]
Decimal	Specific	Based on FAST [11]
Domain	Specific	DSSA [60]
DOORS Extension	Specific	Adaptation of [61]
DREAM	Specific	DREAM [6]
Feature Plugin	Generic	Generative Programming
FeatureIDE	Generic	Feature-Oriented processes
GEARS	Specific	Own Process [41]
Holmes	Specific	Sherlock [62]
Odyssey	Specific	Odyssey DE [48]
Pluss toolkit	Specific	Pluss [49]
PuLSE-BEAT	Specific	PuLSE [12]
Pure::Variants	None	–
RequiLine	Specific	FODA [58]
XFeature	None	–

functional models and architectural models. Among these five tools, three support only the feature model 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 [59]).

Feature Plugin and FeatureIDE tools support generic processes. Their processes are based on a mix of several processes; however, FeatureIDE describes that the supported process has to be feature-oriented. Finally, Pure::Variants and XFeature do not specify which process they support, but they do offer support to an activity similar to feature modeling.

6.2.2. 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 DA process subdivision that some processes have [12,15]. In this review, the same names used in [15] were adopted. The groups are:

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

The functionality of each group is detailed next. Table 5 shows which functionalities each tool offers support to. The roman numbers refer to the functionalities described next and the grey columns separate each group of functionalities (Planning, Modeling, Validation and Extras, respectively). This table facilitates the identification of the gaps in the selected tools, and, in addition, it can help to discover which tool best satisfies the need of the company.

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 obtained from stakeholders and market analysis, constraints, objectives definition and data collection.
(ii) **Matrix of the domain**: represents the relationship between the characteristics of the domain, also called features, and the applications included in the domain. It is represented using rows and columns, where the former represents the

Table 5
Functionalities each tool supports.

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

features and the latter the applications. This matrix aids in the identification of which features are common or variable in the domain.

(iii) **Evaluation functions**: responsible for the process metrics. They are divided into characterization and benefit functions. The former evaluates characteristics for the applications, while the latter uses the results of the former to decide the best characteristics and best products for the domain analysis to cover [12].

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

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

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, trees and others. This model attempts to formalize the 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**: represent the features that will always be in the products.

(viii) **Composition rules**: create 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* [63].

(x) **Relationship types**: provides different types of relationships between the features. They can be *composition*, *generalization/specification* and *implementation*.

(xi) **Feature attributes**: 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 seen in Table 5, almost every tool supports the first four functionalities in this group; because of that, they were further classified as "**common functionalities**". This means that every domain analysis tool should always have them, even if the DA process being supported does not.

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 about the domain, consisting of: description, context, dependencies between systems in the domain, among others. Each tool provides a different set of fields for documentation.

(xiii) **Feature documentation**: provides documentation to every feature of the domain, with information such as: description; rationale; priorities, etc. Each tool provides a different set of fields for documentation.

(xiv) **Requirements management**: provides support for the inclusion of 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 maintain the traceability between the artifacts produced in the domain.

(xvi) **Dictionary**: identifies and defines the words commonly used in the domain and mostly found during pre-analysis.

(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 5), it was also classified as a **common functionality**.

Extra functionalities

In addition to the functionalities described in the previous groups, which are intrinsic to 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.

(xx) **Product documentation**: provides 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 the tool's users. However, they are not presented in Table 5, 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 its 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 other file types, 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 domain analysis. And the lack of training was the case of this review; therefore the tools with these requirements were easily analyzed.

Functionalities summary

In Table 5, apart from the **common functionalities**, the other identified functionalities are rarely implemented in the same tool.

In the planning group, the identified functionalities were available only in **4** tools, and the maximum number of functionalities supported by one tool is **3**. This suggests that many tools would benefit from additional functionalities.

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 [58]. The functionalities description details some of the variabilities encountered. Apart from the common functionalities, the remaining ones for the modeling group are also seldom supported; particularly the *feature group identification*, which is supported by a single tool.

Regarding the validation group, only one tool (*Ami Eddi*) does not support any functionality, in spite of this, not all functionalities are provided by a single tool. *RequiLine* provides most functionalities, but omits a *dictionary*. Otherwise, only **2** tools support more than half of the requirements for this group.

6.2.3. Tools development and usage

Most of the selected tools were developed in the academic environment – 12 – while **4** were developed exclusively in the industry and the remaining three tools were developed in both environments, academic and industrial, as shown in Fig. 2.

However, for the second part of this sub-question – *where the tools were used?* – there were some difficulties in finding this information for some tools. For several tools, especially the ones from the academia, this information was not available at all.

Since there was 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 6 shows in which environments the tools were used, and it includes the reports found confirming them. Whenever a dash appears in the report column, it means that the analyzed reports do not describe any use of the tool.

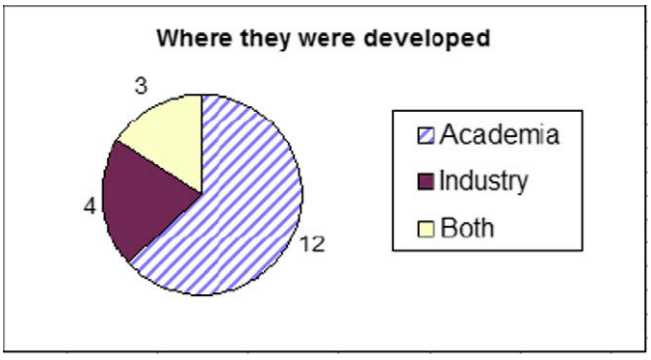

Fig. 2. Where the tools were developed.

Table 6
Where the tools were developed and used.

Tool	Developed	Used	Report
001	A	B	–
Ami Eddi	A	A	–
ASADAL	A	A	–
Captain Feature	A	A	–
DARE	I	B	[28,10]
DECIMAL	A	B	[31]
Domain	I	I	–
Doors Extension	A	B	[35]
DREAM	A	B	[6]
Feature Plugin	A	A	–
FeatureIDE	A	A	–
GEARS	I	I	[64]
Holmes	A	A	–
Odyssey	A	B	[48]
Pluss toolkit	B	B	[51]
PuLSE-BEAT	B	B	–
Pure::Variants	I	I	http://www.pure-systems.com
RequiLine	A	B	[56]
XFeature	B	B	–

The possible results for the *used* and the *developed* columns (Table 6) are: Academia (A), Industry (I) and Both (B).

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

6.2.4. 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 obliges a company that wishes to use the tool to modify or adapt the process it already uses. Thus, the learning curve and the impact it will cause to the company's development life cycle is higher.

Considering the functionalities in **SQ2**, the tools are mainly focused on the modeling phase due to the common functionalities (see Table 5). However, considering only the functionalities not classified as common, just three tools offer some kind of support to the modeling phase. That is similar to the support rate of the planning group functionalities, in which only the *domain matrix* is supported by all the tools with functionalities in this group. Regarding the common functionalities, from the 19 tools supporting the *domain representation* in the modeling phase, seven support this functionality using feature models.

Analyzing Table 5, 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 [65]. In addition, 13 tools support the *product derivation* – among the extra functionalities – for the domain, but only three offer support to *product documentation*.

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 [4], the majority of tools are still being developed and used in an 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 [43,66], tools are still too focused on academic issues, and this is not desirable if the tools are to achieve wider usage.

6.2.5. Discussion and implications

Even though the majority of tools support a specific process, analyzing the processes they support – along with the activities – the Feature Model activity from the FODA process [58] stands out. Excluding the 5 tools that support the FODA, two other tools also have activities similar to the feature model. Although these results suggest that the feature model aspect of FODA is generic, this does not imply that FODA itself is generic.

The idea that the feature model activity can be considered generic is reinforced by the number of tools that implement domain representation through feature models, as described in the previous section.

Other conclusion can be drawn when comparing the black dots in the validation group with the tools' development year (detailed in Section 6.1). It becomes clear that documentation functionalities are being more explored in more recent tools.

Furthermore, based on Table 5, it is clear that there are many functionalities that are supported by only a few tools. No tool provides complete coverage of all functionalities in each group, indicating that existing tools are functionally incomplete.

This lack of full support for DA has a potential impact on tool adoption. Companies wanting tool support for DA may be forced to use several different tools to automate the entire process. 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 saying that the existence of different tools during the process, without standardization on data format, increases staff effort and reduces data traceability and consistency, therefore delaying the project execution [67,68].

Another outcome for this review, based on the functionalities identified, is the definition of the priority to be given to different functionalities. The priorities for different functionalities are discussed below.

7. Functionalities priority

After identifying the functionalities that are supported by the tools, it was clear that not all of them are necessary for domain analysis. Therefore, the following priority categories were used to classify each functionality:

- **Essential**. It represents the indispensable and high-priority requirements that must be carried out. The lack of these functionalities means the application is unable to support domain analysis at all.
- **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 but are not essential for a useful DA tool.

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 Fig. 3. Therefore, the largest number of tools support should indicate that the functionality is mandatory in every domain analysis project.

In Fig. 3, the functionality "*relationship between feature and requirements*" had to be renamed to "*feature and requirements*" due to the lack of space in the graph.

Fig. 3 and the priority classifications provide a prioritized list of functionalities that should be useful for DA tool developers, researchers and companies looking for DA tool support.

7.1. 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 [17,29,69,70]. Additionally, 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 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 an *important priority*.

7.2. Important priority

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

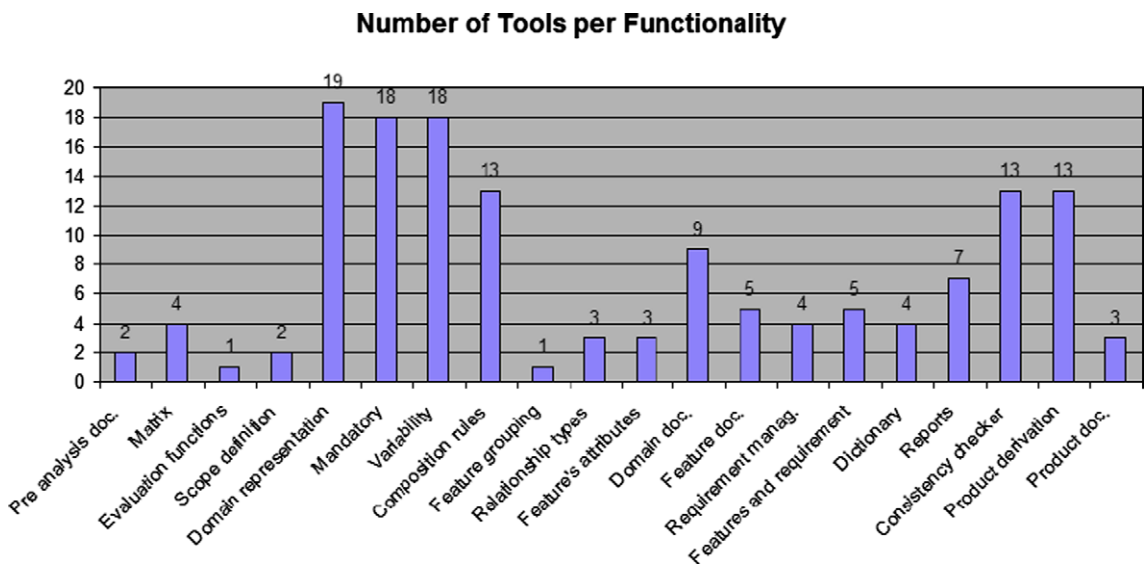

Fig. 3. Number of tools per functionalities.

Table 7
Functionalities each tool support per priority.

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

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 provide an easy-to-learn interface [71].

7.3. 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 have a major impact on the final artifacts of domain analysis.

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

7.4. Priority support per tools

After the definition of which tools support the functionalities and their priority, it was possible to compare the number of priority functionalities each tool delivers. This is depicted in Table 7, in which the tools are ordered according to the number of essential priorities they support. However, this table does not include usability functionalities, i.e. it has only the functionalities shown in Table 5.

The first two rows refer to the tools that fulfill more requirements – *RequiLine* and *Holmes*. Since this report did not focus on comparing how the functionalities are implemented, the evaluation only refers to the number of functionalities they support.

8. Threats to validity

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

Missing important primary studies. The search for the tools information was conducted in several digital libraries and on web search engines and it was focused not only on journals but also on conferences. According to [23,22], the selected digital libraries are some of the most relevant sources in software engineering. In addition, the web search engines were included in order to provide a more complete set of information and references about the tools, specially the ones in the industrial environment.

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 a strict search strategy described in Section 5. Furthermore, the defined criteria intended to select relevant tools to the domain analysis process support and not just tools supporting a few requirements. The only tools we have knowingly omitted are those with inadequate documentation (e.g. marketing white papers). So we do not believe that we have omitted any tools were capable of being analyzed.

Reviewers reliability. All the reviewers of this study are researchers in the software reuse field, focused on the domain engineering and software product line, and none of the tools information was written/developed by us. Therefore, we are not aware of any bias we may have introduced during the analyses, but it might be possible that the conclusions might have been affected by our personal interest and opinions.

Data extraction. Another threat for this review refers to how the data were extracted from the tools, since not all the information was obvious to answer the questions and some data had to be interpreted. Therefore, 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. Furthermore, in the event of a disagreement between the two primary reviewers, a third reviewer acted as an arbitrator to ensure full agreement was reached.

Functionality fulfillment. Even though the support of functionalities is the main goal of this review, it was not verified how they are fulfilled by the tools. In spite of that, the functionality descriptions include some of the ways that they vary among the tools, such as: the *domain representation* can be implemented through a table, tree or model. Consequently, the functionalities fulfillment can be verified in a future review.

Quality assessment. A quality assessment was not performed in this review because its inclusion could restrict even more the available tools and the goal of the review was to identify the tool support to the process, especially their functionalities. However, we are aware that the quality assessment can provide more insights and explanations to the findings of the review; therefore they should be included in a future review.

9. Conclusion

The execution of a domain analysis process without a tool support can lead to an unsuccessful result, due to the complexity of interrelated activities and work products. However, the use of any tool will not necessarily lead to an effective result. A useful tool must assist the whole process, and not just some individual subprocesses.

Thus, aiming to define the first step for identifying what support a tool should provide for the domain analysis process, this paper presented a systematic review of domain analysis tools. It was done according to the defined main question and its sub-questions, which involved the identification of external characteristics about the tools and of the functionalities they support.

Several of the functionalities identified in this review were also identified in previous studies; however other researchers first defined the functionalities to be analyzed in each tool to later verify if the tools supported them. This type of approach restricts a complete mapping of the tool support scenario, such as the one reported here.

Through the results obtained in this review, it is clear that the tools differ in process and functionalities being supported, and confirms that there are gaps in the support for domain analysis process in all the tools we investigated. Such gaps in functionality can make it difficult for industry to adopt a DA tool or to have a successful result in the DA process. Thus our review suggests that there are opportunities for the development of new tools and/or

extending existing tools, and both identifies and prioritizes the required functionalities.

This study, along with RiSE's group expertise, was the rationale for the development of a domain analysis tool called ToolDAy[13] [72] in a cooperation project between CESAR[14] and the Federal University of Pernambuco. This tool is currently being used in an industrial case inside CESAR.

Acknowledgement

We thank Barbara Kitchenham's help for her comments about the report. This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[15]), funded by CNPq and FACEPE, Grants 573964/2008-4 and APQ-1037-1.03/08 and Brazilian Agency (CNPq process number 475743/2007-5).

Appendix A. Journals and conferences

The target journals were:

- ACM Computing Survey;
- Annals of Software Engineering;
- Automated Software Engineering;
- IEEE Software;
- IEEE Transactions on Software Engineering;
- Information and Software Technology;
- Journal of Systems and Software;
- Software and System Modeling;
- 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).

References

[1] C.W. Krueger, Software reuse, ACM Computing Surveys 24 (2) (1992) 131–183.
[2] W.B. Frakes, S. Isoda, Success factors of systematic reuse, IEEE Software 11 (5) (1994) 14–19.
[3] T.J. Biggerstaff, An assessment and analysis of software reuse, Advances in Computers 34 (1992) 1–57.
[4] L. Bass, P. Clements, S.G. Cohen, L. Northrop, J. Withey, Product line practice workshop report, Tech. Rep., Technical Report CMU/SEI-97-TR-003, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1997.
[5] G. Succi, J. Yip, E. Liu, Analysis of the essential requirements for a domain analysis tool, in: ICSE 2000 Workshop on Software Product Lines: Economics, Architectures and Implications, 2000.
[6] M. Moon, K. Yeom, H.S. Chae, 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) (2005) 551–569.
[7] K. Czarnecki, U. Eisenecker, Generative Programming: Methods, Tools, and Applications, Addison-Wesley, 2000.
[8] J. Neighbors, Software construction using components, Ph.D. Thesis, University of California, US, 1981.
[9] R. Prieto-Dfaz, Domain analysis: an introduction, ACM SIGSOFT Software Engineering Notes 15 (2) (1990) 47–54.
[10] W.B. Frakes, R. Prieto-Diaz, C.J. Fox, Dare: domain analysis and reuse environment, Annals of Software Engineering 5 (1998) 125–141.
[11] D. Weiss, C.T.R. Lai, Software Product-Line Engineering: A Family-Based Software Development Process, Addison-Wesley, 1999.
[12] J. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schimd, T. Widen, J.-M. DeBaud, Pulse: a methodology to develop software product lines, in: Symposium on Software Reusability, ACM Press, Los Angeles, California, United States, 1999, pp. 122–131.
[13] M. Kim, H. Yang, S. Park, A domain analysis method for software product lines based on scenarios, goals and features, in: Asia-Pacific Software Engineering Conference (APSEC), Thailand, 2003, pp. 126–136.
[14] H. Mei, W. Zhang, F. Gu, A feature oriented approach to modeling and reusing requirements of software product lines, in: International Conference on Computer Software and Applications (COMPSAC), IEEE Computer Society, USA, 2003, pp. 250–256.
[15] E.S. Almeida, J.C.C.P. Mascena, A.P.C. Cavalcanti, A. Alvaro, V.C. Garcia, S.R.L. Meira, D. LucrTdio, The domain analysis concept revisited: a practical approach, in: M. Morisio (Ed.), International Conference on Software Reuse (ICSR), Turin, Italy, 2006, pp. 43–57.
[16] D. LucrTdio, R.P.M. Fortes, E.S. Almeida, S.R.L. Meira, Performing domain analysis for model-driven software reuse, in: 10th International Conference on Software Reuse (ICSR), Lecture Notes in Computer Science, Beijing, China, Springer-Verlag, 2008, pp. 200–211.
[17] G. Arango, Domain analysis methods, in: E. Horwood (Ed.), Software Reusability, Chichester, England, 1994, pp. 17–49.
[18] H. Gomaa, M.E. Shin, 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, 2004, pp. 73–84.
[19] R. Capilla, A. Sánchez, J.C. Dueñas, An analysis of variability modeling and management tools for product line development, in: Software and Service Variability Management Workshop – Concepts, Models, and Tools, Helsinki, Finland, 2007, pp. 32–47.
[20] M. Sinnema, S. Deelstra, Classifying variability modeling techniques, Information and Software Technology 49 (2007) 717–739.
[21] B.A. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, Tech. Rep. EBSE-2007-01, Keele University, EBSE Technical Report, 2007.
[22] G.H. Travassos, J. Biolchini, Systematic review applied to software engineering, in: Brazilian Symposium on Software Engineering (SBES) – Tutorials, Jopo Pessoa, Brazil, 2007, p. 436.
[23] P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner, M. Khalil, Lessons from applying the systematic literature review process within the software engineering domain, Journal of Systems and Software 80 (2007) 571–583.
[24] M.H. Hamilton, 001: A full life cycle systems engineering and software development environment development before the fact in action, December 15, 2007, 1991. <http://world.std.com/hti/Articles/Full_Life_Cycle.htm>.
[25] R.W. Krut, Jr., Integrating 001 tool support in the feature-oriented domain analysis methodology, Tech. Rep. CMU/SEI-93-TR-11, ESC-TR-93-188, SEI, 1993.
[26] Postech, ASADAL/FORM Tool, User guide, Software Engineering Laboratory, 08/02/2006, 2003.
[27] K. Kim, H. Kim, M. Ahn, M. Seo, Y. Chang, K.C. Kang, Asadal: a tool system for co-development of software and test environment based on product line engineering, in: International Conference on Software Engineering (ICSR), ACM Press, New York, NY, USA, 2006, pp. 783–786.
[28] R. Prieto-Dfaz, W.B. Frakes, B. Gogia, Dare – a domain analysis and reuse environment, Tech. Rep., Reuse, Inc., August 1992.
[29] W.B. Frakes, R. Prieto-Diaz, C. Fox, Dare-cots, a domain analysis support tool, in: Proceedings of the 17th International Conference of the Chilean Computer Science Society (SCCC'97), IEEE Computer Society, 1997, p. 73.
[30] W.B. Frakes, Automating domain engineering, in: Workshop on Institutionalizing Software Reuse (WIRS), 1997.
[31] P. Padmanabhan, Decimal: a requirements engineering tool for product families, Ph.D. Thesis, Iowa State University, US, 2002.
[32] P. Padmanabhan, R.R. Lutz, Tool-supported verification of product line requirements, Automated Software Engineering 12 (4) (2005) 447–465.
[33] J. Dehlinger, M. Humphrey, L. Suvorov, P. Padmanabhan, R. Lutz, 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, IEEE Computer Society, 2007, pp. 49–50.
[34] W. Tracz, L. Coglianese, A DSSA domain analysis tool, Tech. Rep. ADAGE-LOR-94-13, Loral Federal System, Owego, US, 1995.
[35] S. Buhne, K. Lauenroth, P. Klaus, Modelling requirements variability across product lines, in: IEEE International Conference on Requirements Engineering, 2005, pp. 41–52.
[36] J. Park, M. Moon, K. Yeom, Dream: domain requirement asset manager in product lines, in: International Symposium on Future Software Technology (ISFST), Xian, China, 2004.
[37] M. Antkiewicz, K. Czarnecki, Featureplugin: feature modeling plug-in for eclipse, in: OOPSLA Workshop on Eclipse Technology Exchange, ACM Press, New York, NY, USA, 2004, pp. 67–72.
[38] K. Czarnecki, M. Antkiewicz, C.H.P. Kim, S. Lau, K. Pietroszek, fmp and fmp2rsm: eclipse plug-ins for modeling features using model templates, in:

[13] http://www.rise.com.br/english/products_toolday.php.
[14] Recife Center of Advanced Studies and Systems – http://www.cesar.org.br.
[15] INES – http://www.ines.org.br.

Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), ACM, San Diego, CA, USA, pp. 200–201.
[39] T. Leich, S. Apel, L. Marnitz, G. Saake, Tool support for feature-oriented software development: featureide: an eclipse-based approach, in: Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology Exchange, ACM, San Diego, California, 2005, pp. 55–59.
[40] BigLever, Gears user's guide, Tech. Rep., BigLever, 2005.
[41] C.W. Krueger, Software Mass Customization, Biglever, technical white paper ed., 13/04/2006, 2005.
[42] C.W. Krueger, Biglever software gears and the 3-tiered spl methodology, in: Object Oriented Programming Systems and Applications (OOPSLA), ACM, Montreal, Quebec, Canada, 2007, pp. 844–845.
[43] G. Succi, A. Eberlein, J. Yip, K. Luc, M. Nguy, Y. Tan, The design of holmes: a tool for domain analysis and engineering, in: IEEE Pacific Rim Conference, Victoria, BC, Canada, 1999, pp. 365–368.
[44] G. Succi, J. Yip, E. Liu, W. Pedrycz, Holmes: a system to support software product lines, in: International Conference on Software Engineering (ICSE), ACM, Limerick, Ireland, 2000, p. 786.
[45] G. Succi, W. Pedrycz, J. Yip, I. Kaytazov, Intelligent design of product lines in holmes, in: Electrical and Computer Engineering, vol. 1, Canada, 2001, pp. 75–80.
[46] G. Succi, J. Yip, W. Pedrycz, Holmes: an intelligent system to support software product line development, in: International Conference on Software Engineering (ICSE), Toronto, Ontario, Canada, 2001, pp. 829–830.
[47] R.M.M. Braga, C. Werner, M. Mattoso, Odyssey: a reuse environment based on domain models, in: IEEE Symposium on Application-Specific Systems and Software Engineering and Technology (ASSET), Richardson, Texas, 1999, pp. 49–57.
[48] R.M.M. Braga, Components search and retrieval in software reusable environments, Ph.D. Thesis, UFRJ – Federal University of Rio de Janeiro, Brazil, 2000 (in Portuguese).
[49] M. Eriksson, J. Börstler, K. Borg, The pluss approach – domain modeling with features, use cases and use case realizations, in: Software Product Lines Conference, Lecture Notes in Computer Science, vol. 3714, Rennes, France, 2005, pp. 33–44.
[50] M. Eriksson, H. Morast, J. Borstler, K. Borg, 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, ACM, Long Beach, CA, USA, 2005, pp. 300–304.
[51] M. Eriksson, H. Morast, J. Borstler, K. Borg, An empirical evaluation of the pluss toolkit, Tech. Rep. UMINF-06.31, Department of Computing Science, Umea University, 2006.
[52] K. Schmid, M. Schank, Pulse-beat – a decision support tool for scoping product lines, in: Software Architectures for Product Families, International Workshop IW-SAPF-3, Las Palmas de Gran Canaria, Spain, 2000, pp. 65–75.
[53] D. Beuche, O. Spinczyk, 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, ACM, Anaheim, CA, USA, 2003, pp. 108–109.
[54] Pure-systems, Variant Management with Pure::Variants, Pure-systems, 02/02/2006, 2004.
[55] O. Spinczyk, D. Beuche, 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, ACM, Vancouver, BC, Canada, 2004, pp. 18–19.
[56] T.V.D. Massen, H. Lichter, Requiline: a requirements engineering tool for software product lines, in: International Workshop on Product Family Engineering (PFE), Springer-Verlag, Siena, Italy, 2003, pp. 168–180.
[57] RWTH-Aachen, The RequiLine User Manual, Germany, January 2005.
[58] K.C. Kang, S.G. Cohen, J.A. Hess, W.E. Novak, A.S. Peterson, Feature-oriented domain analysis (foda) feasibility study, Tech. Rep. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.
[59] K.C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, M. Huh, Form: A feature-oriented reuse method with domain-specific reference architectures, Annals of Software Engineering 5 (1998) 143–168.
[60] W. Tracz, DSSA (domain-specific software architecture): pedagogical example, SIGSOFT Software Engineering Notes 20 (3) (1995) 49–62.
[61] K. Pohl, G. Bockle, F. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques, Springer-Verlag, New York, 2005.
[62] G. Succi, A. Valerio, T. Vernazza, M. Fenaroli, P. Predonzani, Framework extraction with domain analysis, ACM Computing Surveys (2000) 12. <http://dx.doi.org/http://doi.acm.org/10.1145/351936.35194>.
[63] K. Lee, K.C. Kang, W. Chae, B.W. Choi, Feature-based approach to object-oriented engineering of applications for reuse, Software Practice and Experience 30 (9) (2000) 1025–1046.
[64] C.W. Krueger, Data from soliton's software product line initiative, Tech. Rep. 2002-07-08-1, Biglever, US, 2002.
[65] J. Bayer, D. Muthig, T. Widen, Customizable domain analysis, in: International Symposium on Generative and Component-Based Software Engineering, Springer-Verlag, 1999, pp. 178–194.
[66] M. Jaring, J. Bosch, Representing variability in software product lines: a case study, in: Software Product Line Conference (SPLC), San Diego CA, 2002, pp. 15–36.
[67] M. Steger, C. Tischer, B. Boss, A. Mnller, O. Pertler, W. Stolz, S. Ferber, Introducing pla at Bosch gasoline systems: experiences and practices, in: Software Product Lines (SPLC), Boston, MA, USA, 2004, pp. 34–50.
[68] E.S. Almeida, A. Alvaro, V.C. Garcia, D. LucrTdio, R.P.M. Fortes, S.R.D.L. Meira, An experimental study in domain engineering, in: IEEE EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Component-Based Software Engineering Track, Lubeck, Germany, 2007, pp. 93–100.
[69] D.M. Weiss, 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, Springer-Verlag, London, UK, 1998, pp. 214–222.
[70] E.S. Almeida, Ride: the rise process for domain engineering, Ph.D. Thesis, Federal University of Pernambuco, Brazil, 2007.
[71] A. Fuggetta, A classification of case technology, Computer 26 (12) (1993) 25–38.
[72] L.B. Lisboa, Toolday – a tool for domain analysis, Master's Thesis, Federal University of Pernambuco, Recife, Brazil, January 2009.