ON THE USE OF VISUALIZATION FOR SUPPORTING SOFTWARE REUSE

Marcelo Schots de Oliveira

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadora: Cláudia Maria Lima Werner

Rio de Janeiro
Dezembro de 2015

ON THE USE OF VISUALIZATION FOR SUPPORTING SOFTWARE REUSE

Marcelo Schots de Oliveira

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____
Profa. Cláudia Maria Lima Werner, D.Sc.

_____
Prof. Toacy Cavalcante de Oliveira, D.Sc.

_____
Prof. Claudio Esperança, Ph.D.

_____
Prof. Márcio de Oliveira Barros, D.Sc.

_____
Prof. Alexandre Gonçalves Evsukoff, Ph.D.

RIO DE JANEIRO, RJ - BRASIL
DEZEMBRO DE 2015

Oliveira, Marcelo Schots de

On the Use of Visualization for Supporting Software Reuse/ Marcelo Schots de Oliveira. – Rio de Janeiro: UFRJ/COPPE, 2015.

XV, 224 p.: il.; 29,7 cm.

Orientadora: Cláudia Maria Lima Werner

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2015.

Referências Bibliográficas: p. 165-183.

1. Software Visualization. 2. Software Reuse. 3. Information Visualization. 4. Software Engineering. I. Werner, Cláudia Maria Lima. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

“Perform your works in meekness, and you shall be loved beyond the glory of men. However great you may be, humble yourself in all things, and you will find grace in the presence of God. For only the power of God is great, and He is honored by the humble. The heart of the wise is understood by wisdom, and a good ear will listen to wisdom with all its desire.”

(Ecclus 3:19-21;31. The Holy Bible, Catholic Public Domain Version)

ACKNOWLEDGMENTS

The work presented in this thesis would never have been possible without the strength and inspiration provided by God when I needed the most, besides the support and encouragement of a number of people He has put into my life. Therefore, I must start thanking Him. Despite all the issues, losses, and difficulties Natália and I have been struggling with during this Ph.D. time, He is always with us. I also thank the Holy Mary, for teaching us to "do whatever He tells us" and "reflecting about all the things in her heart", and for her special care of Natália and me. I must not forget to thank my Guardian Angel, who certainly has had a lot of work during all this time!

I would also like to thank my advisor for her guidance and support since the master course. Cláudia, many of us may not always notice your dedication and patience with your students, but you can rest assured that I did, even when I did not demonstrate. I cannot thank you enough for bearing with my defects and failings. I learned with you a lot. Your valuable academic lessons go beyond the work presented in this thesis. I wish you and your family all the best.

I am so grateful for my father Weliton, my mother Maria Helena, and my sister Letícia, for everything I have learned from them, which made me who I am. That small-town boy has come a long way since he left Manhumirim, because of them and their unconditional love and support. Pai, mãe, you made the impossible so that I could make my undergraduate course, and you have made so many sacrifices for Letícia and me that I will never forget. I also thank my brother-in-law Robson, who is such a nice husband to Letícia and has become such a good friend. I hope we can hang out more often.

Very special thanks go to all the members of my lovely family, who have always believed in me, encouraged me, and cared about me. Thank you for all your prayers, each one made the difference. Your heartfelt happiness in each of my progresses was inspiring. My gratitude also goes to Natália's family, for understanding our necessary absences and constantly praying for us.

I thank the members of my dissertation committee for accepting the invitation to evaluate my work. Márcio, thank you so much for dedicating your time for reading this thesis so carefully, for your (positively) challenging questions, and for your pertinent comments. Alexandre, your thoughts as "an outsider" helped me improve the explanation of the thesis context. Claudio, I am very thankful for your enlightening,

interesting, and thought-provoking considerations. Toacy, your points made me provide a better explanation of my work. I also thank all the professors who contributed with my academic formation throughout this journey, in many different ways, helping me define who I should be as a professor and researcher. Thanks especially for the opportunities I was given of teaching and cooperating with industry.

Many thanks to all the organizations where I supported and assessed the implementation of software reuse processes since 2009. A large portion of this thesis is motivated by your feedback about the issues that permeate an effective implementation of reuse. I sincerely hope this work can be a guide towards some solutions, being a starting point for other fruitful academia-industry collaborations for advancing the field.

My sincere thanks to all the volunteers who participated in the experiments I have conducted, related or not with this thesis, for your precious time, genuine interest, and readiness for helping me evaluate my works. I also wish to thank the open source community, especially the authors and maintainers of the projects used in the tools built during this thesis. Much open source software was reused (it could not be any different in such a thesis), and all the openly available data were key to this research.

I am so grateful for the 5 years I spent at UERJ as a professor. Not only I could teach (and learn) with brilliant, amazing students, but also I was blessed for having been invited by some of these to be their advisor, which was such an honor and privilege for me. All the good things we have done in these final projects and papers were thanks to your patience and determination. You are a fundamental part of my academic trajectory.

I am also very lucky for all the professors I have known from the Computer Science and Informatics Department at UERJ, especially Mariluci (who is now watching over us from Heaven) and professors Alexandre Rojas, Vera, and (in the most recent years) Leticia, resulting in very nice collaborations. I also would like to thank the secretaries and staff at UERJ (especially Alice, such a wonderful and important person).

I also thank my colleagues from the Software Engineering research group at COPPE/UFRJ, who were important for my academic trajectory, each in their own ways. Special thanks to Marlon, Renan, André, and Fernanda (a High School scientific initiation student), whose works I had the opportunity to co-supervise, even formally or informally. Thank you for your patience with my defects and limitations. I particularly thank Dr. Claudia Susie, for being a “jack-of-all-trades”, showing how cooperative a person can be in the most simple (and many times unnoticed) things.

Thanks to all the secretaries and staff from the Systems Engineering and Computing Department and the Virtual and Augmented Reality Laboratory (Lab3D) at COPPE, especially Bernadette, for being so supportive. Thanks to CNPq and CAPES for the scholarship provided during the doctoral course. Thanks also to FAPERJ, COPPETEC, and CAPS/SIGSOFT, for the financial support for my expenses when presenting my research in nationwide and worldwide events.

I am thankful for having met many special people during conferences, who have helped me with my research by sharing their opinions and thoughts. Particularly, I would like to thank professors Carolyn Seaman, Fabian Beck, Stephan Diehl, and Andreas Bollin, who provided personal feedback that greatly helped me improve my works. I also thank prof. André van der Hoek for his comments in my previous work, which were helpful for building this one. As I did in my master thesis, I also thank all the anonymous reviewers who provided valuable feedback for improving my research.

I am deeply grateful for all my friends who have prayed and cheered for this thesis to finish as soon as possible. Your support and comprehension surely made the difference. I particularly thank Fr. Luiz Fernando Cintra and Fr. Pedro Barreto (for the spiritual guidance and patience through all these years), Mirna Lewis ("Greg's mum", the most amazing Canadian friend Natália and I could ever have), Dcn. Daniel Kambalame (the WYD pilgrim from Malawi who became my true "br", always keeping in touch), Ivan and Italo (for supporting and encouraging me). I also thank everyone from Tijuca and Marquês de Valença Centers for supporting me and praying for me.

Most of all, I would like to thank my lovely and wonderful wife. Natália, you are such an inspiration for me to be a better person each day, reminding me with your life example that happiness resides in the simplest things. Thank you for all your love, your support, your prayers, the shared smiles and tears, every single thing. Thank you for being so patient with me, and for loving me way more than I deserve. Belar and I love you with all our hearts and beyond. I hope Our Lord may eliminate all the obstacles and bless us again with more children, as we have been waiting and hoping for so long.

Last, but not least, I would like to thank my beloved angel and unborn son Belar for watching over his dad from Heaven. Belar, you know that if I was given the chance to choose between the doctoral title and having you here (not that these options are mutually exclusive), I would have given up this course without thinking twice, with no regrets.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

O USO DE VISUALIZAÇÕES COMO APOIO À REUTILIZAÇÃO DE SOFTWARE

Marcelo Schots de Oliveira

Dezembro/2015

Orientadora: Cláudia Maria Lima Werner

Programa: Engenharia de Sistemas e Computação

A reutilização de software já é parte do dia-a-dia no desenvolvimento de aplicações. No entanto, obstáculos técnicos e não técnicos ainda impedem sua efetiva execução em organizações de software, incluindo a dificuldade de gerenciar e prover visibilidade de cenários de reúso, devido à quantidade e diversidade de dados envolvidos. Este trabalho apresenta APPRAiSER, uma abordagem que visa prover *awareness* por meio de recursos de visualização da informação para apoiar a execução de tarefas de reutilização. Algumas necessidades identificadas a partir de estudos de caracterização do estado da arte e da prática serviram de insumo para a definição da APPRAiSER. A abordagem é composta por ferramentas para a extração, agregação e visualização de dados provenientes de repositórios de software, além de elementos conceituais para a estruturação do conhecimento sobre conceitos da área. Estudos conduzidos com profissionais da academia e da indústria mostraram que o apoio ferramental desenvolvido provê aumento da percepção (*awareness*) na execução de tarefas de reúso, e que os elementos conceituais têm potencial para auxiliar no entendimento dos conceitos para a engenharia de ferramentas interativas de visualização. Por fim, foram também apontadas melhorias para a abordagem.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

ON THE USE OF VISUALIZATION FOR SUPPORTING SOFTWARE REUSE

Marcelo Schots de Oliveira

December/2015

Advisor: Cláudia Maria Lima Werner

Department: Systems Engineering and Computer Science

Software reuse has become part of the day-to-day application development. However, some technical and non-technical obstacles still hinder its effective execution in software organizations, including the difficulty in managing and providing visibility of reuse scenarios, due to the amount and variety of associated data. This work presents APPRAiSER, an approach that aims at providing awareness through information visualization resources for supporting the execution of reuse tasks. Some needs identified through studies for characterizing the state-of-the-art and state-of-the-practice served as input for the definition of APPRAiSER. The approach is composed by tools for extracting, aggregating, and visualizing data from software repositories, as well as some conceptual elements for structuring the knowledge about concepts in the field. Studies conducted with professionals from academia and industry showed that the developed tool support increases awareness on the execution of reuse tasks, and the conceptual elements have the potential to help understanding the concepts for the engineering of interactive visualization tools. Finally, some improvements were also pointed out for the approach.

SUMMARY

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1 – INTRODUCTION

This chapter presents the motivation for this work, the research questions that guided the proposal of the approach, as well as its goals and the adopted research methodology.

1.1 Foreword

Software systems permeate advances in all areas of knowledge, and there is an increasing participation of software in society [Brazilian Computer Society 2006]. Such systems are embedded in everyday devices such as household appliances, and support areas such as communication (e.g., mobile devices, social media etc.) healthcare (e.g., blood pressure and glucose monitors, electronic patient records, clinical exams, artificial pacemakers etc.), context-awareness (e.g., place recommenders based on location awareness, smart houses – for healthcare, energy saving etc.) and so on. In a fast-paced world that is continuously changing across all areas, there is a large demand for new functionalities, and there is a broad field of software-related opportunities that increase each day.

The demands of each of these fields (among others) and the advances of mobile devices, social media, and new technologies over the years have been strongly influencing the way software systems are built. Traditional approaches no longer meet the demands of new circumstances, and have given rise to the emergence of new forms of development, including agile methodologies [Fowler & Highsmith 2001] and the widespread open source practices [Haefliger et al. 2008]. The steady growth of distributed software development has also made the scenario more complex and, at the same time, more stimulating and challenging [Schots et al. 2012].

This scenario not only has led to a large-scale adoption of new technologies, practices and methodologies for software development, but also has required a smaller and ever decreasing time-to-market. Software organizations, in turn, need ways to make software development as efficient as possible in order to cope with the increasing demands.

One of the ways of supporting the demand for delivery of software systems in less time with less effort is through software reuse. Software reuse has become a very

common and widespread concept in software development, and has been a promising paradigm in software engineering since its inception [Benedicenti et al. 1996], given that it can be fully integrated and supported in the software development process, improving the life cycle by reducing the effort and time needed to develop a software system.

Since the beginning of software engineering, much research has been done in developing techniques and tools for supporting software reuse [Naur & Randell 1968] [Mili et al. 1995] [Frakes & Kang 2005], which include cataloging, retrieval and storage of reusable assets[1]. Moreover, several approaches have been proposed or adapted to support development for and with reuse[2], such as domain engineering techniques [Kang et al. 1990], software product lines [Clements & Northrop 2002] [Fernandes et al. 2011], and so on.

1.2 Motivation

Reuse activities are present in the daily routine of software developers, yet mostly in an ad-hoc or a pragmatic way. Despite the maturity of software reuse research and the extensive literature available, organizations still have difficulties in understanding and incorporating some reuse practices, implementing software reuse processes, and establishing an effective reuse program[3].

Part of this problem is caused by the negligence in envisioning non-technical aspects when introducing a software reuse program, e.g., engagement of team members and managerial support [Kim & Stohr 1998]. Sherif and Vinze (2003) highlight that reuse provides better results when all stakeholders are committed to it [Sherif & Vinze 2003]. In this sense, two crucial concerns for facilitating the acceptance/consciousness and adoption of reuse are *how to increase the visibility of reuse results* and *how to provide appropriate awareness for software reuse tasks*. Awareness mechanisms allow stakeholders to be percipient of what goes on in the development scenario [Treude & Storey 2010], and can provide them with the necessary information and support for performing their reuse-related tasks.

---

[1] In this work, any item built for use in multiple contexts, such as software designs, specifications, source code, documentations, test cases etc. can be considered as reusable assets [IEEE 2010].

[2] These concepts are defined in Section 2.1.

[3] A reuse program is an organizational mechanism that establishes the goals, scope, and strategies for addressing issues related to business, people, process, and technology involved in the adoption of software reuse. The term "reuse program" should not be confused with "a software application for reuse".

One of the ways to increase awareness is by employing visualization resources and techniques [Hattori 2010]. Software visualization has been researched as a way to assist software development activities that involve human reasoning, helping people to deal with the large amount and variety of information by providing appropriate abstractions [Lanza & Marinescu 2006] [Diehl 2007]. In the software reuse scenario, its use can allow awareness and comprehension of reuse elements (i.e., assets, developers, and development projects) and their surroundings.

Despite the potential of software visualization on supporting software reuse, little work has been done with this goal, and the existing ones (e.g., [Alonso & Frakes 2000], [Marshall et al. 2003] and [Anslow et al. 2004]) do not take into account the different reuse stakeholders' information needs. Besides, they are very limited in terms of integration with other information sources, not providing enough evidence on their effectiveness [Schots et al. 2014].

Thus, it is believed that a better investigation and exploration of software visualization resources can be performed, in order to provide and increase awareness of reuse scenarios and support reuse-related tasks, such as exploring a reuse repository, obtaining and understanding information regarding reusable assets, and monitoring reuse initiatives. Through visual abstractions, one can better comprehend reuse elements and their surroundings. This is the focus of the approach proposed in this work.

1.3 Hypothesis and Research Questions

Considering that:

1) software reuse brings several benefits to the software development scenario, reducing the cost and effort necessary for the construction of new software systems, thus allowing a "better" time-to-market and bringing a potential increase of quality;
2) the establishment of a reuse program can facilitate reuse management in software development organizations, allowing for an increase of maturity towards systematic reuse;
3) besides supporting stakeholders in their reuse-related tasks according to their roles, it is important to provide them with visibility of the obtained results, so that they can become committed with software reuse by perceiving the benefits brought by it;
4) the difficulty in finding and understanding reusable assets may lead potential consumers to prefer to recreate from scratch an existing solution, due to the lack of

available information (or lack of organization of the existing information) related to such assets;

5) software visualization techniques and resources allow awareness and comprehension of the structure, behavior and evolution of software entities and metadata, assisting software development activities that involve human reasoning;

The hypothesis of this work is:

The use of proper visualization resources for presenting reuse-related information can assist stakeholders in carrying out their software reuse tasks.

For investigating this hypothesis, the following research questions were derived:

- ***RQ1.*** *What are the characteristics and limitations of the visualization approaches that have been proposed to support software reuse?*
- ***RQ2.*** *Which aspects (comprising stakeholders' needs, reuse tasks, and reuse-related data) should be taken into account for a visualization-based approach to support software reuse?*
- ***RQ3.*** *Are the employed visualization resources feasible in helping the targeted stakeholders to be aware of a given reuse scenario, allowing the execution of reuse tasks accurately, with adequate efficiency and efficacy?*

1.4 Goals

The main goal of this work is to investigate, propose, and evaluate the use of visualization resources for supporting software reuse awareness. This generic goal can be decomposed in the following specific goals:

1) Characterize existing works that use visualization for supporting reuse somehow, in order to analyze their features, strengths and limitations;
2) Identify the needs of stakeholders involved in software reuse tasks, taking into account their role in such tasks;
3) Identify the necessary features for an approach that aims to support stakeholders in performing reuse tasks;
4) Define an approach and implement an interactive visualization environment that supports software development organizations by providing reuse awareness, for instigating and monitoring software reuse initiatives;
5) Ensure that the proposed approach meets some of the needs stated by stakeholders, with appropriate efficiency and efficacy, while decreasing the effort and time involved in performing reuse tasks.

This work is also a first step towards meeting the challenges described in [Schots et al. 2012] regarding awareness and comprehension in software and systems engineering, and is related to other works developed at COPPE/UFRJ and the State University of Rio de Janeiro (UERJ), as discussed throughout the text.

1.5 Research Methodology

In order to answer the posed research questions, a research methodology was established, as depicted in Figure 1.1. It shows the main research steps (on the left) and how they are intended to be accomplished (on the right) –, provided that the chosen ways to accomplish each step may vary depending on the findings of the previous steps. The initial steps comprise the characterization of the research topic, and address both RQ1 and RQ2. The results are intended to provide and refine desirable features for proposing and developing a novel approach, which is then evaluated and improved, addressing RQ3.

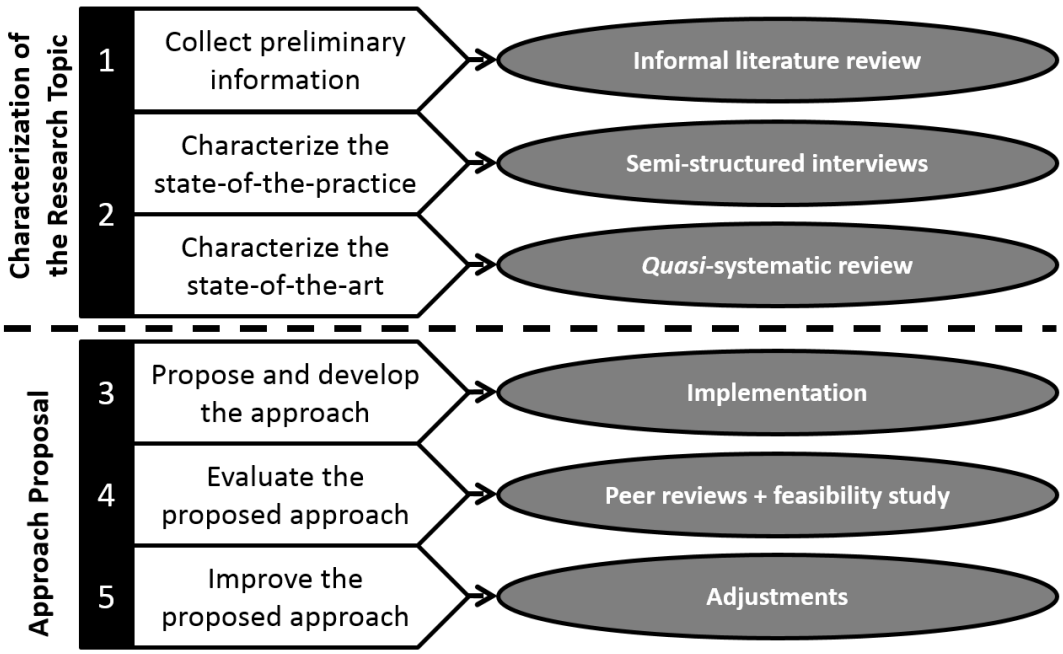


Figure 1.1 – Research methodology

In the first step (*Collect preliminary information*), an *informal literature review* provides the initial/basic knowledge about the research topic, which serves as input for the next step.

A fundamental part of this research process is to ensure the identification of the actual needs of each stakeholder [Schots et al. 2012] from the state-of-the-practice, since such needs may not be properly identified in the technical literature. This is accomplished in part of the second step (*Characterize the state-of-the-practice*) by

means of *semi-structured interviews* with practitioners. Semi-structured interviews "are designed to elicit not only the information foreseen, but also unexpected types of information" [Seaman 1999]. Chapter 2 presents an overview of this study.

A *quasi-systematic review* is conducted in the second step of the research (*Characterize the state-of-the-art*) in parallel to the characterization of the state-of-the-practice, allowing for a broader, more comprehensive view of the topic, and improving the initial understanding obtained in the first step. *Quasi*-systematic reviews use the same rigorous methodological processes from systematic reviews, looking for the identification of relevant evidence in the research field under investigation, but usually no meta-analysis can be applied [Travassos et al. 2008]. The main findings of this step are presented in Chapter 3, and the detailed description of the planning, execution and analysis of the *quasi*-systematic review can be found in [Schots et al. 2014].

The findings from these steps provide and refine desirable features for the third step of the research methodology (*Propose and develop the approach*), and also help building a body of knowledge on the topic, in addition to pointing out research opportunities for other works. Through the *implementation* of the approach, an environment for supporting software reuse by awareness is concretized, according to the defined goals. Chapter 4 describes the definition of the approach and its details.

The feasibility of the proposed approach is evaluated in the fourth step (*Evaluate the proposed approach*), which is intended to assess quantitatively and qualitatively whether the perceptive and cognitive abilities of stakeholders in carrying out software reuse tasks are properly stimulated, with adequate efficiency and efficacy, while decreasing the effort and time spent on such tasks. This involves the planning and execution of studies, described in Chapter 5.

Finally, based on the evaluation results and feedback, the fifth step (*Improve the proposed approach*) takes place, for making the necessary *adjustments* and the identified improvements, if applicable.

1.6 Text Structure

The remaining of this text is organized as follows:

- *Chapter 2 – Software Reuse* presents the main concepts related to software reuse, including how quality standards and maturity models address this topic, as well as some issues related to the establishment of a reuse program. It also presents a study conducted for characterizing the state-of-the-practice, along with its results.

- *Chapter 3 – Software Visualization* introduces some concepts related to software visualization, some challenges in awareness and comprehension, and related works identified from the conduction of a *quasi*-systematic review.
- *Chapter 4 – Proposed Approach: APPRAiSER* describes the proposed solution and its main elements, including their goals and details on their development.
- *Chapter 5 – Evaluation* presents the planning, execution, and results of some studies conducted in order to evaluate the main approach elements.
- *Chapter 6 – Conclusion* summarizes the contributions of this work, and presents a research agenda resulting from open questions and opportunities for improvement.

CHAPTER 2 – SOFTWARE REUSE

This chapter presents an overview of software reuse, including a brief motivation, a description of how some quality standards and maturity models address this topic, and examples of software reuse tasks. Besides, common issues related to the establishment of a reuse program are discussed, along with some problems recognized during the implementation and assessment of reuse processes.

2.1 Contextualization

Software reuse has become a very common and widespread concept in software development. As stated by [Holmes 2008], it has a well-established history in both research literature [Frakes & Kang 2005] and industrial practice [Poulin et al. 1993] [Bauer et al. 2014]. One can state that reuse activities are present in the routine of software developers, yet mostly in an ad-hoc or a pragmatic way[4] [Holmes 2008].

Reuse practices allow achieving a number of benefits, such as reducing the effort and time spent on software development [Krueger 1992] [Poulin et al. 1993] [Mili et al. 1995]. Reusing assets from past projects (i.e., that have been already tested and deployed) also allows developing more reliable applications and decreasing maintenance efforts, since their quality is assured on previous experiences of use [Benedicenti et al. 1996] [Morisio et al. 2002]. Besides, the availability of reusable assets can facilitate newcomers in dealing with new technologies and domains (taking external solutions as a basis for their own development), as well as experienced developers in increasing their productivity by composing existing solutions.

Since the idea of building new software from existing pieces of preexistent software arose [Naur & Randell 1968], it was noticed that several types of artifacts can be reused in software development, such as requirements specifications, software designs, test cases and so on [IEEE 2010]. However, some studies and books point out

[4] Pragmatic reuse is related to an attempt of reusing source code that was not designed explicitly for reuse [Holmes & Walker 2012]. It takes place when an opportunity of reusing an existing code arises, leading developers to collect bunches of code (without modifying the original system where the code comes from) and put them into another system. This can be generalized to other kinds of artifacts.

that reuse of source code is still on the mainstream of software development, such as [Sá et al. 1997], [Haefliger et al. 2008], [Leach 2012], and [Schots & Werner 2014a].

The concept of reuse of source code is frequently linked to software components, i.e., "self-contained, clearly identifiable artifacts that describe and/or perform specific functions and have clear interfaces, appropriate documentation and a defined reuse status" [Sametinger 1997]. Hooper & Chester (1991) classify reusable software components into two categories: horizontal and vertical.

Horizontal reuse refers to "reuse across a broad range of application areas, such as data structures, sorting algorithms, and user-interface mechanisms". According to these authors, the assets are typically utilities that are purposely generic for comprising multiple applications [Hooper & Chester 1991].

Vertical reuse, in turn, refers to components within a given application area that can be reused in similar applications that belong to the same problem domain. Although horizontal reuse is better understood and easier to achieve (thus more frequently employed), the greatest reuse potential leverage comes from vertical reuse, due to its potential to build software product lines [Clements & Northrop 2002] and create competitive advantages to the organization [Hooper & Chester 1991].

Reuse can occur within several activities in software development, which can be divided into two groups [Kim & Stohr 1998]: (i) *producing activities*, involving the identification, classification and cataloging of software resources, and (ii) *consuming activities*, comprising the retrieval, understanding, modification, and integration of those resources into the software product. These groups can also be referred to as *development for reuse* (i.e., build generic assets that can be reused in similar contexts) and *development with reuse* or *by reuse* (i.e., use existing assets to build [parts of the] software), respectively [Moore & Bailin 1991].

The advantage obtained from a reuse-based software development scenario is to develop software assets aiming at their future reuse (if appropriate, according to the organization's goals). Developing assets without taking into account their reuse potential (and consequently without aiming their reuse beforehand) makes it hard to fit them into other contexts beyond the original ones to which they were developed. This is partially due to the lack of systematization in the construction of reusable assets [Prieto-Díaz & Arango 1991].

The view of reuse benefits may vary considerably according to the domain of expertise. While some kinds of organization in which safety is an extremely critical

factor may be reluctant with reusing assets developed by third parties (e.g., those that work in projects for critical domains, such as financial/banking/accounting or healthcare) [Schots & Werner 2014a], other domains largely benefit from such assets.

In fact, performing reuse may be crucial, important, or less relevant depending on the domain of interest. However, although the severity of issues on reusable assets impacts in different levels (depending on the organization/domain/project), some information (such as the license under which an asset version was released) is strongly relevant for all of them for helping to decide whether reuse should occur in a particular case or not.

Introducing reuse in an organization involves tasks that contribute to different purposes. For illustration purposes, Table 2.1 presents some examples of reuse tasks, classified into project tasks – i.e., tasks that are relevant in the context of a specific software project – and organizational tasks – i.e., tasks that either benefit all the projects or are relevant to the organizational structure (and to the reuse initiatives) as a whole.

Table 2.1 – Examples of reuse-related project tasks and organizational tasks

Project Tasks	Explore and/or search the reuse repository
	Obtain general information regarding a reusable asset (in terms of its available metadata, evolution history, developers, reuse occurrences, issues and so on)
	Select an asset according to the project needs
	Understand detailed information of a reusable asset (in terms of the available information regarding its structure, behavior and software metrics)
	Rate and/or report problems on a reused asset
Organizational Tasks	Identify assets candidate to reuse
	Identify reusable assets that need maintenance support
	Evaluate a candidate asset (or a new version of an existing asset) for entering the reuse repository, in terms of organizational criteria
	Identify experts (producers/contributors and consumers) on a reusable asset
	Register usage data of an asset
	Identify potential interested parties and register established interested parties of an asset
	Notify interested parties about changes on the status of an asset
	Evaluate and maintain the reuse repository
	Monitor the reuse activities
	Report reuse results to stakeholders

Some steps must be accomplished before the introduction of a reuse program. According to [Benedicenti et al. 1996], it is necessary to perform an accurate assessment of the organization's current situation, including goals, mission and market strategies defined by top management. The authors also state that the integration of a reuse program to the current development process can only be effective "if the process itself is well defined and structured, and the software life cycle is planned and managed" by the organization.

2.2 Implementing Software Reuse Processes

Software development organizations need to seek continually for improvement of the quality of their products and services, in order to endure in the competitive market. Consequently, they are also aiming at improving the quality of their processes. Due to this increasing demand for software quality, a number of quality standards and maturity models have been proposed, establishing requirements for defining, evaluating, and improving software processes.

In order to promote ways towards its systematization, software reuse is covered by several quality standards (e.g., ISO/IEC 12207 [ISO/IEC 2008], IEEE Std. 1517-2010 [IEEE 2010], and ISO/IEC 15504 [ISO/IEC 2012]), which demonstrate its importance for the maturity of software organizations. Such standards comprise activities related to the management of a reuse program, as well as the storage, retrieval, management and control of assets, among others. They also contain guidelines for integrating reuse to the primary processes of the software life cycle, along with processes for reuse across projects.

These quality standards usually define some outcomes that must be achieved in order to provide evidence of the maturity of organizations. For instance, according to [ISO/IEC 2008], a successful implementation of the Reuse Program Management process should provide the following outcomes as results:

- define the organization's reuse strategy, including its purpose, scope, goals and objectives;
- identify the domains in which to investigate reuse opportunities or in which it intends to practice reuse;
- assess the organization's systematic reuse capability;
- assess each domain to determine its reuse potential;
- evaluate reuse proposals to ensure the reuse product is suitable for the proposed application;
- implement the reuse strategy in the organization;
- establish feedback, communication, and notification mechanisms that operate among reuse program administrators, asset managers, domain engineers, developers, operators, and maintainers; and
- monitor and evaluate the reuse program.

Reuse practices are also integrated into models that aim to measure the maturity level of organizations that produce software, such as MR-MPS-SW [Rocha et al. 2007] [SOFTEX 2012], a program for software process improvement coordinated by the Association for Promoting the Brazilian Software Excellence (SOFTEX). This program aims to define and enhance a model for improvement and assessment of software processes focusing on micro, small, and medium enterprises (MSMEs). The MPS-SW model complies with ISO/IEC 12207 and 15504, is compatible with CMMI-DEV [CMMI Product Team 2010], adopts software engineering best practices, and is appropriate (both from the technical point of view as to costs) to the reality of Brazilian organizations [SOFTEX 2012].

MR-MPS-SW is divided into 7 maturity levels, from level G (lowest maturity level) to level A (highest maturity level), in ascending order. Since its version 1.2 (released in 2007), this model encompasses software reuse as one of the goals to be accomplished by organizations in order to evolve their maturity levels. In this model, two processes define reuse-related outcomes: GRU[5] (Reuse Management), required since intermediary maturity stages (starting from level E), and DRU[6] (Development for Reuse), in more advanced stages (from level C onwards).

The purpose of the Reuse Management process is to manage the life cycle of reusable assets. The process defines that the organizations must have a documented strategy for asset management, including criteria that govern their life cycle (i.e., criteria for acceptance, certification, classification, discontinuity, and evaluation of assets) (GRU 1). In addition, there must be a mechanism for the storage and retrieval of assets (GRU 2). Modifications on these assets must be controlled throughout the life cycle (GRU 4), and usage data shall be recorded (GRU 3), in order to notify users about potential problems detected, modifications carried out, new versions available, and discontinued assets (GRU 5) [SOFTEX 2012].

The purpose of the Development for Reuse process, in turn, is to identify opportunities for systematic reuse of assets in the organization and, if possible, establish a reuse program for developing assets from the application domain engineering. This process starts with the identification of the reuse potential (DRU 1) and reuse capabilities (DRU 2) of the organization. Results from these steps serve as a basis for

[5] Acronym for “Gerência de Reutilização”, in Portuguese.

[6] Acronym for “Desenvolvimento para Reutilização”, in Portuguese.

deciding whether the organization must provide the other outcomes. If so, the ensuing steps are the planning (DRU 3), implementation, monitoring and evaluation (DRU 4) of a reuse program, which comprise the evaluation of proposals for reuse (DRU 5), the development of domain models and domain architectures (DRU 6, DRU 7 and DRU 8), and the specification, development (or acquisition) and maintenance of domain assets (DRU 9) [SOFTEX 2012].

2.3 Issues on Software Reuse Implementations

Achieving effective software reuse is a difficult problem in itself, one that requires proper support in a number of facets, such as managerial aspects [Griss et al. 1994], the aid of tools [Marshall et al. 2003], and adequate mechanisms for retrieval of reusable assets [Braga et al. 2006], among others. In order to be acquainted with the barriers related to effective reuse, it is important to recognize some usual concerns and issues associated to software reuse initiatives.

A number of studies and reports on the implementation of reuse processes in organizations are presented in the literature (e.g., [Kim & Stohr 1998], [Morisio et al. 2002] and [Sherif & Vinze 2003], among others). Some of the frequent issues and challenges pointed out regarding the establishment of a reuse program include the following:

- the difficulty in understanding software reuse concepts and how to effectively apply them [Mili et al. 1995] [Morisio et al. 2002];
- the lack of acceptance of reuse practices by the development team and top management in software organizations [Mili et al. 1995] [Benedicenti et al. 1996] [Sherif & Vinze 2003];
- the lack of knowledge and experience for the creation and management of reuse repositories [Morisio et al. 2002] and the definition, identification and evaluation of reusable assets, as well as making such assets available and findable [Frakes & Kang 2005];
- a long learning curve of understanding a software asset, i.e., its structure, behavior and functionality [Ye & Fischer 2002] [Marshall et al. 2003] [Frakes & Kang 2005];
- the lack of proper tool support for performing software reuse tasks [Mili et al. 1995] [Benedicenti et al. 1996] [Morisio et al. 2002];
- the absence of a culture of development for reuse in development teams and the lack of systematization for the construction of reusable assets [Sherif & Vinze 2003]; and

- the “Not-Invented-Here” (NIH) syndrome [Sherif & Vinze 2003], i.e., the difficulty of accepting and trusting third-party developed assets, resulting in a tendency towards “reinventing the wheel” (recreating something from scratch instead of reusing) based on the belief that in-house developments are inherently better than existing implementations.

Regarding the latter impediment, another study [Frakes & Fox 1995] pointed out that the NIH syndrome has become a minor obstacle, and has included reuse education and the perceived economic feasibility (among others) as factors that affect reuse, in accordance with [Card & Comer 1994]. However, although this syndrome has been alleviated over time, much of the phenomenon is caused by the cognitive difficulties that are inherent in the reuse process [Ye & Fischer 2000].

Many reuse-related issues can be associated to technical aspects, such as the lack of tools and techniques for effectively supporting software reuse, as pointed out by [Kim & Stohr 1998], [Lucrédio et al. 2008] and other works. Particularly, wrong technology choices may considerably hamper the execution of reuse processes [Lucrédio et al. 2008]. However, it is important to emphasize that solving these aspects is not enough for the success of a reuse program.

According to Card & Comer and Morisio et al., a misconception of the reuse needs may lead to the probability of neglecting the importance of assessing the reuse potential at the organizational level and addressing other barriers, treating reuse as a matter of technology acquisition [Card & Comer 1994] [Morisio et al. 2002]. Thus, as with any other software process, a crucial concern to take into account is the envisioning of non-technical aspects, e.g., engagement of team members and managerial support [Kim & Stohr 1998]. Sherif and Vinze highlight that reuse provides better results when all stakeholders are committed to it [Sherif & Vinze 2003].

A crucial concern is how to facilitate the acceptance/consciousness and adoption of reuse. Reuse stakeholders must be aware of the reuse results that are relevant to them and need awareness support for their reuse tasks. Monitoring activities, for instance, allows the early detection (and possibly resolution) of inconsistencies and shortcomings inside the software process, supporting and fostering the real integration of reuse paradigm into the existing software development process, encouraging continuous process improvement [Benedicenti et al. 1996]. Since there is a lot of information involved for performing reuse tasks, the lack of awareness and understanding of such information can hinder obtaining the expected results [Selby 2005] [Gill 2006]. This

non-technical aspect, however, can be handled partially with a proper support to the technical aspects.

2.4 The Importance of a Holistic Reuse Awareness

Introducing reuse practices in an organization nowadays may require new ways of thinking about software development, given that the way software is reused has changed over the years [Holmes & Walker 2012] [Bauer et al. 2014]. The rise and massive use of free/open source software repositories (e.g., BitBucket, GitHub), issue trackers/task managers (e.g., Bugzilla, JIRA, Redmine), release repositories (e.g., Maven Central), among others, have strongly influenced not only software development, but also software reuse.

In fact, reuse has become a data-intensive activity, due to the several sources of information available to which one can resort when performing certain reuse tasks. The listed resources provide several kinds of information about reusable assets, such as examples of use, tutorials, documentation, support, and so on. They also allow for increasing awareness of reuse activities.

A recent case that illustrates the importance of being aware of reuse-related information is an integer overflow vulnerability in the Lempel-Ziv-Oberhumer (LZO) algorithm that was only discovered after 20 years [Ouyang et al. 2014]. The bug fix should be backported to all the innumerous libraries and systems that have incorporated this algorithm for all these years (see [Lab Mouse Security 2014] for a list of examples), including the open-source Linux kernel[7] and its subsystems and variations.

Although it is nearly impossible to know all the libraries that reused this code since its release, this illustrates the importance of (i) enabling consumers to be aware of reusable asset issues as they are discovered, and (ii) enabling producers (or reuse managers, for organizations that have this role) to perform any relevant communication regarding the developed reusable assets. However, this can only be possible if proper information and mechanisms are available to these stakeholders.

It is noteworthy that there is a usual expectation regarding the analysis of the source code structure/behavior to help software reuse. This is indeed important and has been topic of several works (e.g., [Holmes & Walker 2012]). However, high level

[7] There are at least two commits related to this issue (http://git.io/vLRgH and http://git.io/vLRgd); the latter has a message stating, “the fix needs to be backported to all currently supported stable kernels”.

information that is strategic for decision making is often overlooked, in spite of the potential danger and additional expenses in both short-term (bugs found after incorporating an asset in a project) and long-term (lack of asset maintenance or asset discontinuation, preventing necessary upgrades, besides other bugs that may be found in the future). In fact, while finding an appropriate asset and understanding its structure and behavior can be tiresome, seeking for and understanding currently decentralized information that supports taking reuse decisions may become a major problem.

2.5 Software Reuse in Practice: The Brazilian Scenario

After performing the literature review on common issues in implementing software reuse processes, a search was performed for identifying reports concerning Brazilian organizations and characterizing the Brazilian scenario. The results are presented in the next subsections.

2.5.1 Literature reports on software reuse implementations

Sá et al. (1997) report the experience of introducing software reuse in an organization, by measuring aspects related to reuse before and after the implementation. The authors mention technical and cultural obstacles identified during the process. Some of them are: (i) reuse was only understood as code reuse; (ii) there was no technical or managerial commitment to produce reusable assets; (iii) most systems' development was going straight to the implementation phase, because stakeholders did not believe in Software Engineering as presented in the literature; and (iv) the view of profits was immediate (short-sighted) regarding the production of reusable assets [Sá et al. 1997].

Lucrédio et al. (2008) present a survey carried out with industry professionals, involving Brazilian organizations, aiming to relate organizational characteristics with the successful adoption of reuse. The authors did not analyze in depth the reasons why some organizations were not successful. The survey comprised several factors divided into four perspectives: organizational factors, business factors, technological factors, and processes factors. From the 200 contacted organizations, 57 answered the survey. As a result, the main influence factors identified include the development team, the use of tools and quality models, the prior development of reusable assets, the type of these assets, and the existence of a systematic reuse process. The difficulties encountered are

also related to these factors (e.g., an inadequate tool support and the lack of systematization of reuse represent negative influence factors) [Lucrédio et al. 2008].

Silva Filho et al. (2008) describe the implementation of the MR-MPS-SW Reuse Management (GRU) process at the Software Engineering Laboratory of an academic institution. Any software artifact (process asset, source code, or executable) could be considered as reusable assets; they were suggested by the team and evaluated against their quality and reuse potential. Notifications related to the assets' status were made manually by e-mail. The main difficulties mentioned were the definition of a non-intrusive strategy (i.e., which would not impact the usual activities of the organizational unit) and the choice of useful metrics to monitor and control the process. As to technical aspects, the identification of reusable assets was considered the most critical activity regarding the level of intrusion, cost, and effort. Among the lessons learned, the authors mention that the more mature the reuse management process is, the clearer the perception on how it can be automated [Silva Filho et al. 2008].

Santos et al. (2009) describe the experience on implementing MR-MPS-SW Reuse Management (GRU) and Development for Reuse (DRU) processes in a medium-sized, geographically distributed organization. The defined process for GRU is triggered either from the need to assess candidate assets or for implementing enhancements in a particular asset (based on problems or opportunities for improvement identified over time). A research is performed for identifying people potentially interested in a given reusable asset, as well as for defining the role responsible for maintaining such asset. An assessment of the reusable assets base is made periodically for identifying assets subject to discontinuation (e.g., criticized by users or less used). The authors underline that the tools used for supporting reuse were too general, such as text editors and spreadsheets. Regarding DRU, three identified areas of expertise were rated as having some potential for systematic reuse and, therefore, were analyzed in more detail. The assessment of the reuse capabilities of the organization showed that there were limited resources for the establishment of an appropriate reuse program, but a plan was drawn up to overcome this limitation. Nevertheless, DRU was not implemented fully due to the lack of both data and results on the development of reusable assets [Santos et al. 2009].

Although the literature reports on the implementation of reuse processes in the Brazilian scenario present some problems in common, they usually describe isolated cases, and do not aim at comprehensively characterizing usual problems identified during the implementation and assessment of reuse processes. The most comprehensive

one is the work of Lucrédio et al. (2008), but it is not based on a widely used quality standard or maturity model, i.e., it cannot be ensured that all the analyzed organizations perform a set of reuse tasks in common. This is one of the main motivations for conducting the study presented in the next subsections.

2.5.2 A study of the software reuse scenario in Brazil

According to [ABES 2014], in 2013, the software industry in Brazil had an increase of 13.5% on the investments compared to 2012. Overall, software and services grew by 10.1%, above the great majority of other sectors of the Brazilian economy. The use of computer programs developed in Brazil (standard and custom) increased 15.3%, higher than the 12.9% growth identified in the use of such programs developed abroad, reinforcing the trend of growth that comes been appointed since 2004 [ABES 2014].

Approximately 11,230 companies, directed to the development, production, and distribution of software and services, operate the domestic market. Finance, Services and Telecom accounted for almost 51% of the user market, followed by Industry, Government, and Commerce. Considering the size of companies engaged in developing and producing software (around 2,700 at the date of the report), these can be divided as micro (43.9%), small (49.6%), medium (5.2%) and large (1.3%) [ABES 2014].

The Brazilian scenario is very competitive (considering both nationwide and worldwide settings), and software reuse processes play an important role in this regard, due to its well-known benefits. Thus, it becomes important to characterize and obtain more information on reuse practices in Brazilian software organizations. In this sense, a study was performed with MPS.BR implementers and assessors[8], allowing to characterize a considerable subset of the nationwide scenario. The following subsections present an overview of the study planning and execution. Additional details can be found in a technical report [Schots & Werner 2014a].

2.5.2.1 Planning

The study goals are described in the Goal-Question-Metric (GQM) format [Basili et al. 1994] as follows:

[8] MPS implementers are affiliated to Implementing Institutions (II) accredited to render consulting services regarding the implementation of the MR-MPS-SW and MR-MPS-SV reference models, while MPS assessors are affiliated to Assessment Institutions (AI) accredited to render assessment services based on the MA-MPS Assessment Method. According to the MPS organizational structure, the MPS Accreditation Forum is responsible for accrediting such institutions [SOFTEX 2013b].

Analyze software reuse implementations

For the purpose of characterizing

With respect to usual practices, problems, challenges, and opportunities for improvement

Under the point of view of MR-MPS-SW implementers and assessors

In the context of Brazilian software development organizations

Since this study aims at characterizing software reuse in Brazilian organizations based on a set of outcomes in common, MPS.BR implementers and assessors compose the population of this study. The choice for this population is due to the fact that there is a representative number of MPS.BR assessments on level E (60 out of the 488 organizations successfully assessed in MPS.BR[9] are in level E or above, including 38 in level C or above[10]), covering a considerable portion of the nationwide scenario[11].

This population tends to be more homogeneous, since MR-MPS-SW organizations tend to perform a set of similar practices for achieving the necessary outcomes. On the other hand, a downside is the bias it may bring to the study, since this is a subset of the Brazilian software organizations (not being necessarily representative).

In order to obtain more information on the implementation of processes related to reuse in software organizations in Brazil, semi-structured interviews were conducted with the participants of the study. Such kind of interview represents a viable alternative when conducted to obtain or confirm information about a predetermined topic. They "are designed to elicit not only the information foreseen, but also unexpected types of information" [Seaman 1999], which meets our expectations with this study.

For conducting the interviews, some advice from [Seaman 1999] and [Hove & Anda 2005] was used in order to ensure good interaction between the interviewer and interviewees. For analyzing the collected data, the open coding technique [Seaman 2009] is used, by marking and categorizing snippets of interviews, relating them to questions (categories) initially defined.

---

[9] MPS-SW Published Assessments (data from August 23, 2013), extracted from http://www.softex.br/wp-content/uploads/2013/07/Avalia%C3%A7%C3%B5es-MPS-SW.pdf.

[10] It is noteworthy that the DRU process allows the exclusion of most outcomes from an assessment if the organization does not have opportunity and/or ability to perform development for reuse. Thus, one cannot state that all these organizations perform DRU.

[11] Please refer to http://www.softex.br/mpsbr/avaliacoes/mps-sw/mpsbr-ma-mps/ for an overview.

The interview questions were designed to obtain both technical (regarding the decisions for implementing the processes, based on the outcomes) and non-technical information (involving implementers' opinions concerning the assessed organizations, as well as difficulties and frequent problems) with respect to reuse processes. Some questions were directly derived from the MR-MPS-SW reuse outcomes. Table 2.2 shows the questions for Reuse Management (GRU), Development for Reuse (DRU) and other relevant questions for reuse processes in general, along with their corresponding goals. Other aspects related to the outcomes were not directly included in the questions, such as the control of changes in assets (related to GRU 4) and the criteria for acceptance, certification, classification, evaluation and discontinuity of assets (related to GRU 1), among others. These items are very specialized; thus, they were evaluated indirectly through the general questions and the intersection with other outcomes.

Table 2.2 – Interview questions

Questions related to Reuse Management (GRU)		**Goals**
Q1	Which kinds of assets have been considered as reusable by the organizations?	Identify which types of artifacts are considered as reusable by organizations in their projects/ processes. Related to GRU 1.
Q2	Where are the reusable assets usually stored?	Identify mechanisms (tools) used for storing reusable assets. Related to GRU 2.
Q3	Where/how are the reusable assets made available for reuse, i.e., where/how are the stored assets listed so that the interested parties can find them?	Identify the way organizations make their reusable assets available and the mechanism (tool) used to this end. Related to GRU 2.
Q4	How are the usage data about the assets logged?	Identify how organizations record reusable assets' usage data. Related to GRU 3.
Q5	How are interested parties informed of problems detected, modifications made, new versions released, and discontinued assets?	Identify the mechanisms used for notifying interested parties about changes in the status of assets. Related directly to GRU 5 and indirectly to GRU 4.
Questions related to Development for Reuse (DRU)		**Goals**
Q6	What are the application domains of the organizations in which opportunities for reusing assets have been identified, or in which they have intended to practice reuse?	Identify relevant application domains from the viewpoint of the state-of-the-practice. Related to DRU1.
Q7	Are organizations able to plan and establish an effective reuse program?	Check if reuse programs have been properly established in organizations. Related to DRU3 and DRU4.
Q8	How are organizations monitoring the reuse program?	Identify monitoring mechanisms and strategies being used by organizations. Related to DRU4.
Q9	How are reuse proposals (requests for reusing existing domain assets or developing/acquiring new ones) made?	Identify how reuse proposals are made and which kinds of request are more frequent. Related to DRU5.
Q10	How are domain models and domain architectures represented in organizations?	Identify techniques being used by organizations for representing domain models and domain architectures. Related to DRU6, DRU7, and DRU8.
Q11	How are domain assets specified/acquired/developed and maintained?	Identify techniques being used by organizations for specifying, acquiring, and/or developing domain assets. Related to DRU9.

General Questions on Reuse Processes		Goals
Q12	Which comments are made by the organizations regarding the GRU and DRU processes?	Characterize general problems pointed out by organizations. The answers to this question may drive the remainder of the interview for more details (funnel strategy).
Q13	What is the point of view of the diverse stakeholders (developers, project managers, top management) about reuse?	Identify whether there is any cultural resistance by stakeholders and, if so, which roles have such resistance. This information is also relevant for DRU4.
Q14	Which GRU and DRU aspects are more difficult to understand by the organizations?	Obtain more information about difficulties in understanding (including processes, concepts, tasks, tools etc.) pointed out by the respondent. This question is purposely broad.
Q15	Which are the most difficult tasks (particularly, GRU and DRU tasks) for the organizations to perform?	Identify information about the most difficult tasks.
Q16	What are the problems ("required" items) usually identified on GRU and DRU during assessments?	Identify issues that organizations cannot accomplish in GRU and DRU, as well as potential difficulties in implementations.
Q17	Which aspects related to the implementations or assessments of the GRU and DRU processes would you like to add (including the moment in the MR-MPS-SW implementation when you start to implement GRU and DRU processes, and potential difficulties in implementing or evaluating these processes)?	Identify difficulties on the implementations or assessments of the GRU and DRU processes, and ultimately verify how organizations prepare themselves to assessments.
Q18	Is there anything else that has not been asked and you would like to comment on?	Obtain feedback on the process and other aspects that participants would like to add.

2.5.2.2 Execution

Invitation e-mails were sent based on the list of authorized Implementing Institutions (IIs) and Assessment Institutions (AIs) available on the SOFTEX website[12]. The response rate in terms of the IIs and AIs was 38.46%. The criterion for participation in the study was the experience in the implementation and/or assessment of the GRU and/or DRU processes. Participants were interviewed in person during the XII Brazilian Symposium on Software Quality (July 1 to 5, 2013), or remotely, via Skype (between July 6, 2013 and August 25, 2013).

In total, there were 10 respondents, all concomitantly MR-MPS-SW implementers and assessors, having carried out (or accompanied, as leader assessors) at least 1 implementation or assessment of the GRU process (in most cases, more than 3 assessments). Figure 2.1 (left) shows the distribution of the respondents according to the MPS assessor levels (ordered from the lowest to the highest), while Figure 2.1 (right)

[12] Available at: http://www.softex.br/mpsbr/instituicoes-autorizadas/.

shows the year of authorization[13] to perform implementations and assessments of MR-MPS-SW.

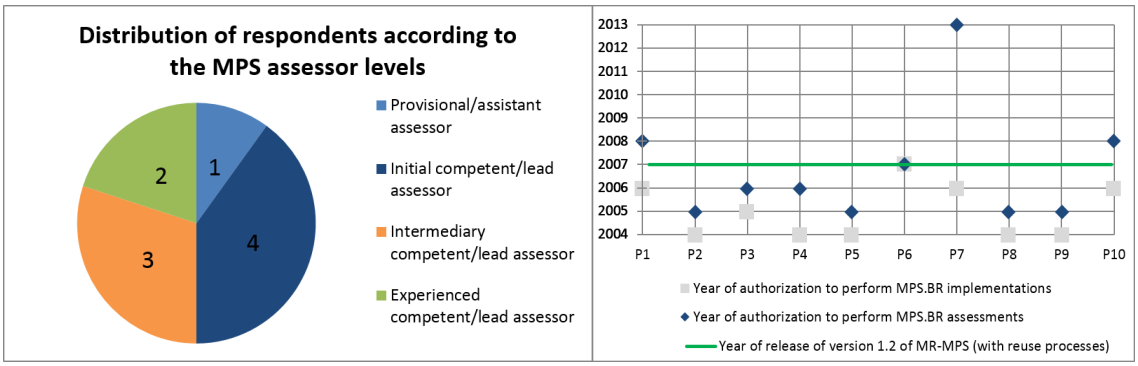


Figure 2.1 – Distribution of respondents according to the MPS assessor levels (left), and participants' experience (based on the year of authorization) in implementations and assessments (right)

As it can be seen, most participants are competent/lead assessors, meaning that they received a specific training from an assessment institution and performed at least 6 assessments as provisional/assistant assessors [SOFTEX 2013b]. Moreover, 2 of them are experienced competent/lead assessors – i.e., besides having competent/lead assessor's skills, they had a specific training on statistical process control and performed at least 4 assessments in levels E, D, and C as competent/lead assessors [SOFTEX 2013b]. Additionally, all the respondents were formed implementers before the release of version 1.2 of MR-MPS. Because a complementary training course is mandatory whenever substantial changes are made in the model, all of them were trained and are allowed to perform implementations of reuse processes from the moment such processes were incorporated into MR-MPS-SW.

More information related to the execution and details on the results can be found in [Schots & Werner 2014a], as well as the threats to validity.

2.5.3 Discussion of the findings

The kind of assets defined by the organizations as reusable (Q1) has a direct influence on the effectiveness and usefulness of reuse initiatives. For instance, one may not identify any benefit in monitoring kinds of assets that are rarely reused (because the visibility of benefits would be indeed impaired) or assets that do not have impact on the software lifecycle. One of the reasons why source code assets are considered more often as reusable assets may be because any issue related to it (e.g., bug or missing feature)

[13] Based on http://www.softex.br/mpsbr/profissionais-habilitados-2/.

may lead to problems in software maintenance and evolution. Thus, it is important to choose assets whose information regarding reuse, evolution, and discontinuation is relevant for the organization. Proper tools play a crucial role in this regard, especially in terms of collecting information about usage data (Q4) and notifying interested parties about changes in the status of assets (Q5). In fact, while the benefits of reuse are significant, many technical challenges remain and must be addressed to realize this potential fully. The management of data related to reuse processes is a major challenge: it is one of the most easily recognized through this study.

It was noticed that, in many cases, notifications about changes in the status of assets are triggered without any distinction of actual interested parties. This may compromise the effectiveness of communication (since information overloading may also adversely affect the perceived benefits of reuse), leading stakeholders to ignore important notifications. Another error-prone situation occurs when the maintenance of the list of interested parties and the sending of e-mails are performed manually (as in [Santos et al. 2009]). This can be partially due to limitations on tracking which team members reused which assets (i.e., collecting usage data), hindering communication.

The fact that some organizations store their reusable assets in version control repositories (Q2) has also been observed in other studies, such as [Morisio et al. 2002] and [Lucrédio et al. 2008]. In this regard, it is noteworthy that each type of repository has features aimed at ensuring the better functioning for their intended purpose, and an inappropriate technology selection can hinder the adoption and implementation of reuse processes. Reuse repositories and configuration management repositories have different purposes (for instance, the former is optimized for searching operations, and should only contain releases of assets), and this must be taken into account when instantiating a repository for an organization. Moreover, the institutionalization of a reuse repository requires proper mechanisms for exploration, search, and retrieval of assets (Q3), allowing potential consumers to obtain information that can be useful for deciding whether a given asset should be reused or not.

For an appropriate awareness of reuse scenarios and communication of results in an effective way, monitoring mechanisms are crucial. Most of the identified ways for collecting and logging information about usage data (Q4) are error-prone, being very dependent on people's feedback. Besides misjudging the importance of monitoring when conducting a reuse program, the lack of availability and trustworthiness of reuse-related data may be some of the reasons why organizations do not keep up with

monitoring practices. Mechanisms for data acquisition, cleaning, integration, aggregation, and representation play an essential role in this regard. In addition, due to the fact that people are the ones who make important decisions on reuse processes, such mechanisms must also account for facilitating analyses from the perspective of humans. Research on visual analytics may provide some guidance in this topic [Thomas & Cook 2006] [Keim et al. 2008].

Concerning Development for Reuse, many organizations were not yet able to implement all of the outcomes, and this makes it hard to draw any conclusions. However, it is believed that organizations should spend more time on domain analysis and, if necessary, ask for consultancy or expert assistance aiming at a better understanding of the gains of implementing this process, as well as identifying which tools best serve this purpose. Technical difficulties or lack of knowledge on a particular notation/methodology should not make organizations avoid this process. Additionally, it seems that industry claims for more evidence (academic and especially industrial) on the benefits of adopting Development for Reuse.

For organizations that do not have the chance to realize the benefits reuse can bring to them, opportunistic reuse may seem to be enough. However, it does not seem fair to assume that the organization itself is the root cause of the problem. Some issues pointed out by this study can be due to the lack of proper preparation (of the organization members, the process implementers, or a combination of both) for the implementation of reuse processes. Neglecting the importance of such processes, putting them at the end of the list of the processes to implement, is also an aggravating factor (among others). This can be either an implementation strategy or an organization's decision. However, this does not seem to be a good choice for the organizations: because of the delayed return on investments associated to reuse [Benedicenti et al. 1996], the benefits of reuse processes may take some time to arise and become noticeable; thus, the earlier they are implemented, the better.

Finally, top management needs more awareness and visibility of relevant information of the reuse processes (as pointed out by [Morisio et al. 2002]), being able to measure and control the impact of a software reuse program. In other words, the value of reuse must be somehow established and communicated to managers [Kim & Stohr 1998], so that they can be aware and become committed to reuse initiatives. Moreover, for a better acceptance of reuse-oriented changes in stakeholders' usual activities with less impact, suitable mechanisms must be identified and developed.

Particularly, because some necessary steps for implementing reuse may be challenging, organizations should try to accomplish them in a progressive way, in order to avoid resistance and allow for a better acceptance by the stakeholders.

2.6 Final Remarks

Since the beginning of software engineering, much research has been done in developing techniques and tools for supporting software reuse [Naur & Randell 1968] [Mili et al. 1995] [Frakes & Kang 2005]. In spite of that, many organizations still have difficulties in understanding and implementing reuse practices. As it can be noticed, many of the findings identified in the study match the literature reports both in the Brazilian and worldwide scenarios, particularly concerning the lack of adequate tool support and the need for more engagement in reuse initiatives. These are recurring problems.

Each day, a software developer needs to answer a variety of questions that require the integration of different kinds of information, and answering these questions can be hard when developers need to manually link and traverse such information step-by-step [Fritz & Murphy 2010]. This is also true for questions related to reuse tasks. The lack of available information not only has a large negative impact on the acceptance of the reuse benefits, but also hampers the proper execution of reuse tasks.

Every organization must keep up with the evolution of the assets they reuse, either developed by them or not. Therefore, it is important to improve their reuse capabilities proportionally to their current maturity stage. Otherwise, they are not likely to endure in the competitive market. Based on the study results, it is expected that some organizations can perceive the need to accurately perform software reuse practices and go a step further, so that they can achieve higher maturity on software reuse practices.

Table 2.3 lists the problems identified and the desirable features for approaches to help solving such problems (providing information for answering RQ2, stated in Section 1.3). These problems were extracted from the results of the performed study [Schots & Werner 2014a]. Whenever another publication points out the same problem, it is cited within the problem listing.

Table 2.3 – Findings and assumptions derived from the semi-structured interviews

ID[14]	Description	Desirable feature
RF1	Source code assets are the most common kinds of reusable asset found in organizations [Haefliger et al. 2008], and managing their reuse is crucial, since any issues (e.g., bugs) not only affect asset consumers, but can also be perceptible by end users of products that incorporate such assets. This ripple effect makes software maintenance even more arduous and challenging.	For properly supporting software reuse tasks, the approach should primarily support managing source code assets.
RF2	Organizations are free to define the kind of assets to be considered as reusable [SOFTEX 2013a], ideally choosing the ones whose information is relevant for them.	The approach should also support different kinds of reusable assets (assuming that there is a corresponding reuse repository with relevant information about them).
RF3	Organizations should be able to track which consumers reused which assets, in order to communicate any problem identified in such assets to their consumers. However, most of the identified ways for collecting and logging information about usage data are error-prone, being very dependent on people's feedback[15]. Besides, organizations must keep up with the evolution of assets they reuse, either developed by them or not.	The approach should provide a way of collecting information regarding reuse (consumption), evolution, and discontinuation of assets, along with the developers involved in the production and consumption of these assets.
RF4	In many cases, organizations trigger notifications about changes in the status of assets without any distinction of actual interested parties[16], due to limitations in the collection of usage data.	The approach should help identifying potential interested parties of an asset based on reuse data and notifying such parties about changes in the status of the assets.
RF5	Some organizations store their reusable assets in version control repositories instead of reuse repositories [Morisio et al. 2002] [Lucrédio et al. 2008].	The approach should provide a reuse repository for the organization, or integrate with an existing one, that allows potential consumers to obtain reusable assets and relevant information about them.

[14] RF refers to "Reuse-Related Finding", while RA means "Reuse-Related Assumption".

[15] In the semi-structured interviews conducted, two ways mentioned by the respondents are highlighted. In one of them, the reuse manager is responsible for capturing such information by analyzing software projects and searching for reuse occurrences, storing results in an Excel spreadsheet or a kind of list. The other way requires the project manager or the development team to inform the reuse manager in case any project reuses an asset [Schots & Werner 2014a].

[16] E-mails are sent manually, either based on a list of interested parties that is maintained manually based on the usage data (which are inaccurate, as mentioned) or to all members of the organization (irrespective of being interested parties) [Schots & Werner 2014a].

ID[14]	Description	Desirable feature
RF6	All the important decisions related to reuse are made by people; thus, there is a need for appropriate awareness in order to facilitate analyses and communicate results in an effective way. The value of reuse must be somehow established and communicated to managers [Kim & Stohr 1998], so that they can be aware and become committed to reuse initiatives.	The approach should present concise information that can help stakeholders in establishing and monitoring the progress of reuse initiatives in the organization, through mechanisms that provide adequate awareness of the reuse scenario.
RA1	For a better acceptance of reuse practices in stakeholders' usual activities, it becomes necessary to identify and develop mechanisms for meeting the specific needs of each of them.	The approach should provide mechanisms with different perspectives to support each stakeholders' needs related to reuse.
RA2	Because some necessary steps for implementing reuse may be challenging, organizations should try to accomplish them in a progressive way, in order to avoid resistance and allow for a better acceptance by the stakeholders.	In order to minimize cultural barriers and allow all stakeholders to become committed with reuse initiatives, there should be a strategy for a gradual introduction of the approach mechanisms, avoiding cognitive overload.
RA3	Integrating different sources of data can provide relevant information about the reuse scenario, especially in terms of giving more confidence to a consumer in deciding whether or not to reuse an asset. The lack of information regarding the assets may inhibit developers to reuse them. This is especially true with respect to assets not developed in the developers' organizations.	In order to show relevant information about the reuse scenario as a whole, particularly providing a better perception of the assets' stability and quality, the approach should collect data from different kinds of source, integrating information from reuse repositories, version control repositories, and change control (bug tracking/task manager) repositories.
RA4	Reuse tasks require handling a large amount of data, which requires the application of mechanisms to represent reuse information, enabling to interact with and manipulate the data, as well as obtain answers to reuse tasks quickly.	The approach should handle a large amount of information through adequate abstractions and interaction techniques.
RA5	For stimulating reuse initiatives in a software development organization, assets reused opportunistically (usually from open source repositories) should be evidenced, in order to demonstrate that the organization already performs some kind of reuse. Likewise, assets developed by the organization should also be taken into account.	The approach should provide the option of tracking reusable assets (and projects), both open source and developed by the software organization.
RA6	Developers do not have tool support to identify candidate assets to be included in the reuse repository.	The approach should integrate with and collect information from version control repositories for suggesting assets that occur in more than one project. This allows a later evaluation for their inclusion on the reuse repository. Collecting usage data and properly identifying producers and consumers help support such decision.

Attempting to implement strategic reuse (and development for reuse) in any organization (regardless of its size) may be useless if its members do not actually perceive the benefits and gains of simpler reuse practices, such as reuse management and the integration of software reuse tasks into the development process. In this sense, it is important to provide reuse managers and developers with proper support from the beginning of reuse initiatives.

To this end, the application of perception and awareness techniques can be useful. For instance, visualization metaphors can represent reuse information, so that users can interact with and manipulate the corresponding data, as well as obtain answers to reuse tasks more quickly, besides decreasing the cognitive overload. Software visualization resources and techniques play an important role on awareness and comprehension, and can be used for supporting a software reuse program, especially in terms of software reuse tasks. This topic is covered in the next chapter.

CHAPTER 3 – SOFTWARE VISUALIZATION

This chapter presents the main concepts related to software visualization. It also describes how visualization resources and techniques can benefit software reuse, as well as how existing works identified from the state-of-the-art have been addressing this issue.

3.1 Contextualization

The large amount and diversity of data generated throughout software development is often difficult to manage and monitor. Organizations have sought for techniques that allow not only to store and process such data, but also to exploit them in order to extract relevant information to support decision-making processes and allow increasing the quality of their services, processes, and products.

The quality and relevance of decision making heavily depend on the understanding, interpretation, and aggregation of organizational data; such factors can become critical while implementing and evaluating organizational strategies, thereby becoming a competitive edge. There is a need for appropriate models and mechanisms for analyzing and monitoring data about software processes and products, as well as studies on how the available resources can support understanding such data.

If data sources with evolution information of software development, such as repositories of version control systems (VCS), are also taken into account, software gains a dimension in time, which increases even further the mass of generated data. In addition to that, there are other data sources, such as issue trackers, measure databases etc., which bring a greater diversity on the nature of data. In order to deal with this scenario, software development requires appropriate mechanisms and tool support that can assist in the extraction and analysis of these data and allow their understanding [Schots et al. 2012]. However, such understanding is not an easy task.

According to Diehl (2007), 75% of all information from the real world is perceived visually [Diehl 2007]. On the other hand, as stated by Brooks Jr. (1987), software is very difficult to visualize; the reality of software is not inherently embedded in space [Brooks Jr. 1987]; hence, it has no ready geometric representation. One of the obstacles for visualizing software information is that data are abstract and, therefore,

have no associated physical structure [Chen 2006]. Thus, it is necessary to consider (i) the use of visual abstractions that are appropriate to the nature of data and their relationships, (ii) representation techniques that allow to emphasize what is relevant in a given context, and (iii) different forms of interaction, allowing to perform exploratory (and, therefore, richer) analyses [Schots & Werner 2012] [Schots et al. 2012].

In this context, software visualization techniques aim to provide a better and faster understanding of the structure, behavior, and evolution of software processes and products [Diehl 2007]. It can be defined as the use of information visualization techniques [Chen 2006] applied to software, i.e., as a branch of information visualization. Data are represented by means of visual metaphors for facilitating the comprehension of different scenarios and contexts, as well as the detection of underlying patterns and the creation of analogies [Lanza & Marinescu 2006] [Diehl 2007].

Software visualization has been exploited as a way to assist software development activities that involve human reasoning, helping people to deal with the large amount and variety of information by providing appropriate abstractions. Software visualization research focuses on the use of computational resources for accelerating and optimizing users' perception, understanding, and assimilation of information *of* software and *about* software, by stimulating the human cognitive capacity (derived from users' memory, perception and reasoning). Perception is the processing of sensory information and thus part of human cognition, which also includes awareness, reasoning, and learning [Lanza & Marinescu 2006] [Diehl 2007].

Several strategies and techniques have been proposed and developed for the representation and interaction with the visual metaphors. Some of these techniques are listed in a previous work [Oliveira 2011], and a more comprehensive set can be found in [Schots et al. 2015]. Software visualization tools make use of these techniques in order to provide a richer representation and exploration of the underlying data, thus better supporting software comprehension and correlated tasks.

In this regard, Diehl (2007) states that, in order to make visualizations effective in their goal, it is important to keep in mind that the visual metaphors and representations to be used must be adapted to the stakeholders' perceptive abilities, not the opposite (as it usually occurs) [Diehl 2007].

Several software engineering fields can be supported by visualizations. Some include requirements engineering [Cooper et al. 2009], software architecture and design

[Lanza & Marinescu 2006] [Gallagher et al. 2008] [Schots et al. 2010], software measurement [Lanza & Marinescu 2006], software evolution [Wettel & Lanza 2008] [Werner et al. 2011], software maintenance, reverse engineering and reengineering [Koschke 2003] [Telea et al. 2010], among others. Software engineering education can also benefit from the use of visual metaphors and other interactive approaches to allow exploration of concepts and enhance learning [Rodrigues & Werner 2011]. One can also highlight the inherent multidisciplinary of the software visualization topic, since it integrates several computer science disciplines, such as data mining, software engineering, computer graphics and human-computer interaction.

Mukherjea & Foley state that visualization is particularly important for allowing people to use perceptual reasoning (rather than cognitive reasoning) in task-solving [Mukherjea & Foley 1996]. In this sense, in addition to the usual understanding goal, it is desirable to make an explicit description of the supported tasks, for facilitating potential users with matching information needs in identifying the visualizations easily.

3.2 The Role of Visualization in Awareness and Comprehension

Since research in software engineering is steadily expanding and investigating different methodologies, processes and techniques, it is also necessary to provide stakeholders of the software development process with a sense of what happens in the scenario in which they are involved, as well as means to explore and understand software artifacts of interest and their properties [Schots et al. 2012]. This requires appropriate awareness and comprehension resources.

Although these concepts are correlated, there is a subtle difference between them. According to [Shi et al. 2011], awareness is "the state or ability to perceive, to feel, or to be conscious of events, objects or sensory patterns", but in this level of consciousness, an observer can confirm sense data without necessarily implying understanding or comprehending. Similarly, program comprehension also encompasses the software development life cycle, but it focuses mainly on software artifacts, rather than the process and its variables. In other words, awareness is related to cognitive reactions to a condition/event (being aware of it), while comprehension involves assimilation of knowledge (understanding a fact) [Schots et al. 2012].

Enhancing awareness and understanding of software information and the software itself requires the identification of adequate abstractions according to the comprehension needs [Schots et al. 2012]. The choice of the visualization abstractions

and techniques for representing the data, as well as the interaction techniques to be employed, heavily depends on contextual information, e.g., the nature of data, the visualization constraints, and the task to be supported (e.g., selecting the most suitable assets from a set of reusable assets). Awareness and comprehension concepts are discussed with more details in the next subsections, along with a brief argument on the role of visualization.

3.2.1 On the awareness of the software development life cycle

The concept of awareness is present in many of today's systems. Context-aware systems offer new opportunities for application developers and for end users by gathering context data and adapting systems behavior accordingly [Baldauf et al. 2007]. In the software development scenario, awareness can be characterized as "an understanding of the activities of others, which provides a context for one's own activities" [Dourish & Bellotti 1992]. Many researchers have recognized awareness as an essential part of collaborative software development and collaborative work in general [Treude & Storey 2010].

Awareness mechanisms allow software development stakeholders to be percipient of what goes on in the development scenario. Each mechanism has its specific purpose, i.e., aims at supporting a particular set of development tasks (e.g., providing information about the detection of potential conflicts in collaborative development, supporting parallel tasks in geographically distributed development, and so on), thus providing different levels of awareness according to the context. Moreover, as stated by [Treude & Storey 2010], depending on the context of the task at hand, the required granularity of awareness can vary significantly.

The inclusion of awareness mechanisms should take into account the tools most commonly used by stakeholders in their usual, daily activities, in order to ease the adoption and use of such mechanisms. For instance, among software developers, the use of IDEs is very common, and most of them are extensible by plug-in systems. Thus, developing awareness facilities as IDE plug-ins can benefit from the available IDE features, including integration with other tools [Hattori 2010] [Schots et al. 2012].

The use of visualizations can enrich development environments to promote awareness [Hattori 2010]. Awareness information can be delivered by means of visual resources especially employed to this end, e.g., dashboards [Treude & Storey 2010] that can summarize important development facts. One important aspect that must be taken

into account is the evaluation of the tradeoff between the usefulness of the visual cues and the level of distraction they may cause [Hattori 2010].

In [Ripley et al. 2007], a 3D (three-dimensional) visualization is presented for providing project managers with a comprehensive view of all project activities, allowing them to intelligently steer development and adjust task assignments. The screenshot shown in Figure 3.1 presents a snapshot of all ongoing changes taking place in a set of workspaces at a particular time [Ripley et al. 2007].

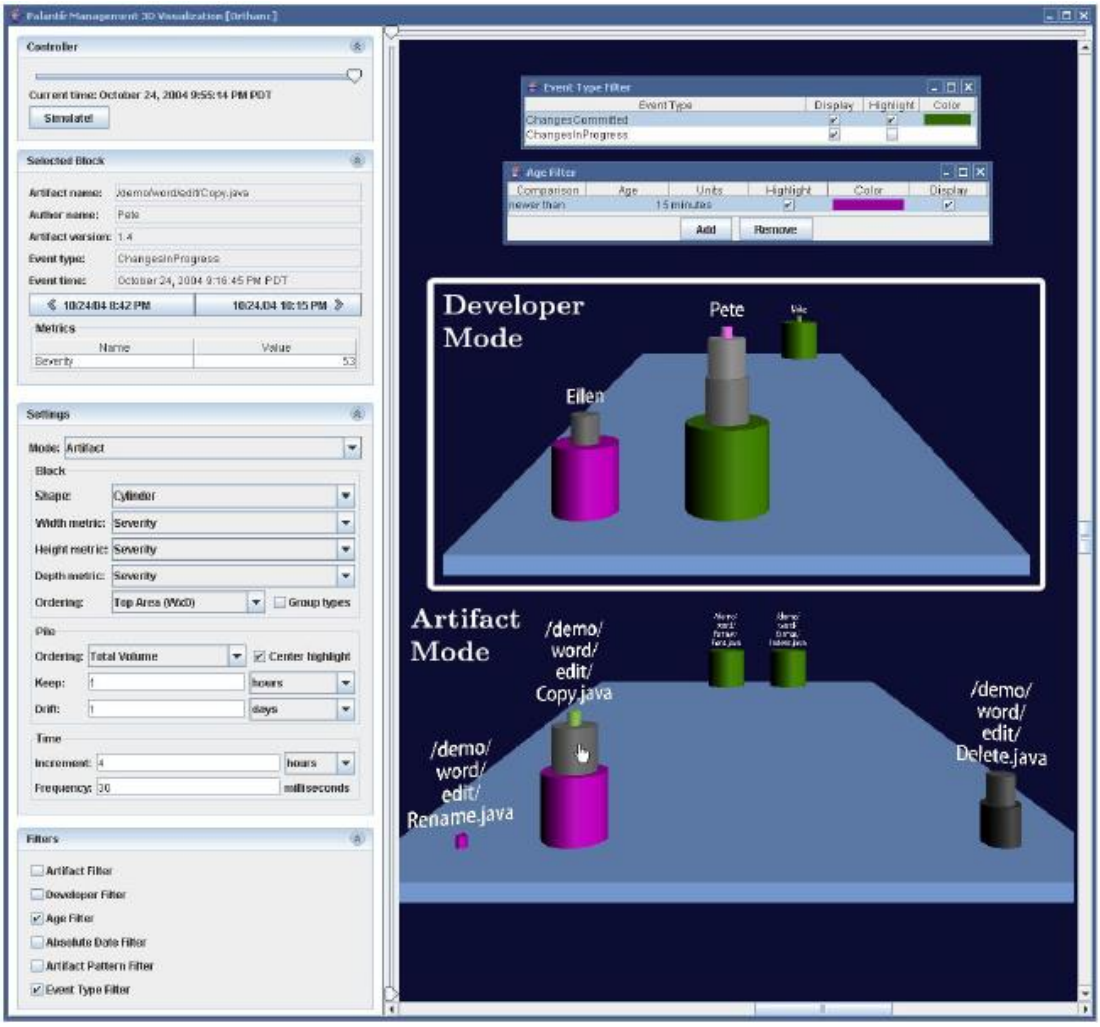


Figure 3.1 – 3D workspace visualization [Ripley et al. 2007]

The stacks of cylinders with the most recent changes are placed in the front of the view and, as time elapses, stacks for workspaces with less recent activity start moving to the back. In the artifact mode, each stack of cylinders represents an artifact, and each cylinder in the stack represents changes to that artifact made by a workspace. In the developer mode, a stack of cylinders represents a developer's workspace.

Another subarea that has emerged is visual analytics, considered as "the science of analytical reasoning facilitated by interactive visual interfaces" [Thomas & Cook

2006]. Its goal is to increase insight into data through the combination of automatic analysis methods with human background knowledge and intuition [Keim et al. 2008].

3.2.2 Program comprehension and visualization

Program comprehension is a vital software engineering activity. It is necessary to facilitate reuse, inspection, maintenance, reverse engineering, reengineering, migration, and extension of existing software systems [Wong et al. 2007], among other software engineering practices. Particularly, it plays a crucial role in software maintenance: according to [Telea et al. 2010], about 40% of the maintenance budget is used for understanding source code.

The mapping of entities, from the software systems domain to graphical representations, aims to support comprehension and development [Gallagher et al. 2008]. In fact, many works that aim at increasing program comprehension make use of visual metaphors, by applying software visualization concepts and techniques. Such works usually try to represent software through a particular point of view, helping stakeholders to focus on the tasks being performed. Duru et al. (2013) state that software visualization tools allow users to synthesize and make sense of vast amounts of information (e.g., regarding the inner organization of software modules and their interactions) [Duru et al. 2013].

An illustrating example of a software visualization tool is the CodeCity tool [Wettel & Lanza 2008], presented in Figure 3.2. It displays source code information mapped into a city metaphor. The visual properties of the city artifacts reflect metric values of classes and packages (in the figure, the number of methods maps to the buildings' height and the number of attributes to their base size). The brown (darker) buildings represent the classes and the blue (lighter) districts represent the packages.

This figure illustrates a version of a system called Jmol. The visualization allows to easily identify outliers, such as the two large platforms (wide and short) in the foreground representing the classes Token and JmolConstants, which define many attributes (large base) and few methods (reduced height), or the "skyscraper" representing the class Viewer with a considerably high number of methods and a much lower number of attributes [Wettel & Lanza 2008].

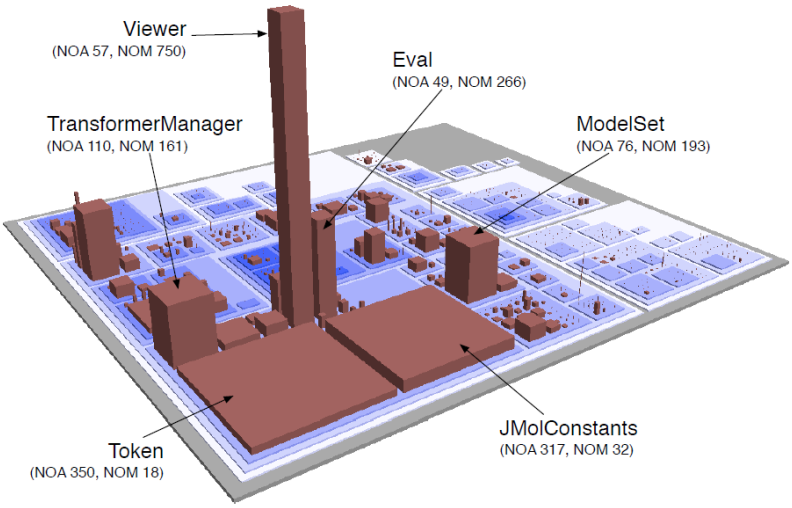


Figure 3.2 – The city metaphor presented in CodeCity [Wettel & Lanza 2008]

Another illustrating example is EvolTrack [Werner et al. 2011], an Eclipse-based extensible mechanism that combines multiple views to provide a better comprehension of the software evolution life cycle through different viewpoints. Its data source and visualization plug-ins allow performing different comprehension tasks, e.g., architectural conformance and co-evolution analyses, social network analysis, and tracking evolution of measurements [Werner et al. 2011].

Figure 3.3 shows EvolTrack and its plug-ins PREViA (for architectural model conformance analyses over time) and SocialNetwork (for socio-technical network analyses) [Werner et al. 2011].

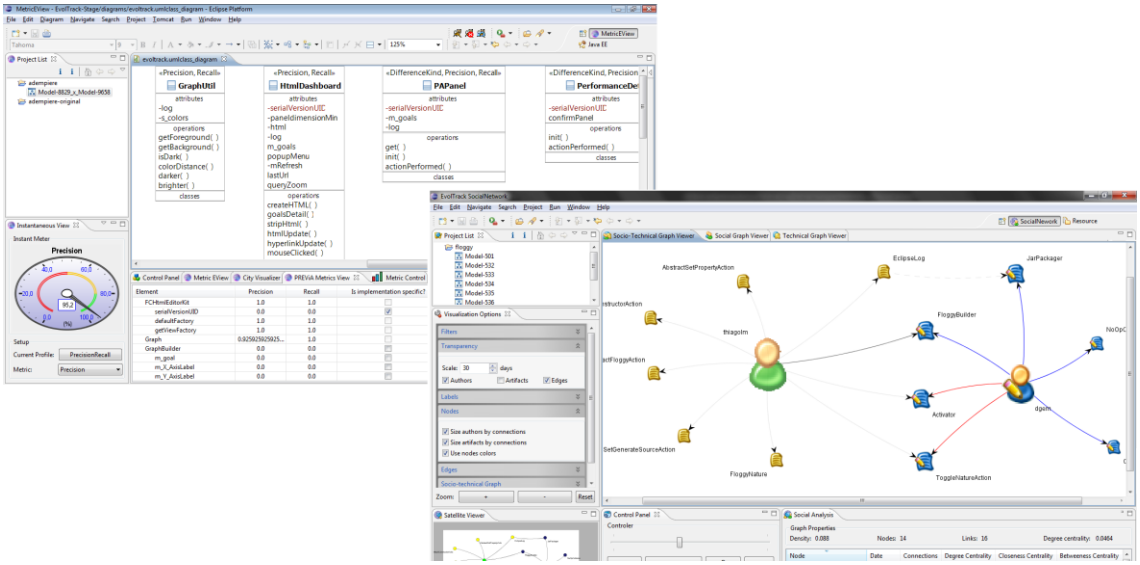


Figure 3.3 – EvolTrack and its plug-ins PREViA and SocialNetwork [Werner et al. 2011]

3.2.3 Awareness and comprehension challenges

The creation of tools, techniques, and methodologies to support the manipulation of large data sets has been receiving special attention of both scientific and industrial communities, in order to discover new ways of dealing with the underlying information, including learning purposes, identification of patterns, decision-making support, among others. However, making use of computing resources to enhance awareness and understanding (of software information and the software itself) is still a challenge in software/systems engineering. It involves the identification of suitable mechanisms, adequate abstractions, and studies on stimulation of the human perceptive and cognitive abilities [Schots et al. 2012].

Among the grand challenges identified by the Brazilian Computer Society for the years 2006-2016 [Brazilian Computer Society 2006], the following somehow relate to these topics:

- The management of information in large volumes of distributed multimedia data, in order to develop solutions for the processing, retrieval and dissemination of relevant information, both narrative and descriptive, from the exponential growth of multimedia data;
- The computational modeling of complex systems (artificial, natural and socio-cultural) and human-nature interactions, particularly the creation of new algorithms and techniques in scientific visualization to enable visually capturing the complexity of the modeled objects and their interactions;
- The quality of technological development, which poses that systems must be available, accurate, secure, scalable, persistent, and ubiquitous – one of the research topics in this sense is the development of tools for supporting the process of implementation and evolution of software.

There is an increasing demand on how to obtain, handle/process, visualize, manipulate, and understand information, particularly data and information about software systems. Research topics that tackle this issue have the potential to deliver promising results, as well as ease and increase the quality of software processes and products. In this sense, some challenges in awareness and comprehension highlighted in [Schots et al. 2012] are listed as follows:

- General comprehension/awareness challenges
  - *Use software tools to seamlessly collect rich data sets on software comprehension activities*: Kagdi & Maletic (2008) highlight the importance of automatic data collection mechanisms (e.g., eye tracking and activity logging) on software comprehension studies [Kagdi & Maletic 2008]. This can be cheaper and more reliable (with respect to data quality) than questionnaires, interviews, and think aloud protocols. IDEs and VCS repositories provide means to collect this type of data. The challenge lies in associating low level (fine-grained) monitoring of actions on software engineering tools with the cognitive actions being executed (e.g., reading, searching or modifying the code). Data mining techniques (sequence and process mining) can help to achieve this.
  - *Build specialized, personalized visualizations according to the comprehension needs*: Software visualizations should present a comprehensive view of the objects under analysis, based on the needs identified in industry and education.
  - *Provide evidence regarding correlations involving people's profiles with respect to quality attributes on program comprehension*: There is little evidence on how (and whether) previous knowledge, skills, and abilities correlate with the efficiency and efficacy of understanding program artifacts. An example is to evaluate the influence of developers' level of expertise on how efficiently they understand code [Von Mayrhauser & Vans 1995]. Studies on stimulation of the human perceptive and cognitive abilities are welcome for understanding scenarios like this [Novais et al. 2012].
  - *Identify and develop suitable mechanisms and adequate abstractions*: If an awareness/comprehension mechanism is not useful in its purpose, it will not be used in practice. It can be, among other reasons, due to its lack of flexibility and integration with other mechanisms. Thus, new tools and visualizations need to consider this.
  - *Strengthen and increase the group of researchers interested in software visualization, awareness, and related areas*: Despite the large number of studies undertaken involving software visualization, the Brazilian community in the area is still scattered. Attempts for establishing a joint research agenda are already in progress, aiming to allow the construction of a collaborative body of knowledge

on software visualization and awareness, besides providing relevant solutions to the community as a whole.

- Industry-related challenges
  - *Understand the real needs of the software development industry stakeholders in terms of awareness and comprehension*: As one of the responsibilities of academia is to provide solutions to existing problems in industry, more studies should be conducted for identifying research opportunities. An example would be by performing primary studies, such as surveys and action-research.
  - *Evaluate the quality of existing data sources and identify relevant data*: The industry is increasingly realizing the importance of having data on the execution of their processes and metrics regarding the product, so that such data can be used to improve the performance of their activities and the quality of the final product. However, it is necessary to ensure that (i) the data are collected, (ii) the data collected are useful and appropriate, and (iii) the data collected allow the analysis and improvement of processes and products.
  - *Bridge the gap and encourage interaction between academia and industry*: Though this challenge is also pertinent for several other areas, research in software visualization and awareness lack evidence of their theories through results of studies performed in real settings. Research initiatives involving industry people with flexible formats could serve as a first step in this direction. Some potential results of such initiatives are the establishment of partnerships and exchange of human resources towards a holistic training for both communities.

As it can be seen, these challenges comprise software/systems engineering in general. This work partially addresses some of them with a focus on software reuse.

3.3 Software Visualization and Reuse

As mentioned in Section 2.4, introducing reuse in an organization may require new ways of thinking about software development. In order to achieve the acceptance/consciousness and successful adoption and institutionalization of software reuse, it is important to take into account how to provide appropriate reuse awareness. Awareness mechanisms allow stakeholders to be percipient of what goes on in the development scenario [Treude & Storey 2010] [Schots et al. 2012], and can provide

them with the necessary information and support for performing their reuse-related tasks.

One of the ways to increase reuse awareness is by employing visualization resources and techniques. It is known that, in general, every visualization system supports understanding of one or more aspects of a software system, and this understanding process in turn supports a particular engineering activity or task [Maletic et al. 2002], such as requirements engineering, software design, or coding. It is believed that most of these software engineering tasks can also be visually supported by software reuse. Visualization resources can be used for allowing awareness and comprehension of reuse elements and their surroundings.

For instance, to reuse a software asset, stakeholders need to understand what it does, how it works, and how it can be reused; however, this is difficult in practice [Marshall 2001] [Marshall et al. 2003]. If software engineers cannot understand assets, they will not be able to reuse them [Frakes & Fox 1996] [Alonso & Frakes 2000]. In contrast, a proper understanding can help developers to decide whether and how the asset can be reused [Marshall 2001] [Marshall et al. 2003], and visualization may play an important role in this context.

Several works aim to assist software engineering stakeholders in their day-to-day activities, but little is known on the role of visualizations in supporting software reuse tasks. Although existing visualization approaches intend to support somehow software reuse, literature lacks of a solid and comprehensive body of knowledge of software visualizations targeted to reuse. Consequently, stakeholders may not be able to choose reuse-oriented visualizations properly (i.e., based on their quality and concrete evidence on their actual effectiveness) for a given scenario.

In this sense, an informal literature review (first step of the research methodology presented in Section 1.5) was conducted for collecting preliminary information and providing the initial/basic knowledge about the research topic. This served as a basis for a secondary study (a *quasi*-systematic review), i.e., a more comprehensive study for characterizing the state-of-the-art (second step of the research methodology), aiming to identify software visualizations targeted to support reuse-related tasks.

All the details about the secondary study, including the full protocol description and the details on the analysis, are described in [Schots et al. 2014]. The next subsections present the approaches and tools identified by means of the informal

literature review, as well as a framework that was created for categorizing visualizations and a brief overview of the planning and execution of the *quasi*-systematic review, as well as the discussion of results.

3.3.1 Findings from the informal literature review

During the informal literature review, a number of works related to visualization and reuse were found, but some of them are not related to software development (e.g., [Klerkx et al. 2006]). For the sake of scope, it was decided to focus the analysis of related works on software development.

Dy-re (Dynamic reuse) [Biddle et al. 1999] supports programming for reuse by displaying dynamic information of the internal structure of the software under development. It aims to make it easy to detect patterns of usage and patterns of dependence within a program – these patterns may help the programmer to determine how best to articulate the structure of a program using components that will be useful and independent for later reuse in other contexts [Biddle et al. 1999].

Dyno [Biddle et al. 1999] [Marshall 2001] [Marshall et al. 2003] is a tool for helping developers in reusing Java code, by means of a view based on their experience of using such code. It allows the use of visualization templates written in Java, which can be generic (for any data type) or specific (for certain data types). Developers can write their own templates. According to [Marshall 2001], the developer himself/herself must map visualizations to data, i.e., must inform "which method in the component maps to which sequence", and this can be a one-to-one or a many-to-one mapping. The author recognizes that this can be a problem, since "a developer may not know enough to know which methods should map to which sequence" [Marshall 2001].

Alonso and Frakes (2000) propose an architecture for visualizing reusable components from a software library, along with an example implementation. The architecture is based on two architectural styles: (i) pipes and filters, and (ii) repository. The repository stores and manages the assets and their metadata; the visual representation displays the data using a visualization metaphor; finally, the intermediate representation enables data interchange between the repository and the visualization [Alonso & Frakes 2000]. There is a strong dependency of the search input query, i.e., the usefulness of the results is closely related to the quality of the search.

The Variant Analysis approach [Duszynski et al. 2011] focuses on recovering and visualizing information about commonalities and differences in the source code of

multiple similar software systems (delivering quantitative information about similarities across system variants). By identifying parts suitable for transformation into reusable assets and planning necessary implementation steps, it aims at supporting the reuse potential assessment and the migration to systematic software reuse, besides providing an overview of commonality distribution in the whole analyzed system family.

These publications compounded an initial data set that served as a preliminary input of control for the *quasi*-systematic review. In addition, the need for organizing the information from the findings for further analyses motivated the extension of a framework for categorizing visualization approaches [Schots & Werner 2014b]. This framework is presented as follows, instantiated to the software reuse scenario.

3.3.2 An extended framework for categorizing visualization approaches

In order to identify the set of data to be extracted from the findings, this work uses the five dimensions of software visualization from [Maletic et al. 2002]. The task-oriented framework proposed by these authors takes into account previous work on taxonomic descriptions for emphasizing general tasks of understanding and analysis during the development and maintenance of large-scale software systems. The framework dimensions reflect the why, who, where, what, and how of the software visualization, as follows [Maletic et al. 2002]:

- **Task:** A visualization system aims at supporting the understanding of one or more aspects of a software system, and this understanding process will in turn support a particular task. Thus, this dimension indicates what particular software engineering tasks are supported by the visualization.
- **Audience:** This defines the attributes of the users of the visualization system. Besides being oriented to distinct roles, different tools can also be tailored towards users with different skills (e.g., experienced versus beginner, developer versus manager etc.). An experienced developer may have different information needs other than a novice team member.
- **Target:** The target of a software visualization system defines which (low level) aspects of the software are visualized, i.e., the work product, artifact, or part of the environment of the software system. Examples include architecture, design, algorithm, source code etc. Other types of target are software metrics, process information, and documentation; this can support the software process and team

management activities. Software development surroundings also provide several aspects that can be visualized.

- **Representation:** This dimension shows how the visualization is constructed based on the available information. An aspect on which the effectiveness of information visualization hinges is its ability to represent information clearly and accurately. The relationship between data values and visual parameters should be univocal[17]; otherwise, it may not be possible to distinguish one value's influence from the other.
- **Medium:** The effectiveness of visualizations also relies on the humans' ability to interact with them to figure out what the information means. The medium is where the visualization is rendered, i.e., some display technology from which the user interacts and perceives the visualization. The medium dictates how interactions may occur; each one has different characteristics and hence is suited for different tasks.

In order to complement the framework with information that is relevant to the visualization users, as well as encompass other aspects related to the findings of this study, two additional, complementary dimensions that are not (or at least not directly) addressed in the original framework are proposed and used in this work. One of them related to the **requirements** of the visualization approaches (***which***) and the other is related to **evidence** on their use (***worthwhile***).

Figure 3.5 depicts the software visualization dimensions. Each dimension maps to a secondary question (*SQ*) shown in Section 3.3.3. Additional details on this extended version of the dimensions and the mapping to the corresponding information fields are described in [Schots et al. 2014] and [Schots & Werner 2014b].

The specific data to be extracted are described in the data extraction form, presented in Table 3.1. This form enables to record full details of the publications under review, besides supporting the construction of an online repository [Schots 2014c] with details about such publications, in order to allow a richer exploration of the findings, as well as establish correlations between the visualization dimensions. The Google Spreadsheets tool[18] is used for supporting the data extraction process. The data extraction fields are identified with their corresponding research questions.

[17] It must be emphasized that the visual encoding is not univocal in some visualizations, e.g., when the defined categories are not mutually exclusive.

[18] http://spreadsheets.google.com/

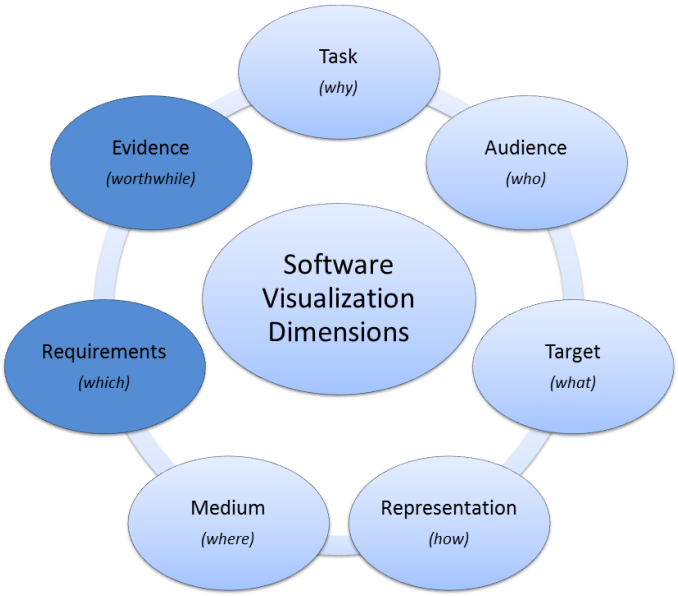


Figure 3.4 – Software visualization dimensions (extended from [Maletic et al. 2002]) [Schots & Werner 2014b]

Table 3.1 – Data extraction form

	Field	Information to be extracted
Publication metadata	**Title**	*[Publication title]*
	Authors	*[List of authors separated by comma, e.g., "Singh, S., Cheung, L. K. Y." – "et al." must be avoided]*
	Publication date (year/month)	*[Year and month of publication, e.g., "September 2000"]*
	Publication type	*[Conference or Article (Journal)]*
	Source	*[Source of the publication, e.g., "Communications of the ACM" or "Proceedings of the International Conference on Software Engineering (ICSE 2007)"]*
	Volume and Edition (for journals)	*[Volume and edition, e.g., "v. 49, n. 10"]*
	Place (for conferences)	*[City and Country of event, e.g., "Washington, USA"]*
	Pages	*[Initial and final pages separated by hyphen, e.g., "pp. 184-191"]*
	Link (if applicable)	*[Link to the publication, preferably the Digital Object Identifier (DOI), e.g., "http://dx.doi.org/10.1109/ICSECOMPANION.2007.8"]*
	Abstract	*[Full abstract text]*
Visualization metadata	**Approach/tool name** *(PQ)*	*[Name of the approach/tool]*
	Screenshot	*[Screenshot of the approach/tool, if available]*

	Field	Information to be extracted
Task (why)	**Approach motivation/Assumptions** *(SQ1)*	*[Problems, motivations or issues that leaded to the development of the approach]*
	Approach goals *(SQ1)*	*[Goals for which the approach was developed]*
	Visualizations' reuse-specific goals *(SQ1)*	*[Description of how the approach goals relate to software reuse, i.e., which goals support or are somehow related to reuse]*
	Software engineering activities addressed by the visualizations *(TQ1.1)*	*[Software engineering activities or development process stages that can be somehow supported by the visualizations (e.g., "requirements engineering", "software design", "software testing", "software maintenance" etc.), including the construction of reusable assets (development for reuse) or the reuse of these assets in a scenario (development with reuse)]*
	Reuse-related tasks supported by the visualizations *(TQ1.2)*	*[Tasks supported by the visualizations, in a fine-grain level, e.g., "integrating reusable assets", "Searching and retrieving reusable assets" etc.]*
Audience (who)	**Visualizations' audience (stakeholders who can benefit from the visualizations)** *(SQ2)*	*[Software development stakeholders who can benefit from the visualizations, e.g., "programmers", "software designers", "end users" etc.]*
Target (what)	**Visualized items/data (what is visualized)** *(SQ3)*	*[Items/data from the software development process that have a visual presentation; examples include source code entities (e.g., "classes and interfaces with their attributes and methods"), high-level artifacts (e.g., "UML diagrams"), metrics (e.g., "coupling", "number of commits" etc.), among others]*
	Source of visualized items/data *(TQ3.1)*	*[Sources from which the items/data are extracted, e.g., "version control system repository", "metrics base", "software tracing log file", "source folder" etc.]*
	Collection procedure/method of visualized items/data *(TQ3.2)*	*[Description on how the items/data are collected and/or aggregated by the approach, e.g., "parsing", "clustering algorithm" etc.]*
Representation (how)	**Visualization metaphors used (how it is visualized)** *(SQ4)*	*[Visual metaphors used for describing the items/data, e.g., "squares and circles", "treemap", "graph" etc.]*
	Data-to-visualization mapping (input/output) *(TQ4.1)*	*[Description on how data are mapped to the visualizations, e.g., "classes are represented as circles and interfaces as triangles", "the color represents the complexity (the darker, the more complex)" etc.]*
	Visualization strategies and techniques *(TQ4.2)*	*[Strategies (e.g., "provide a global view while navigating into specific views") and techniques (e.g., "drill-down", "zoom", "clustering" etc.) used for displaying and interacting with the visualizations; strategies may use a given technique without mentioning it]*
Medium (where)	**Device and/or environment used for displaying the visualizations (where it is visualized)** *(SQ5)*	*[Device used for displaying the visualizations, e.g., "Computer", "Smartphone", "Tablet", "Display wall" etc.]*
	Resources used for interacting with the visualizations *(TQ5.1)*	*[Resources that allow interacting with the visualizations, e.g., "mouse", "keyboard", "pen", "finger touch", "gestures" etc.]*
Requirements (which)	**Hardware and software requirements/dependencies** *(SQ6)*	*[Hardware (e.g., "Quad-core processor", "Graphic card" etc.) and software (e.g., "Eclipse IDE", etc.) required for the approach]*
	Programming languages, APIs, and frameworks used for building the visualization *(TQ6.1)*	*[Programming languages, APIs, and frameworks used for building the approach, e.g., "Java Reflection API", "Prefuse" etc.]*
Evidence (worthwhile)	**Visualization evaluation methods** *(SQ7)*	*[Method applied for evaluating the approach, e.g., controlled experiment, observational study, case study etc.]*
	Application scenarios of the visualizations *(TQ7.1)*	*[Scenarios in which the approach was employed, e.g., "in an industrial setting", "in the context of an academic course" etc.]*
	Evaluated aspects *(TQ7.2)*	*[Evaluated approach aspects, e.g., performance, response time, usefulness, scalability etc.]*
	Visualization evaluation results/outcomes *(TQ 7.3)*	*[Evaluation findings and results]*

3.3.3 Outline of the secondary study (*quasi*-systematic review)

The nature of the study is to investigate existing works in order to characterize a particular field of interest (i.e., visualization approaches that can be used for supporting software reuse, regardless of the focus of support). This kind of investigation can be achieved through the conduction of a systematic literature review (SLR), i.e., a type of secondary study that aims to gather, evaluate and analyze the available literature that is relevant to a particular research question, topic, or phenomenon of interest [Kitchenham et al. 2007] [Kitchenham & Charters 2007]. In contrast with ad-hoc literature reviews, a SLR follows a well-defined sequence of methodological steps, which allows obtaining higher scientific value and more reliable results [Kitchenham 2004]. Moreover, SLRs follow a research protocol that must be defined beforehand, allowing the verification, extension, and replication of the research. These are the main reasons for choosing this research method for this work.

Because this is an exploratory study designed to characterize the state-of-the-art of the research area, and since there is no established baseline for comparison of the results obtained though this study, it is considered a *quasi*-systematic literature review [Travassos et al. 2008]. This kind of study has some similarities to a systematic mapping study, i.e., a study that aims to identify and categorize the research in a fairly broad topic area [Kitchenham et al. 2009]. However, since this study must explore the same rigor and formalism for the methodological phases of protocol preparation and running (except for the fact that no meta-analysis in principle can be applied), the *quasi*-systematic literature review denomination is more appropriate [Travassos et al. 2008].

This study aims at characterizing and identifying visualization approaches that can be used for supporting software reuse, regardless of the focus of support. The study goals are described in the Goal-Question-Metric (GQM) format [Basili et al. 1994]:

Analyze tools and approaches described in publications

For the purpose of characterizing

With respect to visualizations for supporting software reuse

Under the point of view of the researchers

In the context of software development project tasks and organizational tasks

The objects of this study are the publications that present visualizations supporting software reuse. The expected results are (i) the identification of visualizations that can be used for supporting software reuse, as well as their features

and limitations, and (ii) the establishment of a solid body of knowledge on visualizations for software reuse. Based on the findings, it is also expected to identify desirable features for novel approaches.

To achieve this goal, this study aims to answer the following research questions, decomposed into primary (PQ), secondary (SQ), and tertiary (TQ) questions:

- *PQ*: Which visualization approaches have been proposed to support software reuse?
  - *SQ1*: How do visualizations support software reuse?
    - *TQ1.1*: Which software engineering activities are addressed by the visualizations?
    - *TQ1.2*: Which reuse-related tasks are supported by these visualizations?
  - *SQ2*: To which stakeholders are these visualizations intended/targeted?
  - *SQ3*: Which items/data are visually represented?
    - *TQ3.1*: Where do these items/data come from?
    - *TQ3.2*: How are these items/data collected?
  - *SQ4*: Which visualization metaphors are used?
    - *TQ4.1*: How are data mapped to the visualizations?
    - *TQ4.2*: Which visualization strategies and techniques are employed?
  - *SQ5*: Where are the visualizations displayed?
    - *TQ5.1*: Which resources can be used for interacting with the visualizations?
  - *SQ6*: Which hardware/software resources are needed to deploy and execute the visualization tools?
    - *TQ6.1*: Which programming languages, APIs, and frameworks are used?
  - *SQ7*: Which methods are used for assessing the quality[19] of the visualizations (if any)?
    - *TQ7.1*: In which scenarios are the visualizations employed (if any)?
    - *TQ7.2*: Which aspects of the visualizations are evaluated (if any)?
    - *TQ7.3*: What are the results/outcomes of the conducted evaluations (if any)?

These questions map to the data extraction information (shown in Table 3.1), and are partially inspired in [Maletic et al. 2002].

[19] Quality evaluation/assessment encompasses any quality attributes, such as effectiveness, efficacy, amongst others.

The chosen search engine for carrying out the review is Scopus[20], due to its well-known stability, reliability, interoperability with different referencing systems, and high coverage – its database indexes most of the publications that are available in different digital libraries or other search engines (e.g., Compendex, IEEE Xplore, ACM Digital Library, Springer, Web of Science etc.) [Santa Isabel 2011] [França & Travassos 2013]. Besides, it indexes relevant journals and proceedings from the main software engineering conferences that comprise software reuse as a topic of interest. Examples of such conferences include:

- International Conference on Software Reuse (ICSR);
- International Conference on Software Maintenance (ICSM), recently changed to International Conference on Software Maintenance and Evolution (ICSME);
- European Conference on Software Maintenance and Reengineering (CSMR), recently incorporated to the International Conference on Software Analysis, Evolution, and Reengineering (SANER);
- International Conference on Information Reuse and Integration (IRI);
- International Conference on Program Comprehension (ICPC);
- International Conference on Software Engineering (ICSE);
- etc.

Since ACM is the only digital library that contains two of the control publications, it was decided to partially overcome this limitation by visiting the ACM Author Profile Page[21] of the respective authors and searching for the search string terms in the titles, abstracts and keywords of each listed publication. This decision was taken because the research described in these publications belongs to a specific research group and is related to the scope of this work (in terms of goals and features). More details on this issue are discussed in [Schots et al. 2014].

Because Portuguese is the native language of the researchers involved in this study, it was decided that publications in Portuguese should be analyzed as well. The following conferences were considered relevant for the purpose of this research:

- Brazilian Symposium on Software Engineering (SBES);

[20] http://www.scopus.com/

[21] See http://www.acm.org/publications/acm-author-profile-page for details (checked in November 30, 2013).

- Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS) and its predecessor Workshop on Component-Based Development (WDBC);
- Brazilian Symposium on Software Quality (SBQS).

Given that the Brazilian digital library (BDBComp[22]) did not index all the proceedings of any of these conferences until the date of creation of the research protocol, a manual search was required, following the same selection procedure (described in [Schots et al. 2014]).

The search string used was *((software OR system OR program OR asset) AND (reuse OR reusability OR reusable)) AND (visual OR visualization OR visualisation)*. Details on the definition of the search string can be found in [Schots et al. 2014]. Although a large number of publications were obtained, it was decided not to constrain the search string, due to the exploratory nature of this study.

A Portuguese version of the search string was also built: *((software OR sistema OR programa OR ativo) AND (reuso OR reúso OR reutilização OR reusabilidade OR reusável OR reutilizável)) AND (visual OR visualização)*. However, no results were found in the search engine through this search string. Thus, only the manual search on the identified sources should be performed for this language.

The manual search was performed in the following Brazilian proceedings:

- Brazilian Symposium on Software Engineering (SBES): proceedings from 1987 (1st edition) to 2012 (26th edition) (including);
- Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS): proceedings from 2007 (1st edition) to 2012 (6th edition) (including);
- Workshop on Component-Based Development (WDBC): proceedings from 2002 (2nd edition)[23] to 2006 (6th edition) (including);
- Brazilian Symposium on Software Quality (SBQS): proceedings from 2002 (1st edition) to 2012 (11th edition) (including).

The search results are listed in Table 3.2.

[22] http://www.lbd.dcc.ufmg.br/bdbcomp/

[23] The first edition of WDBC has no proceedings; selected works evaluated by the program committee were invited for publication in a book: Gimenes, I. M. S., Huzita, E. H. M. (2005). Component-Based Development: Concepts and Techniques [Desenvolvimento baseado em componentes: conceitos e técnicas] (in Portuguese), 1st ed., 304p., Ciência Moderna.

Table 3.2 – Study selection data (manual search)

	SBES	WDBC / SBCARS	SBQS
Title and abstract reading	556	158	315
Number of accepted publications	30	42	26
Number of rejected publications	526	116	289
Number of duplicate publications	0	0	0
Full reading	30	42	26
Number of accepted publications	0	0	0
Number of rejected publications	30	42	26
Number of duplicate publications	0	0	0

As it can be seen, from the 1030 analyzed publications, no one was selected. Most of the publications selected during the title/abstract reading (98) were related to software reuse; however, in the full reading, it was noticed that no publication mentions the use of visualization resources with the goal of supporting software reuse.

Regarding the search engines, the searches were performed on October 1st, 2012 at 3PM local time (UTC/GMT -3) in both the Scopus search engine and the selected ACM Author Profile Pages. Although no time constraint was set, the publications ranged between 1980 and September 2012. However, publications that had not been indexed until the date of search may have been added to the digital libraries afterwards.

In total, 1159 publications were retrieved from Scopus by performing the search with the chosen search string. The publications were exported from this search engine and formatted in tables. The search performed on the ACM Author Profile Pages was conducted in a different way: all the publications listed in the pages of each key author identified from the control publications (as discussed in [Schots et al. 2014]) were manually exported, and their title, authors and keywords were extracted using regular expressions in a text editing tool (Notepad++[24]). After that, duplicates were semi-automatically identified and removed, resulting in 304 results. Then, a semi-automatic search was performed using the search string terms: these matched with 6 publications.

Table 3.3 summarizes the study selection stages in terms of accepted, rejected, and duplicate publications. From this point, a M.Sc. student (referred to as second researcher) supported the selection stages listed in this table.

[24] http://notepad-plus-plus.org/

Table 3.3 – Study selection data (search engines)

	Scopus		ACM
	1st researcher	**2nd researcher**	**Both researchers**
Title reading	1159	1159	6
Number of rejected publications	740	831	0
Number of duplicate publications	8	8	0
Number of accepted publications	411	320	6
Abstract reading	411	320	6
Number of rejected publications	326	275	0
Number of duplicate publications	8	0	0
Number of accepted publications	77	45	6
Full reading	77	45	6
Number of rejected publications	47	26	0
Number of duplicate publications	1	0	1
Number of accepted publications	**29**	**19**	**5**

During the consensus stage (for conflict resolution), both researchers selected 19 publications, so these did not need to be reanalyzed. From the 15 publications selected only by the first researcher, 13 were included after discussion, and 2 were rejected. From the 5 publications selected only by the second researcher, 2 were included after discussion, and 3 were rejected (being 1 by a third researcher, since consensus had not been achieved). Details on the consensus stage can be found in [Schots et al. 2014].

Beyond one of the control publications, another related publication [Anslow et al. 2004] (found based on the citations of the ACM key authors) was also added manually. It was agreed in the consensus stage to include it, along with the control publications previously included.

In total, 36 publications were selected, describing 34 approach proposals. They are listed in [Schots et al. 2014], along with details on the analysis. The data extraction form presented in Table 3.1 was filled out for each approach, and the results can be found in [Schots et al. 2014] (in tables) and an interactive version is published in an online repository [Schots 2014c], as illustrated in Figure 3.5.

3.3.4 Discussion of the findings

From the moment that the first identified works were published, there was already a concern on supporting reuse of a variety of artifacts (*SQ1*), as can be noticed in [Mancoridis et al. 1993] and [Constantopoulos et al. 1995]. It can also be noticed that most approach goals are artifact-oriented, not taking into account the dynamics of reuse in an organization (i.e., correlating consumers, producers, assets and projects).

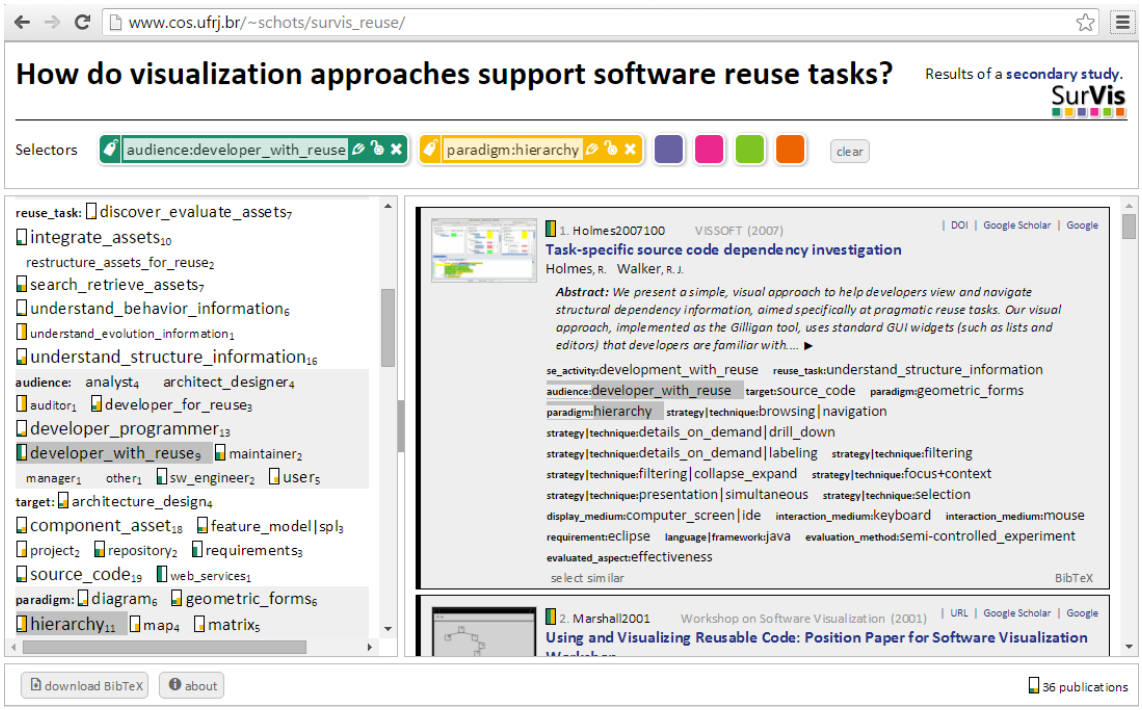


Figure 3.5 – Organization of the secondary study results [Schots 2014c]

Although approaches somehow encompass diverse software engineering activities (*TQ1.1*), only a few of them present integration among activities. In other words, a stakeholder must resort to different approaches to perform activities that might be related, and the communication among them should be seamless.

It can be noticed that understanding assets is by far the most supported reuse task (*TQ1.2*). This is indeed expected, since understanding is a likely benefit in employing visualizations. In terms of the different aspects that can be the focus of comprehension, evolution information about reusable assets is particularly absent from existing works – the only related work deals with comparing refinement sets of different versions of feature models, and it is based on a trace repository; no other evolution aspects are taken into account by any approach. Moreover, repository-related information is only focused on structural characteristics, i.e., the approaches do not handle usage data related to reuse repositories.

Although there is a reasonable variety of stakeholder support (*SQ2*), only a few works support more than one kind of stakeholder simultaneously. This would not be a major problem if different approaches could communicate with each other. The lack of multi-stakeholder approaches hamper the evaluation of how well organization's goals related to reuse are being accomplished, under the perspectives of each reuse stakeholder (e.g., producers, consumers and reuse managers). Particularly, the single approach that mentions support for managers [Mancoridis et al. 1993] only presents

technical details about a software project, which does not seem feasible for management tasks. Managers need more high-level details that can be useful for decision making, so that they can promote actions not only to stimulate reuse, but especially to mitigate potential barriers for performing reuse in their organizations.

The majority of the visualized items and data (*SQ3*) are source code artifacts. In spite of this imbalance, there are many different kinds of artifacts (from different software development stages) that can be visualized. There are few approaches for visualizing software repositories with the intention to promote reuse (providing relevant reuse data), and no repository information or metadata are visually represented aiming to increase awareness. According to [Orso et al. 2000], approaches that employ reuse repositories must store not only the reusable assets, but also the information about them (usually called metadata).

The data sources (*TQ3.1*) are usually the source code of a program and databases. Only a few approaches combine information from different sources (e.g., [Kelleher 2005]), and some are compatible with a limited set of data types. Although many assets have additional related data available online, such data are usually underexplored or overlooked. Moreover, although several kinds of information may be used for supporting reuse, some common data sources are not explored visually by any of the works (e.g., VCS repositories, issue trackers etc.).

Since each visualization technique may have some constraints, each collection procedure (*TQ3.2*) must deal with this issue and make the proper arrangements. For instance, in [Kelleher 2005], some format conversions are mentioned in order to make the data ready to be represented by the intended visualization. During the data collection procedure, the source may still require some transformations to have the data set in the correct format to be used by different representations. Some authors also defend the use of intermediary formats for storing the collected data (e.g., [Alonso & Frakes 2000] and [Anslow et al. 2004]) in order to make them reusable in different visualizations.

As expected, different abstractions are used for representing different data (*SQ4*). Although several types of abstractions are used, publications lack a discussion on how/why a given metaphor was chosen and, more importantly, whether it is effective or not in its purpose. The mapping between data and visualizations (*TQ4.1*) is barely described in most of the publications, so the reader/user has to “guess” it, which can be risky and lead to wrong interpretations of data.

Several visualization strategies and techniques are used (*TQ4.2*), but not in a comprehensive way. This does not mean that every possible technique should be employed, but some approaches might benefit from more interaction facilities for allowing an effective use and understanding of data. Enhancing awareness and understanding of software information and the software itself requires the identification of adequate abstractions according to the comprehension needs [Diehl 2007] [Schots et al. 2012]. The choice of visualization abstractions for representing the data, as well as the interaction techniques to be employed, heavily depends on contextual information, e.g., the nature of data, the visualization constraints, and the task to be supported (e.g., selecting the most suitable assets from a set of reusable assets).

There is a lack of mechanisms to offer flexibility to software stakeholders in customizing their visualizations, so one can focus on relevant data and information to improve the understanding of their activities. Although this can be seen as a downside, on the other hand, letting the user decide which visualization to use may not be adequate, as he/she may not know which metaphors better fit the structure to be visualized. Approaches that require the user to map visualizations to data should provide at least some kind of support for filtering inappropriate visualizations according to underlying restrictions associated with the data.

Regardless of the number of occurrences for each of the strategies, it is unwise to affirm that certain techniques are more important than others. Visualization strategies and techniques must be chosen according to the visualization goals. Moreover, the available data must meet the representation constraints associated to the employed visualizations.

Regarding the medium for displaying visualizations (*SQ5*), most approaches are still computer-based, in spite of the technological advances in displaying and interaction devices. Only a few approaches (explicitly) mention that they work in (or are integrated with) a web environment. This was somehow surprising. Some recent web-based visualization frameworks may help changing this scenario. Publications also lack information in a more detailed level regarding the compatibility of the approaches with different media. For instance, even among approaches proposed more recently, none mentions or focuses on mobile devices as an alternative to execute and interact with the visualizations. Moreover, in spite of the existence of web-based approaches, one cannot state (based solely on the publications) that they are multiplatform, i.e., whether they work in other devices or not, since some devices, such as smartphones and tablets,

contain displaying and interaction constraints that must be accounted for when designing visualizations.

It is not surprising that mouse and keyboard are the main interaction resources (*TQ5.1*), as current information visualization systems still largely focus on these peripherals for interacting with data [Lee et al. 2012]. In spite of that, there has been a constantly growing interest in other research areas for incorporating other natural forms of interaction such as touch, speech, gestures, handwriting, and vision.

It was noticed that software and hardware requirements (*SQ6*) are not discussed properly in the publications, which hampers the proper evaluation of the feasibility of the approaches to particular contexts. The same occurs with information about programming languages, APIs, and frameworks (*TQ6.1*). Such information, if properly discussed, helps to evaluate how up-to-date a tool is, as well as to identify any potential integration constraint. It can be noticed that some of the technologies used by the approaches are already in disuse.

It can be observed that the majority of the works does not present a proper evaluation on their use (*SQ7*): some of them do not present any at all. This can be partially explained by the lack of demand for evidence in publications (a scenario that has been changing in the last years). In many cases, the evaluation is done by the authors themselves, which is subjective and may bring some bias. The absence of proper evaluations may raise questions as regards to meeting the purpose to which the approaches were proposed In order to determine if visualizations are worthwhile, i.e., effective in helping their target users, it is desirable that they are exposed to a proper evaluation [Sensalire et al. 2009]. This can be seen as a major downside.

An interesting finding is that there is a balance between the evaluation scenarios (*TQ7.1*), since not only academic projects are used, but open source projects are also taken into account (which allows the verification of results), as well as commercial/industrial (thus strengthening the interaction with industry). Particularly, since industry stakeholders can directly benefit from the results of such studies, experiments in industry are strongly recommended for strengthening interaction with academia.

In general, the reported data about the evaluations lack additional and useful details, so that one can understand in which scenarios they were conducted (*TQ7.1*), which aspects were evaluated and why (*TQ7.2*), how the analysis was made, and which strengths and opportunities for improvements were identified (*TQ7.3*). It must be

emphasized that the experimental rigor must be correlated with the relevance of the findings, in order to avoid wrong conclusions. Some recent works (e.g., [Feigenspan et al. 2013] and [Yazdanshenas et al. 2012]) present a proper experimental soundness that helps to understand the identified limitations, so that other researches aiming to support reuse can use their evaluation report as a basis.

3.4 Final Remarks

As mentioned in Section 2.6, software visualization can be a useful resource for supporting software reuse demands, especially the ones related to awareness and comprehension. In spite of that, as shown in this chapter, its potential has not yet been thoroughly explored, i.e., there is room for research and development in this regard. Although there are publications in the literature that propose visualization approaches geared specifically for software reuse, few approaches aim at assisting reuse management as a whole, i.e., providing the necessary support to carry on a range of software reuse tasks. This finding contributes to answering RQ1, stated in Section 1.3.

The software engineering community can use the results found in the *quasi*-systematic review as a starting point for future research directions that can be addressed when choosing, instantiating, or developing visualization-based approaches for supporting software reuse. Besides, the presented information can be used as a body of knowledge not only to support the decision making regarding the choice of visualization approaches for software reuse, but also to conduct other secondary studies on software visualization applied to another field of interest (e.g., software maintenance). This study can also be seen as a summarized catalog of the approaches, whose further information can be obtained from the corresponding original publications.

The identified limitations of the current findings and the unexploited research opportunities point out directions and desirable features for a novel approach for providing awareness and visualization support to activities related to software reuse. Table 3.4 summarizes the problems identified and the corresponding desirable features for such an approach to help solving such problems.

Table 3.4 – Findings and assumptions derived from the *quasi*-systematic review

ID[25]	Description	Desirable feature
VF1	Most of the goals of existing visualization approaches geared to software reuse are artifact-oriented, and do not take into account the dynamics of reuse in an organization (i.e., correlating assets, developers, and projects).	The approach should take into account core elements in the reuse scenario (assets, developers, and projects) and data related to them.
VF2	Existing approaches lack proper handling of evolution information about reusable assets, which can show how they have been maintained and improved.	The approach should make use of evolution information about reusable assets, so that their development history can provide useful insights on their maintenance and improvement.
VF3	There are few approaches for visualizing software repositories with the intention to promote reuse (providing relevant reuse data). Most approaches focuses only on structural characteristics of the reuse repositories. Besides, no repository information or metadata are visually represented aiming to increase awareness.	The approach should present information available from reuse repositories that can be relevant and helpful for taking reuse decisions.
VF4	The data sources of the approaches are usually the source code of a program and databases. Although many assets have additional data available online, such data are usually underused, underexplored, or overlooked, not combined to provide useful information.	Already described in RA3.
VF5	The mapping between data and visualizations is barely described in most of the approaches. They lack evidence on how/why each visual metaphor was chosen and, more importantly, whether it is effective or not in its purpose (i.e., if the established goals for the construction of the visualization tools are met).	The approach should ensure that the selected visualizations meet the goals that led to the construction of its visualization tools.
VF6	Regarding the medium for displaying visualizations, most approaches are still computer-based. Only a few approaches (explicitly) mention that they work in (or are integrated with) a web environment. In spite of that, it is not possible to assume that they work in devices such as smartphones and tablets, which contain displaying and interaction constraints that must be accounted for when designing visualizations.	The approach tools should use responsive design whenever necessary (depending on the stakeholders' conventional device) in order to be compatible with different media, such as computers, tablets, and smartphones.

[25] VF refers to "Visualization-Related Finding", while VA means "Visualization-Related Assumption".

ID[25]	Description	Desirable feature
VA1	[Reuse] managers need support for analyzing/monitoring the reuse scenario as a whole in an organization, with high-level information that can be useful for decision making, so that they can promote actions not only to stimulate reuse, but especially to mitigate potential barriers for performing reuse in their organizations.	Already described in RF6.
VA2	Each visualization metaphor may have some constraints on its use that must be taken into account, reflecting on necessary arrangements in the data collection procedure. The collected data should be reusable in different visualizations.	The data to be visualized should meet the representation constraints associated to the corresponding visualizations and use a generic representation in order to be reusable between different visualizations.
VA3	The choice of visualization abstractions and techniques for representing the data, as well as the interaction techniques to be employed, heavily depends on contextual information, e.g., the nature of data, the visualization constraints, and the task to be supported (e.g., selecting the most suitable assets from a set of reusable assets).	The approach should take into account the characteristics and context information of the data, intended for a proper choice of visualization elements.

Based on these desirable features, the approach proposed in this thesis is presented in the next chapter.

CHAPTER 4 – PROPOSED APPROACH: APPRAISER

This chapter introduces the approach proposed in this work (called APPRAiSER), which uses visualization resources for supporting software reuse tasks. The chapter presents its elements, along with relevant aspects regarding their realization/implementation. It also presents some related work.

4.1 Introduction

Based in the literature reports and the semi-structured interviews (both described in Chapter 2) and in the results of the *quasi*-systematic review (described in Chapter 3), it was possible to identify a number of desirable features for an approach to support the implementation of a software reuse program. Furthermore, as discussed in Chapter 2, it is necessary to provide visualization metaphors for representing reuse data, so that stakeholders can interact with and manipulate such data to obtain answers and perform their tasks quickly, besides decreasing the cognitive overload.

Thus, the realization of the third step of the research methodology of this thesis (presented in Section 1.5) is the proposal of APPRAiSER: an **A**pproach for **P**erceiving and **P**romoting **R**euse by **A**wareness **i**n **S**oftware **E**ngineering and **R**eengineering[26] [Schots 2014a] [Schots 2014b].

APPRAiSER aims to assist the execution of some tasks related to software reuse, both at the organizational level and project level, targeting some of the aforementioned desirable features. At the organizational level, it focuses on supporting the management of assets, developers (consumers and producers), projects, and their surrounding metadata, as well as supporting the monitoring of reuse initiatives. At the project level, APPRAiSER aims to support the selection of assets to reuse and the understanding of such assets and their properties, based on information that may be useful with respect to them.

[26] The reason for this name is that an appraiser "has the knowledge and expertise necessary to estimate the value of an asset, or the likelihood of an event occurring, and the cost of such an occurrence" (http://www.investopedia.com/terms/a/appraiser.asp), and such definition relates to the goals of this work.

The current elements of APPRAiSER aim at supporting mainly the following reuse stakeholders:

- *developers* in better exploring software repositories, accessing all the available information about them in the context of reuse, allowing to make informed decisions[27] regarding the reuse of an asset or upgrading/downgrading an asset present in a project, among others;
- *reuse managers*[28] in managing and tracking assets reused in the organization, maintaining the reuse repository with information that allows decision-making regarding its assets, and managing and monitoring the implementation of reuse processes, evaluating the effectiveness of reuse practices in the organization; and
- *both developers and reuse managers* by providing awareness of the reuse scenario as a whole (for perceiving the effects of reuse in an organization) and allowing to perform reuse tasks accurately.

APPRAiSER provides reuse awareness to stakeholders through visualization resources that can help them be aware of the reuse scenario as a whole. In order to provide support to the identified reuse needs, this work also uses and extends some academic (Undergraduate and Master) projects advised or supervised by the author of this thesis. The approach is detailed in the next sections.

4.2 APPRAiSER Overview

APPRAiSER elements compose an environment that collects and correlates reuse-relevant data, presenting them by means of interactive visualizations, aiming to provide reuse analytics for software development organizations and help answering software reuse questions. The concept behind APPRAiSER is broader and more general, contextualized for the particular scenario being handled in this thesis. Figure 4.1 provides an overview of APPRAiSER elements, followed by a brief explanation of each of them.

---

[27] An *informed decision* is a decision made after learning relevant facts (informing oneself) about the focus of the decision.

[28] A reuse manager must manage and monitor the overall reuse program, but instead of being merely a managerial role, a technical profile is needed to deal with specificities of assets and the reuse repository.

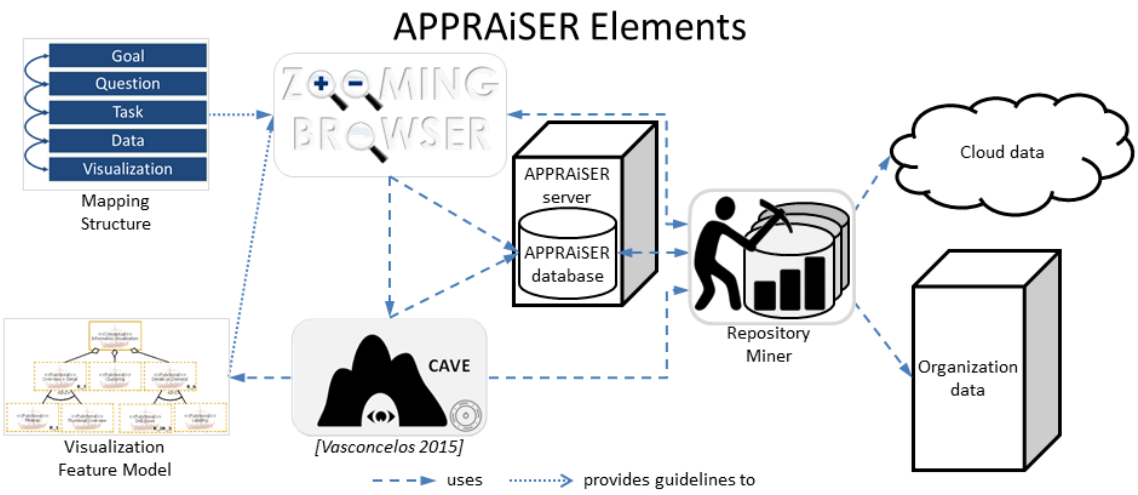


Figure 4.1 – High-level elements of APPRAiSER

The information related to absences in research and practice was obtained both from the semi-structured interviews (reuse-related demands on software organizations) and the secondary study (deficiencies/absences in existing visualization approaches geared to reuse). Such information was the input for the construction of ***Zooming Browser***. This tool is the core of APPRAiSER. It provides visualization and interaction resources aiming to allow users to gain insights about data and perform reuse tasks easily and accurately. Section 4.3 describes its functionalities with more details.

In order to visualize low-level information and help identifying proper visualizations for a given context (not pertaining to Zooming Browser), a tool named ***CAVE (Context-Aware Visualization Engine)*** [Vasconcelos et al. 2014b] [Vasconcelos 2015] was built by a M.Sc. student, in parallel with this thesis, in the context of APPRAiSER. Through context-awareness mechanisms, it checks for the occurrence of context situations (based on data values from sources of interest) and notifies the user about situations that deserves attention, providing visualizations that help analyzing each of them. CAVE relies on a previous work [Queiroz et al. 2013] that indicates visualizations that are most suitable for a given focus of representation/analysis.

Both of the aforementioned works ([Vasconcelos 2015] and [Queiroz et al. 2013]) were developed under the supervision of the author of this thesis as an integral part of APPRAiSER. Because the full contributions of these works go beyond the scope of this thesis, they are described thoroughly in a master thesis [Vasconcelos 2015] and in a technical report [Schots et al. 2015], so that only the relevant elements used or adapted by APPRAiSER are discussed in this thesis. The intersection between these works and Zooming Browser is described throughout Section 4.3.

Reuse tasks usually require a stakeholder to perform some kind of decision-making. This occurs both in the context of a software project (e.g., the decision to incorporate or upgrade a library or any reusable asset) and at the organizational level (e.g., reuse management and reuse monitoring tasks). An informed decision in these contexts depends on reliable data about software development facts. However, when it comes to software reuse, most of these data are barely taken into account by (or even made explicit by tools to) stakeholders, mainly because of the decentralized nature of their sources.

For supporting the data collection and aggregation, APPRAiSER comprises a module called ***Repository Miner***, which extracts information from software repositories (either in the cloud or locally) and delivers it to the APPRAiSER tools. Repository Miner also persists the collected information in the APPRAiSER database for avoiding excessive consumption of API-based cloud services. Section 4.4 provides a detailed explanation of its conception and construction.

In order to develop Zooming Browser, some complementary resources were needed to ensure the proper mapping, organization, and structuring of the information related to the development process (especially to the planning phase). There were two main concerns: (i) choose the visualization and interaction resources to include in the tool, and (ii) make sure that the chosen elements were actually suitable to achieve the goals established in the tool construction. In other words, the main concern was to handle the construction of visualization tools from a software engineering point of view.

Regarding the first concern, the information presented in some dimensions of the extended task-oriented framework obtained from the secondary study (described in Section 3.3.3) provided input for the development of the ***visualization feature model***, one of the APPRAiSER elements. It organizes the concepts and characteristics about the information visualization domain in terms of visualization and interaction features. This model helps planning/choosing the features that can be used in visualization tools (and, particularly, Zooming Browser). The visualization feature model was developed in collaboration with a M.Sc. student [Vasconcelos et al. 2014a] [Vasconcelos 2015] [Schots et al. 2015] and is described in Section 4.5.

With respect to the second concern, in order to ensure the meeting of the established goals through the chosen visualizations, another approach element is the ***staged mapping structure of goals and visualizations*** [Schots & Werner 2015]. This mapping emphasizes decision aspects that could be overlooked, helping to perform a

more focused and cautious decision making towards an anticipated assessment of the usefulness and effectiveness of visualization tools (and, particularly, Zooming Browser). The mapping structure is presented in Section 4.6.

The original architecture of the environment implementation (presented in [Schots 2014a]) included a module for supporting the reengineering of assets, making refactoring recommendations triggered by software metrics. Such module would be an integration with Rec4Reuse [Vital & Krause 2013], a work advised by the author of this thesis at UERJ. However, based on the feedback received when APPRAiSER was proposed (in [Schots 2014a] and [Schots 2014b]) with respect to its large scope, it was decided not to focus on the reengineering or recommendation aspects in this thesis, making it part of a research agenda for future work.

Other elements of the original architecture – ReuseDashboard [Palmieri et al. 2013] and GraphVCS [Pereira & Schots 2011] [Pereira & Schots 2014], supervised (at UFRJ) and advised (at UERJ) by the author of this thesis, respectively – were integrated[29] to Zooming Browser as visualization perspectives (described in Section 4.3) instead of being separate tools.

Table 4.1 lists the realization of the identified desirable features by the APPRAiSER elements and characteristics. They are detailed throughout this chapter.

Table 4.1 – APPRAiSER realization of desirable features

ID[30]	Desirable feature	Realization
RF1	For properly supporting software reuse tasks, the approach should primarily support managing source code assets.	• APPRAiSER defines source code assets as the default kind of reusable asset and provides native support for them.
RF2	The approach should also support different kinds of reusable assets (assuming that there is a corresponding reuse repository with relevant information about them).	• Repository Miner (its modular architecture supports the customization of the identification and extraction strategies in order to allow other kinds of reusable assets).
RF3	The approach should provide a way of collecting information regarding reuse (consumption), evolution, and discontinuation of assets, along with the developers involved in the production and consumption of these assets.	• Repository Miner

[29] Both tools were originally developed in Java. Due to the differences in the frameworks used in each language, it was not possible to adapt the source code, requiring a new implementation. Thus, they served as inspiration for a JavaScript implementation.

[30] RF = "Reuse-Related Finding", RA = "Reuse-Related Assumption", VF = "Visualization-Related Finding", VA = "Visualization-Related Assumption".

ID[30]	Desirable feature	Realization
RF4	The approach should help identifying potential interested parties of an asset based on reuse data and notifying such parties about changes in the status of the assets.	• Repository Miner • Zooming Browser (Metadata Exploration perspective) • APPRAiSER server
RF5	The approach should provide a reuse repository for the organization, or integrate with an existing one, that allows potential consumers to obtain reusable assets and relevant information about them.	• [Communication with] Nexus Repository (organization-specific instance) • Integration with Maven Central (through the Repository Miner)
RF6/ VA1	The approach should present concise information that can help stakeholders in establishing and monitoring the progress of reuse initiatives in the organization, through mechanisms that provide adequate awareness of the reuse scenario. [Reuse] managers need support for analyzing/monitoring the reuse scenario as a whole in an organization, with high-level information that can be useful for decision making, so that they can promote actions not only to stimulate reuse, but especially to mitigate potential barriers for performing reuse in their organizations.	• Zooming Browser (all perspectives, especially the Dashboard[31])
RA1	The approach should provide mechanisms with different perspectives to support each stakeholders' needs related to reuse.	• Zooming Browser (its different perspectives are suitable for different stakeholders) • CAVE[32] (for providing additional information on the assets' low level information)
RA2	In order to minimize cultural barriers and allow all stakeholders to become committed with reuse initiatives, there should be a strategy for a gradual introduction of the approach mechanisms, avoiding cognitive overload.	• APPRAiSER tools as a whole • Recommendations on the use of APPRAiSER in an organization
RA3/ VF4	In order to show relevant information about the reuse scenario as a whole, particularly providing a better perception of the assets' stability and quality, the approach should collect data from different kinds of source, integrating information from reuse repositories, version control repositories, and change control (bug tracking/task manager) repositories. The data sources of the approaches are usually the source code of a program and databases. Although many assets have additional data available online, such data are usually underused, underexplored, or overlooked, not combined to provide useful information.	• Repository Miner (especially its integration with version control and issue tracker repositories)

[31] This perspective is a new implementation of the aforementioned ReuseDashboard.

[32] The role of CAVE in the context of this thesis refers to the Low-Level Data Representation perspective, described in Section 4.3.2.4.

ID[30]	Desirable feature	Realization
RA4	The approach should handle a large amount of information through adequate abstractions and interaction techniques.	• Zooming Browser visualization metaphors and interaction resources • CAVE
RA5	The approach should provide the option of tracking reusable assets (and projects), both open source and developed by the software organization.	• Repository Miner (taking into account both cloud and local repositories)
RA6	The approach should integrate with and collect information from version control repositories for suggesting assets that occur in more than one project. This allows a later evaluation for their inclusion on the reuse repository. Collecting usage data and properly identifying producers and consumers help support such decision.	• Repository Miner (especially its integration with version control repositories and its identification strategies)
VF1	The approach should take into account core elements in the reuse scenario (assets, developers, and projects) and data related to them.	• Repository Miner • Zooming Browser (Metadata Exploration perspective)
VF2	The approach should make use of evolution information about reusable assets, so that their development history can provide useful insights on their maintenance and improvement.	• Repository Miner • Zooming Browser (History perspective)
VF3	The approach should present information available from reuse repositories that can be relevant and helpful for taking reuse decisions.	• Zooming Browser (Metadata Exploration perspective)
VF5	The approach should ensure that the selected visualizations meet the goals that led to the construction of its visualization tools.	• Mapping structure
VF6	The approach tools should use responsive design whenever necessary (depending on the stakeholders' conventional device) in order to be compatible with different media, such as computers, tablets, and smartphones.	• Zooming Browser implementation technologies (HTML + CSS + JavaScript) with responsive design (for developers and reuse managers who need to present results for other stakeholders) • CAVE implementation technologies (HTML + CSS + JavaScript) focused on desktop environments (for developers)
VA2	The data to be visualized should meet the representation constraints associated to the corresponding visualizations and use a generic representation in order to be reusable between different visualizations.	• Visualization feature model (provides the features and constraints) • Repository Miner (uses the JSON format)
VA3	The approach should take into account the characteristics and context information of the data, intended for a proper choice of visualization elements.	• Visualization feature model (organizes visualization features according to their properties) • CAVE

It is important to emphasize that the approach itself is not enough to solve the problems described in Table 4.1. Instead, it provides some support (in different levels) for the listed needs.

It is known that there are several kinds of reusable assets. However, as stated in Section 2.5.3, using source code as reusable assets makes the benefits arising from reuse more noticeable by organizations. Moreover, it can be noticed that reuse of source code artifacts is still on the mainstream of software development [Schots & Werner 2013] [Schots & Werner 2014a]. Thus, APPRAiSER tools currently focus mainly on object-oriented source code artifacts (meeting RF1), including frameworks and libraries, assuming that they are packaged somehow (i.e., in the form of components that are provided to be reused).

The next sections present more details on each of the APPRAiSER elements, indicating the way it handles each of the presented desirable features.

4.3 Zooming Browser

Nowadays, developers write less code and consume more reusable code. Although software reuse has become present in the daily routine of software developers (yet mostly in an ad-hoc or a pragmatic way, as stated previously), it is central to consider the importance of *reuse awareness*, *i.e.*, knowing what is going on in the reuse scenario. It helps deciding whether a given asset should be reused, or communicating problems identified in any kind of reusable asset to its producers and consumers, among other benefits. However, achieving reuse awareness is challenging, especially because of the lack of tools to support this purpose.

The Zooming Browser tool [Schots 2014a] [Schots 2014b] aims at providing reuse awareness to support reuse managers and developers in performing reuse-related tasks, both in the context of a software project (e.g., the decision to incorporate or upgrade a library or any reusable asset) and at the organizational level (e.g., reuse management and reuse monitoring tasks). Zooming Browser provides basic reuse information along with other information that supports reuse awareness, enabling stakeholders to quickly search, navigate, and explore the contents of the reuse repository and its surrounding elements.

Being an integral part of APPRAiSER, Zooming Browser is composed by three core elements (assets, developers, and projects) and the relationships between these elements (meeting VF1), as shown in Figure 4.2.

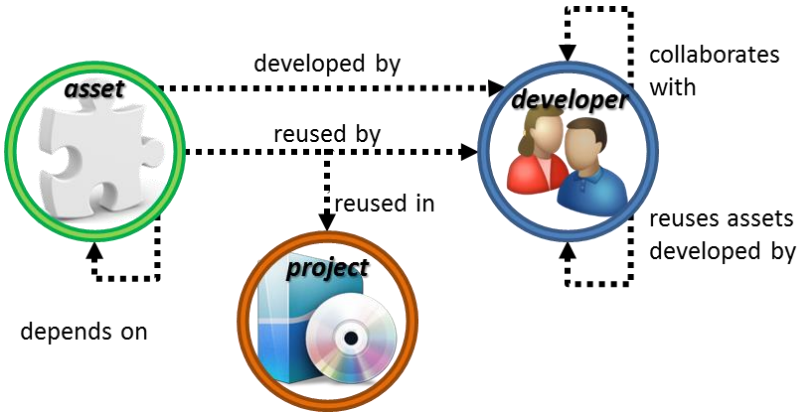


Figure 4.2 – Core elements of Zooming Browser [Schots 2014a]

The role of each element in Zooming Browser is described as follows:

- *Assets* are the core of software reuse. They provide means of solving a problem without going from scratch. They are developed by a producer and are expected to be reused by consumers in software projects.
- *Developers* can be divided into two groups – consumers (the ones who reuse assets) and producers (the ones who develop reusable assets) –, but a developer can be a consumer and a producer at the same time. It is relevant to know their relationship with the assets in order to notify them about status changes or ask for their help in case there are problems with reusing an asset.
- *Projects* make the bridge between assets and consumers. A consumer can only reuse an asset in the context of a software project. The project characteristics may help understanding how and why an asset was reused, and this information may be useful for consumers who do not know how they can reuse a given asset.

These core elements and their relationships drove the selection of some questions related to software reuse. These questions are listed in Appendix A. Some representatives are as follows:

- Which [versions of] assets have ever been reused?
- How often are [versions of] assets reused over time?
- Which projects have ever had an asset included (i.e., contain at least one reusable asset in their development history)?
- Which consumers reused this asset [version]?
- Which producers contributed to the development of this asset [version]?
- Among the reported bugs, improvement suggestions, or feature requests related to this asset [version], are most of them fixed or open?
- Which assets were reused by which consumers in which projects?

Zooming Browser aims at helping to answer these questions, among others, by means of its visualization and interaction resources. The next subsections present the design principles applied to Zooming Browser, the employed visualization perspectives and their characteristics, and some implementation aspects of the tool.

4.3.1 Design principles

Zooming Browser follows some design principles, i.e., some guidelines that help improve viewers' comprehension of visually encoded information [Agrawala et al. 2011]. Instead of being strict rules, they are considered "rules of thumb" that might even oppose and contradict one another, describing how visual techniques affect the perception and cognition of the information in a display [Agrawala et al. 2011].

The design principles presented for Zooming Browser were extracted from a summarized list in a previous work [Vasconcelos et al. 2013], in addition to other broadly used principles derived from literature sources and visualization practice. They are presented in Table 4.2 (along with their sources), and their realization is described throughout the remainder of this section.

Table 4.2 – Zooming Browser design principles

ID	Design principle	Description
DP1	Provide an overview of an entire collection [Shneiderman 1996]*	Visualizing the whole surrounding of a selected item allows understanding the context of an item and increases the comprehension of its current situation.
DP2	Provide a geometric or semantic zoom mechanism on items of interest [Shneiderman 1996] [Buering et al. 2006]*	Zooming in or out is an alternative to reveal more details of an item or put it in perspective with its context, respectively.
DP3	Provide a feature to filter out items that are not of interest [Shneiderman 1996]*	For large volumes of information items, it may be necessary to show only what is relevant to a context.
DP4	Enable details-on-demand to get particularities of selected items [Shneiderman 1996]*	It is convenient to exhibit details of items as they are selected. A focus area should have methods to reveal its content according to the user interaction.
DP5	Present relationships among items [Shneiderman 1996] [Card et al. 1999] [Chen 2006]	Relationships may reveal important details about items and their context.
DP6	Keep a history of actions performed [Shneiderman 1996]	The comparison of information in a view may need performing repetitive actions; a history would store such actions to speed up interaction and, therefore, the analysis. The user should be able to navigate easily through the different perspectives and views, with the option of going back to a previous state. To this end, the tool must "memorize" the flow of information and keep track of it.

ID	Design principle	Description
DP7	Adapt display scale when the data set is dynamically increased [Robertson et al. 2009]	In a dynamic view, the data set can automatically increase, and a fixed display scale becomes inappropriate to understand the visualization, requiring automatic adjustments, such as adapting its information layout to reveal more or less information according to the display size.
DP8	Use colors or texture coding, icons, and sizes to distinguish types and characteristics of items [Card et al. 1999]	Different colors and icons can highlight the diversity of data in a view, thus increasing user comprehension. Sizes bring the assumption that items with greatest size are of greatest interest, thus this attribute requires some handling or user control to be used properly.
DP9	Maintain data order [Card et al. 1999] [Chen 2006]	For improving comprehension, the data must be kept ordered according to an established parameter, unless the user interacts with the view to change it.
DP10	Avoid substantial display changes while data are updated and provide appropriate smooth transitions [Robertson et al. 1989] [Card et al. 1999]	Fast updates in the way data are displayed may increase both difficulty and time to perform this task. Simply showing the beginning and ending states without an animated transition may cause users to misinterpret object transformations. Thus, transitions between visualization elements should be smooth, favoring the immersion flow and avoiding user disorientation. The real-time nature of the interaction process requires the visualization system to use some sort of scheduling mechanism.
DP11	Provide quick responses to the user queries or interactions [Spence 2001]	The response time for interaction requests must not hamper the user immersion flow, i.e., it should not affect the user experience with the visualizations. Quick responses to interactions allow fast comparison of data, improving the quality of analysis results.
DP12	Include metadata describing the meaning of the visualized data [Card et al. 1999]	Metadata allows the comparison of values and is important to improve the interpretation of a view.
DP13	Provide a multi-perspective environment [Wu & Storey 2000][33]	In order to support each stakeholders' needs performing specific tasks, it is advisable to distribute visualization metaphors into different perspectives, so that only the necessary information is presented for performing the task. In multi-perspective views, stakeholders use a main view for general comprehension tasks and, in certain contexts or activities, make use of auxiliary views for additional exploration tasks.
DP14	Take into account the characteristics of data for a proper choice of visualization elements[34]	The mismatch between the data characteristics and the visual abstraction capabilities of representing them may potentially lead to misinterpretations of data.

* These principles are part of the "visual information seeking mantra", proposed in [Shneiderman 1996]. Such mantra is a basic principle that summarizes visual design guidelines: overview first, zoom and filter, then details-on-demand. They also help meeting RA4.

[33] The realization of this design principle meets RA1.

[34] The realization of this design principle meets VA3.

These design principles were taken into account for the definition and implementation of the visualization perspectives, described as follows.

4.3.2 Visualization perspectives

Zooming Browser is composed by a set of interactive perspectives (meeting the design principle DP13 for a multi-perspective environment), which are navigable and keep history of interactions among each other (DP6). In order to meet the design principles DP8 and DP14, the following actions took place:

- The choice of the visualization elements (visualization techniques and interactions) that compose the perspectives uses the visualization feature model (Section 4.5) as a basis.
  - This action was reinforced through a mapping between goals and visualizations (presented in Appendix A and discussed in Section 4.6).
- The choice of charts to be displayed follows the recommendations identified in a study performed previously [Queiroz et al. 2013] [Schots et al. 2015], so that the focus of representation/analysis helps to obtain insights quickly and more easily.

Appendix A presents more details on the mapping between the chosen elements/charts and the mapped data. The following subsections present the perspectives that compose Zooming Browser and their characteristics.

4.3.2.1 Dashboard

The Dashboard perspective provides a "big picture" of what is happening in the context of the reuse scenario (DP1). It is inspired in ReuseDashboard [Palmieri et al. 2013], a previous work originally comprised by APPRAiSER. However, as mentioned in Section 4.2, it was necessary to make a new implementation in Zooming Browser as a visualization perspective, integrating it with the other perspectives.

For communicating the progress of reuse initiatives in a concise and effective way (meeting RF6 and VA1), the Dashboard perspective makes use of visual analytics concepts and practices [Keim et al. 2008], in order to make it informative and actionable. The displayed information aims to generate insights to support and guide reuse managers in decision-making and stimulate developers in keeping up with reuse practices. In this sense, Dashboard can also be seen as a motivational tool, aiming at stimulating the engagement of stakeholders, providing high-level, summarized information about the core elements of APPRAiSER.

In a study conducted by Treude and Storey, developers were often unaware of some of the settings available in their tool, motivating the creation of "advanced default dashboards" [Treude & Storey 2010]. In the design of Dashboard, a mapping was performed to identify the data that would be more relevant to present as default. The main criterion was the displaying of information that summarizes the reuse scenario as a whole instead of data that cannot be interpreted quickly and would require further analyses. Appendix A presents more details on the mapping and the chosen charts.

Figure 4.3 presents the default view[35]. The bar charts highlight the consumers who reuse assets more often, the producers who develop reusable assets more often, and the projects that contain the largest number of reusable assets. The pie charts display the assets most often reused and, by selecting a consumer, the assets reused by such consumer (with the number of reuse occurrences) (DP4, DP5). A line chart depicts how often some assets are reused over time.

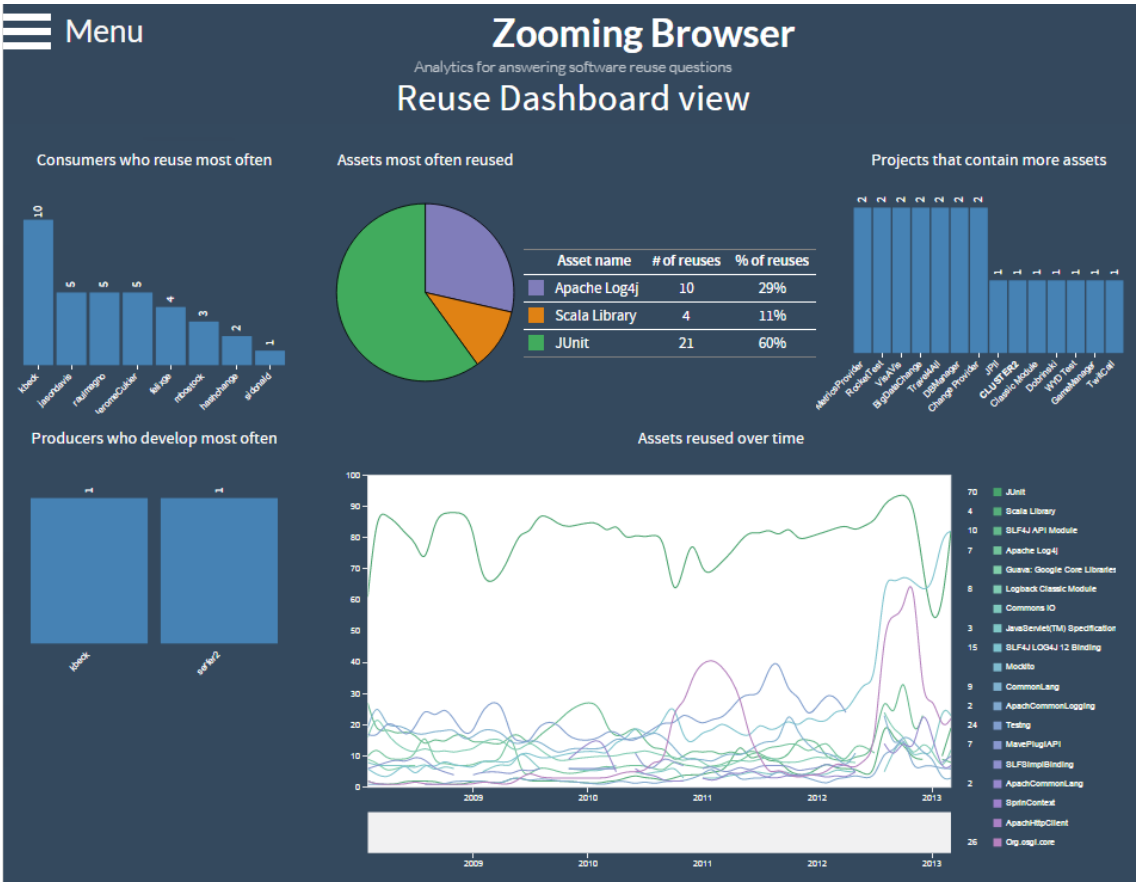


Figure 4.3 – Dashboard's default view

[35] The data presented in these charts are only for illustration purposes. They do not reflect the reality of any real developer (producer/consumer), project, or asset. Although the names used were inspired in actual developers/projects/assets, the displayed data associated to them are by no means representative of real life.

Bar charts present the data sorted (in descending order) by the bar size in order to provide a better understanding of the underlying information and detect outliers easily (DP9). For a consistent interaction pattern, changes in one element/view are propagated to the associated elements/views. Thus, when a given information is selected (e.g., a bar from the bar chart), the other views/charts update their state to reflect the selected data (e.g., a pie chart). This allows for a better exploration of the relationships between the data (DP5). Besides, transitions occur smoothly to avoid user disorientation (DP10). Figure 4.4 shows the result of hovering a slice of the pie chart.

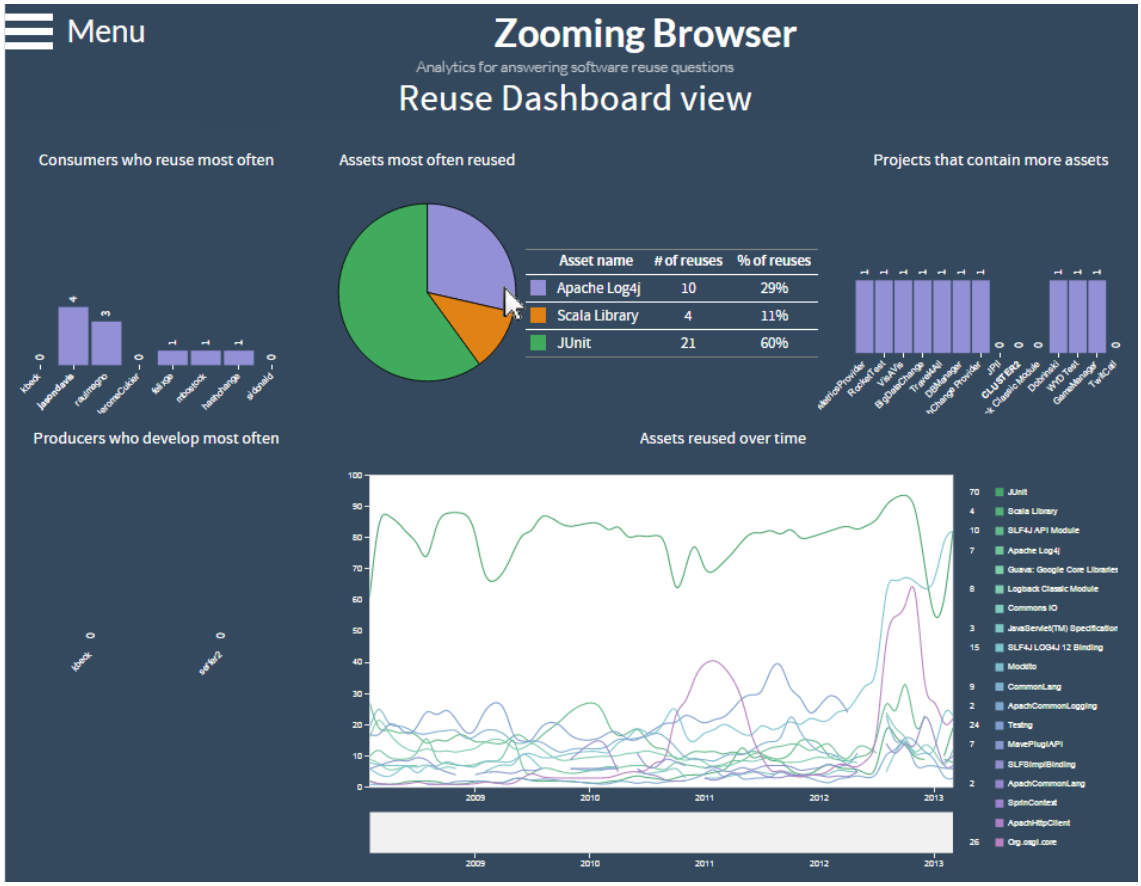


Figure 4.4 – Dashboard's view after hovering a slice of the pie chart

In this figure, it can be noticed that the consumers, producers, and projects related to the selected asset remain in the chart, while the other elements were filtered out. Besides, the consumer who most reuses assets has not reused the selected asset in the monitored projects, and such asset was reused in all 7 "top" projects monitored by Zooming Browser (i.e., those that contain more reusable assets, as shown in Figure 4.3).

Based on this set of information, it is possible to (i) identify which organization members are more experienced in reusing the asset (in case another member has difficulties in doing so), (ii) define which members could be allocated to a project that

requires knowledge on such asset, (iii) decide which projects should be better analyzed in order to understand how to reuse the asset, among others.

Since dashboards are intended to provide information at a glance and to allow easy navigation to more complete information [Treude & Storey 2010], the Dashboard perspective allows filtering (DP3) and drilling-down (DP4) from its overview information (DP1) to the Metadata Exploration perspective (presented in Section 4.3.2.2). This matches the "visual information seeking mantra" [Shneiderman 1996].

The Dashboard view is also composed by a matrix visualization of a reuse map (showing which consumers reused which assets in which projects) (DP4, DP5). The reuse map aims to provide complete, yet summarized overview about the reuse scenario in the organization. Figure 4.5 presents a screenshot of it, showing the reused assets in the intersections between consumers and projects.

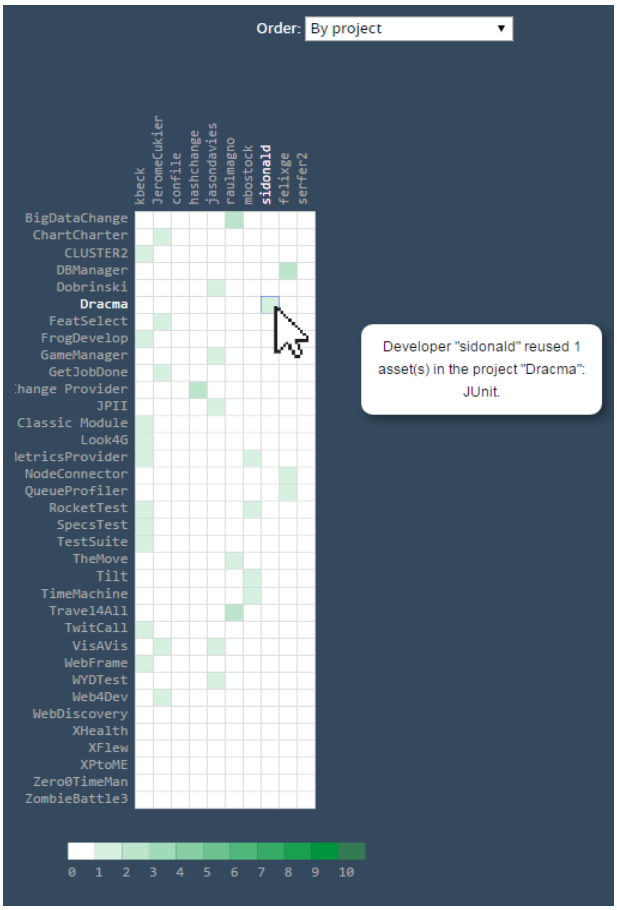


Figure 4.5 – Zooming Browser Reuse Map

4.3.2.2 Metadata Exploration

The Metadata Exploration perspective delivers information that is more specific about the core elements of Zooming Browser (DP12). Most of the questions established for APPRAiSER can be answered through this perspective. Inspired on Shneiderman's

"visual information seeking mantra" [Shneiderman 1996], it provides an overview of the entire collection of each core element (according to DP1), with interactions that allow to zoom in on items of interest (DP2) and filter out uninteresting items (DP3), then enabling to select an item and get details when needed (i.e., on demand) (DP4).

The main visual abstractions employed in Metadata Exploration are bubble charts. These are presented along with additional visualizations and/or interaction options for enriching the analysis to be performed by the user. Selection and filtering are largely applied for navigating throughout the different levels of information. Besides, the selection history is displayed visually, allowing the user to go back to a previous stage (DP6).

The main interaction flow starts by selecting one of the core elements (assets, developers, or projects) (meeting VF1). For example, if one chooses the assets option, the Metadata Exploration shows all the available assets to the user, with options to apply filtering criteria to the elements (e.g., show only assets that have been reused). By selecting a particular element (in this example, a particular asset, say "Metrics2"), all the other elements (assets) are filtered out, the selected element gains focus, and contextual items related to it appear around it – using a "Swiss Army knife" metaphor – to depict the additional information (DP5) available about such element (meeting VF3).

Contextual items vary according to the type of core element, and different actions can take place, depending on the kind of contextual item selected. For instance, when the contextual item "Consumer who reused this asset" is selected in the context of the asset "Metrics2", another bubble chart appears displaying the solicited information. Each visual attribute in the visualizations has a meaning, as described in the mapping presented in Appendix A. In this context, each bubble represents a consumer who reused the asset "Metrics2", and the size of the bubble indicates the number of reuse occurrences. This information presented visually can point to a potential interested party to be notified about changes in the status of the reused asset (meeting RF4).

Another result of selecting a contextual item is the transition between perspectives. The selection of the "Release History" contextual item triggers the History perspective (described in Section 4.3.2.3). Contextual items whose information is not available are visually de-emphasized (i.e., displayed with low opacity) to depict that. Additional visualizations may appear according to the contextual items selected.

Akin to the Dashboard perspective, the Metadata Exploration perspective meets the criterion for smooth transitions (DP10), so that the user does not get lost or confused

when/while the view changes. Figure 4.6 depicts an example of navigation flow, from the selection of one of the core elements to the detailed information about it.

Analogously to the asset core element, the developer core element shows both consumption and production information regarding the developer (e.g., projects in which the consumer reused assets, development collaborations in asset production etc.). The project core element, in turn, brings information about consumers who reused assets in this project, assets reused in the project, amongst others (DP5).

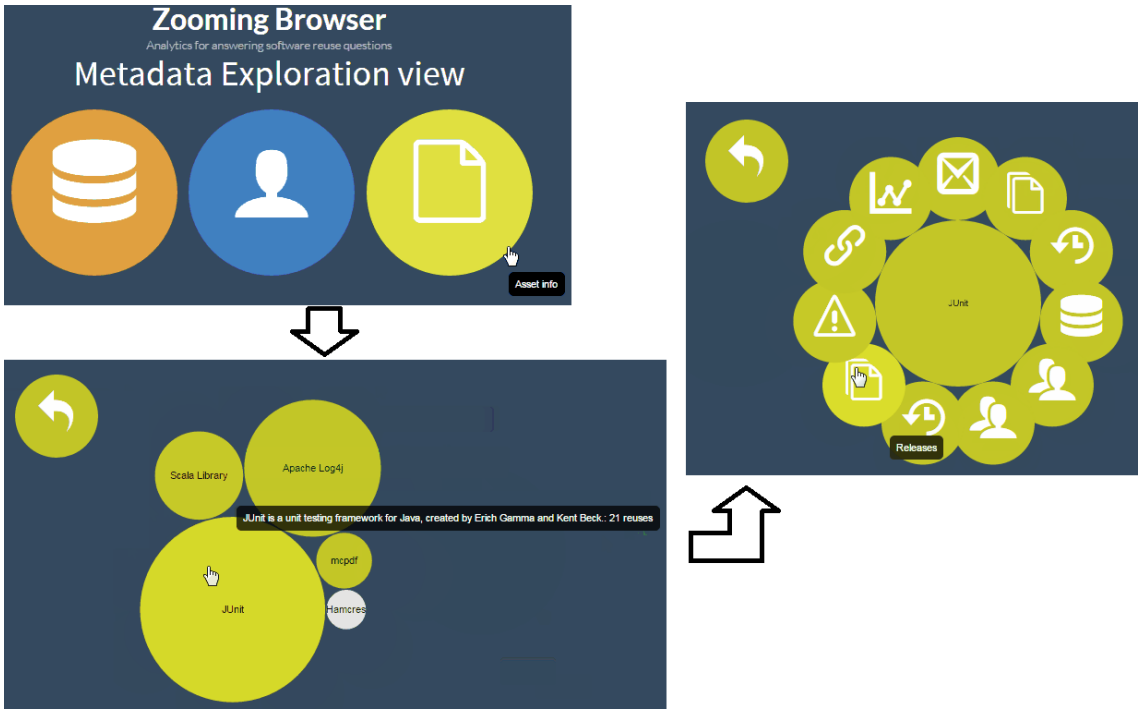


Figure 4.6 – One of the possible paths on the Metadata Exploration navigation flow

4.3.2.3 History

The History perspective presents data related to the evolution of the core elements, using coarse-grained and fine-grained representations (meeting VF2). This perspective is divided into two main views: the VCS Development History Graph and the Release History Graph.

The former is inspired in GraphVCS [Pereira & Schots 2011] [Pereira & Schots 2014], a previous work that aims at visualizing the structure and metadata of VCS repositories. The asset development history is based on its associated VCS repository (if any). The VCS repository structure is represented through visual graphs, in which each commit operation and project milestones (tags) are depicted as nodes, while edges denote the main line of development (trunk/master) or its derivations (branches)

composed by the nodes (DP5). It is possible to perform panning and zooming operations from the overview (DP1, DP2), as well as to drill-down to a given version (DP4).

The Release History Graph, in turn, aims at depicting releases of an asset (also referred to as asset versions) or releases of a software project that contain a reusable asset. The release history information takes into account the semantic versioning[36] commonly used in asset development projects. Such information is obtained from reuse/release repositories (e.g., Maven Central or Nexus). The Release History Graph allows filtering by collapsing/expanding the releases (DP3), and presents tooltips with information related to the release (DP12).

The metadata information depicted in the Metadata Exploration perspective related to an asset (or project) development history can be drilled-down (DP4) to the History perspective. For instance, for the VCS Development History Graph, a graph visualization of the project's history is presented, whose nodes are highlighted (DP3) according to the kind of information solicited in the Metadata Exploration perspective (e.g., highlighting project versions in which an asset is present, for understanding the context in which it was reused).

Figure 4.7 illustrates a hypothetical scenario in which it is possible to observe the moment when an asset was included in the repository (indicated by the "+" sign). It can be noticed that it was added during a bug fix (depicted by the label in the branch) and was later integrated to the main development branch ("master" label). It is not possible to assess (based solely on these data) whether it was fully effective to solve the project needs – this requires a drilling-down operation (potentially to the raw format of the files involved in the commit, in this case) in order to obtain additional details. Thus, this view serves as a guidance indicating where to seek for more information.

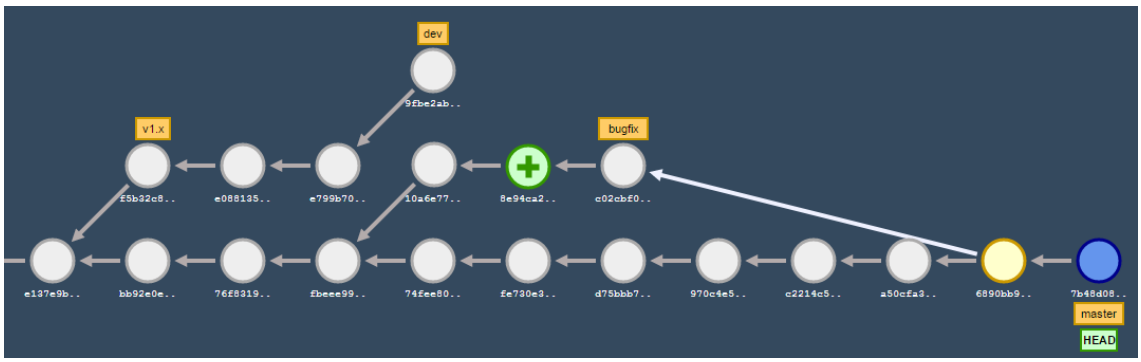


Figure 4.7 – History perspective (VCS Development History Graph view)

[36] http://www.semver.org/

The Release History Graph focuses solely on releases of assets/projects, and shows the frequency of releases and the more up-to-date ones, so that one can be aware of them and consider potential upgrades. The releases are displayed hierarchically (DP5), according to their semantic versioning identification. This information can be extracted from reuse repositories or VCS repositories. Figure 4.8 presents the Release History Graph of the JUnit project (stored in the Maven Central repository).

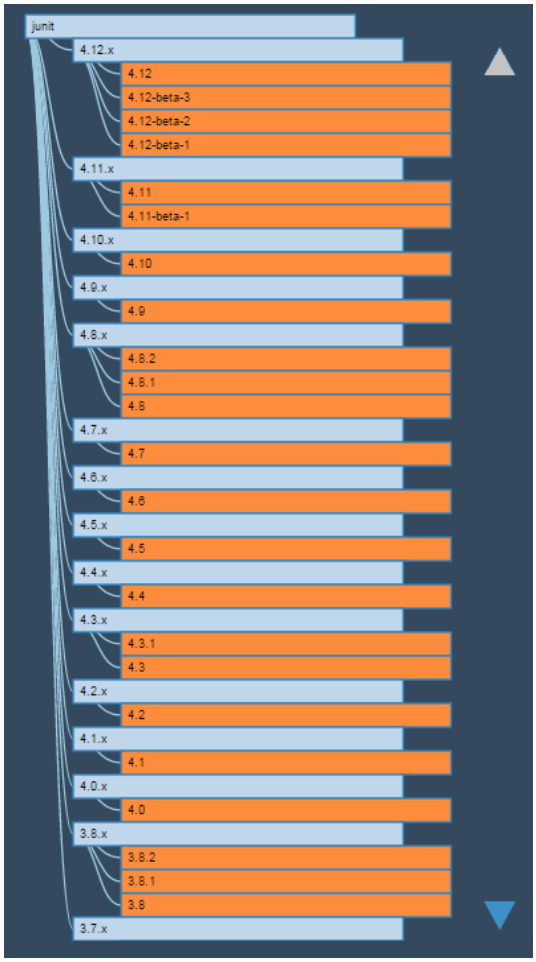


Figure 4.8 – History perspective (Release History Graph view)

In order to ease the location of project information, a search tool is integrated with the visualization features, allowing the filtering of the displayed information (DP3), using smooth transitions (DP10). This facilitates the exhibition of details on demand, and allows for maintaining context without losing focus of the task. The search criteria include date, version ID/number, commit messages, author, and asset/file.

4.3.2.4 Low-Level Data Representation

The choice of the visualization abstractions and techniques for representing the data, as well as the interaction techniques to be employed, heavily depends on

contextual information. Such information includes the nature of data, the visualization constraints, and the task to support (e.g., selecting the most suitable asset for a project from a set of reusable assets) [Queiroz et al. 2013] [Vasconcelos et al. 2014b].

In this sense, CAVE (Context-Aware Visualization Engine) [Vasconcelos et al. 2013] [Vasconcelos et al. 2014b] [Vasconcelos 2015] is a mechanism that displays different visualization metaphors according to the contextual information of the data and the task, which defines the focus of representation (DP14). It takes as input a context-aware feature model [Fernandes et al. 2011] that maps context information and context rules that activate context situations.

The activation of context situations leads to the selection of features from the visualization feature model (described in Section 4.5), also used as input. The association between context situations and visualization features is expressed in CAVE in terms of composition rules, defined as follows:

R_X – <Context Situation> **implies** <Visualization Feature(s)>

The integration between CAVE and Zooming Browser occurs in the following way: Zooming Browser allows drilling-down to a visual representation of low-level information (e.g., the assets' structure) or its raw format (source code, document etc.) (DP4). Such visual representations are provided by CAVE: the tool is invoked by Zooming Browser to depict elements according to the available low-level data and user tasks. It extracts information from these elements to depict them from different granularity levels, using different visual abstractions.

CAVE is composed by three modules [Vasconcelos 2015]. The *Context Manager* checks the information, situations, and rules from the context model used as input and, according to the data present in the data source, it points out situations that are active at a given moment. The *Context Information Connector* presents the context information supported by the tool, with the query mechanism responsible for determining the information value. Finally, the *Visualization Connector* performs the selection of the visualization elements defined in the active context rules (meeting RA4).

One of the context-aware CAVE features is the adaptation of the visual layout based on the data characteristics (e.g., according to the amount of data being displayed, which is one of the context rules) (DP7). For not hampering the user analysis, a notification is shown at the top left part of CAVE to indicate that a new context situation is active.

The reasons behind this notification mechanism are the following: (i) the user may be analyzing an outdated set of data, (ii) the user must be able to choose whether the switching of visualizations should occur, and (iii) visualizations should not change abruptly, otherwise the user would get lost in the analysis process (DP10).

Figure 4.9 and Figure 4.10 show the notification mechanism implemented in CAVE and the visualization of the structure of a project, respectively.

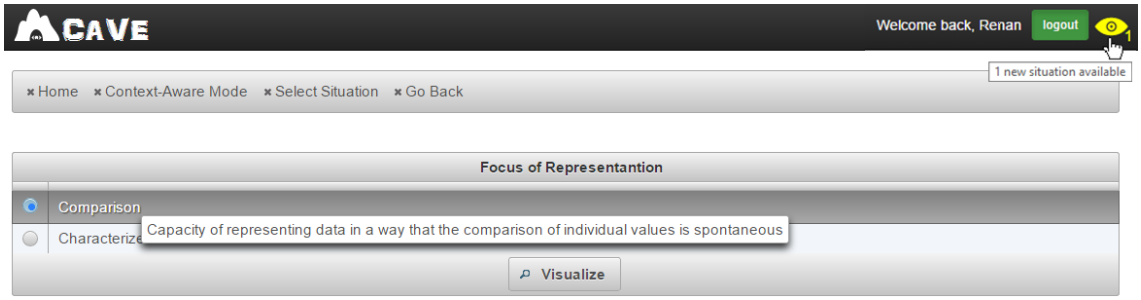


Figure 4.9 – CAVE notification of an active context situation (top left part) (adapted from [Vasconcelos 2015])

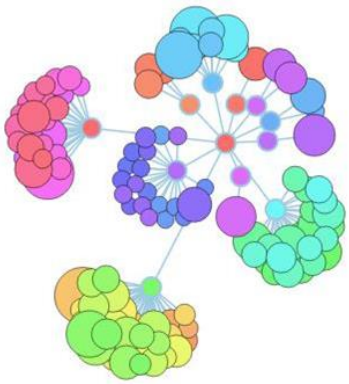


Figure 4.10 – CAVE visualizing a project structure with Code Flowers[37] (each bubble represents the size of a class, in terms of lines of code (LOCs)) [Vasconcelos 2015]

CAVE allows analyzing and comparing properties of the reusable assets. When more than one asset fits the developer/project needs, this feature can support the decision-making regarding which one should be reused (with the support of the other Zooming Browser perspectives, which depict other relevant metadata). With respect to the suitability of visualizations, a research was conducted for correlating types of task and types of visualization in terms of representation constraints [Queiroz et al. 2013] [Schots et al. 2015]. More details about CAVE can be found in [Vasconcelos 2015].

[37] Based on the D3.js implementation available at https://github.com/fzaninotto/CodeFlower.

4.3.3 Implementation details

Figure 4.11 depicts an overview of the Zooming Browser tool, along with its interactions with other APPRAiSER elements[38]. For its execution, it is necessary to use a web browser (to access both organization-independent and organization-specific versions) and an installation of the Node.JS application server (to instantiate the tool to a particular organization).

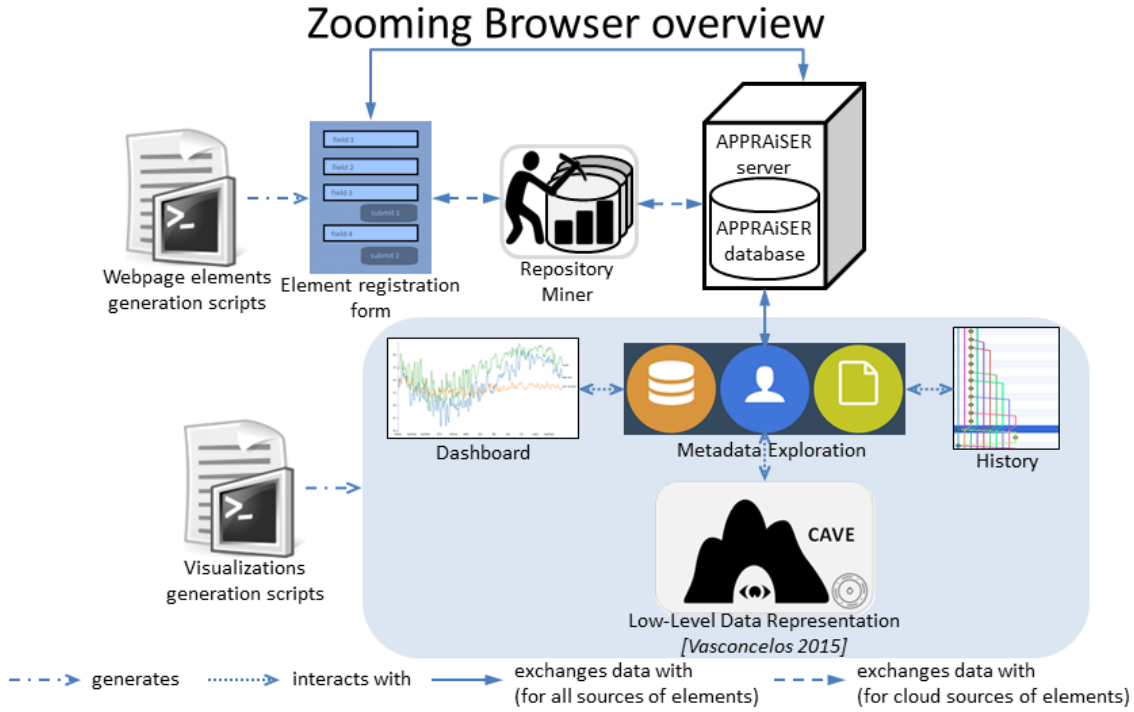


Figure 4.11 – Zooming Browser overview

Zooming Browser was integrated to APPRAiSER using a client-server architecture, with a thin client integrated through the REST architectural style[39] [Fielding 2000], with some characteristics of the presentation-domain-data layering modularization[40]. The server-side architecture of Zooming Browser has some similarities to the one presented in [Gousios et al. 2014], consisting of two loosely coupled parts: a web server (responsible for handling CRUD requests from the client side) and the APPRAiSER server (that performs the data extraction and persists them on

[38] This figure only presents the main elements relevant to this section. For a better understanding of all the APPRAiSER elements, please refer to Figure 4.1.

[39] REST is a hybrid style derived from several of the network-based architectural styles; by separating the user interface concerns from the data storage concerns, portability of the user interface is improved across multiple platforms and scalability is improved by simplifying the server components [Fielding 2000].

[40] http://martinfowler.com/bliki/PresentationDomainDataLayering.html

the APPRAiSER database). The APPRAiSER server routes requests made by the web server, redirecting to a Node module that is able to handle them[41].

On the web server side, a response listener handles incoming results (one for each repository in each request) and updates the web client. The provided responses can be either a message (informative of error, success, or warning) or a JSON (JavaScript Object Notation)[42] object with the solicited metadata. In fact, the data interchange between most of the APPRAiSER elements uses REST and JSON, since they are lightweight solutions. Besides, the JSON format allows reusing data in different visualizations.

The technology chosen for the server is Node.JS[43], an event-driven server-side JavaScript environment that is recommended as a backend for single-page web applications (as in the case of Zooming Browser). Since it processes JavaScript at the server-side, implementation details are hidden from the client, and both client and server use the same programming language. The server also triggers notifications about changes on the status of the assets to the registered interested parties (meeting RF4).

The Zooming Browser visualizations and interactions are implemented with the D3.js [Bostock et al. 2011][44] framework, a JavaScript library that provides useful visualization components, being a data-driven approach to manipulating cross-platform DOMs (Document Object Models). JavaScript-based selections provide flexibility on top of CSS, as styles can be computed dynamically in response to user events or changing data, allowing smooth transitions (as specified in DP10). D3.js has also shown improved scalability among browser-native tools [Bostock et al. 2011], also allowing a faster interaction response time (helping to meet the design principle DP11).

All the scripts for generating the visualizations in APPRAiSER are implemented with this framework. D3.js is also used for generating dynamic elements on the Zooming Browser page (e.g., the element registration forms according to the option selected). The transitions between perspectives use the Reveal.js framework, which provides automatic scaling and orientation according to the device being used for interacting with Zooming Browser (DP7). Besides, some reused scripts were already

---

[41] More details on this implementation aspect can be found in Section 4.4.2 (which presents the Repository Miner implementation).

[42] http://json.org/

[43] https://nodejs.org/

[44] http://d3js.org/

compatible with responsive design directives, and some adjustments were made in Zooming Browser in order to meet one of the APPRAiSER desirable features (VF6).

4.4 Repository Miner

Providing relevant information about reuse integrated from different sources can result in several benefits to software organizations, e.g., assist reuse managers in tracking reuse occurrences and maintaining the reuse repository, support potential consumers by giving more confidence in deciding whether or not to reuse/upgrade an asset, and provide producers with an overview on how their assets are being reused.

The Repository Miner can be seen as a framework that integrates different sources of data (according to RA3 and VF4) for the purpose of gathering information about assets, developers, and projects (meeting RF3 and VF1), as well as tracking reuse assets for proper reuse management (meeting RA5). It is responsible for mining different kinds of software repositories, both organizational (internal) and external, searching for different kinds of reuse-related information, which is delivered to APPRAiSER and its tools [Schots 2014a] [Schots 2014b].

4.4.1 On the sources of data

A subset of data sources related to a software project was selected for analysis, aiming to assess whether and how they could help with providing relevant information to support the execution of some reuse tasks (and thus be integrated to APPRAiSER).

Because reusable assets play the major role in reuse, the first considered source is the *reuse repository*. Reuse repositories contain the release history of a given asset (i.e., its different versions over time) and important metadata about them (e.g., their descriptions, licenses, dependencies, among others), supporting the maintenance of reuse initiatives.

APPRAiSER can handle both in-house reuse (also known as "internal reuse") and reuse from external sources (also known as "external reuse"), based on APIs developed for interacting with the repositories, if available. An organization can reuse both an asset from an external source (e.g., the Log4j[45] component) and an asset developed internally (e.g., a "FormManager" component). The metadata of each asset are extracted from the corresponding repository (either internal or external).

[45] http://logging.apache.org/log4j/2.x/

Another important source of information is the *VCS repository* of each software project. Projects' evolution information (e.g., commit authors, dates etc.) can be used for identifying reuse occurrences, i.e., the presence/inclusion/removal of assets (present in a local or remote reuse repository) in/from the projects' VCS history. This indicates how such assets are reused, and can help suggesting assets that are frequently reused in other projects developed by the organization to be incorporated in the current project. The latter assets can be candidate to the organization's reuse repository (depending on organizational criteria) (meeting RA6 and VF2), in case they were not yet in such repository.

APPRAiSER also collects information from the project history of a reusable asset, aiming to provide the "big picture" of its development, allowing to assess assets' stability and frequency of updates (i.e., how active the development community is).

Project history information and metadata can be obtained from both the organization's portfolio and selected open source projects. The reasons are twofold. In the beginning of the definition of reuse practices, it is unlikely to find enough information about an asset (that encourages its reuse) on local configuration management systems. On the other hand, experiences from the organization itself (even if a reuse-based process has not yet been established) are essential for promoting reuse.

Although these sources of information complement each other, they are handled separately since they allow for different kinds of insight. The fact that an asset is widely reused in open source projects may give more confidence in reusing it, while an asset reused in the organization's projects (successfully or not) gives a clue of the chances of success/failure based on such previous experiences.

Finally, *issue tracker/task manager repositories* provide information that complements the VCS information, including issues that affect reusable assets. Reported issues related to an asset (e.g., problems and feature requests) are relevant to assess the asset's stability and quality, while issues assigned by or to producers can help noticing how they are maintaining their assets.

By mining and combining these sources of information, one can find correlations and identify facts that may be of interest to reuse managers and developers (helping to identify interested parties, as established in RF4).

4.4.2 Implementation details

An overview of Repository Miner is presented in Figure 4.12.

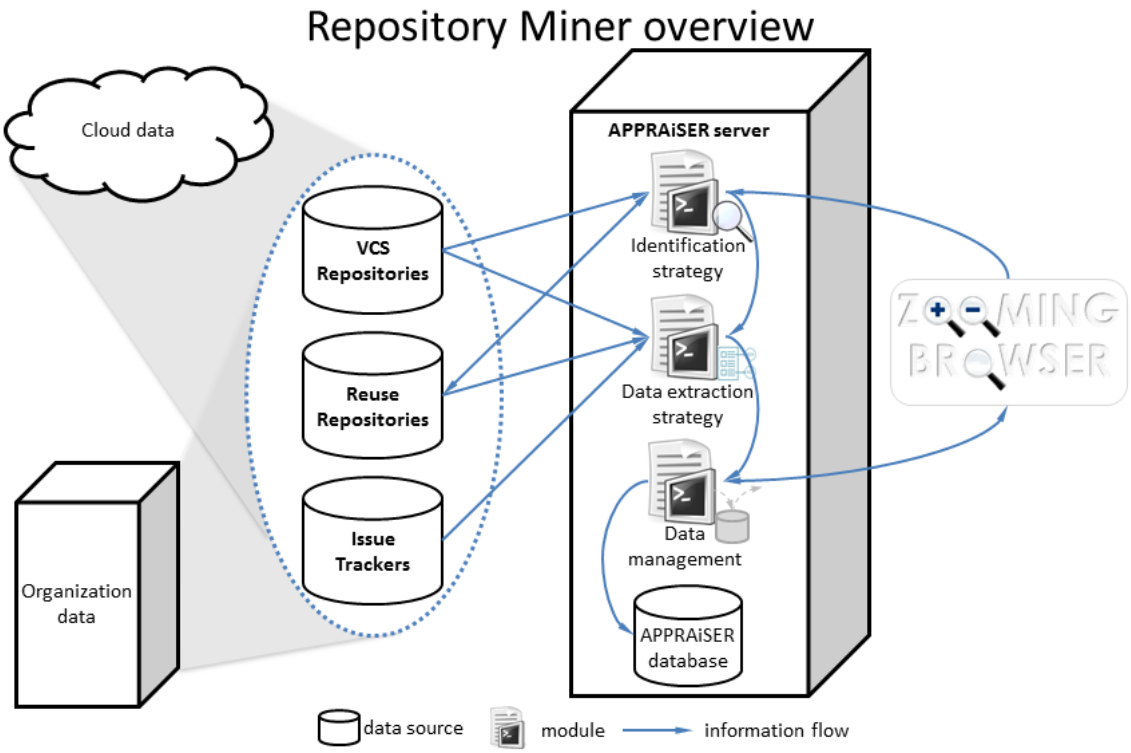


Figure 4.12 – Repository Miner overview

The data sources discussed in Section 4.4.1 are present in both cloud and organization-specific repositories (according to RA5). Each of these sources is mined based on a *data extraction strategy*, i.e., a script that defines how the data from each source is organized, and how it must be parsed to be later added to the APPRAiSER database. There is one miner for each repository (not depicted in Figure 4.12), as follows:

- The *Reuse Repository Miner* collects links to assets and their metadata from reuse repositories, in addition to social information regarding producers (organization, developers, contact information, website etc.);
- The *VCS Miner* captures project information from VCS repositories, such as commits, authors, dates, and so on; it also collects information about projects in which a particular reusable asset was consumed (on demand) and about consumers who reused them in such projects;
- The *Issue Tracker Miner* collects issues and their metadata (status, priority, start and due date, interested parties etc.) from issue trackers.

For detecting the existence of reusable assets in VCS repositories, there is a script that implements an *identification strategy*, which is asset-specific (meeting RA6). Historical data on previous reuse occurrences (commit data) are extracted in a semiautomatic way. Assets are identified automatically, but the project manager or a

project developer must (i) confirm that the identified asset is indeed a candidate to enter the reuse repository (to be assessed by the reuse manager), and (ii) indicate or confirm the version of this asset. Note that some identification strategies work better for assets whose reuse did not required changes to its internal structure, and may benefit from projects that follow a default structure.

If the project history is stored in an organizational VCS, the Repository Miner obtains metadata and information about reuse history from it. Developers who host their projects in online platforms (such as GitHub) can also obtain such metadata and information from it according to what the platform provides via API.

The default (built-in) identification strategy module implemented in the Repository Miner also uses the reuse repository in an attempt to match assets reused in projects and reusable assets present in the repository. Mismatches must be corrected manually if necessary.

Finally, the extracted data are sent to the data management module, which persists them in the APPRAiSER database. As shown in the figure, Zooming Browser can also trigger both the identification of assets and the data management modules.

Figure 4.13 presents the information flow between sources of data, the server, and Zooming Browser, while Figure 4.14 shows the database schema that defines the data extracted by Repository Miner and stored in the APPRAiSER database.

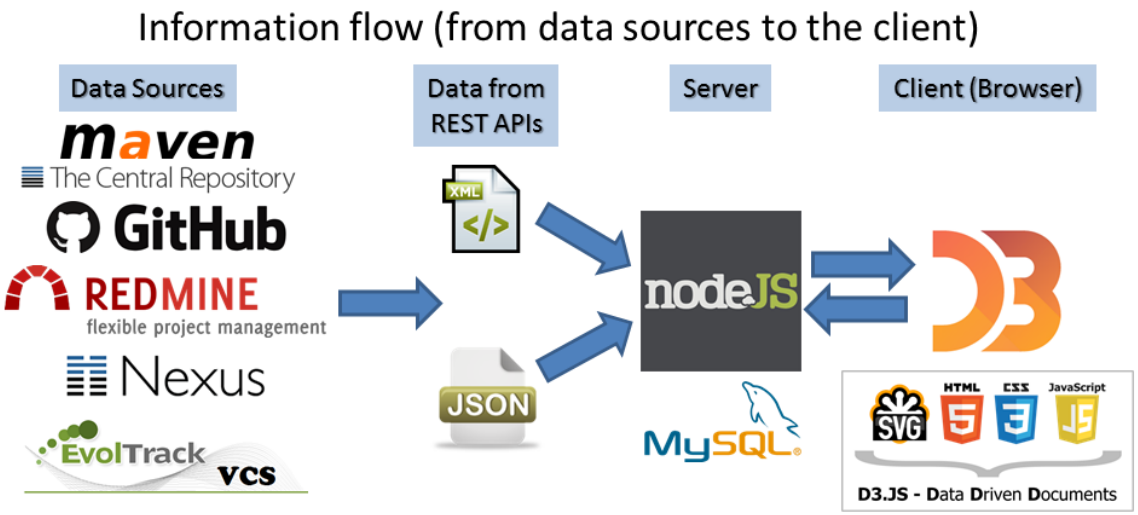


Figure 4.13 – Information flow between Repository Miner and Zooming Browser

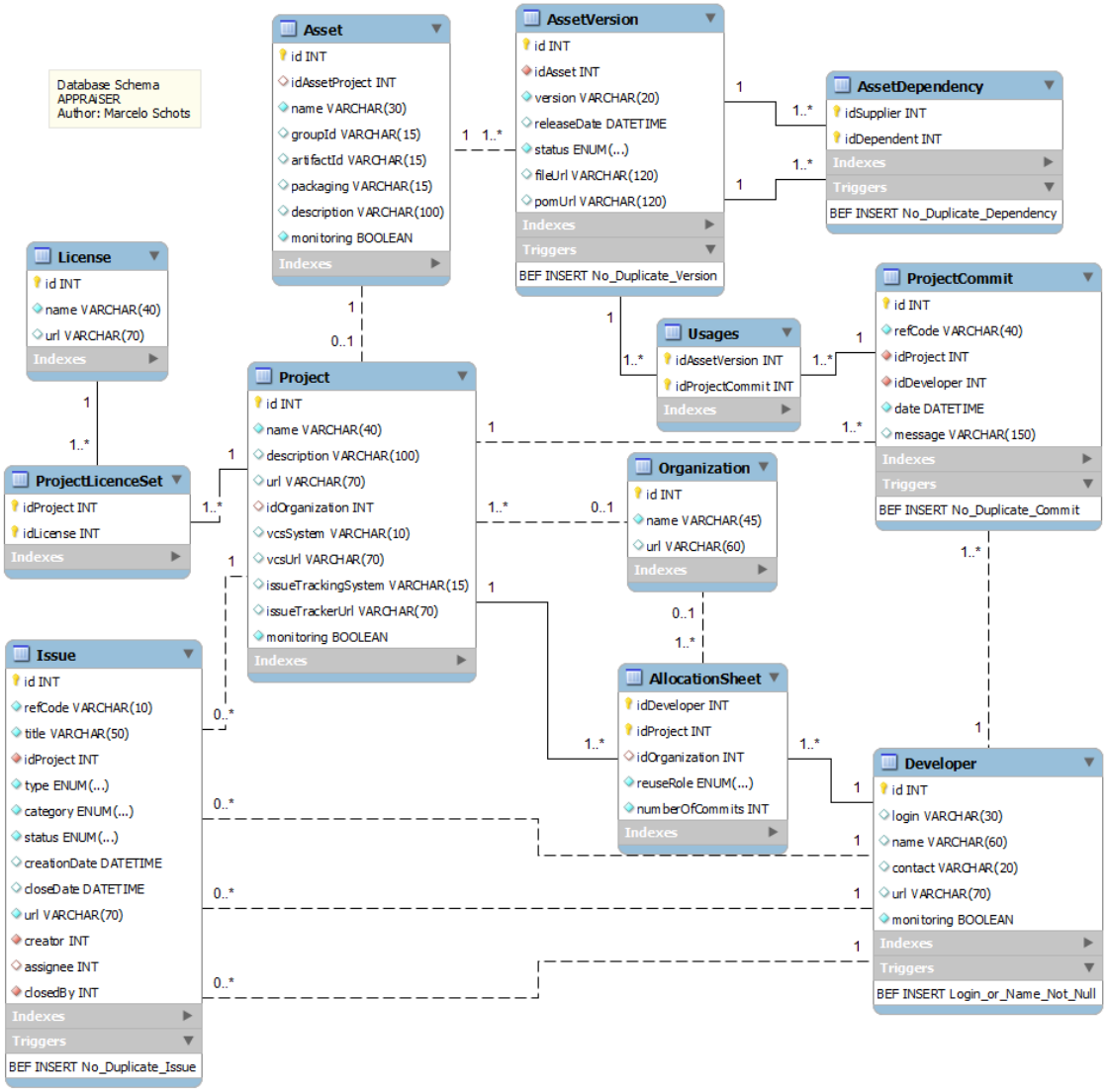


Figure 4.14 – APPRAiSER database schema

As regards to the technologies used, Reuse Repository Miner currently uses Maven Central Repository[46] as its main data source in the cloud and Nexus instances for organizational reuse repositories (meeting RF5). VCS Miner, in turn, currently uses GitHub[47] (for extracting versioning data and metadata from projects versioned in this platform) and EvolTrack-VCS [Werner et al. 2011] (based on the Maven SCM API[48]) for extracting versioning data from native repository implementations (Git, SVN etc.)

[46] The Maven Central Repository (http://search.maven.org/) is one of the most well-known examples of asset repository. There are plugins that provide easy integration with several IDEs. Each Maven asset contains a Project Object Model (POM) file that summarizes the most important metadata about the asset, including the address of its version control and issue tracker repositories (if available).

[47] The current version of Repository Miner extracts available information from Git repositories that do not require authentication.

[48] Available at http://maven.apache.org/scm/scms-overview.html

that are not in a hosting service. Finally, Issue Tracker Miner currently supports GitHub and Redmine (both cloud-based and organization-specific instances).

Each of the modules mentioned in Figure 4.12 is implemented as a Node.JS module in separate files. The decision of modularizing the server functionalities adheres to the principle of separation of concerns, and helps to ease interchanging of modules when necessary (meeting RF2).

The data that support APPRAiSER are extracted from software repositories by means of APIs or REST requests. Because the GitHub API imposes limitations on the number of requests per time interval[49], and aiming to improve performance, APPRAiSER makes use of caching (a strategy adapted from [Gousios et al. 2014]). If the APPRAiSER database already contains data for the referred core element, then the APPRAiSER server first asks GitHub if the solicited information was updated since the last extraction[50]. If there was no update, no GitHub request is made, and the requested metadata for this element are extracted from the APPRAiSER database. Otherwise, both the APPRAiSER database and the information requester are updated with data freshly extracted from the GitHub API.

With respect to the Maven API, the APPRAiSER server only asks for metadata about an asset version if they are not available in the APPRAiSER database, since they are not likely to change. On the other hand, if the metadata are related to an asset, developer, or project, the APPRAiSER server performs request operations (since it does not seem to have limitations on the number of requests).

The data formats used in Repository Miner (JSON and XML) are generic, and supported by several different tools (meeting VA2). Thus, the effort associated to changing any repository (regardless of being a VCS, reuse, or issue tracker repository) or a type of asset (e.g., JavaScript files in the NPM repository) is restricted to:

- the communication with the API of the new repository,
- an adaptation on the identification strategy[51], and

[49] Please refer to https://developer.github.com/v3/rate_limit/ for more information.

[50] This command does not count against the request rate limit. For more information, please refer to https://developer.github.com/v3/#conditional-requests.

[51] In principle, the identification strategy can be applicable to any kind of asset that matches a given pattern, but restrictions apply in terms of the potential number of false positives (e.g., if the reusable asset unit is a Java class).

- the formatting and structuring of data (that must conform to the persistence and visualization mechanisms), which can be done, for instance, through a wrapper.

Although the Repository Miner is not a "one-size-fits-all" tool, it is believed to be flexible enough to accommodate other kinds of reusable assets with few extensions and customizations, assuming that the reuse repository which contains such assets is able to provide the information expected by APPRAiSER. In other words, the Repository Miner architecture is loosely coupled, so that the interaction between the scripts developed in JavaScript can be adapted and interchanged without much effort.

It is known that the kind and amount of information available from local and external repositories may differ considerably, depending on the platform that hosts such information. Besides, some fields of information that are not required for some repositories (e.g., the project description) may have not been filled out.

To overcome this limitation, Zooming Browser's user interface provides *element registration forms* (depicted in Figure 4.15) for filling out information about a core element (asset, developer, or project), allowing to include information that is only available locally or is missing from the data sources (e.g., some contact information that could not be retrieved). Thus, after the extraction process, APPRAiSER allows the end user to complete any missing information (through Zooming Browser) for storing it in its database.

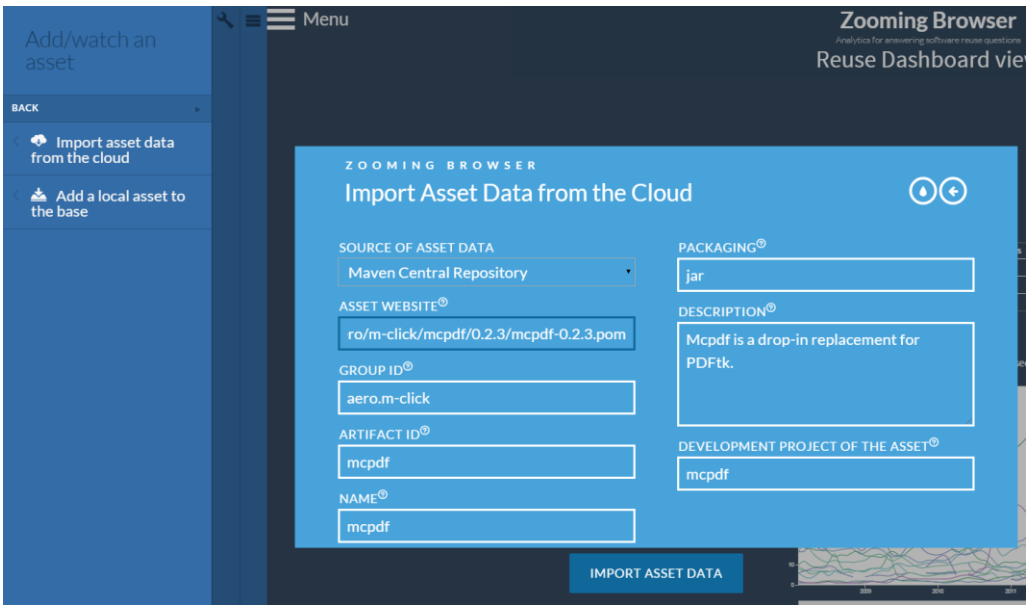


Figure 4.15 – Form filled out by the developer; if a Maven URL is provided, the remaining information is filled out automatically

4.5 Visualization Feature Model

There are several visualization elements[52] available in both the state-of-the-art and the state-of-the-practice, but composing one or more visualizations that can represent everything needed is not a simple task. Since the number of visualization alternatives keeps growing, it is important to adopt some sort of mechanism for organizing their features and allowing the selection of the most suitable ones. The knowledge of existing visualization and interaction features helps to choose only the necessary features and, ultimately, composing visualizations with less effort.

Feature models are a useful way to represent domain knowledge in terms of the elements (features), their relationships, and their constraints of use, facilitating the understanding of such concepts in the domain (meeting VA2). An advantage of feature models lies on the acceptance of features as an effective "media" supporting communication among stakeholders [Lee et al. 2002].

A feature model for the information visualization domain can favor building different views that, regardless of addressing the same kinds of issues, are intended to support different stakeholders (taking into account their particular analysis perspectives). This assumption led to proposing the visualization feature model [Vasconcelos et al. 2014a] as part of APPRAiSER, aiming at providing basic knowledge on the visualization domain based on its features and their restrictions of use. Instead of being a *prescriptive* model (that defines which visualization elements should be used in a given situation), this model has a *descriptive* nature. By presenting the features along with their relationships and constraints, it serves as initial guidance for reasoning about the available alternatives and choosing features for the creation of visualization tools that integrate the approach (and potentially other visualization tools as well, to be built by other researchers and practitioners).

The following subsections present the organization of visualization features based on a domain analysis carried out to identify different characteristics in the visualization domain, resulting in the feature model that organizes the findings and eases the selection of visualization features (meeting VA3).

[52] A visualization element is interpreted as a concept that can be used in the context of a visualization tool, represented by visualization metaphors, paradigms, and techniques.

4.5.1 Domain analysis

According to [Braga et al. 1999], the domain analysis consists in the definition of main domain concepts, standing out similarities and differences among these concepts in a high abstraction level. During the domain analysis, models are divided based on main characteristics (i.e., features) of the domain.

The domain addressed in this study refers to information visualization. The adopted methodology for the analysis is based on three steps:

- an informal literature review for identifying an initial set of visualization and interaction elements (including the ones identified in a previous work [Oliveira 2011]);
- a *quasi*-systematic literature review [Schots et al. 2014], used in the context of this study for confirming the use of the already classified elements and for complementing the model with new candidates; and
- an evaluation (with experts and intermediate-level researchers in the domain) in order to validate the findings and complement the model[53].

Although the object of investigation of the second step was restricted to software visualization approaches proposed to support software reuse, such a literature review enabled to gather initial knowledge from software engineering research. The adopted data extraction methodology was based on the dimensions of software visualization, discussed in Section 3.3.2, particularly with respect to the Representation dimension.

The results from the *quasi*-systematic literature review pointed out interesting findings as regards to the relevance of visualization elements. For instance, from the approaches identified in the publications, 6 visualization and interaction elements are mentioned simultaneously in more than 10 approaches, namely: Selection, Navigation, Drill-Down, Clustering, Highlighting, and Labeling [Schots et al. 2014]. Such candidate elements were selected among others as characteristics of the visualization domain.

As mentioned in [Vasconcelos 2015], it is important to highlight that the domain analysis performed does not aim at being a complete reference for visualization features applicable to any scenario, since other aspects should be considered, such as the mapping between a visualization technique and a data set. A cautious analysis is necessary to define the factors that will define such requirements.

---

[53] Because this is a part of the evaluation of APPRAiSER, this step is described in Section 5.2.

4.5.2 Feature model elements

In order to compose and organize the feature model, the Odyssey-FEX notation [Blois et al. 2006] was used for representing the different types of elements and supporting the domain analysis process, due to the researchers' previous knowledge on its syntax and to its wide-scope representation model.

For better structuring the model, some high-level, conceptual categories were defined to group similar elements. Although all the elements were identified based on works that present visualizations, some of them were strictly related to interaction functionalities on visualizations (*Interaction* category). Another group of visualization elements was interpreted as alternatives for presenting different visualizations (*Presentation* category). Finally, the third proposed group relates to changing the exhibition mode (*Information Visualization* category).

The visualization feature model is a constant work-in-progress and will evolve as new features are identified or changes in their organization must be updated. Figure 4.16 and Figure 4.17 present the most recent version of the model to date[54].

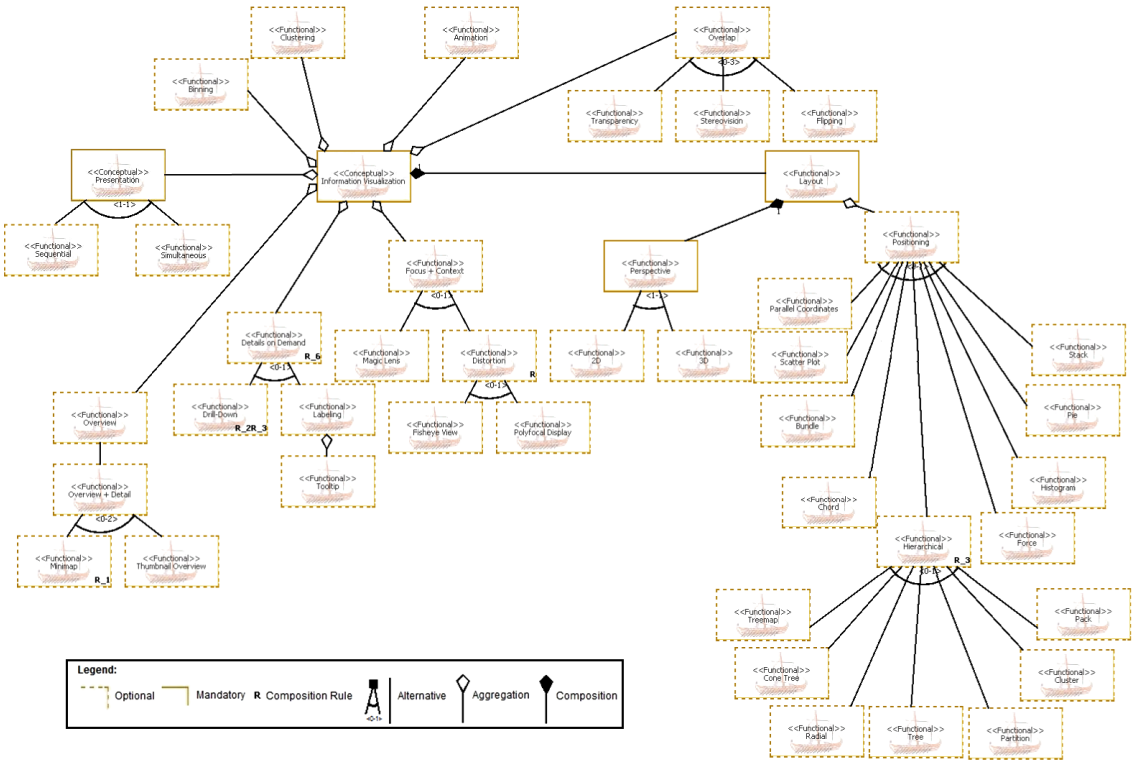


Figure 4.16 – Visualization feature model (presentation and information visualization features) [Vasconcelos et al. 2014a] [Schots et al. 2015]

[54] Due to its composition by many features, please refer to an electronic version (available at http://www.cos.ufrj.br/~schots/papers/featuremodel.png) for a better visualization.

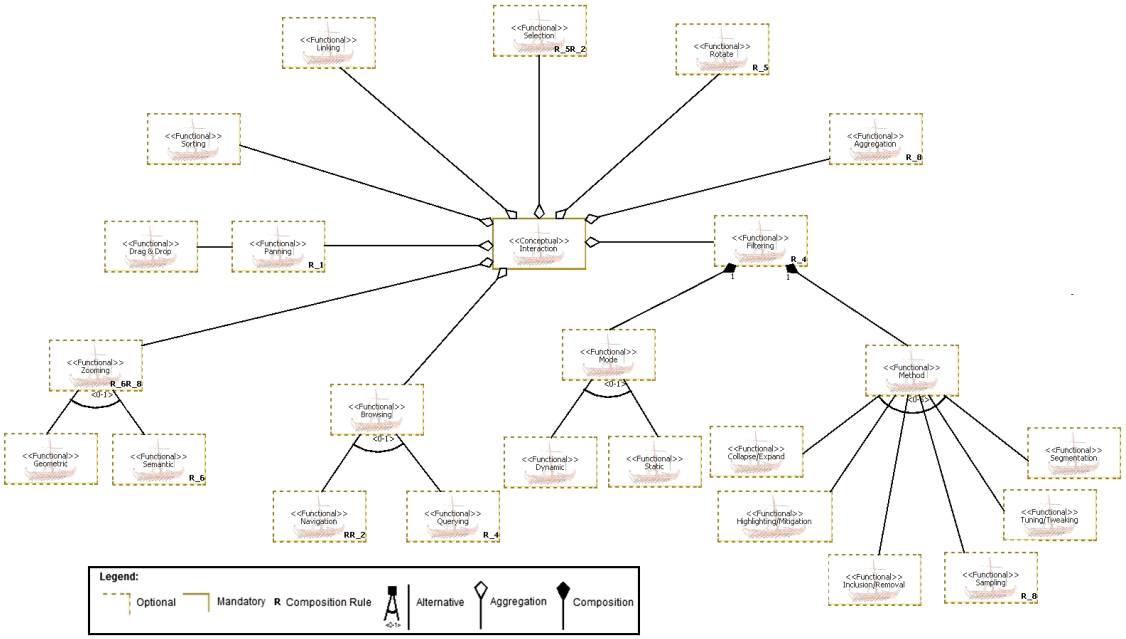


Figure 4.17 – Visualization feature model (interaction features) [Vasconcelos et al. 2014a] [Schots et al. 2015]

The following subsections address only an excerpt of the model, for didactic purposes. The detailed description of each feature can be found in [Schots et al. 2015].

4.5.2.1 Interaction features

For visualizing data, it is essential to map them into visual representations in a way that the result is the most intelligible as possible. However, if the user cannot arbitrarily manipulate a particular visualization, many dataset characteristics may remain hidden [Few 2009]. Thus, interaction techniques, such as the ones presented in [Yi et al. 2007], represent an important feature set that allows the user to manipulate the visual representation for exploring and interpreting the underlying information.

Interaction features are related to actions performed by a user on a view. Figure 4.18 shows some of these features and their relationships. For instance, by selecting the Zooming feature with its Semantic variant for composing a specific visualization, a user can zoom in and out, revealing different visual representations and/or details to information items [Buering et al. 2006] [Cockburn et al. 2008]. In addition, through the Browsing feature along with its Querying variant, the user can perform searches in order to organize or restrict the amount of data displayed in a view.

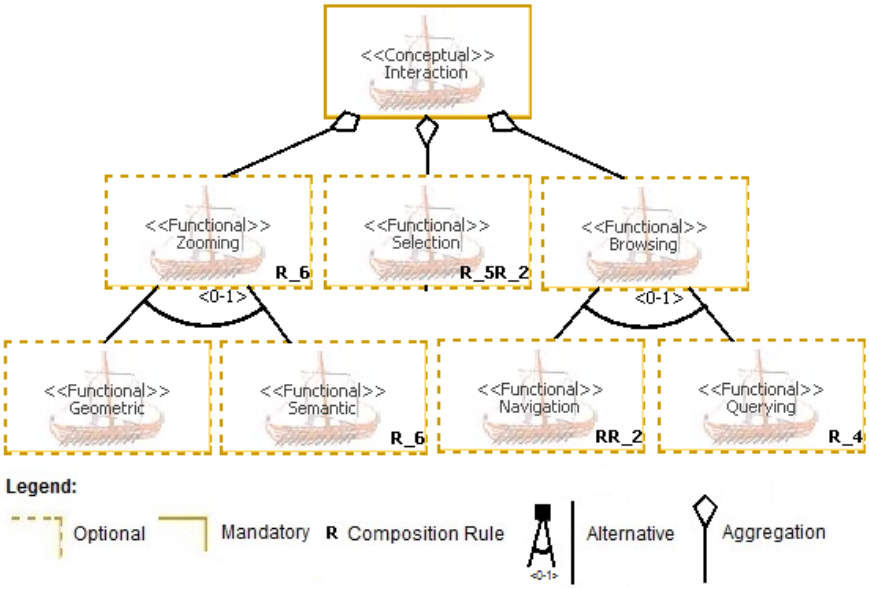


Figure 4.18 – Examples of Interaction features

4.5.2.2 Presentation features

Presentation features map the possible ways of showing multiple visualizations. These features are displayed in Figure 4.19.

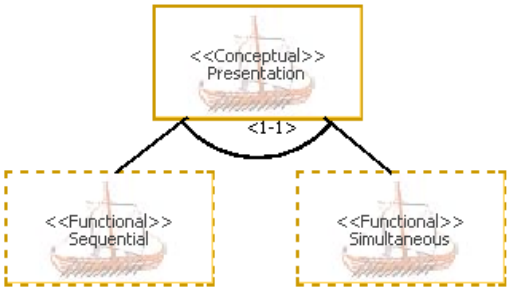


Figure 4.19 – Examples of Presentation features

Typically, the software visualization tool has total control on the application/use of such features on a view, i.e., it is a decision made by the visualization developer instead of a choice by the user. This is an important difference to a usual interaction element.

In terms of the presentation mode, the Sequential feature represents a visualization that displays different views in a sequential order, each at a time. The Simultaneous feature, in turn, allows presenting multiple views at the same time, similarly to a dashboard.

4.5.2.3 Information visualization features

As regards to general exhibition elements, information visualization features map visual methods for changing the views. Figure 4.20 presents some of these features.

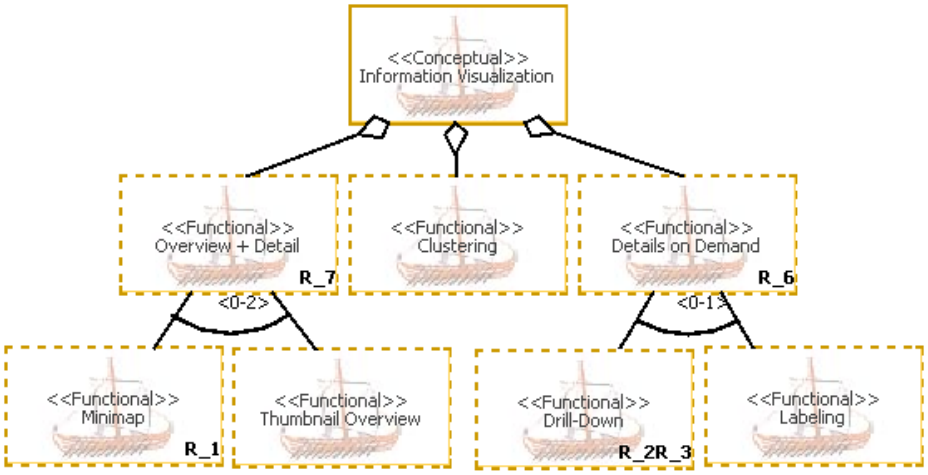


Figure 4.20 – Examples of Information Visualization features

In order to customize the visual attributes of a visualization metaphor, one must choose the information visualization features to apply in the development of the visualization tool. For instance, the Details on Demand feature – with its Drill-Down variant – reveals details according to the user needs, usually following a hierarchical structure [Shi et al. 2005]. The Clustering feature, in turn, aims at splitting a large data set into subgroups based on certain similarity measures to ease the data analysis [Chen 2006].

It is noteworthy that the selection of each feature may impose the selection (or de-selection) of other features. Thus, besides the features and their relationships, composition rules supplement the model with mutual dependency and exclusion relationships [Lee et al. 2002] [Blois et al. 2006].

4.5.2.4 Composition rules

Given the different features in the model mapping to a visualization context, the selection of a single visualization or interaction element may require or exclude the use of another element. Thus, composition rules may apply to the visualization concepts, constraining the selection from optional or alternative features [Lee et al. 2002]. A composition rule between features is specified as follows:

R_X – <Visualization Feature> **requires** <Visualization Feature>

R_Y – <Visualization Feature> **excludes** <Visualization Feature>

As an example of relationship between interaction and information visualization elements, the composition rules R_2 and R_6 present different dependencies between features. R_2 requires the adoption of the Selection feature every time one chooses the Drill-Down or the Navigation feature. This is explained by the fact that a typical user interaction for locating a node in a drill-down method consists of clicking the parent directory (or subtree) in which a node of interest resides [Shi et al. 2005]. Similarly, the Navigation needs a method for selecting options and elements in a view. Another example is rule R_6, which indicates that the selection of the Zooming variant Semantic requires the use of the Details on Demand feature, given that semantic zoom shows new details according to the user demand [Buering et al. 2006], i.e., as the user approaches the visual elements. These rules are described as follows:

R_2 – ((*Drill-Down*) OR (*Navigation*)) **requires** (*Selection*)

R_6 – (*Semantic [Zooming]*) **requires** (*Details on Demand*)

The proposed composition rules compose the feature model and are subject to changes and updates (they are not a complete set). The current visualization features and elements, as well as the explicit relationships between features depicted by the rules, are described in more details in [Schots et al. 2015] with their literature references. Each feature is presented in depth in order to ease the interpretation of the visualization element, with images to illustrate its use whenever possible.

4.5.3 Using the feature model to define Zooming Browser features

The Zooming Browser visualization features were chosen according to (i) the selection of features believed to be important to help users explore the tool, and (ii) the adequacy of such features to the data necessary to answer the established questions. An example of feature selection is presented as follows[55]. The rationale for selecting some of the features is based on the excerpt of the visualization feature model presented in Figure 4.21.

[55] The whole set of selected features for each perspective can be found in the mapping presented in Appendix A.

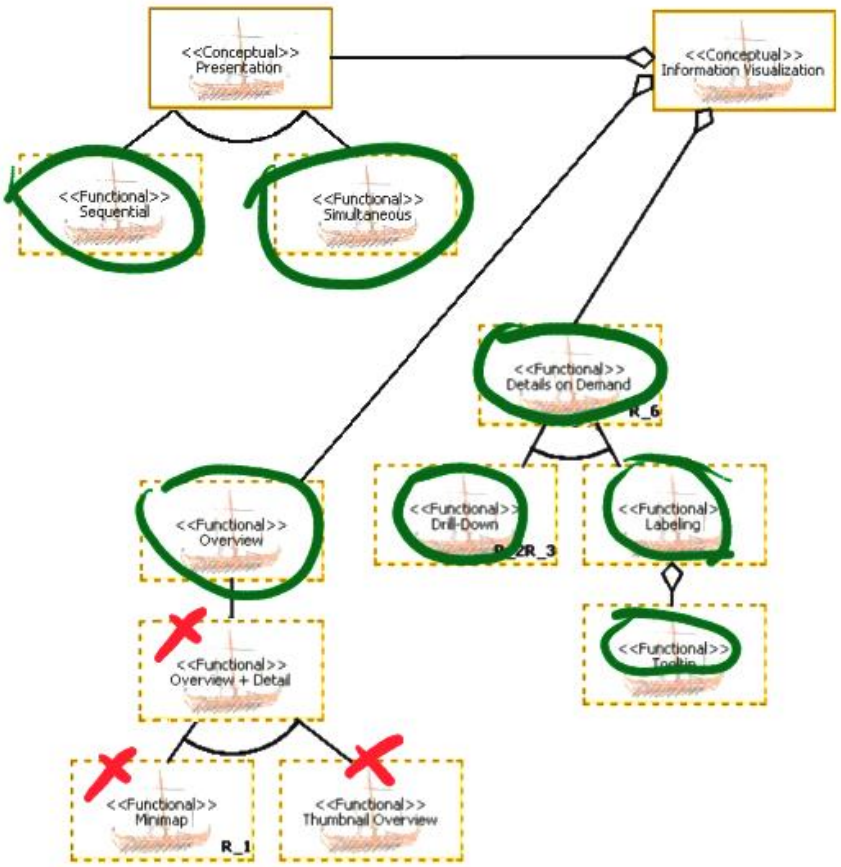


Figure 4.21 – Excerpt of the feature model and the selection of visualization features

Regarding the Presentation feature, there are two possibilities: the views can be portrayed sequentially (i.e., one at a time, separately) or simultaneously (i.e., multiple views are displayed at the same time) [Schots et al. 2015]. The former option is usually employed in dashboard design, which leads to selecting this feature for the Dashboard perspective of Zooming Browser (presented in Section 4.3.2.1). It is also used in the Metadata Exploration perspective (shown in Section 4.3.2.2). The sequential presentation is used for switching between Zooming Browser perspectives and for transitioning between levels of information (both in the Dashboard and the Metadata Exploration perspectives).

With respect to the Information Visualization features, the selection of the Overview feature allows Zooming Browser to provide a general context for understanding the data set, so that users can gain an overview of the entire collection [Shneiderman 1996]. It presents a "picture" of the whole data entity that the information visualization represents [Craft & Cairns 2005] – in Zooming Browser, the data about assets, developers, and projects. Patterns and themes in the data that may be helpful can often be seen only from a viewpoint that comprises the whole view; from this

perspective, major components and their relationships to one another are made evident [Craft & Cairns 2005].

The displaying of details on demand (defined as the design principle DP4), in turn, can be implemented in several ways. The most used feature in Zooming Browser to this end is the drill-down, which reveals details of the data according to the user needs that are made explicit through interactions [Schots et al. 2015]. The Dashboard perspective allows drilling down to the Reuse Map visualization (providing details about which consumers reused which assets in which projects). The Metadata Exploration perspective is fully based on drill down interactions. Some visualizations from the Low-Level Data Representation perspective also employ this feature.

Another variation of details on demand used in Zooming Browser is labeling, to provide an understanding of the context in which the visualized data appear. However, labeling each item cannot be done statically on a dense visualization; in this case, dynamic techniques are advisable, such as interactive tooltips (which are hidden by default and provide access to additional levels of information when interactively requested) [Schots et al. 2015]. In the Dashboard and Metadata Exploration perspective, some items display labels by default, displaying additional information as tooltips when the user interacts through hovering.

After the choice of visualization elements from the feature model, one can implement them in a visualization tool. However, before that, it is important to ensure that the selected elements are suitable for the problem that the visualization tool is expected to solve. This led to the mapping structure presented in the next section.

4.6 Mapping Structure of Goals and Visualizations

During the development of visualization tools, developers[56] recall existing abstractions trying to find out how to (better) depict the available/necessary data based on their characteristics. Furthermore, there is a purpose in mind when creating visualization tools, i.e., there are specific goals to meet. Thus, not only there is the need to represent data using proper abstractions; this must be made in such a way that it can help somebody (audience) to do something (tasks). If this premise is neglected, the tool

[56] In this context, *developers* include roles that take decisions in the *development* of visualization tools (*e.g.*, requirements engineer, designer, programmer etc.), ranging from the tool goals to the visualizations to use.

will be either useless or not fully meet the needs for which it is created, leading its users to resort to other sources of information or additional tools.

As stated previously, it is crucial to ensure, among other aspects, that the visualization tool under development fits the established needs, properly mapping the data required to achieve the goals into corresponding visual attributes. This is not trivial, though, given the abstraction gaps to address and the substantial risk of overlooking important intermediary decisions. Thus, the mapping of goals and visualizations cannot be made instantly; it must instead be decomposed into stages and performed carefully.

This necessity became more evident during the design of Zooming Browser. There was an intention to assure that its visualizations would be actually helpful in answering some reuse-related questions, accomplishing the established goals. To this end, it was necessary to recognize which tasks users should perform to answer these questions, which data would be necessary, and how to map the data into the vast visualization space. While planning its development, some visual metaphors came to mind, but there was no certainty that they were appropriate and whether they would effectively help achieving the established goals. This led to the creation of a mapping structure to guide this process.

The following subsections present the proposed staged set of activities for mapping *user* (or *organization*) *goals* to the *visualizations* that can help achieving such goals [Schots & Werner 2015] (meeting VF5). The purpose of this mapping is not to enforce a set of guidelines on how to perform each stage, nor to point out what would be the best visualization for some goal/task/data. Instead, this mapping structure aims at guiding and encouraging developers to perform a more focused and cautious decision-making process (not tied to any particular methodology) on each mapping stage, towards an anticipated assessment of the usefulness of their visualization tools before evaluating them with the intended audience.

4.6.1 The mapping structure and its application

The structure for mapping goals and visualizations is presented in Figure 4.22. All the relationships are many-to-many, except between data and visual attributes, which should be one-to-one to avoid ambiguity, causing user confusion. This mapping is purposely “open” so that each stage can be accomplished by stakeholders in the most convenient way to them.

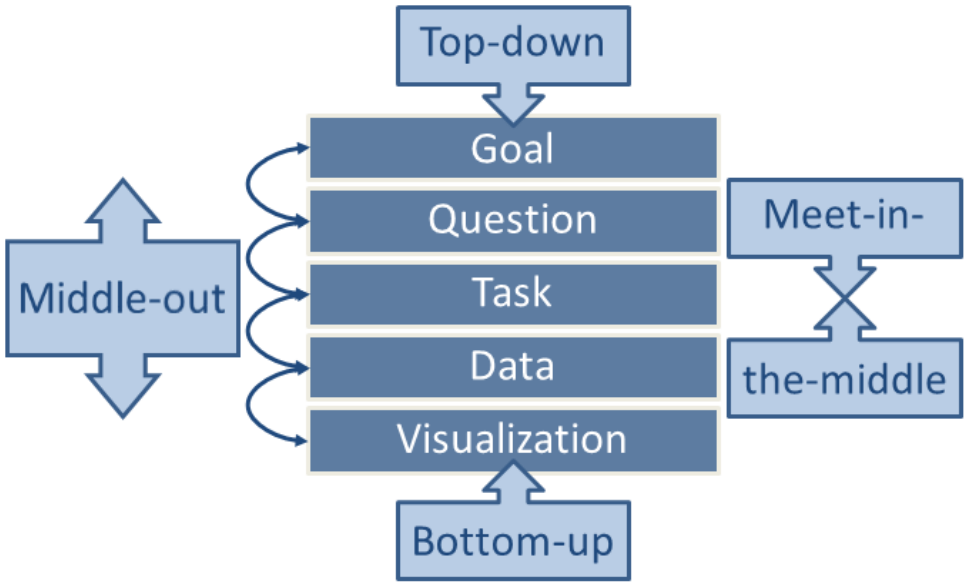


Figure 4.22 – Mapping structure between goals and visualizations [Schots & Werner 2015]

During the execution of the mapping process in the context of Zooming Browser, it was noticed that different strategies could take place: *top-down* (when goals are already set and clear), *bottom-up* (when one wants to find the utility of a set of visualizations), *middle-out* (*i.e.*, starting from an intermediary stage towards achievable goals and assisting visualizations), or *meet-in-the-middle* (*i.e.*, when top-down and bottom-up "join" at some point in their executions). The latter applies when there are goals and visualizations in mind, but some intermediary aspects of the mapping are not clear and require further reflection and analysis. This was the case of Zooming Browser.

The next subsections present an excerpt of the mapping performed in the context of Zooming Browser, the driver of this work. The detailed mapping is presented in Appendix A. For didactic purposes, the mapping is described in terms of the top-down strategy. The whole mapping will be made available in a website for better exploration.

4.6.1.1 Mapping *goals* and *questions*

The mapping between goals and questions has been thoroughly explored in software engineering [Basili et al. 1994]. It consists of associating questions whose answers help achieving a goal. A question may support more than one goal, and a goal usually consists of more than one question. In this stage, the GQM format is not mandatory, but is advisable.

Some asset-centric questions are used for illustration purposes. They are derived from PG03, one of the Zooming Browser project-related goals (*Decide whether an*

existing project that already contains a given asset version should upgrade/downgrade to a newer/older asset version) (presented in [Schots 2014b] and listed in Appendix A). They are listed in Table 4.3.

Table 4.3 – Mapping between goals and questions

Goal ID	Question ID	Question
PG03	Qa	How often is this asset [version] reused over time?
	Qb	Which consumers reused this asset [version]?
	Qc	In which projects was this asset [version] reused?
	Qd	Which projects contain this asset [version] at some point of the development life cycle but do not contain such asset [version] afterwards?
	Qe	Which projects contain, among their releases, [a version of] this asset?
	Qf	Which [versions of] assets does this asset [version] depend on?
	Qg	Among the reported bugs related to this asset [version], are most of them fixed or open?
	Qh	How often do producers of this asset fix reported bugs?
	Qi	How long does it take for producers of this asset to fix reported bugs?
	Qj	How often do producers of this asset implement improvement suggestions or feature requests?

It is advised to keep record of the rationale that relates each question to the goals, since this helps executing the next stage. For instance, *Qc*-*Qe* point to reuse attempts of a given asset version (which can be successful or not), while *Qf*-*Qj* provide awareness on the commitment of the asset development team regarding problems identified and features requested, among others.

4.6.1.2 Mapping *questions* and *tasks*

One could expect a mapping between questions and metrics, as defined in the GQM approach [Basili et al. 1994]. There is no doubt that metrics can be useful for answering questions, but their interpretation is more intuitive when they are tied to visual representations [Lanza & Marinescu 2006]. Besides, when it comes to interactive visualization tools, it seems more natural to map questions to project or organizational tasks[57] that must be performed to obtain the answers being sought. It is noteworthy that many questions or tasks are based on literature reports, but it is imperative to assess

[57] Interaction tasks with the visualizations (such as filtering, browsing, drill-down etc.) are handled separately in an upcoming stage. Developers of visualization tools may resort to the visualization feature model to this end.

their relevance to the current state-of-the-practice [Novais et al. 2014]. In the Zooming Browser design, the association between tasks and questions is presented in Table 4.4.

Table 4.4 – Mapping between questions and tasks

Task ID	Task	Related Questions
Ta	Check [the successfulness of] reuse attempts of [a given version of] an asset in existing projects	Qa, Qc, Qd, Qe
Tb	Identify experts (producers/ contributors and consumers) on a reusable asset [for communication needs]	Qb
Tc	Understand/Evaluate asset dependencies	Qf
Td	Check if producers have been keeping up with the development of a reused asset (community participation)	Qg, Qh, Qi, Qj

4.6.1.3 Mapping *tasks* and *data*

In order to perform software development tasks, it is necessary to resort to data, usually available from different sources. Thus, at this point, one should find out what data are required to support executing such tasks (for the proper identification of relevant data sources and the filtering of unnecessary data). Some processing (data cleaning, integration, aggregation etc.) is usually necessary. Other data may become necessary for complementing the visualization (e.g., due to positioning and organization of data), so it is likely that this stage is revisited afterwards.

Some data for performing the tasks defined for Zooming Browser are listed in Table 4.5. They are extracted from reuse repositories, version control repositories, and issue tracker/task manager systems (as described in Section 4.4). Links to original sources or other representations (e.g., HTML websites) are also stored, allowing to drill-down to additional information.

Table 4.5 – Mapping between tasks and data

Source of information	Data	Related Tasks
Project VCS history (both from assets' projects and other projects in which assets were reused)	Project name	Ta, Tb
	Commit author	Tb, Td
	Commit date	Tb, Td
	Added assets*	Ta
	Removed assets*	Ta
Reuse repository	Asset name	Ta, Tc
	Asset versions	Ta, Tc
	Asset dependencies**	Tc

Source of information	Data	Related Tasks
Issue tracker/Task manager repository	Issue status	Td
	Issue type	Td
	Issue creation date	Tb, Td
	Issue close date	Tb, Td
	Issue assignee	Tb, Td
	Issue resolver	Tb, Td

* Data filtering is applied in order to retrieve only commits related to assets.
** There are different kinds of software dependencies; the current scope is limited to dependencies made explicit (e.g., described in a build configuration file).

4.6.1.4 Mapping *data* and *visualizations*

This is one of the most important parts of the mapping, because an inappropriate or ambiguous mapping may impair the effectiveness of the visualization tool as a whole, as stated previously. The results of one of the studies conducted in the scope of this thesis (presented in Section 3.3.4) showed that the mapping between data and visualizations is barely described in publications, so users have to "guess" it, which can be risky and lead to wrong interpretations of data [Schots et al. 2014]. Because this is a more complex and most crucial stage, it can be divided into three different steps, described as follows.

- Firstly, one or more visualizations must be already in mind based on the established data and their characteristics; thus, there must be a *pre-selection of candidate visualizations* that may be confirmed later based on the subsequent steps.
- Secondly, since the visual attributes are responsible for linking data to visualizations, one must *decompose the visual attributes that constitute the visualizations* (such as size, color, position, shape etc.) in order to recognize the data type required by such visual attributes. For instance, different colors enable the representation of categorical data, while color scales require the data type to be continuous.
- Finally, *mapping each datum to each visual attribute* involves analyzing the available data types to attest their suitability to the visual attributes that compose the visual metaphor.

These steps (especially the first two) may require support from skilled visualization experts. After that, it is possible to ensure that the visual attributes are both

necessary and sufficient to the data[58]. Thus, one can be more confident to determine which visualization(s) will be actually used. However, if a data-to-visualization mapping cannot be fully performed, it can be due to three causes: (i) the available data cannot be mapped to the intended visualization due to incompatibilities (restrictions on data or on visual attributes); (ii) the intended visualization is not sufficient to comprise the necessary data; or (iii) the visualization requires more data than the available ones.

A solution for all these cases can be the choice of different visualizations. An alternative solution for (i) and (ii) is to combine another visualization with the existing one(s) (for instance, through interaction resources or by creating a multi-perspective environment [Carneiro et al. 2010]). In this case, it is important to keep in mind that, ideally, a visual attribute should keep its semantics in a consistent way among different visualizations, in order to avoid misleading interpretations. Finally, for (iii), one may need to collect more data to make the most of a visual metaphor and its resources.

For illustration purposes, an excerpt of the design of the *issues* visualization (from Zooming Browser's *asset-centric* view based on the Metadata Exploration perspective) is depicted in Figure 4.23, while Table 4.6 shows how some of the aforementioned data were mapped to visual attributes of this visualization. It is noteworthy that many considerations may be taken into account, e.g., the color and layout schemes, legibility, cultural and aesthetic aspects, among others.

Although it is not explicit as a separate stage in the mapping, this stage also requires the choice of interaction resources to employ, aiming to allow users of the visualization tools to explore the data and perform their tasks[59]. In Figure 4.23, for instance, one can filter issues by type through a filtering mechanism.

These steps should ideally map all the elements (goals, questions, tasks, data, visual attributes, and visualizations) involved in the design of the visualization tool. As stated previously, the detailed mapping performed in the context of Zooming Browser is presented in Appendix A. The use of this mapping provided more confidence for implementing Zooming Browser, since it enabled to check whether the tool could help answering the established questions before evaluating it with its intended stakeholders.

---

[58] Note that this does not discard the need for performing evaluations (such as usability studies) with the audience of the visualization tools.

[59] This can be done with the support of the visualization feature model.

Figure 4.23 – Initial visualization design for representing the issues of an asset

Table 4.6 – Mapping between data and visualizations

Visualization	Visual Attribute	Data	Value	Description
Adapted bubble layout	Geometric shape	Issue	Circle/Bubble	A circle represents an issue.
	Size	Issue lifetime	For closed issues (issue status = closed): (issue close date – issue creation date)	The size of an issue is proportional to the time it remains open, i.e., the longer the time an issue has been opened (and not closed) the greater it appears.
			For open issues (issue status = open): (system current date – issue creation date)	
	Color	Issue status	Green, for closed issues (issue status = closed)	The use of colors helps finding out the status of the issues. Colors provide an overview of how developers are handling issues as they appear.
			Red, for open bug issues (issue status = open AND issue type = bug)	
			Yellow, for other open issues (issue status = open AND (issue type = feature request OR issue type = improvement suggestion))	
	Icon	Issue type	Exclamation mark (issue type = bug)	Icons facilitate differing issue types. Since bugs are usually more severe or relevant than feature requests, many open bugs may indicate lack of support for the asset's users.
			Lamp bulb (issue type = feature request OR issue type = improvement suggestion)	

4.7 Illustrative Scenarios

In order to demonstrate how APPRAiSER tools can support performing reuse tasks and aiming to make evident the utility of metadata and project history information (highlighted in italics) and its sources (underlined), two reuse-related goals (*decision-making with respect to asset reuse* and *maintenance of a reuse repository*) are presented in the next subsections, contrasting their achievement with and without APPRAiSER.

4.7.1 Making informed reuse decisions in a project

In order to decide whether an asset or an asset version can/should be reused in or incorporated to a project (PG01 in the APPRAiSER mapping, as shown in Appendix A), the developer, as a potential consumer, might consider the information and sources listed as follows (assuming that the asset has already been pre-selected).

By analyzing the reuse repository, the developer can obtain a set of information about an asset, such as its *status*, its *release date*, its *released versions*, its *license*, its explicit *dependencies* and *exclusions*, among others. The latter ones particularly help saving time by assessing the asset's compatibility with the project under development/maintenance beforehand. However, the effort of extracting these data from a reuse repository may lead the developer to ignore this source of information, either making a risky reuse decision or ending up building the asset from scratch.

APPRAiSER obtains this information automatically when the developer fills out the Zooming Browser form (accessible through the menu), informing the asset about which he/she wants additional information (as shown in Figure 4.15). Repository Miner extracts the data, which are visually presented in the Zooming Browser perspectives, particularly in the Metadata Exploration. Through this perspective, the developer can select the asset and obtain all the aforementioned information from the reuse repository, in addition to other information discussed as follows.

The asset's version control system provides information about *consumers* and *producers*, and allows verifying, through its *commit history data*, whether the asset development project remains active and its development community is participative. It is also relevant to know who are the *producers* (and the *organization*, if any) who develop the asset, since reputation in development communities plays an important role in trusting what has been developed. If the potential consumer realizes that the project did not receive any *changes* for a reasonable period, chances are that he/she may

struggle in obtaining any support or bug fixes. However, such awareness is not easy to achieve with traditional VCS tools. Some websites (such as GitHub) can provide some clues in this regard, but part of this set of information may require several interactions throughout the options provided by the website, which may cause user disorientation.

Through APPRAiSER, the Repository Miner automatically collects the assets' VCS history, and the developer can obtain information regarding the development of the asset through the Zooming Browser's History perspective. In addition, the developer can understand the project's branching strategy (e.g., to observe if new features are developed in a separate branch or how long it takes for branched bug fixes to be merged back to the trunk/main development branch). By using the filtering resource, it is also possible to observe which developers contributed to which parts of the project, and how participative the development community of the asset is.

One of the most important information when reusing an asset is to check for previous *reuse occurrences*, i.e., whether it (or its version of interest) was reused before, in which projects, and by which consumers. Previous reuse occurrences provide examples on how to reuse the asset. Besides, they assure whether there are successful cases of reuse (e.g., if the asset version was incorporated to a *project release*). This information, among others, can be obtained through the projects' version control systems, both from open source repositories and organizational repositories. However, the developer may face the same problems listed for the asset's VCS if he/she does not have appropriate tool support. Besides, the process of identifying and matching asset versions becomes cumbersome if done manually.

APPRAiSER automatically identifies reusable assets from imported projects, with the support of the Repository Miner's identification strategy (discussed in Section 4.4.2). As discussed in the same section, APPRAiSER requires the project manager or a project developer to indicate or confirm the version of this asset, being a semiautomatic process.

A potential asset consumer is usually concerned with the *issues history* from the issue tracker/task manager of the asset, to find out if *bugs* are fixed (and how long it takes for that) and how often producers make *improvements* and implement *feature requests*. This information is derived from the issues' *status*, *creation* and *close date*, *assignee* and *developers* involved in it. These items are also relevant for deciding if a project that contains a version of the asset should be upgraded/downgraded to a newer/older version of such asset. The developer should then navigate through the issue

tracking system in order to obtain this information (which is usually presented textually).

After the Repository Miner has extracted the issues information with the Issue Tracker Miner, the Zooming Browser's Metadata Perspective provides information about issues at a glance, through visual attributes that build an overview of them (based on the design presented in Figure 4.23). This allows for a faster understanding of the asset project status and for a better exploration of details of the issues. For instance, the developer may be interested in understanding details on an issue that has been open for months and has a high severity for the asset's project, and drill-down to it in order to assess whether and how it would affect the asset reuse.

An important concern when reusing an asset is its *license(s) of use*, since this may constrain or establish some conditions under which it can be reused. There are cases in which private projects have to make their source available[60], due to a requirement of the license under which the asset reused by them was released. This information is not always emphasized enough in order to draw the developer's attention (oftentimes it is at the end of a large text description, which may be unnoticed by the developer). To this end, Zooming Browser's Metadata Exploration perspective has a "Profile" option for each of its core elements (assets, developers, and projects) so that general information such as this one can be found easily in a single place.

With this set of information at hand presented intuitively in a centralized place, it may become easier for potential consumers to make decisions about reusing/upgrading an asset, among others.

4.7.2 Maintaining organizational reuse repositories

Creating and maintaining a reuse repository in an organization are not easy tasks [Ye & Fischer 2002]. They require reuse managers to include, exclude, request maintenance, or discontinue/deprecate asset versions, besides keeping metadata information for communication purposes. In other words, reuse managers must be aware of what is going on in the reuse scenario in order to support the organization in properly conducting their projects and keep up with reuse initiatives.

---

[60] TechTudo, June 2014. "ZapZap has its source code released after controversy about legality", available (in Portuguese) at http://www.techtudo.com.br/noticias/noticia/2014/06/zapzap-tem-codigo-fonte-liberado-apos-polemica-sobre-legalidade-entenda.html.

For communication matters, reuse managers must first track *reuse occurrences* (which consumers reuse assets in which projects). Ideally, this information should be based on the projects' version control systems. However, as shown in Section 2.5.3, this is not the reality in many organizations. Besides, as stated in Section 4.7.1, identifying and matching asset versions collected from this source is not trivial.

APPRAiSER's Dashboard perspective allows drilling-down to the reuse map, which presents this information visually in order to ease reuse management in the organization. It also provides reuse awareness for producers (in terms of the [attempts of] reuse of their assets) and consumers (helping them to recall which assets they already reused, since they may lose sight of it [Ko et al. 2007]).

Reuse managers must also have *contact information* of producers and consumers (as well as appropriate mechanisms) to notify them about asset status changes, problems detected, modifications carried out, new versions available, and discontinued assets. This is usually done by e-mails written manually in many organizations, which can lead to mismatches.

Based on the contact information present in the developer profile (also available in the Metadata Exploration perspective), the APPRAiSER server can notify all the consumers and producers about these events. Besides, Zooming Browser also shows the events related to each core element.

The issue tracker/task manager can help reuse managers to check whether there are *bugs* whose *severity* may require asset maintenance. Additional information pointed out in the previous scenario, such as the time it takes for fixing bugs or the period an asset is not reused, may lead to asset discontinuation from the organizational repository (according to organizational criteria). The problems discussed in Section 4.7.1 also apply to this scenario, as well as the solution provided by APPRAiSER. Moreover, the information related to activeness and participation of the development community (obtained from the asset's version control system) can be used for decision-making regarding an asset's discontinuation.

Besides providing evidence on the reuse of an asset, showing that it has been actually included at some point of the project development, the asset's version control system and the issue tracker/task manager systems together can show social information involving developers (in terms of *collaboration* and *communication* in asset development) and the *status* of *bugs* that affected the asset. However, these sources of

information are usually decentralized, making it hard to analyze their information in an aggregated way.

Through Zooming Browser, APPRAiSER allows the navigation to the asset issues (as described in Section 4.7.1) and establishes different kinds of correlations between developers: which producers collaborate with which producers, which consumers reuse assets developed by which producers, and so on. This can also be helpful for project management in terms of team allocation, especially when it comes to the development of reusable assets for a specific need and, in an increased maturity level, development for reuse.

The proper integration of these sources of information and the presentation of their data in an intuitive way can help reuse managers to gather some analytics of the reuse scenario. Examples comprise how often an asset is reused, how often a consumer reuses assets, and how often a producer develops assets, among other interesting findings. This can help evaluating the effectiveness of reuse practices (progresses and efforts) in the context of local projects/assets/developers (i.e., belonging to the organization), as well as stimulate in-house reuse.

4.8 Related Work

This section presents some works related to each of the elements presented in the previous sections.

4.8.1 Zooming Browser

The results from the *quasi*-systematic review pointed out the lack of works that aim to support performing reuse tasks through visualization resources, especially using metadata information. There are a number of related works providing visualizations for the structural or evolutionary aspects of different artifacts in repositories. The scope of analysis in this section is limited to the ones whose goals are somehow similar to Zooming Browser's goals, particularly the ones that visually represent or explore contents of reuse repositories for supporting decision making about reuse.

The only work not related to software development is ALOCOM [Klerkx et al. 2006], which aims to visualize a large repository of learning objects in the form of small reusable content components. The disaggregation of legacy content creates such components, and some metadata are added to each of them. The visualization gives an

overview of the components in the repository, including how they are put together, in terms of "is part of" and "has part" relations.

Figure 4.24 shows a screenshot of the ALOCOM visualization. The user interface consists of a right panel (that visualizes all the components in the repository), a left top part (with options to filter out components that are not of interest), and a left bottom part (which displays textual metadata on the components).

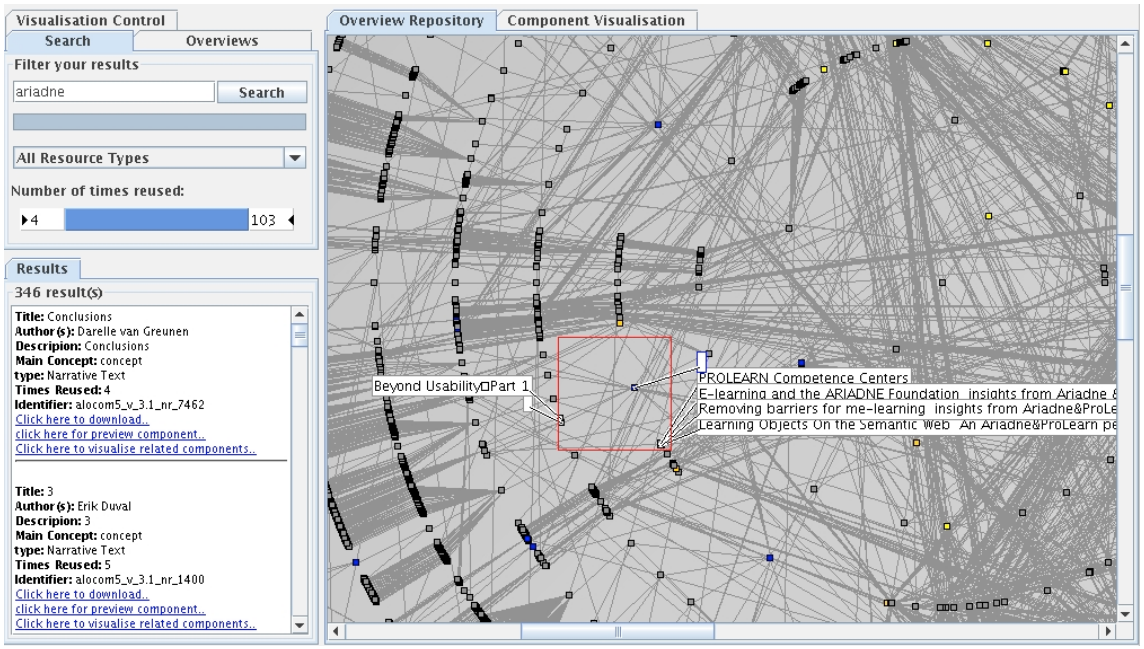


Figure 4.24 – The ALOCOM repository visualization [Klerkx et al. 2006]

The concept of component is very small-grained: examples include images, definitions, slides, and text fragments. Thus, the semantics of what can be characterized as a reusable asset is very wide. Besides, other visualization/interaction resources could have been employed to reduce the amount of information displayed textually.

The works by [Kula et al. 2014] and [Yano et al. 2015] (which seem to be collaborative, since they involve some authors in common) discuss that, as libraries on which a system depends evolve (with bug fixes and new features), the system maintainer needs to decide if, when and what to update [Yano et al. 2015]. In [Kula et al. 2014], the authors state that novice maintainers may lack the historical knowledge required to manage an inherited system efficiently.

The work presented in [Kula et al. 2014] proposes visualizations of the co-evolution of a system and its library dependencies, in order to ease the understanding of this phenomenon and help deciding to upgrade systems to a newer version of an outdated library [Kula et al. 2014]. Before introducing the visualizations used, the

authors present the concepts related to the systems' adoption of libraries, depicted in Figure 4.25.

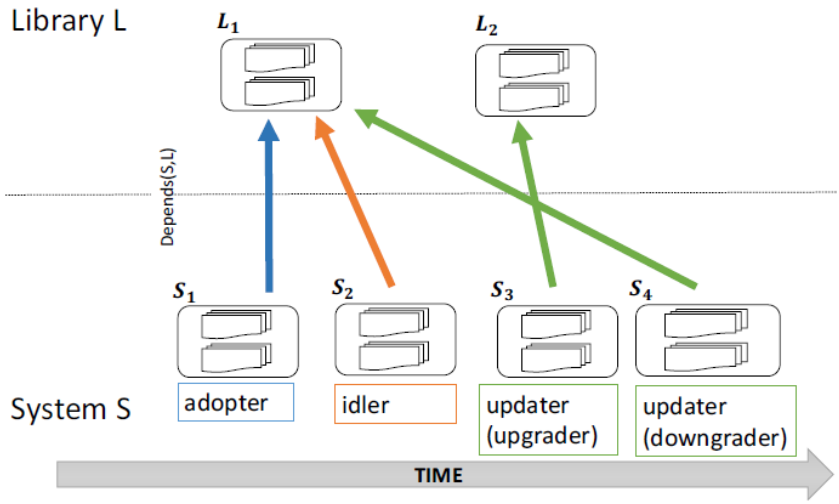


Figure 4.25 – Dependency relations between system versions and library versions [Kula et al. 2014]

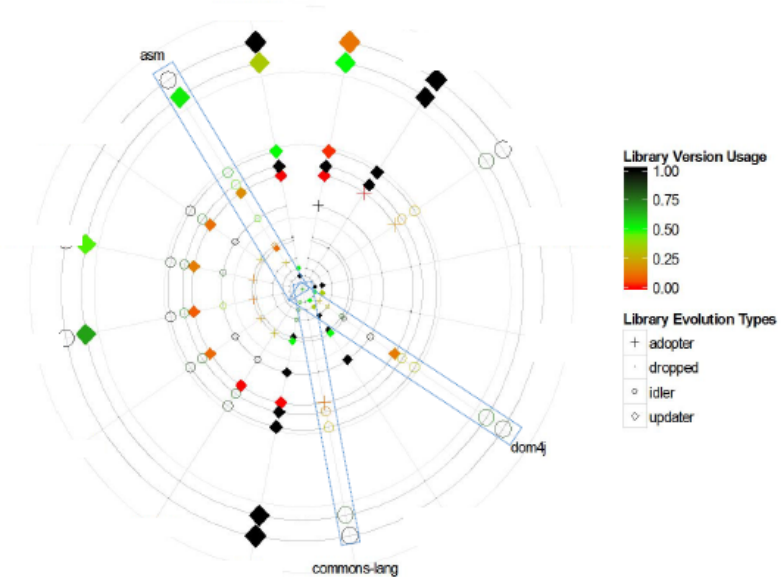
The relations presented in this figure are explained as follows [Kula et al. 2014]:

- an *adopter* system version is a version that starts using a library for the first time (i.e., it has not used previous versions of it;
- an *idler* is a system version that depends upon the same library version as its immediate predecessor;
- an *updater* is a system version of which the previous version depended upon a different library version (so an updater is either an *upgrader* or a *downgrader*);
- finally, a *dropper* (not depicted in the figure) is a system version of which the current version ceases the dependency relationship. A *dropper* can revert to an *adopter* of a different library version or resume being an *idler* of a previous version.

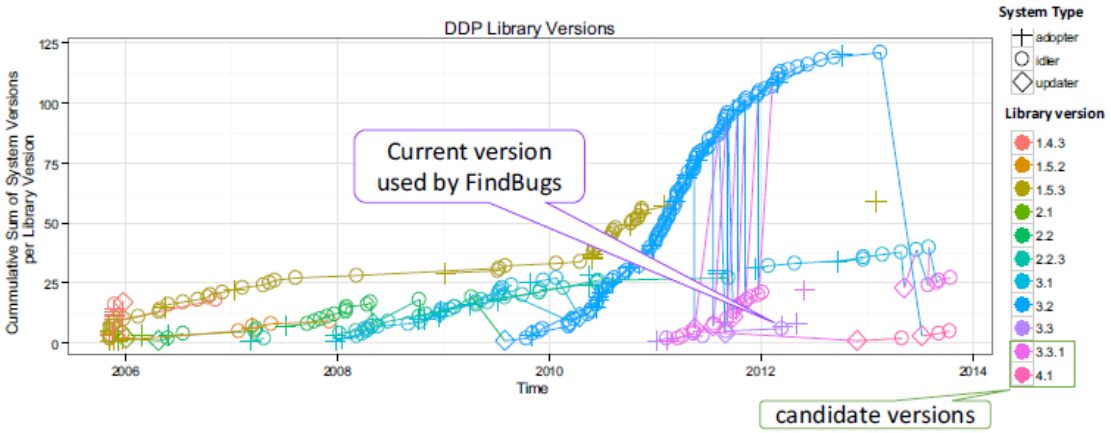
This work aims at visualizing how the dependency relation between a system and its dependencies evolves from two perspectives: the system-centric dependency plots (SDP) and the library-centric dependents diffusion plot (LDP) [Kula et al. 2014]. While the former shows successive library versions on which a system depends over time, the latter shows the diffusion of users (systems) across the different versions of a library [Kula et al. 2014].

Figure 4.26 presents the SDP visualization of the FindBugs system – part (a) – and the transition to the LDP visualization of the ASM library – part (b) – after selecting its axis, highlighted in the upper part of (a). In (a), starting from the center, each ring represents a system release, and the relative distance between rings indicates the time between releases (weeks). In (b), the time-series displays the popularity of library

versions at any point in time – the x-axis indicates the time, while the y-axis presents the aggregation of system versions that reused a given library version depicted along the curve [Kula et al. 2014].



(a) SDP for FINDBUGS system



(b) LDP for ASM library

Figure 4.26 – SDP visualization presenting an overview of the evolution of the dependencies of a system as it evolves [Kula et al. 2014]

The authors state that they believe the visualizations are scalable, and they envision the implementation of filtering mechanisms to help managing the data [Kula et al. 2014].

Apparently as an evolution of [Kula et al. 2014], Yano et al.'s work mines data from similar systems to obtain "wisdom of the crowd" [Yano et al. 2015] (although the authors do not make clear what accounts for similarity). The mined sources are GitHub and Maven Central (also used in APPRAiSER by the Repository Miner, described in Section 4.4).

A visualization tool called VerXCombo was developed aiming to allow interaction with the mined data in order to find the "best-fit" combination of libraries, determined by the popularity of use and the latest version release [Yano et al. 2015]. Figure 4.27 shows a screenshot of the tool.

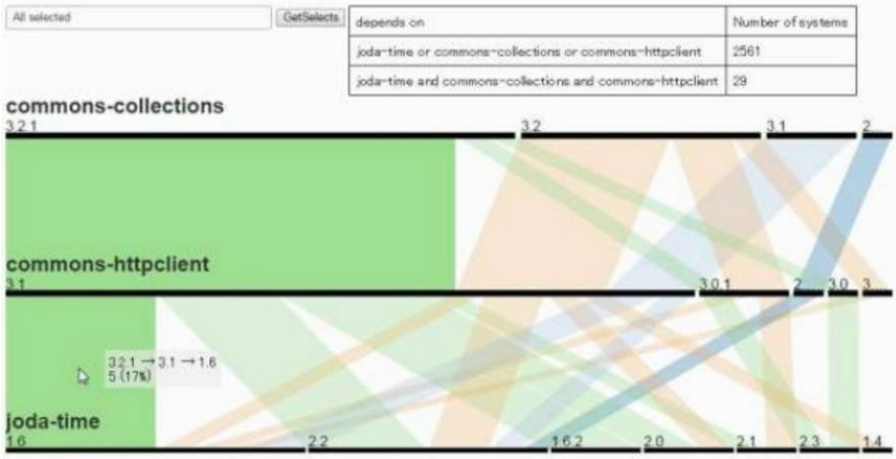


Figure 4.27 – The VerXCombo tool (source: http://www.slideshare.net/augai9/verxcombo-an-interactive-data-visualization-of-popular-library-version-combinations)

After candidate libraries are chosen from a drop-down list, they are presented in a parallel sets visualization. Different combinations can be highlighted through mouse interactions. The link thickness between library versions indicates their frequency of usage (extracted from similar projects). Users can sort libraries by version or popularity of use.

The major limitations of these approaches are:

- The "wisdom of the crowd" knowledge is restricted to a number (summarized in the "best-fit" calculation), which limits the user exploration of data – in other words, there is no additional information to the user;
- The tools do not allow drilling-down to understand the context in which the libraries have been included or used in the similar systems; thus, users are limited to the contents of these views, hampering to obtain additional information (such as the "relevance" of the projects that reused the library versions);
- As opposed to the Zooming Browser, the tools do not take into account developers' information (also available in the mined sources), which could increase confidence in decision-making (since developers' influence also accounts for reuse confidence).
- None of the tools present an experiment for an evaluation of their use.

4.8.2 Repository Miner

Several works have investigated sources of information and information needs in software development, each under a particular perspective. Some examples are listed as follows.

Ko et al. performed a field study of software developers to understand their information needs [Ko et al. 2007]. They observed groups across the corporation focusing on (i) what information software developers seek, (ii) where they seek this information, and (iii) what prevents them from finding such information.

Regarding *writing code* tasks, developers had questions related to data structures or functions. To answer them, they searched documentation and inspected other code for examples (which can be thought of as a search through the space of existing reusable code) [Ko et al. 2007]. Once they had a candidate, they sought its syntactic usage rules (e.g., which method should be called, what data structures are required etc.). Documentation was used when available, but sometimes they needed to use code whose author was the only person who could fully understand it [Ko et al. 2007].

Regarding *maintaining awareness* tasks, developers worked to keep track of hardware, people, and information needed for their tasks. Some awareness information was "pushed" to them through clients, alert tools, and check-in emails; they also obtained other types of awareness by actively seeking it. Groups had meetings to keep aware of problems on which teammates were working and issues on which they were blocked. Because developers were often interrupted, they also sought awareness about their own work. The most common needs were coworker awareness, finding out what code caused a given program state, and how resources they depend on have changed [Ko et al. 2007].

Treude & Storey (2010) aim at achieving awareness of projects, developers, and tasks using dashboards and feeds, with a focus on project development tasks. The source of data used by the authors is the status of the project, which arises from an aggregation of data on open and closed development tasks, successful and failed builds, delivered and pending changes, and successful and failed tests as well as evolutionary information. The authors state that there is a lack of understanding of how to achieve high-level awareness (of project management issues) with low-level awareness (of more fine-grained activities, such as source code changes and development task creation). They also claim for better visualization support as an enhancement for dashboards.

Panichella et al. analyzed written communication between developers recorded through mailing lists, issue trackers, IRC chat logs, and code co-changes [Panichella et al. 2014]. The primary goal was to obtain evidence that a single communication channel offers a misleading portrait of developers' interaction, and that different combinations of sources may provide different views of it.

According to their findings, not all developers use all communication sources, and the use of different communication channels in studies and tasks (e.g., identifying key project roles such as developers with a high communication degree or mentors) can lead to different results. Another finding is that the overlap of communication links between various sources is relatively low (generally below 30%-40%) and varies depending on the project. Therefore, one should merge data from multiple channels to have a better view of developers' interactions [Panichella et al. 2014].

These works demonstrate the importance and usefulness of analyzing how different sources of information can support different aspects of software development: Ko et al. focus on general information from collocated development teams, Treude and Storey aim at project development tasks, and Panichella et al. correlate collaboration aspects and code changes. In the scope of software reuse, the most related works identified are the ones proposed by [Ye & Fischer 2002], [Holmes & Walker 2012], and [Kula et al. 2014].

Ye and Fischer aim at reducing the difficulty of locating components from a large reuse repository [Ye & Fischer 2002]. To this end, they propose a tool (called CodeBroker) that uses doc comments and signatures to extract queries from partially written programs and retrieve matching components. For filtering results, CodeBroker uses as information the knowledge about components, taking into account components known to individual software developers. It also relies on previous interactions with the system, excluding components that developers have explicitly indicated that are of no interest in the current development session [Ye & Fischer 2002].

Holmes and Walker state that, to investigate a pragmatic reuse task, a developer must navigate through, and reason about, source code dependencies in order to identify program elements that are relevant to the task and to decide how those elements should be reused. The tool proposed by the authors uses a reuse model that aims at investigating low-level details of an originating system to transform the selected source code from such system and integrate it into the developer's system in a semiautomatic way [Holmes & Walker 2012].

Kula et al. present visualizations of the co-evolution of a system and its library dependencies, in order to ease the understanding of this phenomenon and help deciding to upgrade systems to a newer version of an outdated library [Kula et al. 2014]. They focus solely on releases of project and libraries, not taking into account fine-grained elements, e.g., commits from VCSs, and other sources of history information. An apparent evolution of this work [Yano et al. 2015] encompasses information from GitHub repositories, but it seems to focus solely on releases.

These works do not thoroughly explore available sources of information regarding software reuse (e.g., production and consumption information). Additionally, in contrast to [Holmes & Walker 2012], APPRAiSER does not directly aim at helping to identify if a given project is reusable or not. The Repository Miner goes a step beyond, providing other sorts of metadata (especially high-level ones) that are important for decision-making about reuse in different levels. No other work discussing or exploring different kinds of information was found, specifically aiming at supporting a broad range of software reuse tasks.

Regarding the state-of-the-practice, there are several web-based version control repositories (such as SourceForge, BitBucket, GitHub etc.) and release repositories (e.g., Maven Central, Node Package Manager – NPM –, etc.) available. These version control repositories provide a wealth of information and resources that help software developers in keeping up with development tasks (many of them also include an integrated issue tracker). However, although they store a huge amount of open source assets, they do not provide quantitative or qualitative information that helps developers decide whether or not to reuse an asset. In GitHub, for instance, it is up to the repository owner to incorporate a generated label showing test results in the repository homepage.

Release repositories, in turn, do not aim at supporting reuse management (i.e., their focus is on storing software releases and related information), requiring consumers to check periodically the status of the reused assets – which does not seem to be a wide-ranging practice, given the effort involved. Besides, some of these repositories do not provide consumption information of a given release[61].

[61] MvnRepositoy (http://mvnrepository.com/) collects information from the Maven Central Repository to indicate which assets listed on Maven Central depend on which assets (probably based on the declared dependencies in the POM file). The website does not provide an API for information exchanging.

4.8.3 Visualization Feature Model

Some studies in the literature propose means to organize a body of knowledge in terms of techniques and algorithms, recognizing the importance and utility of information visualization. Although none of these works are feature model proposals, they seek to comprise a set of techniques related to visualizations. Some of these works are listed as follows.

The Data State Model (DSM) approach [Chi 2000] shows the similarities in terms of operation steps that can be reused. Its goal is to support programmers in understanding the broad application possibilities of visualization techniques. The presented taxonomy groups the techniques into several data domains, facilitating the analysis. The included techniques were chosen according to their relevance to information visualization systems.

Another approach presents a high-level visualization taxonomy [Tory & Moller 2004], classifying algorithms instead of data. The taxonomy is considered flexible because it is based on assumptions that algorithms make about the data to visualize. These assumptions are categorized based on (i) whether they are continuous or discrete, and (ii) according to how much they constrain display attributes. This taxonomy helps organizing the visualization literature to address future research.

The InfoVis Wiki project [INFOVIS WIKI 2015] is a website that provides general information about information visualization elements. It also provides other kinds of information, such as general news about the visualization field. The website does not present how visualization elements relate to each other, and does not show constraints on their use. To be fair, since the Wiki philosophy usually does not enforce filling out predefined sections, this is not required or suggested when someone includes an information).

There are also other works aiming to organize information about visualization elements, such as Treevis.net [Schulz 2011] and SurVis [Beck et al. 2016][62]. These works are restricted to a specific design space (e.g., dynamic graph visualizations), and are not intended to provide a "big picture" of general visualization features. Besides, they have a descriptive nature (presenting what has been done in a given design space) instead of a prescriptive one (showing how elements can or cannot be used/combined).

[62] Available at https://github.com/fabian-beck/survis. An instance of SurVis was used for organizing the findings of the secondary study presented in Section 3.3.3 – available at [Schots 2014c]

These related works propose different taxonomies for the visualization domain. Despite presenting different modes of classifying the techniques, none of these studies focuses on the identification of features and their constraints to compose visualizations. Besides, although several studies mention the application of visualization elements in software tools, no work categorizing a large set of elements in terms of feature models was found, in order to perform a proper comparison.

4.8.4 Mapping Structure of Goals and Visualizations

Before elaborating the mapping structure, an analysis of related work was performed to find out whether existing solutions would meet the aforementioned needs. Some of these works are presented as follows.

The well-known and largely applied Goal-Question-Metric (GQM) approach [Basili et al. 1994] comprises the setting of goals, the derivation of questions from such goals, and the choice of metrics to answer these questions. GQM was the first attempt to derive the necessary data for Zooming Browser. However, it was soon recognized that a metric or a set of metrics is usually not enough to answer all the questions developers have. It becomes necessary to perform some kind of task to find out the answers to those questions. Besides, GQM does not aim at mapping metrics and visualizations.

Some approaches offer a customizable mapping between visual elements and data. CogZ [Falconer et al. 2009], for instance, is a set of tools that provides a module for the rapid development of specific visualizations for ontologies. It provides drag and drop mechanisms for mapping concepts (ontology terms) to visual representations. Apart from the benefits and the customization facilities, the mapping process starts from the data, not focusing on the goals that led to choosing such data.

Beck et al. (2013) aim at helping visualization experts to choose visualization techniques for dynamic graph visualization scenarios. Profiles reflecting different aesthetic criteria describe both the techniques and the application: their similarity shows how appropriate a visualization technique is for such application (it may be necessary to refine profiles or consider other criteria to achieve a best match). A solid knowledge of visualization techniques and significant experience in visualization design are required [Beck et al. 2013].

These approaches support particular stages of the mapping process, but none of them provide the full picture on the mapping between a set of goals and visualizations.

4.9 Final Remarks

This chapter described APPRAiSER, including its tools and conceptual elements, aiming to assist the execution of some tasks related to software reuse, both at the organizational level and project level, providing reuse awareness and visually supporting the execution of these tasks. The approach was based on some desirable features obtained from two studies: a semi-structured interview with practitioners and a secondary study with respect to the state-of-the-art. Through its core elements, APPRAiSER intends to support reuse managers and developers in performing their tasks with less effort, quickly and easily.

A quote attributed to Cicero[63] is that "the causes of events are ever more interesting than the events themselves". This maxim leads to exploring the facts that made an event happen. However, the information about these facts is often unavailable or not explicit. In order to enable the understanding of the causes of any problem detected in the reuse scenario, further investigation and exploration of the available data are required. APPRAiSER makes it possible by allowing to obtain useful contextual information that may help understanding reuse events more precisely.

This work also aims to encourage a better exploration of software-related information that can be helpful in solving, at least partially, some of the still remaining software reuse issues. In this sense, it is easier to perform analyses by querying and interacting with the available data. By turning such data into visual abstractions, it is possible to build visualization tools (such as Zooming Browser) that can considerably enhance the exploration and understanding of a particular event of interest.

As mentioned previously, APPRAiSER aims at promoting software reuse in a progressive way, so that cultural barriers can be gradually overcome and all stakeholders can become committed with the reuse initiatives by perceiving the benefits brought by them, without causing cognitive overload. In this sense, for meeting RA2, Table 4.7 presents a suggestion on how APPRAiSER (particularly Zooming Browser and its perspectives) could be used in different stages of reuse initiatives, i.e., the expectations with its use.

[63] Marcus Tullius Cicero (106 BC – 43 BC) was a Roman philosopher, politician, lawyer, orator, political theorist, consul and constitutionalist.

Table 4.7 – The use of APPRAiSER in different stages of reuse initiatives

	Initial stages	Intermediate stages	Advanced stages
Dashboard perspective	Supports awareness and communication of first reuse achievements, communicating results in a fast and effective way.	Keeps stakeholders' motivation and help institutionalizing a reuse program.	Provides relevant data for supporting decision-making.
Metadata Exploration perspective	Provides information about reusable assets in a centralized way, easing and stimulating their reuse.	Supports exploring the sources of information, avoiding user disorientation when there are a reasonable number of reusable assets.	Provides a better understanding and exploration of relationships between developers, assets, and projects, while avoiding information overloading (when there is a large amount of information). Also helps identifying kinds of projects in which assets are usually reused (for identifying domains with greater reuse potential) and top producers (to lead development for reuse).
History perspective	Supports finding occurrences of candidate reusable assets in the organization projects, helping to populate the reuse repository and stimulating further reuse.	Supports deeper analyses of the assets' history, when the organization is already used to the reuse practices.	Supports allocating teams for the development of reusable assets and monitoring the development history aiming to identify how producers work together.
Low-Level Data Representation perspective	Helps understanding underlying details (e.g., structure) of assets in order to [better] reuse them.	Helps understanding common characteristics of reusable assets and potentially detecting refactoring opportunities.	Helps analyzing how assets are being developed and detecting potential bottlenecks for their reuse.
Zooming Browser as a whole	Helps engaging stakeholders in the establishment of a reuse program.	Helps identifying key consumers and domains in which there are more reuse opportunities, towards consolidating a reuse program.	Provides information for supporting continuous improvement of reuse activities and processes.

It is important to highlight that APPRAiSER can only partially support the achievement of these results, and it may be necessary to extend it in order to provide a better support, especially for more advanced stages.

It is known that some repository data may be sometimes incorrect or do not necessarily represent actual development data (e.g., data from issue trackers may be outdated or not be filled out at all). However, the Repository Miner implementation assumes that the development team uses software repositories properly, and that the

data about assets, developers, and projects are up-to-date. If this is not the case, this will only count against the projects/developers/assets (i.e., it will be an evidence against reuse). The purpose of APPRAiSER is not to handle this problem: it may at most point it out (which is believed to be relevant for stakeholders). Thus, such scenario is not considered as a limitation for this work.

Other potential sources not listed in this work should be explored as well. Social media tools geared to software development such as forums, mailing lists of development communities, and Q&A (question-and-answer) websites (e.g., StackOverflow) have been receiving a growing attention in the last years. An insight for future research is the analysis of the role of these sources of information for better supporting software reuse and other software engineering fields.

Finally, the following considerations are concerned with the conceptual contributions of APPRAiSER:

- Although the composition process of the visualization feature model is based on different literature references, with the confirmation of some applications of the features through a *quasi*-systematic review, it is important to check the categorization options and the model completeness in terms of visualization elements. Since novel visualization and interaction resources are steadily under development by several researchers and practitioners, it is expected to constantly expand this model based on findings of new studies.
- The mapping structure has proved to be useful in the design of Zooming Browser, and it is expected that it can serve as an initial guidance to future developments of other visualization tools as well, emphasizing the importance of performing each stage carefully. In order to deepen the understanding of the usefulness of this mapping, it is important to investigate with the visualization community answers to the following questions (and others that may emerge through other applications of the mapping):
    - Does this mapping structure actually support developers of visualization tools in better planning the visualization metaphors and resources to use?
    - How can each stage of this mapping be facilitated, i.e., what additional support can be provided for easing such stages?
    - Is it relevant (or crucial) to build tools to help performing this mapping?

These points, along with the problems, limitations, open issues, and features not handled by the current implementation of APPRAiSER (including the recommendation

of assets and the suggestions for refactoring an asset to improve its reusability), are part of a research agenda, to be conducted in collaboration with the software engineering research community.

In order to help with understanding to which extent APPRAiSER can support reuse tasks, and whether the provided information and visualizations are useful to answer some of the questions listed in Appendix A, it becomes necessary to perform some kind of evaluation. This topic is presented in the next chapter.

CHAPTER 5 – EVALUATION

This chapter describes the evaluation of APPRAiSER main elements, including the planning, execution, results, and conclusions.

5.1 Evaluation Scope

The evaluation of APPRAiSER consists of 3 steps:

- the evaluation of the visualization feature model,
- the evaluation of Zooming Browser, and
- the evaluation of some outputs from the mapping structure applied to the Zooming Browser (in terms of their relevance).

The visualization feature model encompasses the knowledge acquired through the performed studies. Thus, in the scope of this thesis, experts and intermediate-level researchers in the visualization domain evaluated some aspects of the features through a peer review. This evaluation is the third step of the domain analysis mentioned in Section 4.5.1, and is described in Section 5.2.

The evaluation of the performed mapping in the context of Zooming Browser is a result of the evaluation and use of the tool. This can eventually provide information on what aspects are lacking or could be mapped differently. Besides, some outputs of the mapping are evaluated indirectly through the Zooming Browser evaluation questions.

The Zooming Browser evaluation (presented in Section 5.3) consisted of a feasibility study with reuse managers and developers from the industry. Repository Miner is integrated to Zooming Browser (which visually presents the information collected by it), and the relevance of the collected data composes one of the steps of the Zooming Browser evaluation (through a survey with the participants). The CAVE tool, in turn, was already evaluated in [Vasconcelos 2015].

Table 5.1 summarizes the aforementioned aspects. Other aspects that were not evaluated in the scope of this thesis may be object of evaluation as part of a research agenda. More details can be found in Section 5.4.

Table 5.1 – Evaluation scope

Elements presented in this thesis	Considerations on their evaluation
Visualization Feature Model	The evaluation of the model features is described in Section 5.2.
Mapping Structure	The relevance and usefulness of the outputs were already discussed in Sections 4.6 and 4.9. Some outputs of the mapping are evaluated indirectly through the Zooming Browser evaluation questions.
Zooming Browser	The evaluation is described in Section 5.3.
Repository Miner	The relevance of the outputs is described as a constituent part of Section 5.3.
CAVE	Evaluated in [Vasconcelos 2015].

The next sections present the steps for evaluating the selected approach elements. No gender distinction is made during the description due to the imbalance of male and female participants, which could enable their identification among them.

5.2 Evaluation of the Visualization Feature Model

A preliminary evaluation of the feature model was conducted, aiming at assessing its syntax/form and content. The study was designed taking some elements of a feature model checklist [Mello et al. 2014] as a basis. Some researchers that have publications in the information visualization field or related fields were invited to perform a peer review by means of a questionnaire (presented in Appendix B).

Due to its size and complexity, the model was divided into categories (listed in Appendix B and throughout this section), so that each potential participant could evaluate a smaller set of features. Each of the evaluated features is composed by a description, an illustrating figure (if applicable), the possible constraints on the use of such feature (if applicable), and the references used for these aspects.

5.2.1 Planning

Five parts compose the questionnaire. The *characterization* (first part) allows obtaining the participants' background on the topics involved in the study, as well as their academic degree. The *overview of the visualization feature model* (second part) provides a "big picture" of the model, with all its features and constraints, for contextualization purposes; in this part, the participant must select which category of features is going to be evaluated.

The third and fourth parts compose the *evaluation of the feature model* itself, which encompasses questions regarding the clarity of descriptions, appropriateness of figures, assessment of the identified constraints, and some aspects for its inspection (such as checking for inconsistency, omission, ambiguity, and so on). Finally, the last part comprises some follow-up questions for identifying potentialities of the model and opportunities for further research.

A pilot study was run with one master student and one undergraduate student. They filled out the questionnaire in order to identify any potential problems in the study and specify an estimated time average for participants. They suggested the inclusion of animated images (e.g., .gif files) for illustrating some features whose understanding would require such resource, and a better explanation of some instructions. Besides, one participant identified inconsistencies between what was presented and what was asked. The corrections were made before sending the invitations to potential participants.

5.2.2 Execution

The division of categories between participants followed the criterion of keeping at least one participant with Ph.D. (or D.Sc.) degree in each category. However, since some of the invited participants did not reply to the invitation e-mail, some changes were made so that each category would have at least one participant with ongoing or finished Ph.D. course. Some master students that were working with visualization and/or interaction resources were also invited for participation; for proper balance, these stayed in a category that included a participant with at least an ongoing Ph.D. course.

Table 5.2 – Participant's academic background according to the categories

Category	**Participants' Academic Background**	
Information Visualization – Focus + Context, Overview + Detail, and Details on Demand features	Ph.D. Degree	Master Student
Information Visualization – Hierarchical Layout and Perspective features	Ph.D. Degree	Master Student
Information Visualization – Layout features	Ph.D. Student	Ph.D. Student
Information Visualization – Other features	Ph.D. Degree	Ph.D. Student
Interaction – Filtering features	Ph.D. Degree	Master Student
Interaction – Panning, Browsing, and Zooming features	Ph.D. Degree	Ph.D. Student
Interaction – Other features	Ph.D. Student	Master Degree

Each category had two respondents. The participants' level of experience on some topics related to the study is presented in Figure 5.1.

Figure 5.1 – Participant's level of experience on related topics

4 out of 7 categories had at least one participant who stated to have practical knowledge about information visualization in industry projects, while other 2 categories had at least one participant who declared to have practical knowledge based on personal projects in the topic. In general, participants had a medium- to high-level of experience in information visualization.

Regarding the level of experience in Computer-Human Interaction (CHI), participants had mostly an intermediary level: in 4 categories, there was at least one participant with practical experience (being two in industry projects and two in personal projects). Finally, with respect to feature modeling, most participants had a superficial knowledge on the topic. Only one participant stated to have practical experience (in industry projects).

5.2.3 Analysis and Results

The participants who evaluated the category "*Information Visualization – Focus + Context, Overview + Detail, and Details on Demand features*" agreed most of the times that the features had a proper description. The only exception is the Focus + Context feature. According to the participant, "for concepts such as this, it is very useful to use examples, so one might understand what 'focus' and 'context' mean".

Regarding the figures used for representing the features and the associated restrictions, there was no consensus in most of the cases. One of the participants stated that he/she did not understand the goal of using the figure (which is an illustration example of the feature). Besides, he/she was in doubt if "the term 'constraints' meant that the feature can always be present" or "the idea of something else (other features?), imposing constraints on this feature". Either way, the participant was not sure if this was true. The other participant missed a self-explanatory legend in the figures for improving their understanding.

In the "Magic Lens" feature, both participants agreed that the figure is not adequate; besides the lack of a legend, one of the participants stated that the image presented is more close to a "magnifying glass" than the Magic Lens concept. However, the figure is similar to the one in the original publication that proposed the concept.

With respect to the relationships between the features from this category, one of the participants considered adequate, while the other stated that he/she "did not know" or "was not sure". By their comments, it was noticed that the concept of "feature" might have been misleading in the context of this study, which might have hampered the

evaluation (since the participant asked what was the purpose of the model, which was not clear in his/her opinion). The participant also asked if the goal was aiding in the analysis of existing visualizations or helping to build new ones. This misunderstanding also impacted his/her answers to check for ambiguity, inconsistency, and omission.

The description of most features from the "*Information Visualization – Hierarchical Layout and Perspective features*" category was considered adequate by the participants. The only exception was the 3D feature, whose description was not considered adequate by one of the participants, since he/she stated that there could be more information about the disadvantages on the use of the third dimension for enriching the description.

Regarding the images, one participant considered that some were not adequate, as he/she stated, "features associated with visualization should make use of images to better understand the spatial placement of its elements". However, he/she only made this comment for abstract features, whose concretization had an illustrating figure. Other comments related to the figures include the presentation of other Cone Tree, Tree, and Cluster representations, and a less complex example for the Treemap feature.

One of the participants was not sure about the concept of constraints, since in this category all of them were described as "N/A"; thus, he/she asked these questions with the "I don't know/I am not sure" option.

All the relationships between features in this category were considered adequate. However, one participant suggested restructuring some parts (e.g., including Tree in a more generic level which would be derived in items such as Cone Tree, Treemap, Partition, and so on). Furthermore, this participant suggested an additional level on the feature model to present the different layouts of a feature. Finally, the participant did not feel comfortable with the use of the term "Perspective" as being a generalization of 2D and 3D, but he/she stated that he/she did not find a more suitable term.

In the "*Information Visualization – Layout features*" category, one of the participants did not consider the description of some features (Scatter Plot, Parallel Coordinates, and Bar Chart/Histogram) adequate. He/she suggested a literature reference for improving the description of the Scatter Plot feature, and pointed out some improvements for making the description more clear. One participant suggested making a better differentiation between Pie Chart and Bar Chart.

With respect to the figure, with the exception of the Stack feature, all the other features contained adequate images. The participants also suggested other figures. The

constraints defined in the Scatter Plot and the Parallel Coordinates features were considered incorrect. One participant did not agree that Details on Demand is a mandatory constraint for the Scatter Plot ("maybe a desirable one"), since this can be produced on paper, without interactivity. For the Parallel Coordinates feature, one participant suggested interaction constraints for allowing the reordering of dimensions.

One participant did not agree with the relationship of the Bar Chart/Histogram; however, the description of the problem could not be identified. Another participant suggested including other kinds of visual paradigms (i.e., Graph-based, Hierarchical, Geometric Projection, Pixel-Oriented technique, and Iconographic). Both participants provided literature references for supporting these improvements.

Regarding the description of features of the "*Information Visualization – Other features*" category, in most cases participants agreed that the description was correct; only the description of Overlap, Stereovision, and Presentation features were pointed out as incorrect (or, in most cases, incomplete).

Both the figures and the constraints were considered adequate in this category, with the exception of the Binning: one participant stated that Binning should require the Interaction feature, since the user could zoom in and out. Both participants agreed that the relationships between features were adequate and did not point out problems regarding ambiguity, inconsistency, omission, or extraneous information.

In the "*Interaction – Filtering features*" category, participants agreed, for most of the features, that the description, figure, and constraints identified were adequate. Regarding the description, one participant did not consider 3 features adequate: Mode, Method, and Tuning/Tweaking. In Mode and Method, the participant suggested the presentation of the subtypes of the respective features for making their description adequate. In Tuning/Tweaking, the participant stated that the definition was not clear, since it employed other terms that were not previously presented "(e.g., 'tweaking' and 'stand from the screen')".

According to one participant, only the Tuning/Tweaking and Segmentation figures were not considered appropriate, since the former did not have a higher resolution and the latter was not clear enough to represent Segmentation. Both participants agreed that the relationships between features were adequate, and there were no problems with ambiguity, inconsistency, and extraneous information. However, one participant thought there was omission of some features, such as "Filter by keyword" and those that contain information regarding augmented reality.

For the "*Interaction – Panning, Browsing, and Zooming features*" category, only the Drag and Drop feature was pointed out as inadequate by one participant. However, the problems pointed out by him/her refer to its relationship with Panning, not its description. Regarding the figures, both participants affirmed that the "Drag and Drop" and "Querying" were not clear, since (i) they did not reflect the description of the feature, and (ii) some elements used in the figure were not clear. One participant also did not consider appropriate the Browsing, Navigation, and Zooming figures. He/she asked for "more clarity" in the two former, since some elements of the figure made it confusing. Regarding the latter, according to him/her, the figure should present the same object with three different zoom levels to make the concept clearer. The Semantic variation of the Zooming feature presented such example, but maybe the participant expected this to be on the higher-level feature.

Four feature constraints were not adequate for one participant. Most of the times, the participant suggested the addition of some constraints (e.g., Expand/Collapse for the Browsing feature, Selection for Querying, and Zooming for Geometric). Regarding the relationships between features, one of the participants seemed to disagree with the established "hierarchies" (e.g. Drag and Drop as a child of Panning), although he/she did not provide any explanation for this.

No participant identified problems with respect to ambiguity, inconsistency, and extraneous information. Regarding omissions, one participant stated that the feature "Details on Demand" could be incorporated in this category as well.

Finally, for the "*Interaction – Other features*" category, both participants found the description of the Aggregation feature as not adequate. One of them found the definition vague and imprecise, while the other suggested that the definition should inform that aggregation is very important for the summarization of large data.

One participant did not consider adequate the description of Linking (since it uses the term 'highlighting', which makes some confusion with the Highlighting/Mitigation filtering feature) and Selection (due to the "poor" description of the feature, emphasizing its frequency of use and relevance instead).

According to one participant, the Selection and Aggregation figures were not appropriate. In the Selection figure, "the mouse pointer does not look like it is selecting some of the graph nodes", while in the Aggregation feature, "the figure does not demonstrate the concept of aggregation, and the correspondence between figures is not clear".

With respect to the constraints, both participants agreed they were adequate, except the Aggregation constraint, since one participant did not agree that Zooming mandatorily requires Aggregation, "although it would be very useful". Regarding the relationship between features, participants agreed that they were adequate. One participant informed, during the check for ambiguity, that the clustering feature appeared two times (as an information visualization method and as a hierarchical positioning feature); the participant stated, "it is better to say that it is an information visualization method".

During the follow-up questions, when asked about the scenarios in which the feature model could be used, five participants mentioned its contribution for knowledge management:

- "it might be useful for analyzing existing visualizations";
- "it is a way of organizing the knowledge in the field";
- "it could be used as a basis for the creation of an ontology or a Wiki about information visualization";
- "it shows different ways of visualizing data";
- "it shows the different possibilities on combining visualization features".

One participant also pointed out the potential of using it in "teaching/training for beginners in the field".

Four participants stated that it could be used in practice in different ways:

- "it can be a driver towards identifying reusable parts for accelerating the development of visualization tools";
- "it would be very useful for building a visualization product line and for general decision about which visualization can be more useful in a given scenario";
- "it could be used in projects in which the interaction between the user and the system are of great importance, and for contextual application design";
- "it is useful for building systems with large amounts of data".

As it can be noticed, two participants mentioned the model as an initial step for supporting the reuse of visualization features. If one could build systems or frameworks in a "feature-based" way, this could potentially ease and accelerate the construction of visualization tools. However, one participant pointed out that the feature model "seems of little use in building new visualizations". The same participant stated that the model concepts and examples "are useful in analyzing and thinking about visualizations".

Regarding suggestions for improving the description of the feature model, one participant said that there should be more examples of use of each visual metaphor. Another participant stated that the model needs to keep growing, and it would be interesting to represent that somehow.

Based on the comments and suggestions made by the participants, an improved version of the visualization feature model will be prepared and made available through an interactive website (a kind of visual catalog of features) for allowing a better exploration of features and their relationships/constraints. This might also help performing further evaluations. For performing the improvements, the answers pointed out by all participants will be discussed for ponderation.

Some limitations and threats to validity of this study include: (i) the closeness to some participants, which may have caused bias, (ii) the low familiarity of many participants with feature modeling (shown in Figure 5.1), (iii) the lack of answers when a participant disagreed with something, since it is not possible to improve the model without an explicit statement of the problems, (iv) the low number of participants, and (v) the size of the questionnaire, which may have tired some participants to properly analyzing the questions.

5.3 Evaluation of Zooming Browser

This section presents the study for evaluating the feasibility of Zooming Browser. The goals of this study are described in Table 5.3, according to the GQM approach [Basili et al. 1994].

Table 5.3 – Identification of the study goals

Questions	Answers
Object of study (what is going to be analyzed?)	the Zooming Browser tool
Purpose (why / for which purpose the object is going to be analyzed?)	characterizing
Quality focus (what properties of the object will be analyzed?)	feasibility in supporting the execution of reuse tasks
Viewpoint (who will use the collected data?)	software developers and reuse managers
Context (in which context the analysis will be performed?)	software development tasks and organizational tasks

Thus, this study can be defined as follows:

Analyze the Zooming Browser tool

For the purpose of characterizing

With respect to the feasibility in supporting the execution of reuse tasks

Under the point of view of software developers and reuse managers

In the context of software development project tasks and organizational tasks

Similarly to [Fritz & Murphy 2010], it was decided not to perform a comparative study, as it was not possible to identify any other approach close to Zooming Browser that provides a similar support. The only analogous alternative to the execution of most tasks is to search for the information provided by the tool in several different websites and perform data aggregation manually, which is an obviously time-consuming approach.

Thus, due to the exploratory nature of the study, there is no baseline for comparison (i.e., there is no setting for the execution of such tasks without visual support). The goal of this study is to obtain quantitative and (mainly) qualitative information regarding the extent to which Zooming Browser can achieve the purposes for which it was built.

5.3.1 Planning

The next subsections describe details on the planning.

5.3.1.1 Study research questions

In order to consider Zooming Browser useful, developers and reuse managers must be able to easily use its visualization and interaction resources to answer questions of interest about software reuse. The main research question of this study aims to assess the following: *Is the use of Zooming Browser feasible in supporting the execution of reuse tasks?* Some study research questions (SRQs) were derived from it for investigating some aspects in more detail, as follows:

- *SRQ1. Is the information provided by Zooming Browser useful for supporting the execution of reuse tasks?*
- *SRQ2. Are the visualizations and interaction resources employed in Zooming Browser useful for supporting the execution of reuse tasks?*
- *SRQ3. By performing the reuse tasks through Zooming Browser, is the efficacy considered satisfactory?*
- *SRQ4. By performing the reuse tasks through Zooming Browser, is the efficiency considered satisfactory?*
- *SRQ5. To which extent does Zooming Browser ease the execution of the reuse tasks?*

5.3.1.2 Variables

The evaluation uses the following quantitative variables: **efficacy** (measured through **precision** and **recall**) and **efficiency**. They are described as follows:

- **Precision** (exactness)
  - number of correct answers given by the participant divided by the total number of answers given by the participant.
- **Recall** (completeness)
  - number of correct answers given by the participant divided by the total number of expected correct answers.
- **Efficacy** (correctness)
  - the F-measure[64] is used in the context of this work for analyzing efficacy as a tradeoff between precision and recall; it is calculated as two times the precision times the recall, divided by the sum of the precision and the recall.
- **Efficiency** (time needed to answer each question)
  - number of answers given by the participant (including incorrect and duplicate answers) divided by the time spent.

When measuring time, two different aspects are considered: (i) the time spent *for starting to answer*, and (ii) the time spent *on answering*. The former is accounted since the moment the reading of the question is finished until the moment the participant starts answering verbally. Thus, it encompasses interactions performed and the rationale to answer the question. The latter is accounted since the moment the participant starts answering verbally until he/she finishes answering. This decision is based on previous experiences, which showed that people might find the answer quickly, but take some time to elaborate it orally or textually. The opposite also applies.

The assessment of the adequacy of both efficiency and efficacy values obtained is made qualitatively, since there is no baseline for comparison and this is not a comparative study. Thus, although it is measured quantitatively for providing the efficiency of executing reuse tasks with Zooming Browser, its adequacy is evaluated in terms of the participants' perception with respect to this variable.

The qualitative information is composed by the following variables:

- **Perceived usefulness** of the presented **information**

[64] The traditional F-measure (or balanced F-score, or F1 score) is known as the weighted harmonic mean of the precision and the recall, with equal balance between these variables.

- **Perceived usefulness** of the presented **visualizations**
- **Perceived usefulness** of the employed **interaction** resources
- **Perceived efficiency** of the tool (to contrast with the efficiency variable)
- **Perceived easiness** in performing the tasks
- **Perceived difficulties** identified in performing the tasks

With the exception of the perceived difficulties, all these variables are measured through a 5-point Likert scale [Likert 1932]. In addition, the researcher may take notes related to the difficulties not made explicit by the participants, based on his observation.

5.3.1.3 Study population

Reuse managers and software developers compose the population of this study. To be eligible to participate in the study, reuse managers should have participated in an MR-MPS-SW implementation of levels E or above or have led reuse initiatives on their organization, being aware of the assignments of this role. Software developers, in turn, must have a minimum knowledge of software reuse practices, i.e., must have at least tried to reuse an asset in the context of a project. The selection of participants is based on convenience sampling, i.e., because of their convenient accessibility and proximity to the researcher. Potential participants are selected based on their availability, and can also recommend other participants.

The tasks to be performed by reuse managers are different from the ones targeted to software developers (although some questions are common to both roles, as shown in Section C.3 of Appendix C). The characterization of the participants defines the role that they will perform.

5.3.1.4 Study method and instruments

This study uses a *replicated project study design*, i.e., it employs multiple subjects (or teams of subjects), all working on the same application and scenario [Seaman 1999]. The advantage of keeping the application constant is "to isolate the effect of differences between subjects, especially, it is hoped, the treatment effect" [Seaman 1999].

For the study setup, a set of questions was selected from the original questions for this study (presented in Appendix A) and grouped in tasks. They were chosen based on the performed mapping (also presented in Appendix A) to cover the Zooming Browser goals in a hypothetical setting. Similarly to [Fritz & Murphy 2010], each

question was made more specific in this study for two reasons: (i) to reduce the range of interpretations of the questions (so that the different approaches of participants to answer the questions could be compared), and (ii) to match the data in the study setup.

The participants are oriented not to interact with the tool, neither while the tasks and questions are being presented to them nor while asking clarification questions, for a proper accounting of the time spent. Each question is presented for the participants orally, one at a time; otherwise, in the later analysis of the sessions, it would not be possible to differentiate when the participant is reading the question or using the tool.

The study is composed of four steps, each with a corresponding instrument (all of them are presented in Appendix C). First, participants must fill out a characterization questionnaire (presented in Section C.1). This is followed by the application of a survey on the relevance of some information on executing reuse tasks (presented in Section C.2)[65]. The third step is the execution of a list of tasks (presented in Section C.3) with Zooming Browser, according to the performed role. Finally, participants fill out a follow-up questionnaire (presented in Section C.4). Some of the questions from the characterization questionnaire were inspired in previous works ([Oliveira 2011] [Pötter et al. 2014]) and in [Bauer et al. 2014].

Each task is composed by a set of questions. The initial question of each task requires the participant to inform how he/she would perform the task in his/her day-to-day activities without the tool support. For its answer, the time was not taken into account, because no interaction is made with the tool. The last question of each task, in turn, involves taking a decision related to the task after having access to the data provided by the tool. Although it did not involve tool interactions, the answer was accounted in the time variable because it is a result of the previous interactions. Both kinds of questions do not have an expected answer, since different participants can have different opinions and take different decisions based on their background experience.

5.3.1.5 Data collection methods

The data collection methods are (i) the *participant observation* (whose idea is to capture firsthand behaviors and interactions that might not be noticed otherwise [Seaman 1999]), and (ii) the application of the instruments mentioned in Section 5.3.1.4. Because much of software development work takes place inside a person's

[65] This survey evaluates the relevance of the information collected by Repository Miner.

head, such activity is difficult to observe. Thus, *think aloud protocols* are used in addition to the participant observation. These protocols require the participant to verbalize his/her thought process so that the observer can understand the process going on [Seaman 1999].

The recording of study sessions is based on audio recording (if authorized by the participant) to save participant's time and to keep track of some reactions and observations more carefully, improving the quality of the data and the analysis.

5.3.1.6 Evaluation setting

The study setup is based on a set of tasks elaborated for each role (reuse manager and developer), described in Appendix C (Section C.3). Such tasks use a subset of real data (collected from open source projects) and some hypothetical situations, inspired on previous experiences of the author of this thesis in implementing and assessing reuse processes in software organizations[66]. Since the resulting scenario is a combination of real data and hypothetical situations, it is characterized as fictitious, i.e., neither the problems listed in the tasks nor the answers and solutions provided to them represent the reality of any organization or any open source project.

5.3.1.7 Analysis procedure

The mapping presented in Table 5.4 shows the strategy used for answering the study research questions based on the variables presented in Section 5.3.1.2. All these questions contain a direct measure of the variables in the follow-up questionnaire, with the exception of SRQ3, which is evaluated solely through the results of the execution of the tasks. SRQ4, in turn, involves both the results of the execution of the tasks and a direct measure of the perceived efficiency through the follow-up questionnaire.

Table 5.4 – Mapping for the analysis procedure

Study Research Questions	Variables	Questionnaire Items
SRQ1. Is the information provided by Zooming Browser useful for supporting the execution of reuse tasks?	• Perceived usefulness of the presented information	Corresponding question in the follow-up questionnaire AND Results of the questionnaire on the relevance of the information

[66] Due to that, there is a field in the follow-up questionnaire asking if participants think that the executed tasks in the scenario match the day-to-day reality of their performed role.

Study Research Questions	Variables	Questionnaire Items
SRQ2. Are the visualizations and interaction resources employed in Zooming Browser useful for supporting the execution of reuse tasks?	• Perceived usefulness of the presented visualizations • Perceived usefulness of the employed interaction resources	Corresponding questions in the follow-up questionnaire
SRQ3. By performing the reuse tasks through Zooming Browser, is the efficacy considered satisfactory?	• Efficacy (Precision and Recall)	Results of the execution of the tasks
SRQ4. By performing the reuse tasks through Zooming Browser, is the efficiency considered satisfactory?	• Efficiency • Perceived efficiency of the tool	Results of the execution of the tasks AND Corresponding question in the follow-up questionnaire
SRQ5. To which extent Zooming Browser eases the execution of the reuse tasks?	• Perceived easiness in performing the tasks • Perceived difficulties identified in performing the tasks	Corresponding questions in the follow-up questionnaire

The open coding technique [Seaman 2009] is also used for better organizing the qualitative data obtained from the study.

5.3.1.8 Threats to validity identified during the study design

Some threats to validity of this study were identified during its design. Each description is followed by the decision made in its regard. Table 5.5 lists the identified threats and the planned actions for mitigation each of them, when possible.

Table 5.5 – Identified threats to validity and actions for mitigating them

Threat	Action(s) for mitigation
The observational nature of the study may cause anxiety or influence the behavior of some participants (causing observation bias[67]).	This is something hard to control. The participants must be comfortable to perform the study and aware that what is under evaluation is the tool, not the participant. Besides, they are free to ask for pausing or stopping the recording at any time, as well as make questions.
Think aloud protocols are limited by the comfort level of the participants and their ability to articulate their thoughts.	This is an underlying threat to validity when using this protocol. However, although automatic data collection is sometimes more suitable (as posed by [Kagdi & Maletic 2008]), participants' reactions and difficulties (which are highly relevant for this characterization study) would not be properly tracked. Through the audio recording and the participants' observation, it is possible to detect how this impacts each participant.

[67] This is also known as *the Hawthorne effect* (also referred to as *the observer effect*). It is a type of reactivity in which individuals modify or improve some aspect of their behavior in response to their awareness of being observed [Franke & Kaul 1978].

Threat	Action(s) for mitigation
The researcher created the scenario setting and chose the set of questions for the study; they may not be representative of the software development scenario.	These aspects must be assessed in terms of representativeness of a software development scenario. In this sense, a follow-up question is added to ask if the scenario setting reflects the reality of the software development scenario (i.e., if the identified needs are actually relevant), in the participants' opinion.
As the information provided by Zooming Browser was tailored towards the questions asked, participants may have had to spend less time than otherwise to answer a question of interest.	A similar threat was pointed out in [Fritz & Murphy 2010]. This is a limitation that cannot be overcome at the time, because an evaluation in an industry setting would require both collecting industry data and selecting an organization that is currently implementing reuse processes (i.e., that faces the needs that the tool is intended to support). It was not possible to identify (at the time of the evaluation) an organization with such characteristics and availability.
Participants may not be representative of the population to which Zooming Browser is intended.	It was decided to take the risk. The researcher did not consider selecting subjects that are representative of the entire population due to the difficulty and effort involved in this procedure. Instead, the selection of participants is based on convenience sampling, as stated previously. Thus, it is known beforehand that results cannot be generalized.
External factors may distract the participant while performing the task.	This is not under total control of the researcher, as the experiment cannot take place in a single locus, due to restrictions on the availability of potential participants. The only criterion is that the place for the evaluation sessions must be an environment that avoids interruptions as much as possible.

5.3.1.9 Pilot study

A pilot study was run with one master student and one specialization student (i.e., both with an undergraduate degree). The former has 3 years of practical experience in object-oriented (OO) development, while the latter has 6 months. Both stated that they have advanced knowledge in software reuse, and while the master student stated he/she has an advanced knowledge in software development (programming), the specialization student has a deeper knowledge in the reuse management process of MR-MPS-SW. Thus, the former (hereafter referred to as PilotSD) performed the software developer role and the latter performed the reuse manager role (hereafter referred to as PilotRM). The list of executed tasks can be found in Section C.3 of Appendix C.

The pilot study followed the expected flow of the evaluation (described in Section 5.3.1.4). A short explanation of each perspective was given to the participants before the actual execution of the tasks. During this explanation, they were not allowed to interact with the tool, in order to allow verifying the intuitiveness, affordance, and "actionability" of visualizations.

The participants of the pilot study indicated two questions that sounded confusing in the characterization questionnaire, and pointed out a question that was out of place. During the execution of the tasks, PilotSD (which has an intermediate expertise level in software visualization) indicated the lack of affordance because of the mouse cursor. “When hovering some items, the cursor remains as an arrow, but should be a pointer, which indicates that there is something else there to be explored”. None of them could answer one of the questions properly, due to the lack of information in a tooltip. Some questions asked during the tasks were not clear and should be rewritten.

According to both participants, the text size was too small for the screen size, and the title of each perspective was disproportionally large, in spite of the responsive design. Some texts were cropped in some charts and in the reuse map. Besides, participants complained about the impossibility of interacting with the tool before executing the tasks. According to PilotSD, “it is ok not to provide great level of details in the explanation, for not anticipating answers and for assessing the intuitiveness of the tool. However, participants need some immersion time with the tool; otherwise, they will not be able to achieve good efficiency”.

All these items were subsequently corrected before the execution with the participants. Based on the last comment by PilotSD, an additional time of interaction (without supervision by the researcher) was included in the task execution.

5.3.2 Execution

The characterization questionnaire and the survey were sent by e-mail to the participants who agreed with participating in this study. They were answered before the execution of the tasks. 12 participants answered the questionnaire and participated in the full study. From those, 3 participants performed the role of reuse managers (RMs), while 9 performed the role of software developers (SDs), according to their profiles.

The sessions were scheduled and executed in places that were as free of interruptions as possible. The experiment was run in a 13.3’’ notebook with a Core-i5 processor, CPU @ 1.60GHz, 4GB RAM memory, running 64-bit Windows 7 Home Premium, with maximum brightness level. All the study sessions were run on the same machine, in order to avoid issues on hardware performance differences. In addition, all the participants used the same initial configuration of views.

All participants agreed with audio recording their sessions. They were instructed to express themselves according to the think-aloud protocol. They were told they could

make any clarification questions whenever they wanted; however, in some cases, the researcher could ask what they thought was the answer (e.g., to assess if they interpreted the underlying concepts in the visualizations). They had a brief explanation of Zooming Browser and its goals, along with a basic training of its perspectives. Following the presentation of each perspective, participants were given some limited time to interact with them and check the visual responses to their interactions.

After that, participants started executing the tasks, according to the role they performed. When the tasks sessions were done, participants filled out the follow-up questionnaire, without the presence of the researcher.

5.3.3 Analysis

The analysis of this study was divided into four parts: characterization data, results from the survey (on the relevance of information for software reuse), analysis of the task executions, and follow-up data. The detailed analysis of each aspect will be published as a technical report, due to the broadness of results. The main results of this analysis are presented in the next subsections.

5.3.3.1 Characterization data

The academic level background of the participants is shown in Figure 5.2. All RMs were at least coursing a specialization. All SDs had at least finished the undergraduate course, but most of them (5) range between an ongoing master course and a finished specialization/master course.

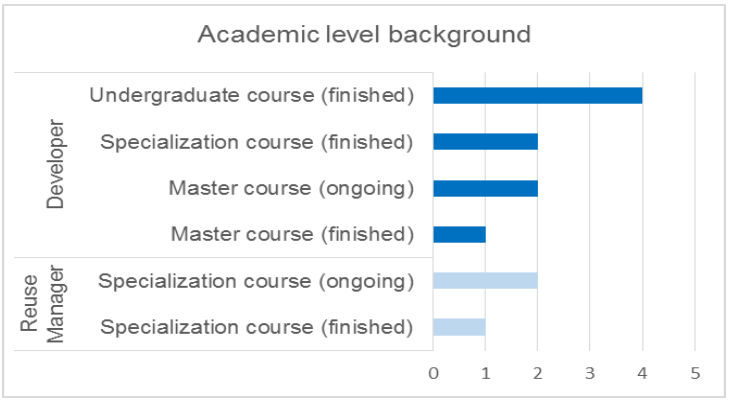


Figure 5.2 – Participants' academic level background

Figure 5.3 shows the participants' OO development experience. 2 out of 3 RMs and 7 out of 9 SDs have developed OO as part of a team in the industry. Besides, participants in general have at least four years of OO development experience, and 4 of them have 10 or more years.

The organizations in which they work include a Brazilian government-owned corporation of IT services, a rendering services organization for a semi-public Brazilian multinational energy corporation, two IT consulting organizations (one is a provider of consulting services, development of business solutions, and marketing, and the other is a management consulting and technology services). There are also specialized companies that provide products on-demand (one of them develops websites for e-commerce and marketing sectors, another is specialized in cloud computing, outsourcing, and services). Finally, other two organizations use IT as a medium for improving their activities (i.e., software development is not their core business).

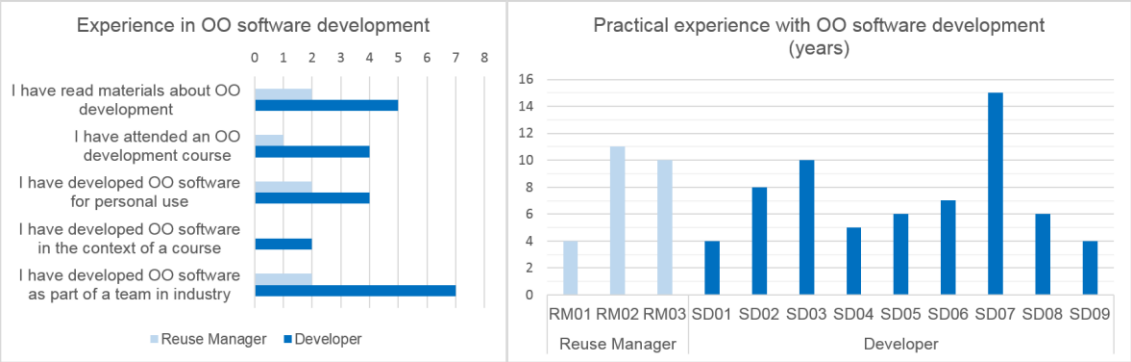


Figure 5.3 – Participants' OO development experience

The level of familiarity with some topics related to the evaluation is depicted in Figure 5.4. Three participants (all of them SDs) have no expertise in issue tracking systems, and three participants (one RM and two SDs) have no expertise in software visualization. All participants have at least a basic expertise in the other topics.

Level of familiarity	Reuse Manager			Developer								
	RM01	RM02	RM03	SD01	SD02	SD03	SD04	SD05	SD06	SD07	SD08	SD09
Software development (programming)												
Software reuse												
Version control systems												
Issue tracking or task management systems												
Software project management												
Software visualization												
English fluency (reading)												

I have no expertise in this topic	I believe I have a basic expertise level in this topic	I believe I have an intermediate expertise level in this topic	I believe I have an advanced expertise level in this topic

Figure 5.4 – Participants' level of familiarity with topics involved in the study

Regarding the knowledge about the MR-MPS-SW, only two participants (all RMs) participated in an implementation or assessment involving reuse processes, and both executed the role of reuse managers. The third RM led reuse initiatives on his/her organization (not related to MR-MPS-SW), being aware of the assignments of this role.

When asked about what they consider for deciding whether or not to reuse an asset, responses were the following:

- Meeting of project needs (3 participants, all SDs);
- Asset compatibility (2 participants, being 1 RM and 1 SD);
- Asset documentation (2 participants, both SDs);
- Asset development/release history (2 participants, both SDs);
- Asset reviews by consumers (2 participants, both SDs);
- Asset issues (2 participants, both SDs);
- Asset popularity (1 participant, RM);
- Asset stability (1 participant, RM); and
- Projects in which the asset was reused successfully (1 participant, RM).

Some of these responses seem to have some intersection. For instance, the reviews by consumers and projects that reused the asset relate to the asset popularity, while the asset development/release history and its issues partially denote the asset stability. Zooming Browser already encompasses some of these data, as shown in the next subsection. The other mentioned information can be taken into account for improvements of the approach.

5.3.3.2 Results of the survey (on the relevance of information for software reuse)

Participants' responses to the relevance of information for software reuse are presented in Figure 5.5. Values range between 1 (totally irrelevant) and 4 (totally relevant), in an ordinal scale. Responses are ordered (from top to bottom) according to the relevance assigned by the participants for each information, and the median values depict the most common opinion among them.

Relevance of information for taking a reuse decision (e.g., for deciding whether to reuse an asset)	Reuse Manager			Developer									
	RM01	RM02	RM03	SD01	SD02	SD03	SD04	SD05	SD06	SD07	SD08	SD09	Median
Asset license	2	4	4	4	4	3	4	4	3	4	4	4	4.000
Asset issues (bugs, feature requests etc.)	3	4	3	4	3	3	2	4	3	4	4	4	3.500
Asset dependencies of other assets	4	3	4	3	4	2	4	2	3	4	2	4	3.500
Asset release history	3	4	3	2	3	3	3	3	4	4	4	2	3.000
Asset producers' contact information	3	3	3	4	2	3	3	4	2	3	2	4	3.000
Number of reuse occurrences of the asset	2	3	4	2	3	3	3	4	3	3	2	4	3.000
Asset development history (commit history)	3	4	4	3	2	3	3	2	2	4	4	2	3.000
Asset producers	3	3	4	4	3	3	2	3	3	2	3	2	3.000
Organization that developed the asset	4	4	4	2	3	3	2	2	3	2	3	2	3.000
Asset consumers	3	2	3	2	4	3	2	4	2	2	3	3	3.000
Projects in which the asset was reused	3	4	3	2	4	3	2	3	2	3	1	2	3.000
Asset consumers' contact information	3	2	2	2	2	2	2	4	1	3	3	2	2.000
Other assets that depend on the asset	3	3	2	4	2	2	1	2	1	4	2	2	2.000
Median	3.000	3.000	3.000	3.000	3.000	3.000	2.000	3.000	3.000	3.000	3.000	2.000	

Totally irrelevant	Somewhat irrelevant	Quite relevant	Totally relevant

Figure 5.5 – Participants' opinion on the relevance of information for software reuse tasks

Results show that the *asset license* is one of the most relevant information for these participants, followed by *asset issues*, since in both cases only one participant considered these kinds of information "somewhat irrelevant", while the others considered at least "quite relevant". In fact, asset licenses must be checked due to their conditions and restrictions, which may incur in legal concerns. Besides, asset issues represent an important indicator of its quality and frequency of corrections, which are crucial aspects when it comes to reuse. The *asset dependencies on other assets* and the *asset release history* also play an important role according to these participants.

The less relevant information, according to the participants, is the list of *other assets that depend on the asset*, followed by the *asset consumers' contact information*. Interestingly, 2 out of the 3 RMs informed that such contact information is "somewhat irrelevant", although it is necessary to communicate them about changes in the status of the assets.

An observation extracted from these results is that each single information in considered "totally relevant" by at least one participant, which shows that some information may be considered of no importance for some, but is very important for others. Another remark is that RMs considered a larger number of information as relevant (along with two developers).

Comparing with the information that participants considered relevant (asked in the characterization questionnaire), the items *asset development/release history*, *asset issues*, and *projects in which the asset was reused* are the only ones that directly match the set of information listed in the survey. However, participants seem to agree with most of the information presented in the survey.

5.3.3.3 Analysis of the task executions

The duration of the individual sessions ranged between 35 minutes and 1 hour, without considering the preparation time. The differences are due to several reasons: some participants found it difficult to start using the tool, but ended up getting used to it during the study. Besides, some participants requested for repeating the questions more than once, because they were missing the question goal while interacting with the tool and exploring the available options.

For illustration purposes, Figure 5.6 shows the performance of the three RMs for each question based on the time spent (in seconds).

Figure 5.6 – Participants' performance (in seconds) in executing Reuse Management (RM) questions

It can be noticed that there is a considerable variation among participants not only in the time for finding the information and starting to answer, but also in the time spent on answering. In addition, occasionally the time for starting to answer is close to or equals zero because the participant already realized (based on previous interactions) where the information is, or because the information is in the last perspective he/she used. However, even in cases like this, some participants spent an additional time searching for the information in other places.

Results for precision, recall, and efficacy per RM question are presented in Figure 5.7. A zoomed copy of the varying parts is positioned on the right side of the figure.

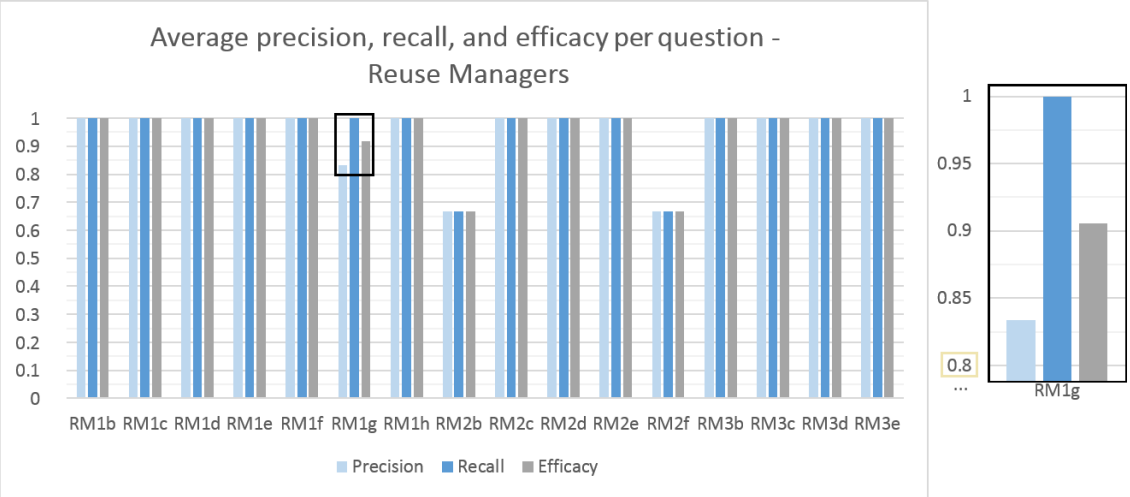


Figure 5.7 – Precision, recall, and efficacy per question (reuse managers)

In RM1g, participants were asked to answer which assets (among the most reused ones) were reused by the 3 most active asset producers. The question might have been "tricky" to answer using the dashboard, because one could interact with the pie chart (so that the producer bar chart is updated) or the opposite way, depending on the interpretation of the necessary steps to answer the question. However, when hovering a slice of the pie chart, the producer chart updates itself with *who produced that asset*, while when hovering a bar of the producer bar chart, the pie chart updates itself with the *assets reused* by the producer (as the title of the pie chart indicates), which was the expected answer. Two out of three RMs gave an incomplete answer (lowering their precision), but the answers they gave were correct (keeping a high recall).

Questions RM2b and RM2f required the participant to make an estimation of the frequency of releases and the time it takes for asset producers to fix reported bugs, respectively, by interacting with a bubble chart. Since this information was not readily available visually (it required reading the bubble tooltips), all RMs took some interaction time for answering the question. Some tried to be more precise than others. The bubble size (which indicates the time) should be used as a starting point, so that the participant could answer the question based on a small number of similar bubbles.

One of the RMs provided an incorrect answer to both RM2b and RM2f. In the former, he/she did not provide a quantitative or qualitative answer, just a statement that there were several releases, while in the latter he/she provided an incorrect answer

(overestimating the time, because he/she mixed up bugs and improvement requests). The other two RMs answered the questions correctly.

Figure 5.8 presents results for precision, recall, and efficacy per SD question. A zoomed copy of the varying parts is positioned on the right side of the figure.

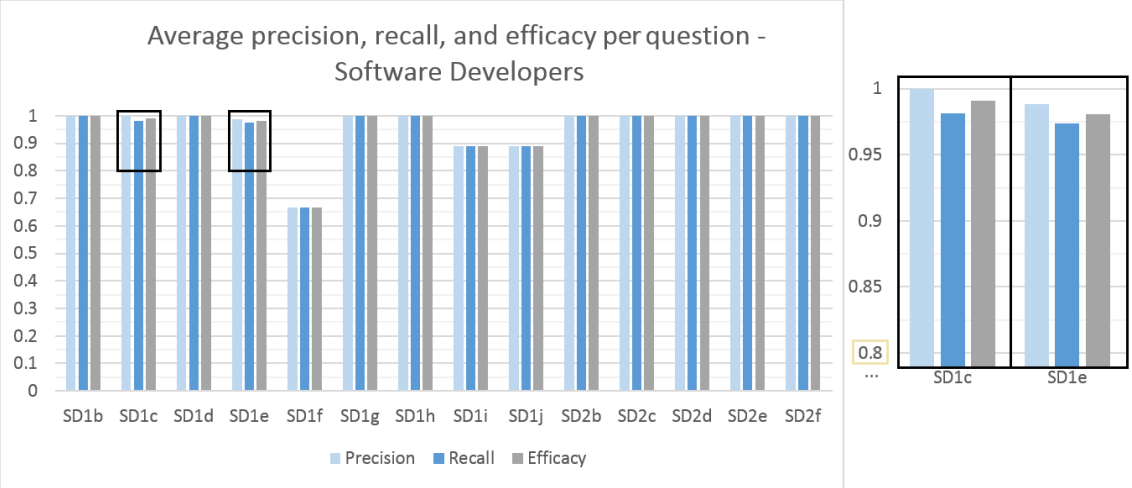


Figure 5.8 – Precision, recall, and efficacy per question (software developers)

Regarding SDs, questions SD1c and SD1e asked which version of the asset were reused and which consumers reused it in which projects, respectively. These questions required the participant to answer orally each of the reused versions and each of the asset reuse occurrences, which can be considered as error-prone tasks due to the amount of presented information. The latter also involved several interactions with the tool, since the information was presented in a tooltip.

The answers provided by SDs for SD1c were all correct (maximum precision), but two participants missed one asset version each (decreasing the recall). This can be due to the amount of bubbles (i.e., versions) that were part of the answer – there were 12 reused versions in total. Regarding SD1e, one SD missed a reuse occurrence (decreasing recall). Another SD provided 19 answers (there were 21 expected answers): he/she missed 4 reuse occurrences (decreasing recall) and provided the same answer twice for 2 reuse occurrences (decreasing the precision).

Questions SD1f, SD1i and SD1j also had a decrease in both precision, recall, and efficacy, at the same proportion. Since SD1f is the same as RM2d and SD1i is the same as RM2f, the reasons stated for these RM questions also apply to SDs. SD1j, in turn, refers to the frequency with which producers implement improvement suggestions. For SD1f, two SDs did not provide a valid answer, and one overestimated the release frequency. For SD1i, one SD overestimated the time (he/she also mixed up bugs and improvement requests). In SD1j, one SD underestimated the implementation frequency.

For all the other questions, all RMs and SDs achieved maximum precision and recall (therefore, maximum efficacy). Many participants provided oral feedback about question SD1e being tiresome; they lacked a filter mechanism to ease the answering.

Because there is no baseline for comparison and since there is not an "ideal" pre-established value for the efficiency, this variable is analyzed by comparing the participants' results with respect to the average of their results. However, results can only be compared in the context of the same question, since the efficiency value does not take into account the different underlying levels of complexity involved in each question, neither the different number of answers when comparing the questions.

The measures for the efficiency variable values per question for each RM are presented in Figure 5.9. The perceived efficiency is analyzed in Section 5.3.3.4.

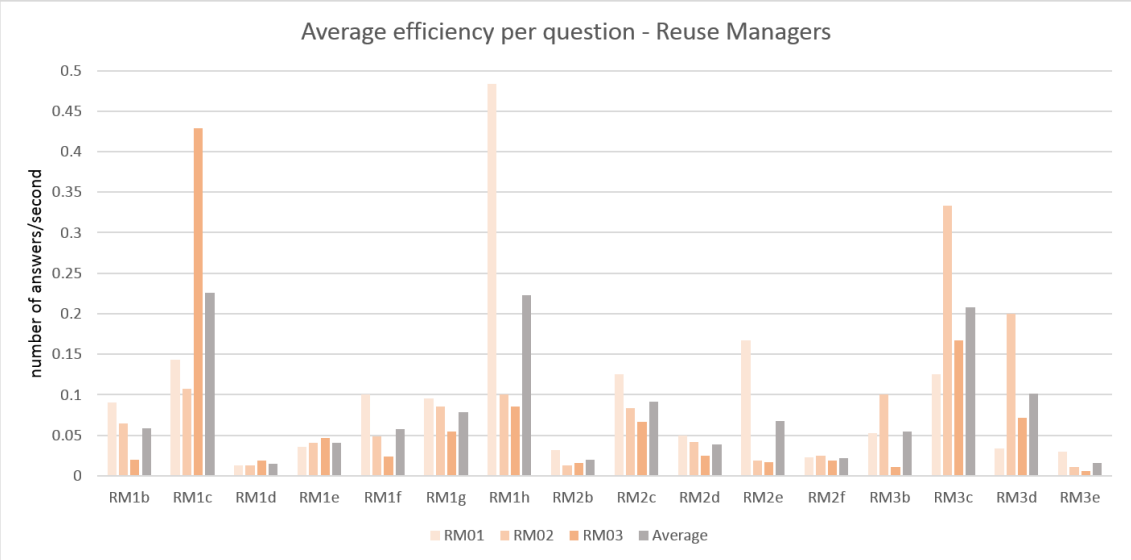


Figure 5.9 – Efficiency values per question for each reuse manager

Although some RMs may seem to be outliers in some of the questions, each participant achieved a considerably greater efficiency (in isolate questions) in different questions. Besides, the sample is too small to perform outlier analyses.

RM1 stayed above the average in 10 out of the 16 questions, and had a particular high efficiency in question RM1h. This question involved the reuse map, and the other participants spent a significant time trying to use another metaphor (due to the lack of a filter option, according to them). RM2 stayed above the average in 7 questions, outperforming the other participants in RM3c. Although all participants spent the same time for starting to answer RM3c, RM2 answered it faster. However, the difference in time to RM1 was not too significant (only 5 seconds).

Finally, RM3 stayed above the average efficiency in only 3 questions. In many cases, he/she stayed under half of the average. This RM stated that he/she lacked familiarity with the tool, and had a particular analysis profile: for answering most questions, he/she kept interacting with the tool in order to explore the available options, even in cases in which the answer was already in the currently open perspective. It is not possible, though, to state that he/she is an outlier: many developers are curious for exploring and understanding the tool features, and the lack of familiarity with the tool impacted his/her performance (as stated in the follow-up questionnaire).

The measures for the efficiency variable values per question for each SD are presented in Figure 5.10. An excerpt of this figure with a detailed view of questions SD1e to SD2b is depicted in Figure 5.11, with a different y-axis scale.

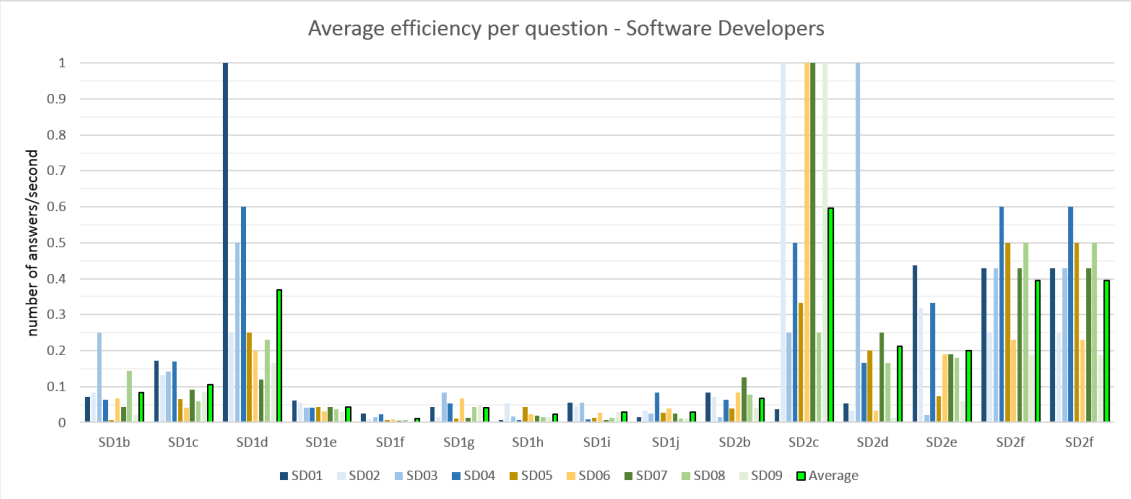


Figure 5.10 – Efficiency values per question for each software developer

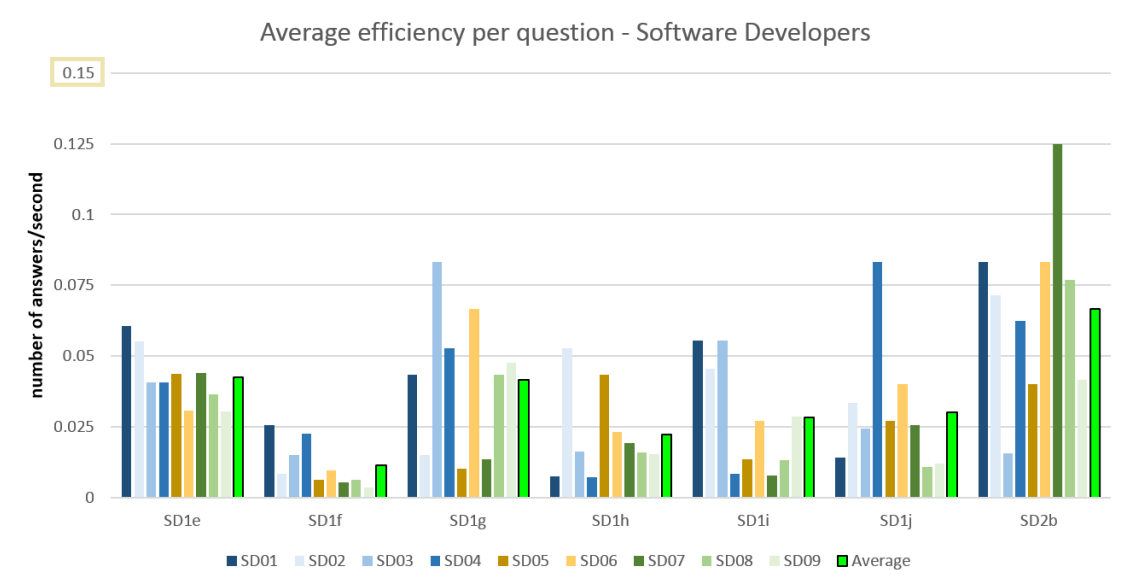


Figure 5.11 – Efficiency values per question for each software developer (excerpt)

It was not possible to identify patterns between participants. In most questions, there was no homogeneity of efficiency. An observation was that most participants did not stay too far from the average in SD1e. In the other questions, there was a large variance between participants' efficiency.

Participant SD01, for instance, stayed above the average in 6 questions, being an isolated case especially in SD1d and SD2e. However, he/she also had the worst efficiency in SD1h and SD1j (related to estimating the frequency with which producers fix bugs and implement improvement suggestions, respectively). Another SD09 had low efficiency values when compared to the others in each question, except in SD2c (for identifying the 3 main producers of an asset), in which he/she achieved the greatest efficiency. In fact, it can be noticed that there is no SD who is the most or the least efficient in all the questions.

One of the questions (asking which consumers reused a given asset in which projects) took a longer time because it required participants to identify a large set of information by interacting with the tool. Moreover, some participants initially refused to use the available view (because it would require several interactions with the reuse map, due to the lack of a filter resource) and kept searching for the information in other views/perspectives. Some participants selected other options more than twice trying to find an alternative way of answering the question, but without success.

Other questions (common to RMs and SDs) that some participants found tiresome were the ones related to the issues bubble chart. Participants in general had difficulties in finding the average time that producers had taken to fix a bug, which should be based on the average size of the bubbles and the tooltip that showed the exact time. The interpretation of the color scheme varied a lot between participants: some thought that red bubbles were the ones that were delayed, other stated that they represented urgent issues, and a few identified that it represented a bug. When participants saw this bubble chart, no legend was initially presented (intentionally) for capturing what they thought the colors meant. After they stated their opinions, a legend was given with an explanation (the time variable did not consider the explanation time).

During the evaluation, there were some interesting findings. Some participants who evaluated the relevance of a given information as "somewhat irrelevant" and "totally irrelevant" ended up using the information for answering some questions (related to the information taken into account for performing the task). For instance, in SD2a, one of the participants who found the consumers' contact information as "totally

irrelevant" stated, "I could also ask for some help from developers, in order to integrate it to the project (preferably developers who already reused the asset); I would identify these developers by asking if they already reused it".

A curiosity is that one SD mentioned that he/she thought the information presented in the tool was more administrative. This was mentioned when answering SD1j. However, for answering SD2a, he/she said that the first thing he/she would do is "to look at the Dashboard and check who is using the asset more often and for a longer time, in order to contact these people to know where to start from". The participant added that he/she "would also check what are the versions most reused". This reinforces the fact that the information provided by the tool is not limited to administrative roles, but it is also a way to provide reuse awareness that can promote communication between team members.

Besides, in RM1h, all RMs noticed that more than 3 projects contained the same amount of assets, so they were not sure about which ones to consider for answering this question. A similar situation occurred in RM1f: RMs were asked to indicate the "top 3" consumers, but a fourth consumer reused the same amount of assets compared to the second and third consumers. After noticing these situations, they were told to consider the ones in order of appearance. This demonstrates that they have not answered the questions "automatically", but paid attention to surrounding information for providing the answer (which could be relevant in a real scenario).

Most participants were able to understand the semantics of the visual attributes used in the Metadata Exploration perspective (especially size and color) and the reuse map (the color scale, without using the available legend). They either stated this explicitly in the think-aloud protocol or provided this information when asked by the researcher ("what did you use to answer this question?") after answering the question.

Finally, three SDs noticed that, in one of the tasks, the suggestion of reusing the asset was given by two developers, but one of them had never reused it, according to the data presented in the perspectives. Interestingly, each participant realized it using a different perspective: one used the Dashboard, another used the Metadata Exploration, and the third one noticed while interacting with the Reuse Map. Besides, 5 participants (1 RM and 4 SDs) figured out that the tool drilled down to the issue webpage for additional information, by selecting a bubble in the issues bubble chart.

5.3.3.4 Follow-up data

Some aspects defined in the planning were evaluated in the follow-up questionnaire. The results are depicted in Figure 5.12.

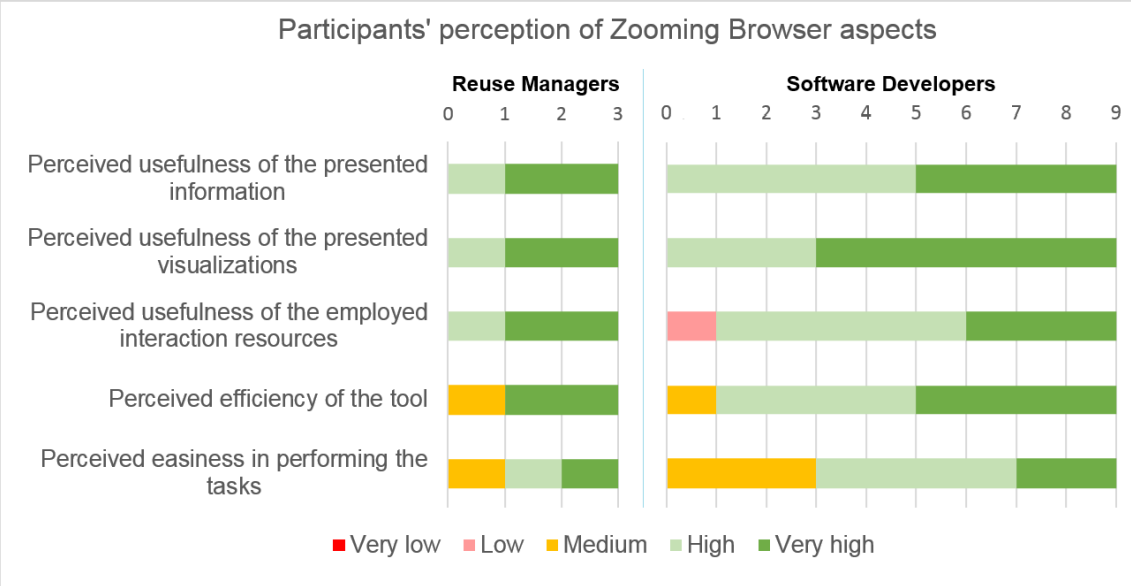


Figure 5.12 – Participants' perception on some aspects of Zooming Browser

As it can be seen, all the participants considered the usefulness of the presented information at least high. One participant complained, "the tool is missing auxiliary data that can be tabulated to complement the displayed data". The usefulness of the presented visualizations was the most well evaluated aspect (8 out of the 12 participants considered its usefulness as very high and the other 4 considered it high).

Regarding the usefulness of the employed interaction resources, one participant (a SD) found it low, while all the others found it at least high. In other follow-up questions, this participant stated that his/her "greatest difficulty was the lack of a steady 'Back' button". He/she stated, "At first, I experienced some difficulty in navigating and changing context between the available perspectives, especially in interpreting and deciding which perspective or visualization would help me perform the tasks the best way".

The efficiency of the tool was considered very high by 6 participants, high by 4 participants, and medium by 2 participants (a RM and a SD). The RM stated in another comment that the "unfamiliarity with the tool" was one of his/her perceived difficulties, which may be the reason for this perception. The SD, in turn, complained about the "lack of clarity and organization in the issues bubble chart". Indeed, he/she was the participant who took the longest time to answer one of the questions related to this

chart, and the second longest time in another question on the same chart. One participant added, “If I already knew the interface, I would probably be significantly more efficient”.

Regarding the easiness on executing the tasks, only 3 participants considered it very high, while 5 found it high and 4 stated it was medium (one of them is the RM who felt unfamiliar with the tool). Other comments include “not knowing in advance (*a priori*) where to find the required information” and “the lack of auxiliary tabulated data to complement the displayed data” (as stated previously).

When asked if the executed tasks match the day-to-day reality of the role they performed, all participants answered positively, but one of them (a RM) mentioned “partially”. According to him/her, “in the day-to-day, the developers’ feedback would be useful combined with the static analysis of the dependencies and other information provided by the tool”.

Most of the identified difficulties seem to be related to the lack of familiarity with Zooming Browser. Participants perceived the following difficulties:

- Some tool issues (mentioned in the improvements for the tool);
- “Navigating and changing context between the available perspectives” (as stated previously);
- “At first, because there is too much information on the tasks, I was a little confused; however, when I got used to the tool, I could perform tasks more easily”;
- “Unfamiliarity with the tool” (as mentioned previously);
- “The first contact with the tool; at the beginning of use, I was more apprehensive (mood), but over time I’ve had a better understanding in practice. In addition, the tool has several features, so recalling where I could obtain an information and in which perspective was more difficult at first”.

With respect to the benefits of Zooming Browser, the following answers were given:

- Regarding the provided information:
  - “Detailed and quantitative information about reuse”;
  - “With the tool, it was possible to identify all the information required to manage, maintain, and execute some reuse tasks without any kind of problem or difficulty”;

- "The tool allows to obtain relevant information about an asset, such as usage history, developers who used it etc., linked with the asset repository. Such information greatly helps in choosing an asset for reuse";
- "The tool favors the decision making for choosing the asset, based on a consolidated view. The fact that the information is made available in a summarized and totalized form is itself a benefit (for example, in the dashboard)".

- Regarding the interface and intuitiveness:
  - "The tool has an intuitive and simple interface, so that within a few minutes I was already familiar with it";
  - "The tool is intuitive and allows for a more practical analysis of assets with respect to their reuse";
  - "The ease of discovering how the interaction between the assets, the organization, its projects, and developers occurs".
- Regarding its contribution to software organizations:
  - "I believe that this tool would help a lot in the context of a large organization, or even for small ones";
  - "This tool presents in a simple way the assets that can be reused. Many organizations do not have control or do not provide visibility over the assets they own and that could be reused in other projects";
  - "There is nothing on the market that brings information with so much completeness as this tool does; by using a tool of this kind in the reuse process, such process would no longer be complex as regards to the audits, asset analyses, and asset provisioning".
- Regarding the visualizations:
  - "I found the tool very useful. The visual forms from which I could get information about the assets were very interesting. It was very easy to use the tool. The icons and tooltips displayed made it easier for me to achieve the desired goals and answer to the asked questions";
  - "The different visualizations about the assets are very useful because they allow understanding a lot of information concisely. I would highlight the visualization of issues, producers, and consumers [as useful]";

- "The visualizations of the information are clear and help making decisions quickly. Finding information is easy after the adaptation time";
- "In a single view, I can get a variety of information, such as when analyzing the issue tracker";
- "The interactive dashboard stands out, from which one can check, in a summarized manner, information regarding reuse within the organization very efficiently".

The pointed drawbacks (besides the aforementioned ones in the overview of the participants' perceptions) are the following:

- "Maybe some people might have difficulties or a greater learning curve to deal with a tool that presents all in such a visual way";
- "The absence of the legends" (4 participants) "when visualizing issue tracker information" (2 participants);
- "In some cases the tooltip appeared over the items" (5 participants), e.g., in the reuse map (4 participants) and in the release history view when collapsing/expanding items (1 participant). According to one participant, "although there is no loss of information, the visualization part becomes impaired".

Finally, they pointed out the following improvements:

- Some participants (4) still complained about the font size (which had been increased after the pilot study);
- Two participants suggested improvements on the navigation between items. Particularly, one participant stated that he/she "was hoping to find a 'starting point' perspective for the application (i.e., a 'Home' perspective). Having only the separate perspectives hindered understanding the purpose of each of them";
- One participant suggested "a qualification or rating of assets (1 to 5 stars, for example) from the opinion of the developers, as well as other attributes inferred by the experience of use" (which meets some ideas from [Caldiera & Basili 1991] that are not yet fully put into practice);
- One participant suggested to "improve the legend color scheme of the issues bubble charts", which in fact was interpreted differently by several participants;
- Creating additional filters (for instance, for the reuse map visualization) (2 participants).

With the exception of one participant (a SD)[68], none of the participants stated that they would change their answer regarding reuse-related information. This may be because their answers were not available for them when this question was asked, or because they believe their answers indeed should not change. However, most participants who stated that some information are of little of no relevance ended up mentioning the use of such information for taking a decision in the execution of tasks.

5.3.3.5 Considerations regarding the study

The study research questions listed in Section 5.3.1.1 are answered in Table 5.6.

Table 5.6 – Analysis of the Zooming Browser study research questions

Study Research Questions	Variables	Study results
SRQ1. Is the information provided by Zooming Browser useful for supporting the execution of reuse tasks?	• Perceived usefulness of the presented information	In the follow-up questionnaire, participants confirmed that the information presented by the tool was highly useful (as depicted in Figure 5.12). Besides, in the relevance questionnaire, the majority of information was pointed out as relevant (all of them – even the less relevant ones – were considered quite relevant for at least one participant). Finally, although only one participant stated that he/she would change his/her answer regarding reuse-related information, most participants ended up mentioning the use of such information for taking a decision in the execution of tasks.
SRQ2. Are the visualizations and interaction resources employed in Zooming Browser useful for supporting the execution of reuse tasks?	• Perceived usefulness of the presented visualizations • Perceived usefulness of the employed interaction resources	The usefulness of the presented visualizations was the most well evaluated aspect, with very positive feedback as one of the benefits of the tool. All participants, except one, considered the interaction resources highly relevant.
SRQ3. By performing the reuse tasks through Zooming Browser, is the efficacy considered satisfactory?	• Efficacy (Precision and Recall)	The efficacy variable was smaller than 1 in 3 RM tasks and 5 SD tasks (given that two of these tasks were common to SDs and RMs). In one RM task and two SD tasks, the efficacy was greater than 0.9, while in the other three tasks it stayed between 0.6 and 0.7. For all the other 26 tasks, all participants achieved total efficacy.

[68] This participant stated that he/she would increase the relevance of information about *asset consumers* and *projects in which the asset was reused*. One of them was previously considered *somewhat irrelevant*.

Study Research Questions	Variables	Study results
SRQ4. By performing the reuse tasks through Zooming Browser, is the efficiency considered satisfactory?	• Efficiency • Perceived efficiency of the tool	The quantitative results of the efficiency variable did not support to draw any conclusions, since values varied considerably in the same question. It only allowed making some observations. The perceived efficiency was considered very high by 6 participants, high by 4 participants, and medium by 2 participants (mostly due to the lack of familiarity with the tool).
SRQ5. To which extent Zooming Browser eases the execution of the reuse tasks?	• Perceived easiness in performing the tasks • Perceived difficulties identified in performing the tasks	Eight participants considered the easiness as at least high, while four considered it medium. Most of the difficulties were due to the lack of familiarity with the tool, as pointed out by many participants. Other difficulties are related to some interaction issues, in spite of the relevance of the interaction resources.

Thus, to sum up, it is believed that the main study research question (*Is the use of Zooming Browser feasible in supporting the execution of reuse tasks?*) can be answered positively, based on the provided information and assuming that the familiarity with the tool increases its benefits. However, some aspects need improvements for increasing the support for reuse managers and software developers.

Regarding the potential threats to the study validity listed in Table 5.5, no problems with the think-aloud protocol were identified by the researcher. Besides, participants agreed that the scenario setting represents to some extent the reality of software development scenarios (although it is known that an evaluation in a real setting can provide concrete evidence to such statement). Since participants were selected by convenience sample, results cannot be generalized; new evaluations are necessary in this regard. There was only one interruption among all the experiment sessions, which did not seem to impact the study – the participant kept performing the task and achieved good results. Finally, one participant seemed a little nervous at the beginning, but no great impact on his/her performance was observed.

A threat that was not considered previously is the fact that the notebook in use for the experiments had problems with its battery charger in two SD sessions. This was noticed before the conduction of one session, and it required the researcher to "hold" the charger, which may have disturbed the participants. The analysis of their data does not allow drawing any parallel in this regard, so it is not possible to know the influence of this event in the experiment, if any. The charger was replaced in the remaining sessions.

5.4 Final Remarks

The evaluations performed with some elements of APPRAiSER provided positive evidence on their use and important consideration for their improvement. Results also provide positive evidence to RQ3, stated in Section 1.3.

Participants of the evaluation of the visualization feature model pointed out opportunities for improving the description and organization of features, which will be taken into consideration in future works. This is also an opportunity for collaborative research on the topic. Besides, most participants believe the model has the potential of organizing and structuring the knowledge related to visualization and interaction.

Through the evaluation of Zooming Browser, participants stated (and demonstrated though the execution of tasks) that the tool can help raising reuse awareness in software organizations. The major contribution of Zooming Browser is the use of visualizations for presenting reuse-related information – in fact, according to the evaluation data, the visualizations were the top rated aspect of the tool.

The performed study with Zooming Browser was also a source of several opportunities for improvement that will be considered in future work. Some include the incorporation of additional interaction resources in some views (e.g., filtering), the redesign or combination of some views that required many interactions for answering a question (e.g., including a summarization view with aggregated information that could only be found individually through tooltips), the creation of a "Home" perspective displaying all available perspectives in a nutshell, and so on.

Other aspects of the APPRAiSER contributions were not evaluated in the scope of this thesis. For instance, a proper evaluation of the proposed mapping structure (used for the design of Zooming Browser and presented in Section 4.6) would require its application by other researchers in the development of other visualization tools. The integration between CAVE and Zooming Browser would require the setting of more complex scenarios, which was not feasible for the context of this thesis.

CHAPTER 6 – CONCLUSION

This chapter presents the contributions and results obtained from this work, as well as a research agenda for handling open questions and opportunities for improvement.

6.1 Epilogue

Software reuse provides several benefits throughout the software development process, such as the decrease of implementation efforts, the reduction on time-to-market, and the amortization of test and inspection costs, favoring an increase of quality. Nevertheless, organizations still find difficulties in implementing reuse due to several reasons, including technical and non-technical aspects. It can be noticed, though, that many of these difficulties, if not most of them, are recurring throughout the years, as illustrated in the study with Brazilian organizations, presented in Chapter 2.

A comprehensive study (described in Chapter 3) was conducted for identifying software visualization approaches targeted to reuse-related tasks. Results pointed out that no work addressed a number of reuse tasks in an integrated way, and the existing ones that address particular tasks are limited in terms of collecting information from different data sources and lack support for reuse management. Besides, most of them do not provide properly evaluated evidence on their effectiveness.

To this end, APPRAiSER was defined, (described in Chapter 4) encompassing interactive visualization tools for assisting stakeholders (mainly reuse managers and developers) in executing software reuse tasks, such as obtaining and understanding information regarding assets, developers, and projects, and being aware of reuse initiatives. The realization of APPRAiSER was achieved through its tools, which incorporate elements for gathering, processing and visually presenting information that is relevant for software reuse. APPRAiSER also contains conceptual elements to understand visualization concepts and support the construction of visualization tools.

The evaluations performed (Chapter 5) showed that the integrated APPRAiSER tools have the potential to enhance software reuse tasks and awareness, while the visualization feature model is promising for organizing visualization knowledge.

In summary, this thesis (i) revealed some recurring software reuse issues in some Brazilian organizations, (ii) pointed out drawbacks and gaps in current visualization tools for supporting reuse tasks, (iii) presented an approach composed by tools for collecting and presenting data related to the reuse scenario, raising awareness on information that can be used for taking informed reuse decisions, and (iv) provided initial evidence on the use of such tools. In addition, the approach presented in this thesis also comprises conceptual elements for engineering interactive visualization tools, to be used in the evolution of APPRAiSER and in other fields of research.

6.2 Contributions and Results

The research and work described in this thesis has the following contributions:

- *A primary study on issues related to software reuse in some Brazilian organizations* (Section 2.5.2): The results from the semi-structured interviews conducted with Brazilian implementers and assessors of MR-MPS-SW allowed the definition of some reuse tasks that need more support. This can be used not only for other research initiatives, but also for the development of additional tool support for the implementation of reuse processes in software organizations.
- *A secondary study on visualization approaches geared to software reuse* (Section 3.3.3): The results from the *quasi*-systematic review, available in a website [Schots 2014c], can be used as a starting point for future research directions to be addressed by the software engineering community, as well as for other secondary studies correlating visualization with another software engineering field of interest. Besides, the presented information can be used as a body of knowledge to support the decision making regarding the choice of visualization approaches for software reuse.
- *The implementation of the APPRAiSER tools* (described in Chapter 4 and its sections): Zooming Browser and Repository Miner, both contributions from the author of this thesis, help obtaining and visualizing pertinent information about assets, developers, and projects. The collected information can also be used for the construction of other visualization tools and for performing a variety of studies and analyses in the software engineering field, especially on software reuse. The visualization perspectives can also be adapted to represent other kinds of reusable content, not limited to software assets. Finally, the flexible architecture of APPRAiSER allows adapting Zooming Browser and Repository Miner to handle different kinds of information/visualizations.

- *The APPRAiSER conceptual elements (namely the visualization feature model and the mapping structure)* (Sections 4.5 and 4.6): These elements can be used for supporting the selection of features and the construction of other visualization tools, and are intended to be improved along time based on other research results.
- *The conducted evaluation studies* (described in Sections 5.2 and 5.3): The evaluations provide evidence on the usefulness of Zooming Browser and the visualization feature model. They also pointed out suggestions for improving APPRAiSER. It is believed that the presented details of the studies allow replication and adaptation to other studies related to visualization in software development.

Although not considered as a contribution, the framework extension for categorizing visualization approaches (presented in Section 3.3.2 and in [Schots & Werner 2014b]), with the two new dimensions and the definition of research questions for all dimensions, can help the planning and construction of novel approaches, besides indicating information that should ideally be described in publications. Such framework may also support the conduction of other secondary studies on software visualization applied to another field of interest.

Regarding the awareness and comprehension challenges listed in Section 3.2.3, the listed contributions aim to address the following ones in the context of this work:

- The semi-structured interviews, as well as the experiments on industry, represent a step towards *understanding the real needs of the software development industry stakeholders in terms of awareness and comprehension*, and *bridging the gap and encouraging interaction between academia and industry*;
- In terms of *identifying and developing suitable mechanisms and adequate abstractions* and *building specialized, personalized/customizable visualizations according to the comprehension needs*, APPRAiSER and its tools provide abstractions that were considered (in general) useful in supporting reuse tasks;
- Repository Miner and the survey performed in the context of the evaluation of Zooming Browser allowed *identifying relevant data*, and the integration between Zooming Browser and Repository Miner is a step towards *evaluating the quality of existing data sources* in future works.

6.2.1 Research achievements

The conduction of this research allowed the following research achievements:

- Publications and research projects related to the APPRAiSER tools:
  - *CAVE* is part of a M.Sc. thesis at COPPE/UFRJ developed in the context of this work [Vasconcelos 2015] informally co-supervised by the author of this thesis; some preliminary results focusing on the use of a context model for supporting context-aware visualizations are described in [Vasconcelos et al. 2013] and [Vasconcelos et al. 2014b];
  - *VCS Miner* uses an infrastructure previously developed [Werner et al. 2011] [Silva 2012] in collaboration with the author of this thesis. The GitHub integration was developed by the author of this thesis;
  - *Reuse Repository Miner* was inspired by MPS-Reuse[69] [Chaves 2013], an Undergraduate Final Project at UERJ advised by the author of this thesis;
  - *Issue Tracker Miner* was produced in the context of an Undergraduate Research at COPPE/UFRJ [Queiroz et al. 2012] and extended by [Vasconcelos 2015] to export Redmine data in the JSON format. The GitHub integration was developed by the author of this thesis;
  - *GraphVCS* (which partially inspired the History perspective) was developed in the context of an Undergraduate Final Project at UERJ advised by the author of this thesis. It was presented as an ongoing work in [Pereira & Schots 2011] and as a complete work in [Pereira & Schots 2014];
  - *ReuseDashboard* (which partially inspired the Zooming Browser's Dashboard perspective) is described in [Palmieri et al. 2013], in the context of a M.Sc. thesis proposal informally co-supervised by the author of this thesis;
  - *Rec4Reuse*, a tool that composed APPRAiSER's original proposal, was developed in the context of an Undergraduate Final Project [Vital & Krause 2013] at UERJ advised by the author of this thesis.
- Publications and research projects related to the APPRAiSER conceptual elements:
  - Details on the construction of the visualization feature model are described in [Vasconcelos et al. 2014a] and in a technical report to be published [Schots et al. 2015]. The original idea of using feature models for supporting information visualization comes from a previous work [Silva 2012] [Silva et al. 2012] conducted under informal co-supervision of the author of this thesis;

[69] MPS-Reuse was developed in another technology and did not contain visualization resources, so it only served as inspiration.

- The extension and usage of the task-oriented framework to characterize visualization approaches was presented in [Schots & Werner 2014b];
- The mapping structure that correlates goals and visualizations was presented in [Schots & Werner 2015].

- Research on software visualization:
  - The research on software visualization allowed the identification of awareness and comprehension challenges in a special track of the Brazilian Symposium on Software Engineering (SBES) [Schots et al. 2012];
  - Furthermore, an introductory tutorial in software visualization was presented at the Brazilian Conference on Software (CBSoft) [Schots & Werner 2012].
- Performed studies:
  - The preliminary results from the semi-structured interviews are described in [Schots & Werner 2013]. A technical report with all the details is available at [Schots & Werner 2014a];
  - A preliminary study on mapping visualizations according to the focus of representation/analysis was developed in the context of an Undergraduate Research [Queiroz et al. 2013] and included in a technical report [Schots et al. 2015];
  - The full protocol and the results from the *quasi*-systematic review are described in a technical report [Schots et al. 2014] and in a website [Schots 2014c] built for this purpose.
- Research proposal:
  - The research proposal of this thesis [Schots 2014] was presented at the Doctoral Symposium of the 36th International Conference on Software Engineering.
- Other works not directly related to the scope of this research:
  - The collaboration between members of the Reuse group and the Experimental Software Engineering group resulted in a publication [Mello et al. 2014] that was used as basis for the evaluation of the visualization feature model.

6.3 Open Questions and Research Agenda

The research conducted during the Ph.D. course, along with the feedback received from some submitted papers, provided input for building an initial research agenda. This will also allow for the improvements expected in the last step of the

research methodology presented in Section 1.5 towards engineering interactive visualization tools for providing awareness in software reuse tasks.

The APPRAiSER elements will be evolved for improving their contributions, subject to further evaluations. The visualization feature model can be improved in collaboration with experts in the field, based on the currently identified information [Schots et al. 2015] along with the corrections and suggestions made by the participants of the study. With the help of an intuitive user interface (e.g., a visual catalog) that explains each feature, its constraints, and so on, it is possible to evaluate how it can contribute to the understanding of visualization concepts. As one participant of the feature model study pointed out, some improvements can be made to make it adequate to help education and training in information visualization.

The visualization feature model can also be evaluated by experts in feature modeling (through the complete checklist [Mello et al. 2014]) for improving the results. Besides, its use in combination with the mapping structure for choosing visualization features can be evaluated in scenarios that demand the construction of visualization tools (not necessarily related to software). The use of visualization resources is continuously growing in the industry, so there may be opportunities in this scenario. Another possibility is the integration with the extended version of the task-oriented framework.

The relevance and usefulness of the two novel dimensions of such task-oriented framework (*requirements* and *evidence*, presented in Section 3.3.2) were discussed in [Schots & Werner 2014b], and this version was used for categorizing the results of the secondary study presented in Section 3.3.3. Other studies on information/software visualization might benefit from this extension, but an evaluation of this hypothesis requires its practical use by other researchers. This also helps improving the framework.

Regarding the APPRAiSER tools, the use of Repository Miner for obtaining other kinds of information from other relevant sources can also be explored. Besides, based on the extracted data, the use of data mining techniques can help extracting unnoticed facts, as well as creating clusters of core elements to ease the navigation.

The detection strategies of Repository Miner can be improved through the use and application of search-based algorithms and techniques. Such techniques can also be used to provide suggestions and recommendations of assets.

APPRAiSER can also benefit from the creation of recommendation systems [Robillard et al. 2010] that take into account consumers' usage data of assets in projects

and producer's contribution data, combined with their development profile. This can provide interesting suggestions, for instance, for project team allocation or collaborations in asset development.

Finally, the amount and variety of available software-related information has opened opportunities for investigating the use of software engineering in Big Data, and vice-versa. Improvements in the collection and storage mechanisms can help managing data from software repositories with different focuses, supporting awareness and decision-making in several kinds of software development activities.

REFERENCES

[ABES 2014] ABES (2014). “Brazilian Software Market: Scenario and Trends”. Associação Brasileira das Empresas de Software, 1st ed., São Paulo, June. ISBN 978-8586700-03-3. Available at http://www.abessoftware.com.br/dados-do-setor/.

[Agrawala et al. 2011] Agrawala, M., Li, W., Berthouzoz, F. (2011). “Design principles for visual communication”. *Communications of the ACM*, v. 54, n. 4, pp. 60-69, April.

[Alonso & Frakes 2000] Alonso, O., Frakes, W. B. (2000). “Visualization of Reusable Software Assets”. In: *6th International Conference on Software Reuse (ICSR 2000)*, Vienna, Austria, pp. 251-265, June.

[Anslow et al. 2004] Anslow, C., Marshall, S., Noble, J., Biddle, R. (2004). “Software visualization tools for component reuse”. In: *2nd Workshop on Method Engineering for Object-Oriented and Component-Based Development, 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004)*, Vancouver, Canada, pp. 1-11, October .

[Baldauf et al. 2007] Baldauf, M., Dustdar, S., Rosenberg, F. (2007). “A survey on context-aware systems”. *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, v. 2, n. 4, pp. 263-277.

[Basili et al. 1994] Basili, V., Caldiera, G., Rombach, H. (1994). “Goal Question Metric Paradigm”, *Encyclopedia of Software Engineering*, v. 1, edited by John J. Marciniak, John Wiley & Sons, pp. 528-532.

[Bauer et al. 2014] Bauer, V., Eckhardt, J., Hauptmann, B., Klimek, M. (2014). “An Exploratory Study on Reuse at Google”. In: *1st International Workshop on Software Engineering Research and Industrial Practices (SER&IPs 2014)*, Hyderabad, India, pp. 14-23, June.

[Beck et al. 2013] Beck, F., Burch, M., Diehl, S. (2013). “Matching application requirements with dynamic graph visualization profiles”. In: *17th International Conference on Information Visualisation (IV)*, London, UK, pp. 11-18, July.

[Beck et al. 2016] Beck, F., Koch, S., Weiskopf, D. (2016). “Visual analysis and dissemination of scientific literature collections with SurVis”. *IEEE Transactions on Visualization and Computer Graphics*, v. 22, n. 1, January (*in press*).

[Benedicenti et al. 1996] Benedicenti, L., Succi, G., Valerio, A., Vernazza, T. (1996). “Monitoring the efficiency of a reuse program”. *SIGAPP Applied Computing Review*, v. 4, n. 2, pp. 8-14, September.

[Biddle et al. 1999] Biddle, R., Marshall, S., Miller-Williams, J., Tempero, E. (1999). “Reuse of debuggers for visualization of reuse”. In: *Proceedings of the 5th Symposium on Software Reusability (SSR 1999)*, Los Angeles, USA, pp. 92-100, May.

[Blois et al. 2006] Blois, A. P. T. B., Oliveira, R. F., Maia, N., Werner, C., Becker, K. (2006). “Variability modeling in a component-based domain engineering process”. *Reuse of Off-the-Shelf Components*, pp. 395-398. Springer Berlin Heidelberg.

[Bostock et al. 2011] Bostock, M., Ogievetsky, V., Heer, J. (2011). D^3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, v. 17, n. 12, pp. 2301-2309, December.

[Braga et al. 1999] Braga, R. M., Werner, C. M. L., Mattoso, M. (1999). “Odyssey: A reuse environment based on domain models”. In: *2nd IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET 1999)*, Richardson, USA, pp. 50-57, March.

[Braga et al. 2006] Braga, R. M. M., Werner, C. M. L., Mattoso, M. (2006). “Odyssey-Search: A Multi-Agent System for Component Information Search and Retrieval”. *Journal of Systems and Software*, v. 79, n. 2, pp. 204-215, February.

[Brazilian Computer Society 2006] Brazilian Computer Society (2006). “Grand Challenges of Computing Research in Brazil – 2006-2016”. Available at http://www.sbc.org.br/.

[Brooks Jr. 1987] Brooks, Frederick P. (1987). “No Silver Bullet: Essence and Accidents of Software Engineering”. *Computer*, v. 20, n. 4, pp. 10-19, April.

[Buering et al. 2006] Buering, T., Gerken, J., Reiterer, H. (2006). “User Interaction with Scatterplots on Small Screens – A Comparative Evaluation of Geometric-Semantic

Zoom and Fisheye Distortion". *IEEE Transactions on Visualization and Computer Graphics*, v. 12, n. 5, pp. 829-836, September.

[Caldiera & Basili 1991] Caldiera, G., Basili, V. R. (1991). "Identifying and qualifying reusable software components". *Computer*, v. 24, n. 2, pp. 61-70, February.

[Card et al. 1999] Card, S. K., Mackinlay, J. D., Shneiderman, B. (1999). *Readings in information visualization: using vision to think*. 1st ed., Morgan Kaufmann Publishers, San Francisco, USA, 712p.

[Card & Comer 1994] Card, D., Comer, E. (1994). "Why do so many reuse programs fail?" *IEEE Software*, v. 11, n. 5, pp. 114-115, September.

[Carneiro et al. 2010] Carneiro, G. F., Sant'Anna, C., Mendonça, M. (2010). "On the Design of a Multi-Perspective Visualization Environment to Enhance Software Comprehension Activities". In: *7th Workshop on Modern Software Maintenance (WMSWM)*, Belém, Brazil, pp. 61-68, June.

[Chaves 2013] Chaves, V. B. C. (2013. "MPS-Reuse: A tool for supporting the execution of reuse management tasks and reusable assets management" [MPS-Reuse: Uma ferramenta de apoio à execução de tarefas de gerência de reutilização e à gestão de ativos reutilizáveis] (in Portuguese). Undergraduate Final Project, Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, Brazil.

[Chen 2006] Chen, C. (2006). *Information Visualization: Beyond the Horizon*. 2nd ed. Springer.

[Chi 2000] Chi, E. H. H. (2000). "A taxonomy of visualization techniques using the data state reference model". In: *IEEE Symposium on Information Visualization (InfoVis 2000)*, Salt Lake City, USA, pp. 69-75, October.

[Clements & Northrop 2002] Clements, P., Northrop, L. (2002). *Software product lines*. Addison-Wesley, Boston.

[CMMI Product Team 2010] CMMI Product Team (2010). "CMMI for Development, Version 1.3", Technical Report CMU/SEI-2010-TR-033, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA. Available at http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm.

[Cockburn et al. 2008] Cockburn, A., Karlson, A., Bederson, B. B. (2008). “A Review of Overview+Detail, Zooming, and Focus+Context Interfaces”. *ACM Computing Surveys*, v. 41, n. 1, pp. 1-31, December.

[Constantopoulos et al. 1995] Constantopoulos, P., Jarke, M., Mylopoulos, J., Vassiliou, Y. (1995). “The software information base: A server for reuse”. *The VLDB Journal*, v. 4, n. 1, pp. 1-43.

[Cooper et al. 2009] Cooper, J. R., Lee, S.-W., Gandhi, R. A., Gotel, O. (2009). “Requirements Engineering Visualization: A Survey on the State-of-the-Art”. In: *4th International Workshop on Requirements Engineering Visualization (REV 2009)*, Atlanta, USA, pp. 46-55.

[Craft & Cairns 2005] Craft, B., Cairns, P. (2005). “Beyond guidelines: what can we learn from the visual information seeking mantra?” In: *Proceedings of the 9th International Conference on Information Visualization (IV 2005)*, London, UK, pp. 110-118, July.

[Diehl 2007] Diehl, S. (2007). *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*, 1st ed., Springer-Verlag Heidelberg New York.

[Dourish & Bellotti 1992] Dourish, P., Bellotti, V. (1992). “Awareness and Coordination in Shared Workspaces”. In: *1992 ACM Conference on Computer-Supported Cooperative Work (CSCW 1992)*, Toronto, Canada, pp. 107-114, November.

[Duru et al. 2013] Duru, H. A., Çakır, M. P., İşler, V. (2013). “How Does Software Visualization Contribute to Software Comprehension? A Grounded Theory Approach”. *International Journal of Human-Computer Interaction*, v. 29, n. 11, pp. 743-763, November.

[Duszynski et al. 2011] Duszynski, S., Knodel, J. Becker, M (2011). “Analyzing the source code of multiple software variants for reuse potential”. In: *Proceedings of the 18th Working Conference on Reverse Engineering (WCRE 2011)*, Limerick, Ireland, pp. 303-307, October.

[Falconer et al. 2009] Falconer, S. M., Bull, R. I., Grammel, L., Storey, M.-A. (2009). “Creating Visualizations through Ontology Mapping”. In: *International Conference*

on Complex, Intelligent and Software-Intensive Systems (CISIS), Fukuoka, Japan, pp.688-693, March.

[Feigenspan et al. 2013] Feigenspan, J., Kästner, C., Apel, S., Liebig, J., Schulze, M., Dachselt, R., Papendieck, M., Leich, T., Saake, G. (2013). "Do background colors improve program comprehension in the #ifdef hell?" *Empirical Software Engineering*, v. 18, n. 4, pp. 699-745, February.

[Fernandes et al. 2011] Fernandes, P., Werner, C., Teixeira, E. (2011). "An Approach for Feature Modeling of Context-Aware Software Product Line". *Journal of Universal Computer Science*, v. 17, n. 5, pp. 807-829, March.

[Few 2009] Few, S. (2009). *Now You See it: Simple Visualization Techniques for Quantitative Analysis*, 1st ed., Analytics Press, 327p.

[Fielding 2000] Fielding, R. T. (2000). "Architectural styles and the design of network-based software architectures", Ph.D. Thesis, University of California, Irvine, USA.

[Fowler & Highsmith 2001] Fowler, M., Highsmith, J. (2001). "The agile manifesto". *Software Development*, v. 9, n. 8, pp. 28-35.

[Frakes & Fox 1995] Frakes, W. B., Fox, C. J. (1995). "Sixteen questions about software reuse". *Communications of the ACM*, v. 38, n. 6, pp. 75-87, June.

[Frakes & Fox 1996] Frakes, W., Fox, C. (1996). "Quality Improvement Using a Software Reuse Failure Modes Model". *IEEE Transactions on Software Engineering*, v. 22, n. 4, pp. 274-279, April.

[Frakes & Kang 2005] Frakes, W. B., Kang, K. (2005). "Software reuse research: Status and future". *IEEE Transactions on Software Engineering*, v. 31, n. 7, pp. 529-536.

[França & Travassos 2013] França, B. B. N., Travassos, G. H. (2013). "Are We Prepared for Simulation Based Studies in Software Engineering Yet?" *CLEI Electronic Journal*, v. 16, n. 1, paper 8, pp. 1-25, April.

[Franke & Kaul 1978] Franke, R. H., Kaul, J. D. (1978). "The Hawthorne experiments: First statistical interpretation". *American sociological review*, v. 43, n. 5, pp. 623-643, October.

[Fritz & Murphy 2010] Fritz, T., Murphy, G. C. (2010). "Using information fragments to answer the questions developers ask". In: *32nd ACM/IEEE International*

Conference on Software Engineering (ICSE 2010), Cape Town, South Africa, pp. 175-184, May.

[Gallagher et al. 2008] Gallagher, K., Hatch, A., Munro, M. (2008), "Software Architecture Visualization: An Evaluation Framework and its Application", *IEEE Transactions on Software Engineering*, v. 34, n. 2, pp. 260-270.

[Gill 2006] Gill, N. S. (2006). "Importance of Software Component Characterization for Better Software Reusability". *SIGSOFT Software Engineering Notes*, v. 31, n. 1, pp. 1-3, January.

[Gousios et al. 2014] Gousios, G., Vasilescu, B., Serebrenik, A., Zaidman, A. (2014). "Lean GHTorrent: GitHub data on demand". In: *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014)*, Hyderabad, India, pp. 384-387, May.

[Griss et al. 1994] Griss, M. L., Favaro, J., Walton, P. (1994). "Managerial and organizational issues – Starting and Running a Software Reuse Program". In: Schäfer, W., Prieto-Díaz, R., Matsumoto, M. (eds.), *Software Reusability*, pp. 51-78, Ellis Horwood Ltd.

[Haefliger et al. 2008] Haefliger, S., von Krogh, G., Spaeth, S. (2008). "Code Reuse in Open Source Software". *Management Science*, v. 54, n. 1, pp. 180-193, January.

[Hattori 2010] Hattori, L. (2010). "Enhancing collaboration of multi-developer projects with synchronous changes". In: *32nd ACM/IEEE International Conference on Software Engineering*, Cape Town, South Africa, pp. 377-380, May.

[Holmes 2008] Holmes, R. (2008). "Pragmatic software reuse". Ph.D. Thesis, University of Calgary, Calgary, Canada, November.

[Holmes & Walker 2012] Holmes, R., Walker, R. J. (2012). "Systematizing pragmatic software reuse". *ACM Transactions on Software Engineering and Methodology (TOSEM 2012)*, v. 21, n. 4, November.

[Hooper & Chester 1991] Hooper, J. W., Chester, R. O. (1991). "Software Reuse: Guidelines and Methods", Springer, 180p.

[Hove & Anda 2005] Hove, S. E., Anda, B. (2005). "Experiences from conducting semi-structured interviews in empirical software engineering research". In: *11th IEEE International Software Metrics Symposium*, Como, Italy, pp. 1-10, September.

[IEEE 2010] IEEE (2010). “IEEE Std. 1517-2010: IEEE Standard for Information Technology – System and Software Life Cycle Processes – Reuse Processes”, 39p, Institute of Electrical and Electronics Engineers.

[INFOVIS WIKI 2015] The InfoVis:Wiki Team (2015). “The InfoVis:Wiki project”. Available at http://www.infovis-wiki.net/.

[ISO/IEC 2008] ISO/IEC (2008). “ISO/IEC 12207:2008 – Systems and software engineering – Software life cycle processes”, 123p, International Organization for Standardization and the International Electrotechnical Commission, Geneva, Switzerland.

[ISO/IEC 2012] ISO/IEC (2012). “ISO/IEC 15504-5:2012 – Process assessment, Part 5: An exemplar software life cycle process assessment model”, 196p, International Organization for Standardization and the International Electrotechnical Commission, Geneva, Switzerland.

[Kagdi & Maletic 2008] Kagdi, H., Maletic, J. I. (2008). “Expressiveness and effectiveness of program comprehension: Thoughts on future research directions”. In: *Proceedings of the IEEE Frontiers of Software Maintenance (FoSM 2008)*, Beijing, China, pp. 31-37, October.

[Kang et al. 1990] Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., Peterson, A. S. (1990). *Feature-oriented domain analysis (FODA) feasibility study* (No. CMU/SEI-90-TR-21). Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, USA.

[Keim et al. 2008] Keim, D. A., Mansmann, F., Schneidewind, J., Thomas, J., Ziegler, H. (2008). “Visual analytics: Scope and challenges”. In: Simoff, S. J., Böhlen, M. H., Mazeika, A. (eds.), *Visual Data Mining*, pp. 76-90, Springer Berlin Heidelberg.

[Kelleher 2005] Kelleher, J. (2005). “A reusable traceability framework using patterns”. In: *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2005)*, Long Beach, USA, pp. 50-55, November.

[Kim & Stohr 1998] Kim, Y., Stohr, E.A. (1998). “Software reuse: survey and research directions”. *Journal of Management Information Systems*, v. 14, n. 4, pp. 113-147, March.

[Kitchenham 2004] Kitchenham, B. (2004). “Procedures for Performing Systematic Reviews”, Keele University Technical Report TR/SE-0401. Available at http://www.scm.keele.ac.uk/ease/sreview.doc.

[Kitchenham et al. 2007] Kitchenham, B. A., Mendes, E., Travassos, G. H. (2007). “Cross versus Within-Company Cost Estimation Studies: A Systematic Review”, *IEEE Transactions on Software* Engineering, v. 33, n. 5, pp. 316-329, May.

[Kitchenham & Charters 2007] Kitchenham, B., Charters, S. (2007). “Guidelines for performing Systematic Literature Reviews in Software Engineering”. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, July.

[Kitchenham et al. 2009] Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., Linkman, S. (2009). “Systematic literature reviews in software engineering – A systematic literature review”, *Information and Software Technology*, v. 51, n. 1, pp. 7-15, January.

[Klerkx et al. 2006] Klerkx, J., Verbert, K., Duval, E. (2006). “Visualizing Reuse: More than Meets the Eye”. In: *6th International Conference on Knowledge Management (I-KNOW)*, Graz, Austria, pp. 489-497.

[Ko et al. 2007] Ko, A. J., DeLine, R., Venolia, G. (2007). “Information needs in collocated software development teams”. In: *29th International Conference on Software Engineering (ICSE 2007)*, Minneapolis, USA, pp. 344-353, May.

[Koschke 2003] Koschke, R. (2003). “Software visualization in software maintenance, reverse engineering, and re-engineering: a research survey,” *Journal of Software Maintenance and Evolution: Research and Practice*, v. 15, n. 2, pp. 87-109.

[Krueger 1992] Krueger, C. W. (1992). “Software reuse”. *ACM Computing Surveys*, v. 24, n. 2, pp. 131-183, June.

[Kula et al. 2014] Kula, R. G., De Roover, C., German, D., Ishio, T., Inoue, K. (2014). “Visualizing the evolution of systems and their library dependencies”. In: *2nd IEEE Working Conference on Software Visualization (VISSOFT 2014)*, Victoria (BC), Canada, pp. 127-136, September.

[Lab Mouse Security 2014] Lab Mouse Security (2014). “Raising Lazarus – The 20 Year Old Bug that Went to Mars”. Available at http://blog.securitymouse.com/2014/06/raising-lazarus-20-year-old-bug-that.html.

[Lanza & Marinescu 2006] Lanza, M., Marinescu, R. (2006). *Object-Oriented Metrics in Practice*. Springer-Verlag Berlin Heidelberg.

[Leach 2012] Leach, R. J. (2012). Software Reuse: Methods, Models, and Costs, 2nd ed., ISBN-13: 978-0-9853685-1-7, 656p.

[Lee et al. 2002] Lee, K., Kang, K. C., Lee, J. (2002). "Concepts and Guidelines of Feature Modeling for Product Line Software Engineering". In: *Software Reuse: Methods, Techniques, and Tools*, pp. 62-77. Springer Berlin-Heidelberg.

[Lee et al. 2012] Lee, B., Isenberg, P., Riche, N. H., Carpendale, S. (2012). "Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions". *IEEE Transactions on Visualization and Computer Graphics*, v. 18, n. 12, pp. 2689-2698, December.

[Likert 1932] Likert, R. (1932). "A technique for the measurement of attitudes". *Archives of Psychology*, v. 22, n. 140, pp. 1-55.

[Lucrédio et al. 2008] Lucrédio, D., Brito, K. S., Alvaro, A., Garcia, V. C., Almeida, E. S., Fortes, R. P. M., Meira, S. L. (2008). "Software reuse: The Brazilian industry scenario". *Journal of Systems and Software*, v. 81, n. 6, pp. 996-1013, June.

[Maletic et al. 2002] Maletic, J. I., Marcus, A., Collard, M. L. (2002). "A task oriented view of software visualization". In: *Proceedings of the 1st International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2002)*, Paris, France, pp. 32-40, June.

[Mancoridis et al. 1993] Mancoridis, S., Holt, R. C., Penny, D. A. (1993). "Conceptual framework for software development". In: *Proceedings of the 1993 ACM Computer Science Conference*, Indianapolis, USA, pp. 74-80, February.

[Marshall 2001] Marshall, S. (2001). "Using and Visualizing Reusable Code: Position Paper for Software Visualization Workshop". In: *Workshop on Software Visualization, 2001 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2001)*, Tampa, USA, pp. 1-4, October.

[Marshall et al. 2003] Marshall, S. Jackson, K., Anslow, C., Biddle, R. (2003). "Aspects to visualising reusable components". In: *Proceedings of the Australasian Symposium*

on Information Visualisation (InVis.au 2003), Adelaide, Australia, pp. 81-88, February.

[Mello et al. 2014] Mello, R. M., Teixeira, E. N., Schots, M., Werner, C. M. L., Travassos, G. H. (2014). "Verification of Software Product Line Artefacts: A Checklist to Support Feature Model Inspections". *Journal of Universal Computer Science (J.UCS)*, v. 20, n. 5, pp. 720-745.

[Mili et al. 1995] Mili, H., Mili, F., Mili, A. (1995). "Reusing software: Issues and research directions". *IEEE Transactions on Software Engineering*, v. 21. n. 6, pp. 528-562.

[Moore & Bailin 1991] Moore, J. M., Bailin, S. C. (1991). "Domain Analysis: Framework for reuse". In: Prieto-Díaz, R., Arango, G. (eds.), *Domain Analysis and Software System Modeling*, pp. 179-202, IEEE Computer Society Press, Los Alamitos, USA.

[Morisio et al. 2002] Morisio, M., Ezran, M. and Tully, C. (2002). Success and failure factors in software reuse. *IEEE Transactions on Software Engineering*, v. 28, n. 4, pp. 340-357.

[Mukherjea & Foley 1996] Mukherjea, S., Foley, J. (1996). "Requirements and Architecture of an Information Visualization Tool". In: *Database Issues for Data Visualization*, Springer Berlin/Heidelberg, pp. 57-75.

[Naur & Randell 1968] Naur, P., Randell, B. (1968). "Software Engineering", Scientific Affairs Division, NATO, Brussels, Garmisch, Germany, Report on a conference sponsored by the NATO Science Committee, October.

[Novais et al. 2012] Novais, R., Nunes, C., Lima, C., Cirilo, E., Dantas, F., Garcia, A., Mendonça, M. (2012). "On the proactive and interactive visualization for feature evolution comprehension: An industrial investigation". In: *34th International Conference on Software Engineering (ICSE 2012) – Software Engineering in Practice (SEIP) Track*, Zürich, Switzerland, pp. 1044-1053, June.

[Novais et al. 2014] Novais, R., Brito, C., Mendonça, M. (2014). "What Questions Developers Ask During Software Evolution? An Academic Perspective". In: *2nd Workshop on Software Visualization, Evolution, and Maintenance (VEM 2014)*, Maceió, Brazil, pp. 14-21, September.

[Oliveira 2011] Oliveira, M. S. (2011). “PREViA: An Approach for Visualizing the Evolution of Software Models” [PREViA: Uma Abordagem para a Visualização da Evolução de Modelos de Software] (in Portuguese). M.Sc. Thesis, COPPE/UFRJ, Rio de Janeiro, Brazil, March.

[Orso et al. 2000] Orso, A., Harrold, M. J., Rosenblum, D. S. (2000). “Component Metadata for Software Engineering Tasks”, *2nd International Workshop on Engineering Distributed Objects*, Davis, USA, pp. 126-140, November.

[Ouyang et al. 2014] Ouyang, Y., Zeng, S., Yang, C., Wang, Q. (2014). “Improving guide-based vulnerability detection with hybrid symbolic execution”. In: *2nd International Conference on Systems and Informatics (ICSAI 2014)*, Shanghai, China, pp. 1038-1043, November.

[Palmieri et al. 2013] Palmieri, M., Schots, M., Werner, C. (2013). “ReuseDashboard: Supporting Stakeholders in Monitoring Software Reuse Programs” [ReuseDashboard: Apoiando Stakeholders na Monitoração de Programas de Reutilização de Software] (in Portuguese). In: *1st Brazilian Workshop on Software Visualization, Evolution, and Maintenance (VEM 2013)*, Brasília, Brazil, pp. 54-61, September.

[Panichella et al. 2014] Panichella, S., Bavota, G., Di Penta, M., Canfora, G., Antoniol, G. (2014). “How Developers’ Collaborations Identified from Different Sources Tell Us about Code Changes”. In: *30th IEEE International Conference on Software Maintenance and Evolution (ICSME 2014)*, Victoria (BC), Canada, pp. 251-260, September.

[Pereira & Schots 2011] Pereira, T. A., Schots, M. (2011). “GraphVCS: An Approach for Visualizing and Understanding Version Control Repositories” [GraphVCS: Uma Abordagem para a Visualização e Compreensão de Repositórios de Controle de Versão] (in Portuguese). In: *1st Brazilian Workshop on Software Visualization (WBVS 2011)*, São Paulo, Brazil, pp. 1-8.

[Pereira & Schots 2014] Pereira, T. A., Schots, M. (2014). “GraphVCS: A Visualization Approach for Supporting the Understanding of Version Control Repositories” [GraphVCS: Uma Abordagem de Visualização para Apoio à Compreensão de Repositórios de Controle de Versão] (in Portuguese). In: *1st Minas Gerais*

Symposium on Software Engineering (SMES 2011), Belo Horizonte, Brazil, pp. 72-81.

[Pötter et al. 2014] Pötter, H., Schots, M., Duboc, L., Werneck, V. (2014). "InspectorX: A game for software inspection training and learning". In: *27th IEEE Conference on Software Engineering Education and Training (CSEE&T 2014)*, Klagenfurt, Austria, pp. 55-64, April.

[Poulin et al. 1993] Poulin, J. S., Caruso, J. M., Hancock, D. R. (1993). "The Business Case for Software Reuse". *IBM Systems Journal*, v. 32, n. 4, pp. 567-594, October.

[Prieto-Díaz & Arango 1991] Prieto-Díaz, R., Arango, G. F. (1991). *Domain Analysis and Software Systems Modeling*. IEEE Computer Society Press.

[Queiroz et al. 2012] Queiroz, A. R., Oliveira, M. S., Werner, C. M. L. (2012). "Supporting Project Management through Visual Analytics of Scenario Data" [Apoio ao Gerenciamento de Projetos por Meio da Análise Visual de Dados de Cenários]. In: *XXXIV Jornada Giulio Massarani de Iniciação Científica, Tecnológica, Artística e Cultural UFRJ*, pp. 229.

[Queiroz et al. 2013] Queiroz, A. R., Oliveira, M. S., Werner, C. M. L. (2013). "Systematic Support for the Choice of Information Visualizations based on Representation Constraints" [Apoio Sistemático à Escolha de Visualizações de Informação Baseada em Restrições de Representação]. In: *XXXV Jornada Giulio Massarani de Iniciação Científica, Tecnológica, Artística e Cultural UFRJ*, pp. 102.

[Ripley et al. 2007] Ripley, R. M., Sarma, A., Van der Hoek, A. (2007). "A Visualization for Software Project Awareness and Evolution". In: *Proceedings of the 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2007)*, Banff, Canada, pp. 137-144, June.

[Robertson et al. 1989] Robertson, G., Card, S. K., Mackinlay, J. D. (1989). "The cognitive coprocessor architecture for interactive user interfaces". In: *Proceedings of the 2nd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST 1989)*, Williamsburg, USA, pp. 10-18, November.

[Robertson et al. 2009] Robertson, G., Ebert, D., Eick, S., Keim, D., Joy, K. (2009). "Scale and complexity in visual analytics". *Information Visualization*, v. 8, n. 4, pp. 247-253, July.

[Robillard et al. 2010] Robillard, M., Walker, R., Zimmermann, T. (2010). "Recommendation System for Software Engineering". *IEEE Software*, v. 27, n. 4, pp. 80-86, July.

[Rocha et al. 2007] Rocha, A. R. C., Montoni, M., Weber, K. C., Araujo, E. E. R. (2007). "A Nationwide Program for Software Process Improvement in Brazil". In: *6th International Conference on Quality of Information and Communications Technology (QUATIC 2007)*, Lisbon, Portugal, pp. 167-176, September.

[Rodrigues & Werner 2011] Rodrigues, C. S. C., Werner, C. M. L. (2011). "Making the comprehension of software architecture attractive". In: *24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2011)*, Honolulu, Hawaii, pp. 416-420.

[Sá et al. 1997] Sá, M. L. B., Werner, C. M. L., Goldman, I. (1997). "Introduction of Reuse in a Brazilian Enterprise on Software Production" [Introdução da Reutilização em uma Empresa Brasileira de Produção de Software] (in Portuguese). In: *11th Brazilian Symposium on Software Engineering (SBES 1997)*, Fortaleza, Brazil, pp. 233-248, October.

[Sametinger 1997] Sametinger, J. (1997). *Software Engineering with Reusable Components*. Ed. 1997. Springer-Berlin, 288p.

[Santa Isabel 2011] Santa Isabel, S. L. (2011). "Selection of Testing Approaches for Web Applications" [Seleção de Abordagens de Teste para Aplicações Web] (in Portuguese). M.Sc. Thesis, COPPE/UFRJ, Rio de Janeiro, Brazil, July.

[Santos et al. 2009] Santos, G., Zanetti, D., Maciel, M., Simões, C. A., Werner, C., Rocha, A. R. (2009). "The Experience of the Implementation of Reuse Management and Development for Reuse Processes in Synapsis-Brazil" [A Experiência de Implantação dos Processos Gerência de Reutilização e Desenvolvimento para Reutilização na Synapsis-Brasil] (in Portuguese). In: *Proceedings of the 5th Annual Workshop of MPS (WAMPS)*, Campinas, Brazil, pp. 128-135, October.

[Schots et al. 2010] Schots, M., Silva, M. A., Murta, L. G. P., Werner, C. M. L. (2010). "Adherence Checking between Conceptual and Emerging Architectures by Using the PREViA Approach" [Verificação de Aderência entre Arquiteturas Conceituais e Emergentes Utilizando a Abordagem PREViA] (in Portuguese). In: *7th Brazilian*

Workshop on Modern Software Maintenance (WMSWM 2010), Belém, Brazil, pp 1-8, June.

[Schots & Werner 2012] Schots, M., Werner, C. (2012). "Exploiting the Intangible: An Overview of Software Visualization and its Applications" [Explorando o Intangível: Um Panorama da Visualização de Software e suas Aplicações] (in Portuguese). Tutorial. In: *III Brazilian Congress on Software: Theory and Practice (CBSoft 2012)*, Natal, Brazil.

[Schots et al. 2012] Schots, M., Werner, C., Mendonça, M. (2012). "Awareness and Comprehension in Software/Systems Engineering Practice and Education: Trends and Research Directions". In: *26th Brazilian Symposium on Software Engineering (SBES)*, Natal, Brazil, pp. 186-190.

[Schots & Werner 2013] Schots, M., Werner, C. (2013). "Characterizing the Implementation of MR-MPS-SW Reuse Processes: Preliminary Results" [Caracterizando a Implementação de Processos de Reutilização do MR-MPS-SW: Resultados Preliminares]. In: *Proceedings of the 9th Annual Workshop of MPS (WAMPS 2013)*, Campinas, Brazil, pp. 44-53, October.

[Schots 2014a] Schots, M. (2014a). "On the Use of Visualization for Supporting Software Reuse". Qualifying Examination, COPPE/UFRJ, Rio de Janeiro, Brazil, April.

[Schots 2014b] Schots, M. (2014b). "On the Use of Visualization for Supporting Software Reuse". In: *36th International Conference on Software Engineering (ICSE 2014), Doctoral Symposium*, Hyderabad, India, pp. 694-697, June.

[Schots 2014c] Schots, M. (2014c). "How do visualization approaches support software reuse tasks?" Available at http://www.cos.ufrj.br/~schots/survis_reuse/.

[Schots et al. 2014] Schots, M., Vasconcelos, R., Werner, C. (2014). "A *Quasi*-Systematic Review on Software Visualization Approaches for Software Reuse", Technical Report ES-748/14, COPPE/UFRJ, Rio de Janeiro, Brazil, June.

[Schots & Werner 2014a] Schots, M., Werner, C. (2014a). "Characterizing the Implementation of Software Reuse Processes in Brazilian Organizations", Technical Report ES-749/14, COPPE/UFRJ, Rio de Janeiro, Brazil, August.

[Schots & Werner 2014b] Schots, M., Werner, C. (2014b). “Using a Task-Oriented Framework to Characterize Visualization Approaches”. In: *2nd IEEE Working Conference on Software Visualization (VISSOFT 2014) – New Ideas and Emerging Results (NIER) track*, Victoria (BC), Canada, pp. 70-74, September.

[Schots & Werner 2015] Schots, M., Werner, C. (2015). “On Mapping Goals and Visualizations: Towards Identifying and Addressing Information Needs”. In: *3rd Workshop on Software Visualization, Evolution, and Maintenance (VEM 2015)*, Belo Horizonte, Brazil, pp. 25-32, September.

[Schots et al. 2015] Schots, M., Vasconcelos, R., Queiroz, A. R., Werner, C. (2015). “Organization of Visualization Knowledge for Supporting the Choice of Software Visualizations”. Technical Report, COPPE/UFRJ, Rio de Janeiro, Brazil, December (to appear).

[Schulz 2011] Schulz, H. J. (2011). “Treevis.net: A tree visualization reference”. *IEEE Computer Graphics and Applications*, v. 31, n. 6, pp. 11-15, November.

[Seaman 1999] Seaman, C. B. (1999). “Qualitative methods in empirical studies of software engineering”. *IEEE Transactions on Software Engineering*, v. 25, n. 4, pp. 557-572, July.

[Seaman 2009] Seaman, C. (2009). “Using Qualitative Methods in Empirical Studies of Software Engineering”. Short course. In: *VI Experimental Software Engineering Latin American Workshop (ESELAW 2009)*, São Carlos, Brazil, November.

[Selby 2005] Selby, R. W. (2005). “Enabling reuse-based software development of large-scale systems”. *IEEE Transactions on Software Engineering*, v. 31, n. 6, pp. 495-510, June.

[Sensalire et al. 2009] Sensalire, M., Ogao, P., Telea, A. (2009). “Evaluation of software visualization tools: Lessons learned”. In: *5th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2009)*, Edmonton, Canada, pp. 19-26, September.

[Sherif & Vinze 2003] Sherif, K., Vinze, A. (2003). “Barriers to adoption of software reuse: A qualitative study”. *Information & Management*, v. 41, n. 2, pp. 159-175, December.

[Shi et al. 2005] Shi, K., Irani, P., Li, B. (2005). “An Evaluation of Content Browsing Techniques for Hierarchical Space-Filling Visualizations”. In: *IEEE Symposium on Information Visualization (InfoVis 2005)*, Minneapolis, USA, pp. 81-88, October.

[Shi et al. 2011] Shi, Z., Wang, X., Yue, J. (2011) “Cognitive Cycle in Mind Model CAM”. *International Journal of Intelligence Science*, v. 1, n. 2, pp. 25-34.

[Shneiderman 1996] Shneiderman, B. (1996). “The eyes have it: A task by data type taxonomy for information visualizations”. In: *Proceedings of the IEEE Symposium on Visual Languages*, Boulder, USA, pp. 336-343, September.

[Silva 2012] Silva, M. A. (2012). “Software Visualization Product Line: An Infrastructure for Supporting Comprehension Activities by Visualization Mechanisms Generation” [Linha de Produtos para Visualização de Software: Uma Infraestrutura para Apoiar Atividades de Compreensão por meio da Construção de Mecanismos de Visualização] (in Portuguese). M.Sc. Thesis, COPPE/UFRJ, Rio de Janeiro, Brazil, December.

[Silva et al. 2012] Silva, M., Schots, M., Werner, C. (2012). “Supporting Software Maintenance Activities through a Software Visualization Product Line Infrastructure”. In: *9th Workshop on Modern Software Maintenance (WMSWM 2012)*, Fortaleza, Brazil, pp. 1-8, June.

[Silva Filho et al. 2008] Silva Filho, R. C., Katsurayama, A. E., Santos, G., Murta, L., Rocha, A. R. C. (2008). “Deploying Software Reuse Management at COPPE/UFRJ Software Engineering Laboratory”. In: *1st Workshop on Software Reuse Efforts (WSRE), 2nd RiSE Summer School on Software Product Lines (RiSS 2008)*, Recife, Brazil, pp. 1-5, November.

[SOFTEX 2012] SOFTEX (2012). “MPS.BR – Brazilian Software Process Improvement – General Guide” [MPS.BR – Melhoria de Processo do Software Brasileiro – Guia Geral] (in Portuguese), August. Available at http://www.softex.br/mpsbr.

[SOFTEX 2013a] SOFTEX (2013a). “Implementation Guide – Part 3: Reasoning for the Implementation of Level E of MR-MPS-SW:2012” [Guia de Implementação – Parte 3: Fundamentação para Implementação do Nível E do MR-MPS-SW:2012] (in Portuguese), September. Available at http://www.softex.br/mpsbr/guias/.

[SOFTEX 2013b] SOFTEX (2013c). “Assessment Guide” [Guia de Avaliação] (in Portuguese), September. Available at http://www.softex.br/mpsbr/guias/.

[Spence 2001] Spence, R. (2001), *Information Visualization*, Addison Wesley Publisher, 1st edition.

[Telea et al. 2010] Telea, A., Ersoy, O., Voinea, L., (2010). “Visual Analytics in Software Maintenance: Challenges and Opportunities”. In: *1st European Symposium on Visual Analytics (EuroVAST)*, Bordeaux, France, pp. 65-70, June.

[Thomas & Cook 2006] Thomas, J. J, Cook, K. A. (2006) “A visual analytics agenda”. *IEEE Computer Graphics and Applications*, v. 26, n. 1, pp. 10-13, January.

[Tory & Moller 2004] Tory, M., Moller, T. (2004). “Rethinking visualization: A high-level taxonomy”. In: *IEEE Symposium on Information Visualization (InfoVis 2004)*, Austin, USA, pp. 151-158, October.

[Travassos et al. 2008] Travassos, G. H., Santos, P. S. M., Mian, P. G., Dias Neto, A. C., Biolchini, J. (2008). “An environment to support large scale experimentation in software engineering”. In: *13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2008)*, Belfast, Northern Ireland, pp. 193-202, April.

[Treude & Storey 2010] Treude, C., Storey, M.-A. (2010). “Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds”. In: *32nd ACM/IEEE International Conference on Software Engineering (ICSE 2010)*, Cape Town, South Africa, pp. 365-374, May.

[Vasconcelos et al. 2013] Vasconcelos, R. R., Schots, M., Werner, C. (2013). “Recommendations for Context-Aware Visualizations in Software Development”. In: *10th Workshop on Modern Software Maintenance (WMSWM 2013)*, Salvador, Brazil, pp. 41-48, July.

[Vasconcelos et al. 2014a] Vasconcelos, R., Schots, M., Werner, C. (2014). “An Information Visualization Feature Model for Supporting the Selection of Software Visualizations”. In: *22nd International Conference on Program Comprehension (ICPC 2014), Early Research Achievements Track*, Hyderabad, India, pp. 122-125, June.

[Vasconcelos et al. 2014b] Vasconcelos, R., Schots, M., Werner, C. (2014). “On the Use of Context Information for Supporting Software Visualizations”. In: *2nd Workshop on Software Visualization, Evolution, and Maintenance (VEM 2014)*, Maceió, Brazil, pp. 54-61, September.

[Vasconcelos 2015] Vasconcelos, R. R. (2015). “A Context-Aware Approach for Information Visualization: An Example in Software Development Scenarios” [Uma Abordagem Sensível ao Contexto para Visualização de Informação: Um Exemplo em Cenários de Desenvolvimento de Software] (in Portuguese). M.Sc. Thesis, COPPE/UFRJ, Rio de Janeiro, Brazil, April.

[Vital & Krause 2013] Vital, G. B., Krause, V. S. (2013). “Rec4Reuse: A system for performing evaluations and recommendations based on desirable properties of reusable software” [Rec4Reuse: Um sistema de avaliação e recomendação baseado em propriedades desejáveis a software reutilizável] (in Portuguese). Undergraduate Final Project, Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, Brazil.

[Von Mayrhauser & Vans 1995] Von Mayrhauser, A., Vans, A. M. (1995). “Program comprehension during software maintenance and evolution”, *Computer*, v. 28, n. 8, pp. 44-55, August.

[Werner et al. 2011] Werner, C., Murta, L., Schots, M., Magdaleno, A., Silva, M., Cepeda, R., Vahia, C. (2011). “EvolTrack: A Plug-in-Based Infrastructure for Visualizing Software Evolution”. In: *1st Brazilian Workshop on Software Visualization (WBVS 2011)*, São Paulo, Brazil, pp. 1-8.

[Wettel & Lanza 2008] Wettel, R., Lanza, M. (2008). “Visual Exploration of Large-Scale System Evolution”. In: *15th Working Conference on Reverse Engineering (WCRE 2008)*, Antwerp, Belgium, pp. 219-228, October.

[Wong et al. 2007] Wong, K., Stroulia, E., Tonella, P. (2007). “Message from the Chairs”. In: *15th IEEE International Conference on Program Comprehension (ICPC 2007)*, Banff, Canada, pp. ix, June.

[Wu & Storey 2000] Wu, J., Storey, M.-A. D. (2000). “A Multi-Perspective Software Visualization Environment”. In: *Conference of the Centre for Advanced Studies on Collaborative Research*, Mississauga, Canada, pp. 1-10, November.

[Yano et al. 2015] Yano, Y., Kula, R. G., Ishio, T., Inoue, K. (2015). “VerXCombo: An interactive data visualization of popular library version combinations”. In: *23rd IEEE International Conference on Program Comprehension (ICPC 2015), Tool Demo Session*, Firenze, Italy, pp. 291-294, May.

[Yazdanshenas et al. 2012] Yazdanshenas, A. R., Moonen, L. (2012). “Tracking and visualizing information flow in component-based systems”. In: *20th IEEE International Conference on Program Comprehension (ICPC 2012)*, Passau, Germany, pp. 143-152, June.

[Ye & Fischer 2000] Ye, Y., Fischer, G., Reeves, B. (2000). “Integrating Active Information Delivery and Reuse Repository Systems”. In: *ACM SIGSOFT Symposium on Foundations on Software Engineering (FSE 2000)*, San Diego, USA, pp. 60-68, November.

[Ye & Fischer 2002] Ye, Y., Fischer, G. (2002). “Supporting reuse by delivering task-relevant and personalized information”. In: *24th International Conference on Software Engineering (ICSE 2002)*, Orlando, USA, pp. 513-523, May.

[Yi et al. 2007] Yi, J. S., Kang, Y., Stasko, J. T., Jacko, J. A. (2007). “Toward a Deeper Understanding of the Role of Interaction in Information Visualization”, *IEEE Transactions on Visualization and Computer Graphics*, v. 13, n. 6, pp.1224-1231, December.

APPENDIX A – MAPPING BETWEEN GOALS AND VISUALIZATIONS

This appendix presents the results of the mapping performed for the construction of Zooming Browser. It lists the elements that have been derived from the meet-in-the-middle strategy (mentioned in Section 4.6.1).

Table A.1 presents the reuse goals, divided into project goals (PG) and organizational goals (OG). It is noteworthy that Zooming Browser only provides partial support for each of these goals. One must resort to other resources (e.g., CAVE) in order to perform analyses in a more detailed level.

Table A.1 – Reuse goals

Goal ID	Description
PG01	Identify an asset that fits the project needs
PG02	Decide whether an asset or an asset version can/should be reused in (incorporated to) a project or not
PG03	Decide whether an existing project that already contains a given asset version should upgrade/downgrade to a newer/older asset version
OG01	Maintain the reuse repository/library (i.e., include, exclude, request maintenance or discontinue/deprecate asset versions, as well as keep metadata information for communication purposes)
OG02	Manage and monitor the implementation of reuse processes and evaluate the effectiveness of reuse practices (progresses and efforts), in the context of local projects/assets/developers (i.e., belonging to the organization)

These goals were broken down into a set of questions, presented in tables (from Table A.2 to

Table A.**5**). The relationship/mapping between goals and questions is presented in Table A.6, and the main elements from the mapping between questions and visualizations are presented in Table A.7.

Table A.2 – Questions on general asset/consumer/project information (related to reuse occurrences)

<table>
<tr><th colspan="2">Asset-Centric</th><th colspan="2">Project-Centric</th><th colspan="2">Developer-Centric</th></tr>
<tr><th>Asset-Developers</th><th>Developer-Assets</th><th>Asset-Projects</th><th>Project-Assets</th><th>Developer-Projects</th><th>Project-Developers</th></tr>
<tr><td colspan="2">Which [versions of] assets have ever been reused?</td><td colspan="2" rowspan="3">Which projects have ever had an asset included (i.e., contain at least one reusable asset in their development history)?</td><td colspan="2" rowspan="2">Which developers are consumers (i.e., have ever reused an asset)?</td></tr>
<tr><td colspan="2">Which versions of this asset were reused?</td></tr>
<tr><td colspan="2">Which [versions of] assets are most often reused?</td><td colspan="2" rowspan="2">Which consumers reuse assets more often?</td></tr>
<tr><td colspan="2">Which versions of this asset are most often reused?</td><td colspan="2" rowspan="2">In which projects assets are most often reused (i.e., which projects contain the largest number of assets)?</td></tr>
<tr><td colspan="2">How often are [versions of] assets reused over time?</td><td colspan="2" rowspan="2">How active is this consumer in terms of number of reuses?</td></tr>
<tr><td colspan="2">How often is this asset [version] reused over time?</td><td colspan="2"></td></tr>
</table>

Table A.3 – Questions on project/producer information related to asset development/release history and asset maintenance

	Project information	Producer information
Asset development/ release history	Which assets have the most active development project?	Which developers are producers (i.e., have ever produced an asset or contributed to the development of an asset)?
	Which versions of this asset were released?	Which consumers are also producers?
	How active is the development project of this asset?	Which producers develop assets more often?
		How active is this producer in terms of assets' development?
Asset maintenance	Among the reported bugs, improvement suggestions, or feature requests related to this asset [version], are most of them fixed or open?	Among the reported bugs, improvement suggestions, or feature requests assigned to this producer, are most of them fixed or open?
	How often do producers of this asset fix reported bugs?	How often does this producer fix reported bugs?
	How long does it take for producers of this asset to fix reported bugs?	How long does it take for this producer to fix reported bugs?
	How often do producers of this asset implement improvement suggestions or feature requests?	How often does this producer implement improvement suggestions or feature requests?

Table A.4 – Asset/consumer/project information (related to reuse occurrences)

<table>
<tr><th colspan="2">Asset-Centric</th><th colspan="2">Project-Centric</th><th colspan="2">Developer-Centric</th><th rowspan="2">Asset-Assets</th><th rowspan="2">Developer-Developers</th><th rowspan="2">Project-Projects</th></tr>
<tr><th>Asset-Developers</th><th>Developer-Assets</th><th>Asset-Projects</th><th>Project-Assets</th><th>Developer-Projects</th><th>Project-Developers</th></tr>
<tr><td colspan="6">Which assets were reused by which consumers in which projects?</td><td rowspan="2">Which [versions of] assets were reused in the development of which [versions of] assets? (CAVE)</td><td rowspan="2">Which consumers reused the same [version(s) of] asset(s) reused by which consumers?</td><td rowspan="2">Which projects contain the same [version(s) of] asset(s) reused in another project?</td></tr>
<tr><td colspan="2">Which consumers reused which assets?</td><td colspan="2">Which assets were reused in which projects?</td><td colspan="2">Which consumers reused assets in which projects?</td></tr>
<tr><td rowspan="2">Which consumers reused this asset [version]?</td><td>Which assets did this consumer reuse?</td><td rowspan="2">In which projects was this asset [version] reused?</td><td>Which assets were reused in this project?</td><td>In which projects did this consumer reuse [versions of] assets?</td><td>Which consumers reused [versions of] assets in this project?</td><td>Which [versions of] assets were reused in the development of this asset [version]? (CAVE)</td><td>Which consumers reused the same [version(s) of] asset(s) reused by this consumer?</td><td>Which projects contain the same [version(s) of] asset(s) reused in this project?</td></tr>
<tr><td>Which versions of this asset did this consumer reuse?</td><td>Which versions of this asset were reused in this project?</td><td>In which projects did this consumer reuse this asset [version]?</td><td rowspan="2">Which consumers reused this asset [version] in this project?</td><td>Which [versions of] assets does this asset [version] depend on?</td><td>Which consumers reused [version(s) of] asset(s) developed by which producers?</td><td rowspan="2"></td></tr>
<tr><td></td><td></td><td>Which [versions of] assets did this consumer reuse in this project?</td><td>Which [versions of] assets depend on this asset [version]?</td><td>Which consumers reused [version(s) of] asset(s) developed by this producer?</td></tr>
</table>

<table>
<tr><th colspan="2">Asset-Centric</th><th colspan="2">Project-Centric</th><th colspan="2">Developer-Centric</th><th rowspan="2">Asset-Assets</th><th rowspan="2">Developer-Developers</th><th rowspan="2">Project-Projects</th></tr>
<tr><th>Asset-Developers</th><th>Developer-Assets</th><th>Asset-Projects</th><th>Project-Assets</th><th>Developer-Projects</th><th>Project-Developers</th></tr>
<tr><td>Who are the main consumers of this asset [version] (i.e., the consumers who reused this asset [version] more often)?</td><td>Which [versions of] assets does this consumer reuse most often?</td><td>Which project contains the largest number of reuse occurrences of this asset (i.e., the largest number of distinct versions of this asset)?</td><td>Which assets are most often reused in this project (i.e., have the largest number of distinct versions included in this project)?</td><td>In which projects did this consumer include the largest number of [versions of] assets?</td><td>Who are the main consumers in this project (i.e., the consumers who included different [versions of] assets in this project more often)?</td><td colspan="3" rowspan="8"></td></tr>
<tr><td colspan="2">How often does this consumer reuse this asset [version]?</td><td colspan="2">How often is this asset reused in this project (i.e., are there distinct versions of this asset reused in this project)?</td><td colspan="2">How often does this consumer reuse [versions of] assets in this project (i.e., are there distinct [versions of] assets reused in this project by this consumer)?</td></tr>
<tr><td colspan="6">For how long (over time) does this project contain this asset included by this consumer?</td></tr>
<tr><td colspan="2">For how long (over time) do projects contain this asset included by this consumer?</td><td colspan="2">For how long (over time) does this project contain this asset?</td><td colspan="2">For how long (over time) does this project contain assets included by this consumer?</td></tr>
<tr><td colspan="4" rowspan="3"></td><td colspan="2">Which projects contain reusable assets among their releases?</td></tr>
<tr><td>Which projects contain, among their releases, [a version of] this asset?</td><td>Which [versions of] assets are among the releases of this project?</td><td>Which projects contain, among their releases, assets included by this consumer?</td><td>Which consumers included assets that are among the releases of this project?</td></tr>
<tr><td colspan="2">Which [versions of] assets included by this consumer are among the releases of this project?</td></tr>
</table>

Table A.5 – Questions on production information (related to the asset development/release history)

Asset-Centric		Developer-Developers
Asset-Developers	**Developer-Assets**	
Which [versions of] assets were developed by which producers?		Which producers contribute/collaborate with which producers in assets development?
Which producers contributed to the development of this asset [version]?	Which [versions of] assets contain [which kinds of] development contributions made by this producer?	Who are the producers with whom this producer contributes/collaborates (in assets development)?
		Which producers contribute/collaborate in the development of assets developed by this producer?
To which parts of the development history of this asset [version] did each producer contribute?	To which parts of the assets development history did this producer contribute?	
Who are the main producers of this asset [version] (i.e., the producers who most contributed to the development of this asset [version])?	What are the [versions of] assets to which development this producer has most contributed?	
Which producers contributed to which releases of [versions of] assets?		
Which producers contributed to a release of this asset [version]?	Which released [versions of] assets contain development contributions made by this producer?	
Which released versions of this asset contain development contributions made by this producer?		
Which producers contributed to the development of this asset [version] but stopped contributing afterwards?	Which [versions of] assets contain some contribution made by this producer but such producer stopped contributing afterwards?	

Table A.6 – Mapping between questions and reuse goals

Question ID	Description	Goals supported by the question
GENERAL ASSET INFORMATION RELATED TO REUSE OCCURRENCES		
Q01.R.A	Which [versions of] assets have ever been reused?	OG01;OG02
Q02.R.A	Which versions of this asset were reused?	PG02;PG03;OG01;OG02
Q03.R.A	Which [versions of] assets are most often reused?	PG02;PG03;OG01;OG02
Q04.R.A	Which versions of this asset are most often reused?	PG02;PG03;OG01;OG02
Q05.R.A	How often are [versions of] assets reused over time?	OG01;OG02

Question ID	Description	Goals supported by the question
Q06.R.A	How often is this asset [version] reused over time?	PG02;PG03;OG01;OG02
GENERAL PROJECT INFORMATION RELATED TO REUSE OCCURRENCES		
Q07.R.P	Which projects have ever had an asset included (i.e., contain at least one reusable asset in their development history)?	OG01;OG02
Q08.R.P	In which projects assets are most often reused (i.e., which projects contain the largest number of assets)?	OG02
GENERAL CONSUMER INFORMATION RELATED TO REUSE OCCURRENCES		
Q09.R.D	Which developers are consumers (i.e., have ever reused an asset)?	OG01;OG02
Q10.R.D	Which consumers reuse assets most often?	OG02
Q11.R.D	How active is this consumer in terms of number of reuses?	OG02
GENERAL PROJECT INFORMATION RELATED TO ASSET DEVELOPMENT/RELEASE HISTORY		
Q12.P.A	Which assets have the most active development project?	OG02
Q13.P.A	How active is the development project of this asset?	PG02;PG03;OG02
Q14.P.A	Which versions of this asset were released?	PG02;OG02
GENERAL PRODUCER INFORMATION RELATED TO ASSET DEVELOPMENT/RELEASE HISTORY		
Q15.P.D	Which developers are producers (i.e., have ever produced an asset or contributed to the development of an asset)?	OG01;OG02
Q16.P.D	Which consumers are also producers?	OG02
Q17.P.D	Which producers develop assets most often?	OG01;OG02
Q18.P.D	How active is this producer in terms of assets' development?	OG01;OG02
GENERAL ASSET INFORMATION RELATED TO ASSET MAINTENANCE		
Q19.M.A	Among the reported bugs, improvement suggestions, or feature requests related to this asset [version], are most of them fixed or open?	PG02;PG03;OG02
Q20.M.A	How often do producers of this asset fix reported bugs?	PG02;PG03;OG02
Q21.M.A	How long does it take for producers of this asset to fix reported bugs?	PG02;PG03;OG02
Q22.M.A	How often do producers of this asset implement improvement suggestions or feature requests?	PG02;PG03;OG02
Q19.M.A	Among the reported bugs, improvement suggestions, or feature requests related to this asset [version], are most of them fixed or open?	PG02;PG03;OG02
GENERAL PRODUCER INFORMATION RELATED TO ASSET MAINTENANCE		
Q23.M.D	Among the reported bugs, improvement suggestions, or feature requests assigned to this producer, are most of them fixed or open?	PG02;PG03;OG02

Question ID	Description	Goals supported by the question
Q24.M.D	How often does this producer fix reported bugs?	PG02;PG03;OG02
Q25.M.D	How long does it take for this producer to fix reported bugs?	PG02;PG03;OG02
Q26.M.D	How often does this producer implement improvement suggestions or feature requests?	PG02;PG03;OG02
ASSET/CONSUMER/PROJECT INFORMATION (ALL RELATED TO REUSE OCCURRENCES)		
Q27.R.*	Which assets were reused by which consumers in which projects?	PG01;PG02;PG03;OG01;OG02
Q28.R.(AD)	Which consumers reused which assets?	OG01;OG02
Q29.R.AD	Which consumers reused this asset [version]?	PG02;PG03;OG01;OG02
Q30.R.AD	Who are the main consumers of this asset [version] (i.e., the consumers who most often reused this asset [version])?	PG02;PG03;OG01;OG02
Q31.R.DA	Which assets did this consumer reuse?	OG01;OG02
Q32.R.DA	Which versions of this asset did this consumer reuse?	OG01;OG02
Q33.R.DA	Which [versions of] assets does this consumer reuse most often?	PG02;PG03;OG01;OG02
Q34.R.(AD)	How often does this consumer reuse this asset [version]?	OG01;OG02
Q35.R.(AP)	Which assets were reused in which projects?	PG01;PG02;PG03;OG01;OG02
Q36.R.AP	In which projects was this asset [version] reused?	PG01;PG02;PG03;OG01;OG02
Q37.R.AP	Which project contains the largest number of reuse occurrences of this asset (i.e., the largest number of distinct versions of this asset)?	PG03;OG02
Q38.R.AP	Which projects contain this asset [version] at some point of the development life cycle but do not contain such asset [version] afterwards?	PG02;PG03;OG01;OG02
Q39.R.PA	Which assets were reused in this project?	PG01;PG02;PG03;OG01;OG02
Q40.R.PA	Which versions of this asset were reused in this project?	PG01;PG02;PG03;OG01;OG02
Q41.R.(AP)	Which assets are most often reused in this project (i.e., have the largest number of distinct versions included in this project)?	PG03;OG02
Q42.R.PA	How often is this asset reused in this project (i.e., are there distinct versions of this asset reused in this project)?	PG03;OG02
Q43.R.(AP)	Which [versions of] assets were included at some point of the development life cycle of this project but were removed afterwards?	PG03;OG02
Q44.R.(AP)	Which projects contain reusable assets among their releases?	PG02;PG03;OG01;OG02
Q45.R.AP	Which projects contain, among their releases, [a version of] this asset?	PG02;PG03;OG01;OG02
Q46.R.PA	Which [versions of] assets are among the releases of this project?	PG02;PG03;OG01;OG02
Q51.R.(DP)	Which consumers reused assets in which projects?	PG01;PG02;OG01;OG02

Question ID	Description	Goals supported by the question
Q52.R.DP	In which projects did this consumer reuse [versions of] assets?	PG01;PG02;OG01;OG02
Q53.R.DP	In which projects did this consumer reuse this asset [version]?	PG02;OG01;OG02
Q54.R.(DP)	Which [versions of] assets did this consumer reuse in this project?	PG01;PG02;OG01;OG02
Q55.R.DP	Which projects contain, at some point of the development life cycle, [versions of] assets included by this consumer that were removed afterwards?	PG02;OG01;OG02
Q56.R.DP	In which projects did this consumer include the largest number of [versions of] assets?	OG02
Q57.R.PD	Which consumers reused [versions of] assets in this project?	OG01;OG02
Q58.R.PD	Which consumers reused this asset [version] in this project?	OG01;OG02
Q59.R.PD	Who are the main consumers in this project (i.e., the consumers who most often included different [versions of] assets in this project)?	OG02
Q60.R.(DP)	How often does this consumer reuse [versions of] assets in this project (i.e., are there distinct [versions of] assets reused in this project by this consumer)?	OG02
Q63.R.DP	Which projects contain, among their releases, assets included by this consumer?	PG03;OG01;OG02
Q64.R.PD	Which consumers included assets that are among the releases of this project?	PG03;OG01;OG02
Q65.R.(DP)	Which [versions of] assets included by this consumer are among the releases of this project?	PG03;OG01;OG02
Q69.R.*	For how long (over time) does this project contain this asset included by this consumer?	OG02
Q70.R.(AD)	For how long (over time) do projects contain this asset included by this consumer?	PG02;OG02
Q71.R.(AP)	For how long (over time) does this project contain this asset?	PG02;OG02
Q72.R.(DP)	For how long (over time) does this project contain assets included by this consumer?	PG02;OG02
Q72.R.AA	Which [versions of] assets were reused in the development of which [versions of] assets? (CAVE)	PG02;PG03
Q73.R.AA	Which [versions of] assets were reused in the development of this asset [version]? (CAVE)	PG02;PG03
Q74.R.AA	Which [versions of] assets does this asset [version] depend on?	PG02;PG03
Q75.R.AA	Which [versions of] assets depend on this asset [version]?	OG01
Q76.R.DD	Which consumers reused the same [version(s) of] asset(s) reused by which consumers?	OG02
Q77.R.DD	Which consumers reused the same [version(s) of] asset(s) reused by this consumer?	PG02; OG02
Q78.R.DD	Which consumers reused [version(s) of] asset(s) developed by which producers?	OG02
Q79.R.DD	Which consumers reused [version(s) of] asset(s) developed by this producer?	OG02
Q80.R.PP	Which projects contain the same [version(s) of] asset(s) reused in another project?	PG01

Question ID	Description	Goals supported by the question
Q81.R.PP	Which projects contain the same [version(s) of] asset(s) reused in this project?	PG01
PRODUCTION INFORMATION RELATED TO ASSET DEVELOPMENT/RELEASE HISTORY		
Q82.P.(AD)	Which [versions of] assets were developed by which producers?	PG02;OG02
Q83.P.AD	Which producers contributed to the development of this asset [version]?	PG02;PG03;OG01;OG02
Q84.P.AD	To which parts of the development history of this asset [version] did each producer contribute?	PG02;PG03;OG01;OG02
Q85.P.AD	Who are the main producers of this asset [version] (i.e., the producers who most contributed to the development of this asset [version])?	PG02;PG03;OG01;OG02
Q86.P.AD	Which producers contributed to the development of this asset [version] but stopped contributing afterwards?	PG02;OG02
Q87.P.DA	Which [versions of] assets contain [which kinds of] development contributions made by this producer?	PG02;OG02
Q88.P.DA	To which parts of the assets development history did this producer contribute?	PG02;OG02
Q89.P.DA	What are the [versions of] assets to which development this producer has most contributed?	OG02
Q90.P.DA	Which [versions of] assets contain some contribution made by this producer but such producer stopped contributing afterwards?	OG02
Q91.P.(AD)	Which producers contributed to which releases of [versions of] assets?	PG02;OG02
Q92.P.AD	Which producers contributed to a release of this asset [version]?	PG02;OG02
Q94.P.DA	Which released [versions of] assets contain development contributions made by this producer?	OG02
Q95.P.(AD)	Which released versions of this asset contain development contributions made by this producer?	OG02
Q97.P.DD	Which producers contribute/collaborate with which producers in assets development?	PG02;OG02
Q98.P.DD	Who are the producers with whom this producer contributes/collaborates (in assets development)?	PG02;OG02
Q99.P.DD	Which producers contribute/collaborate in the development of assets developed by this producer?	PG02;OG02

Table A.7 – Mapping between questions, data, visual attributes, and visualizations

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
Dash-MRA-BarC	Dashboard	--	--	Bar/Stacked bar chart	--	Size (indicates the number of reuses of the asset)	• Which [versions of] assets are most often reused?
Dash-MADP-BarC	Dashboard	--	--	Bar/Stacked bar chart	--	Size (indicates the activeness of the development project)	• Which assets have the most active development project?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
Dash-MAC-BarC	Dashboard	--	--	Bar/Stacked bar chart	--	Size (indicates number of assets reused by the consumer)	• Which consumers reuse assets most often?
Dash-MAP-BarC	Dashboard	--	--	Bar/Stacked bar chart	--	Size (indicates number of assets produced by the producer)	• Which producers develop assets most often?
Dash-PLNRA-BarC	Dashboard	--	--	Bar/Stacked bar chart	--	Size (indicates the number of reuse occurrences in the project)	• In which projects assets are most often reused (i.e., which projects contain the largest number of assets)?
Dash-MAP-MRAC-PieC	Dashboard	--	--	Pie chart	Select consumer of interest (from the MAP bar chart)	Pie slice (each slice represents a [version of] asset reused by this consumer, and its size represents the percentage of times he/she reused it against all his/her reuse occurrences)	• Which [versions of] assets does this consumer reuse most often?
Dash-PLNRA-MRAP-PieC	Dashboard	--	--	Pie chart	Select project of interest (from the PLNRA bar chart)	Pie slice (each slice represents a [version of] asset reused in this project, and its size represents the percentage of times it was reused against all the reuse occurrences in the project)	• Which assets are most often reused in this project (i.e., have the largest number of distinct versions included in this project)?
Dash-RAOT-LC	Dashboard	--	--	Line chart	--	Increasing/Decreasing pattern of the line path	• How often are [versions of] assets reused over time?
Dash-Matrix-ACP	Dashboard (not sure)	--	Full Reuse Map	Matrix	Perform custom association of rows, columns, and cells to assets, developers (consumers), and projects.	Matrix cell content matching row and column	• Which assets were reused by which consumers in which projects?
Dash-Matrix-AC	Dashboard (not sure)	--	Asset-Consumers Map	Matrix	--	Matrix cell content matching row and column represents projects in which a consumer reused an asset.	• Which consumers reused which assets?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
Dash-Matrix-AProd	Dashboard (not sure)	--	Asset-Producers Map	Matrix	--	Matrix cell content matching row and column represents assets to which development a producer contributed.	• Which [versions of] assets were developed by which producers?
Dash-Matrix-AProj	Dashboard (not sure)	--	Asset-Projects Map	Matrix	--	Matrix cell content matching row and column represents consumers who reused an asset in a project.	• Which assets were reused in which projects?
Dash-Matrix-CP	Dashboard (not sure)	--	Consumer-Projects Map	Matrix	--	Matrix cell content matching row and column represents assets that were reused by a consumer in a project.	• Which consumers reused assets in which projects?
A-BC-Ov	Metadata Exploration	Assets	--	Bubble Chart	Each asset can be exploded to depict its versions	Overview of bubbles (depicts assets or asset versions)	No question associated
A-BC-HMF	Metadata Exploration	Assets	--	Bubble Chart	Each asset can be exploded to depict its versions / Filter by Reused Assets	Highlight filter (showing assets that have been reused) OR Mitigation (hiding the ones who do not match the filtering criteria) (could use color too)	• Which [versions of] assets have ever been reused?
A-CBC-S	Metadata Exploration	Assets	--	Cartesian Bubble Chart (with asset versions)	Switch to asset versions view	Size (indicates number of reuses) AND Row (shows an asset [version] reuse history)	• Which [versions of] assets are most often reused?
A-RH-RHG-SI	History (from Metadata Exploration)	Assets	Reuse History	VCS Release History Graph	Filter by Consumption Data	Star Icon (depicts asset versions)	• Which versions of this asset were reused?
A-RH-RHG-S	History (from Metadata Exploration)	Assets	Reuse History	VCS Release History Graph	Filter by Consumption Data OR Production Data	Size (indicates number of reuses (for Consumption Data) and "Activeness" of development (for Production Data))	• Which versions of this asset are most often reused? • Which assets have the most active development project?
A-RH-LC	Metadata Exploration	Assets	Reuse History	Line chart	--	Line path (each line depicts an asset or an asset version)	• How often is this asset [version] reused over time?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
A-C-BC	Metadata Exploration	Assets	Consumers	Bubble Chart	--	Bubble (depicts the consumers)	• Which consumers reused this asset [version]?
A-C-BC-S	Metadata Exploration	Assets	Consumers	Bubble Chart	--	Size (indicates number of reuses performed)	• Who are the main consumers of this asset [version] (i.e., the consumers who most often reused this asset [version])?
A-C-BC-RCC	History (from Metadata Exploration)	Assets	Consumers	Range Column Chart	Select consumer of interest, then choose "Permanence of this asset in projects over time (included by this consumer)" option	Each line represents a project; Length and position of the column-bars in the x-axis represent the time during which the project contains the asset	• For how long (over time) do projects contain this asset included by this consumer?
A-C-Proj-BC	Metadata Exploration	Assets	Consumers	Consumer's Bubble Chart, drilling down to Projects Bubble Chart	Select consumer of interest	Bubble (depicts projects in which the consumer reused the asset [version])	• In which projects did this consumer reuse this asset [version]?
A-C-Proj-BC-RCC-DHG-HMF	History (from Metadata Exploration)	Assets	Consumers	Consumer's Bubble Chart, drilling down to Project's Range Column Chart, drilling down to VCS Development History Graph	Select consumer of interest, then choose "Permanence of this asset in projects over time (included by this consumer)" option, then select project of interest from the Range Column Chart	Highlight filter (showing the versions since the moment the asset was included in this project by this consumer until the moment it was removed, if so) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• For how long (over time) does this project contain this asset included by this consumer?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
A-P-BC	Metadata Exploration	Assets	Producers	Bubble Chart	--	Bubble (depicts the producers)	• Which producers contributed to the development of this asset [version]?
A-P-BC-S	Metadata Exploration	Assets	Producers	Bubble Chart	--	Size (indicates amount of contribution)	• Who are the main producers of this asset [version] (i.e., the producers who most contributed to the development of this asset [version])?
A-P-BC-HMF	Metadata Exploration	Assets	Producers	Bubble Chart	Filter by "Show only producers who contributed to asset releases"	Highlight filter (showing producers who contributed to asset releases) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• Which producers contributed to a release of this asset [version]?
A-P-BC-DHG-PIContrib	History (from Metadata Exploration)	Assets	Producers	Bubble Chart, drilling down to VCS Development History Graph	Filter by producers that started contributing and stopped afterwards AND Select producer of interest	Bubble (each bubble represents a producer in these conditions) AND Producer Icon with + or - visible signs (depicts a VCS project version in which the producer started/stopped contributing)	• Which producers contributed to the development of this asset [version] but stopped contributing afterwards?
A-Proj-BC	Metadata Exploration	Assets	Projects in which this asset was reused	Bubble Chart	--	Bubble (depicts projects that contain this asset [version])	• In which projects was this asset [version] reused?
A-Proj-BC-S	Metadata Exploration	Assets	Projects in which this asset was reused	Bubble Chart	--	Size (indicates number of reuse occurrences)	• Which project contains the largest number of reuse occurrences of this asset (i.e., the largest number of distinct versions of this asset)?
A-Proj-BC-LC	Metadata Exploration	Assets	Projects in which this asset was reused	Line chart	Select project of interest	Line path (each line depicts an asset version)	• How often is this asset reused in this project (i.e., are there distinct versions of this asset reused in this project)?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
A-Proj-BC-HMF	History (from Metadata Exploration)	Assets	Projects in which this asset was reused	Bubble Chart	Filter by "Project releases that contain this asset"	Highlight filter (showing which project releases contain this asset) OR Mitigation (hiding the ones who do not match the filtering criteria) (could use color too)	• Which projects contain, among their releases, [a version of] this asset?
A-Proj-C-BC	Metadata Exploration	Assets	Projects in which this asset was reused	Project's Bubble Chart, drilling down to Consumers Bubble Chart	Select project of interest	Bubble (depicts consumers who reused the asset [version] in the project)	• Which consumers reused this asset [version] in this project?
A-Proj-BC-DHG-AIContrib	History (from Metadata Exploration)	Assets	Projects in which this asset was reused	Bubble Chart, drilling down to VCS Development History Graph	Filter by projects that have assets included and removed afterwards AND Select project of interest	Bubble (each bubble represents a project in these conditions) AND Asset icon with + or - visible signs (depicts a VCS project version in which the asset was included/excluded)	• Which projects contain this asset [version] at some point of the development life cycle but do not contain such asset [version] afterwards?
A-Proj-C-BC-DHG-HMF	History (from Metadata Exploration)	Assets	Projects in which this asset was reused	Project's Bubble Chart, drilling down to Consumer's Bubble Chart, drilling down to VCS Development History Graph	Select project of interest, then select consumer of interest	Highlight filter (showing the versions since the moment the asset was included in this project by this consumer until the moment it was removed, if so) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• For how long (over time) does this project contain this asset included by this consumer?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
A-Proj-BC-DHG-HMF	History (from Metadata Exploration)	Assets	Projects in which this asset was reused	VCS Development History Graph	Select project of interest, then choose "Permanence of this asset in this project over time" option	Highlight filter (showing the versions since the moment the asset was included in this project until the moment it was removed, if so) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• For how long (over time) does this project contain this asset?
A-DH-DHG-Pos	History (from Metadata Exploration)	Assets	Development history	VCS Development History Graph	--	Position of elements in the x-axis (which represents time)	• How active is the development project of this asset?
A-DH-DHG-Color	History (from Metadata Exploration)	Assets	Development history	Line chart	--	Line path (represents the frequency of commits over time)	• How active is the development project of this asset?
A-DH-DHG-Circle	History (from Metadata Exploration)	Assets	Development history	VCS Development History Graph	--	Circle (depicts a VCS project version)	No question associated
A-DH-DHG-SI	History (from Metadata Exploration)	Assets	Development history	VCS Development History Graph	Filter activating option "Show releases"	Star Icon (depicts a VCS project version that was released)	• Which versions of this asset were released?
A-DH-DHG-PI_Color_HMF	History (from Metadata Exploration)	Assets	Development history	VCS Development History Graph	Optionally filter by producer / Filter by axis segment (slider?) or, if this information is available, filter by version interval	Producer Icon (depicts a VCS project version committed by a producer) AND Color (differentiates each producer) AND Highlight filter (showing parts of the development history that match the filtering criteria) OR Mitigation (hiding the ones who do not match the filtering criteria) (could use color too)	• To which parts of the development history of this asset [version] did each producer contribute?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
A-R-RHG-SI	History (from Metadata Exploration)	Assets	Releases	VCS Release History Graph	--	Star Icon (depicts a VCS project version that was released)	• Which versions of this asset were released?
A-I-BC-Color	Metadata Exploration	Assets	Issues	Bubble Chart	Filter by type (already depicted by the bubble icon)	Color (indicates the status)	• Among the reported bugs, improvement suggestions, or feature requests related to this asset [version], are most of them fixed or open?
A-I-BC-S	Metadata Exploration	Assets	Issues	Bubble Chart	Filter by type (already depicted by the bubble icon)	Size (indicates the time since the issue was created)	• How long does it take for producers of this asset to fix reported bugs?
A-I-LC	Metadata Exploration	Assets	Issues	Line chart	Filter by type (already depicted by the bubble icon)	Line path	• How often do producers of this asset fix reported bugs? • How often do producers of this asset implement improvement suggestions or feature requests?
A-D-EG	Metadata Exploration	Assets	Dependencies	Egocentric graph	--	Nodes (depict assets that are depended or dependent on other assets) AND Arrows (point to the depended asset)	• Which [versions of] assets were reused in the development of this asset [version]? (CAVE) • Which [versions of] assets does this asset [version] depend on? • Which [versions of] assets depend on this asset [version]?
A-D-COM	Metadata Exploration	Assets	Dependencies	Co-Occurrence Matrix	Filter by asset	Matrix cell content matching row and column represents assets that depend on other assets - create arrows to point the dependency direction (↑ or ←)	• Which [versions of] assets were reused in the development of which [versions of] assets? (CAVE)
D-BC-Ov	Metadata Exploration	Developers	--	Bubble Chart	--	Bubble (depicts the developers (producers and/or consumers))	No question associated

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
D-BC-HMF	Metadata Exploration	Developers	--	Bubble Chart	Filter by consumers, producers, or both AND Filter by "Consumers who included reusable assets among project releases / Producers who contributed to asset releases"	Highlight filter (showing which are consumers/producers/both) OR Mitigation (hiding the ones who do not match the filtering criteria) (could use color too)	• Which developers are consumers (i.e., have ever reused an asset)? • Which developers are producers (i.e., have ever produced an asset or contributed to the development of an asset)? • Which consumers are also producers? • Which consumers included reusable assets that are among project releases? • Which producers contributed to which releases of [versions of] assets?
D-BC-Cons-S	Metadata Exploration	Developers	--	Bubble Chart	Filter by consumers	Size (indicates number of reuses)	• Which consumers reuse assets most often?
D-BC-Prod-S	Metadata Exploration	Developers	--	Bubble Chart	Filter by producers	Size (indicates number of developments)	• Which producers develop assets most often?
D-AC-BC	Metadata Exploration	Developers	Assets Consumed (Reuse Occurrences)	Bubble Chart	Each asset can be exploded to depict its versions	Bubble (depicts consumed assets)	• Which assets did this consumer reuse? • Which versions of this asset did this consumer reuse?
D-AC-BC-S	Metadata Exploration	Developers	Assets Consumed (Reuse Occurrences)	Bubble Chart	Each asset can be exploded to depict its versions	Size (indicates number of reuses)	• Which [versions of] assets does this consumer reuse most often?
D-AC-BC-LC	Metadata Exploration	Developers	Assets Consumed (Reuse Occurrences)	Line chart	Select asset of interest / Each asset can be exploded to depict its versions	Line path (each line depicts an asset or an asset version)	• How often does this consumer reuse this asset [version]?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
D-AC-A-BC	Metadata Exploration	Developers	Assets Consumed (Reuse Occurrences)	Asset's Bubble Chart, drilling down to Projects Bubble Chart	Select asset of interest / Each asset can be exploded to depict its versions	Bubble (depicts projects in which the consumer reused the asset [version])	• In which projects did this consumer reuse this asset [version]?
D-AC-LC	Metadata Exploration	Developers	Assets Consumed (Reuse Occurrences)	Line chart	Filter by custom interval (month, week etc.)	Increasing/Decreasing pattern of the line path	• How active is this consumer in terms of number of reuses?
D-AC-Proj-BC-RCC-DHG-HMF	History (from Metadata Exploration)	Developers	Assets Consumed (Reuse Occurrences)	Asset's Bubble Chart, drilling down to Project's Range Column Chart, drilling down to VCS Development History Graph	Select asset of interest, then choose "Permanence of this asset in projects over time (included by this consumer)" option, then select project of interest from the Range Column Chart	Highlight filter (showing the versions since the moment the asset was included in this project by this consumer until the moment it was removed, if so) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• For how long (over time) does this project contain this asset included by this consumer?
D-AC-BC-RCC	History (from Metadata Exploration)	Developers	Assets Consumed (Reuse Occurrences)	Range Column Chart	Select asset of interest, then choose "Permanence of this asset in projects over time (included by this consumer)" option	Each line represents a project; Length and position of the column-bars in the x-axis represent the time during which the project contains the asset	• For how long (over time) do projects contain this asset included by this consumer?
D-Proj-BC	Metadata Exploration	Developers	Projects in which this consumer reused assets	Bubble Chart	--	Bubble (depicts projects in which the consumer reused assets)	• In which projects did this consumer reuse [versions of] assets?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
D-Proj-BC-S	Metadata Exploration	Developers	Projects in which this consumer reused assets	Bubble Chart	Filter "Granularity:" by "Assets" or "Asset versions"	Size (indicates the number of [versions of] assets)	• In which projects did this consumer include the largest number of [versions of] assets?
D-Proj-BC-HMF	History (from Metadata Exploration)	Developers	Projects in which this consumer reused assets	Bubble Chart	Filter by "Show only project releases"	Highlight filter (showing only project releases) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• Which projects contain, among their releases, assets included by this consumer?
D-Proj-BC-LC	Metadata Exploration	Developers	Projects in which this consumer reused assets	Line chart	Select project of interest	Line path (each line depicts an asset or an asset version)	• How often does this consumer reuse [versions of] assets in this project (i.e., are there distinct [versions of] assets reused in this project by this consumer)?
D-Proj-BC-RCC	History (from Metadata Exploration)	Developers	Projects in which this consumer reused assets	Range Column Chart	Select project of interest, then choose "Permanence of assets in this project over time (included by this consumer)" option	Each line represents an asset in a given project; Length and position of the column-bars in the x-axis represent the time during which the project contains the asset	• For how long (over time) does this project contain assets included by this consumer?
D-Proj-A-BC	Metadata Exploration	Developers	Projects in which this consumer reused assets	Project's Bubble Chart, drilling down to Assets Bubble Chart	Select project of interest	Bubble (depicts [versions of] assets reused by this consumer in this project)	• Which [versions of] assets did this consumer reuse in this project?
D-Proj-A-BC-HMF	History (from Metadata Exploration)	Developers	Projects in which this consumer reused assets	Project's Bubble Chart, drilling down to Assets Bubble Chart	Select project of interest AND Filter by "Show only project releases"	Highlight filter (showing only project releases) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• Which [versions of] assets included by this consumer are among the releases of this project?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
D-Proj-RA-BC-RCC-DHG-HMF	History (from Metadata Exploration)	Developers	Projects in which this consumer reused assets	Project's Bubble Chart, drilling down to Asset's Range Column Chart, drilling down to VCS Development History Graph	Select project of interest, then choose "Permanence of assets in this project over time (included by this consumer)" option, then select asset of interest from the Range Column Chart	Highlight filter (showing the versions since the moment the asset was included in this project by this consumer until the moment it was removed, if so) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• For how long (over time) does this project contain this asset included by this consumer?
D-Proj-BC-DHG-AIContrib	History (from Metadata Exploration)	Developers	Projects in which this consumer reused assets	Bubble Chart, drilling down to VCS Development History Graph	Filter by projects that have assets included and removed afterwards AND Select project of interest	Bubble (each bubble represents a project in these conditions) AND Asset icon with + or - visible signs (depicts a VCS project version in which the asset was included/excluded)	• Which projects contain, at some point of the development life cycle, [versions of] assets included by this consumer that were removed afterwards?
D-AP-BC_CI	Metadata Exploration	Developers	Assets Produced	Bubble Chart	Each asset can be exploded to depict its versions	Bubble (depicts produced assets or asset versions) AND Contribution Icon (indicates whether the developer owns the asset project or has just contributed to it)	• Which [versions of] assets contain [which kinds of] development contributions made by this producer?
D-AP-BC-S	Metadata Exploration	Developers	Assets Produced	Bubble Chart	Each asset can be exploded to depict its versions	Size (indicates amount of contribution)	• What are the [versions of] assets to which development this producer has most contributed?
D-AP-BC-HMF	Metadata Exploration	Developers	Assets Produced	Bubble Chart	Filter by "Show only asset releases that contain development contributions made by this producer"	Highlight filter (showing asset releases that contain development contributions made by this producer) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• Which released [versions of] assets contain development contributions made by this producer?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
D-AP-BC-RHG-HMF	History (from Metadata Exploration)	Developers	Assets Produced	Bubble Chart, drilling down to VCS Release History Graph	Select asset of interest	Highlight filter (showing releases in which there are development contributions made by this producer) OR Mitigation (hiding the ones who do not match the filtering criteria) (could use color too)	• Which released versions of this asset contain development contributions made by this producer?
D-AP-BC-DHG-PI	History (from Metadata Exploration)	Developers	Assets Produced	Bubble Chart, drilling down to VCS Development History Graph	Select asset of interest	Producer Icon (depicts a VCS project version committed by a producer)	• To which parts of the assets development history did this producer contribute?
D-AP-LC	Metadata Exploration	Developers	Assets Produced	Line chart	Filter by custom interval (month, week etc.)	Increasing/Decreasing pattern of the line path	• How active is this producer in terms of assets' development?
D-AP-BC-DHG-PIContrib	History (from Metadata Exploration)	Developers	Assets Produced	Bubble Chart, drilling down to VCS Development History Graph	Filter by assets whose projects contain contributions from this producer that started contributing and stopped afterwards AND Select asset [version] of interest	Bubble (each bubble represents an asset in these conditions) AND Producer Icon with + or - visible signs (depicts a VCS project version in which the producer started/stopped contributing)	• Which [versions of] assets contain some contribution made by this producer but such producer stopped contributing afterwards?
D-PI-BC-Color	Metadata Exploration	Developers	Production Issues	Bubble Chart	Filter by type (already depicted by the bubble icon)	Color (indicates the status)	• Among the reported bugs, improvement suggestions, or feature requests assigned to this producer, are most of them fixed or open?
D-PI-BC-S	Metadata Exploration	Developers	Production Issues	Bubble Chart	Filter by type (already depicted by the bubble icon)	Size (indicates the time since the issue was created)	• How long does it take for this producer to fix reported bugs?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
D-PI-LC	Metadata Exploration	Developers	Production Issues	Line chart	Filter by type (already depicted by the bubble icon)	Line path	• How often does this producer fix reported bugs? • How often does this producer implement improvement suggestions or feature requests?
D-Corr-BC	Metadata Exploration	Developers	Correlations with Consumers	Bubble Chart	Filter by "Consumers who reused the same assets", "Consumers who reused assets developed by this producer"	Bubble (depicts developers that correlate somehow with other developers, according to the defined filter	• Which consumers reused the same [version(s) of] asset(s) reused by this consumer? • Which consumers reused [version(s) of] asset(s) developed by this producer?
D-Corr-COM	Metadata Exploration	Developers	Correlations with Consumers	Co-Occurrence Matrix	Filter by "Consumers who reused the same assets", "Consumers who reused assets developed by producers" / Filter by specific consumer/producer	Matrix cell content matching row and column represents developers that correlate somehow with other consumers, according to the defined filter - for producers, create arrows to point the producer starting from the consumer (↑ or ←)	• Which consumers reused the same [version(s) of] asset(s) reused by which consumers? • Which consumers reused [version(s) of] asset(s) developed by which producers?
D-PCollab-EG	Metadata Exploration	Developers	Development Collaborations (in Asset Productions)	Egocentric graph	Filter by "Collaboration in projects owned by this producer" and "Collaborations in projects owned by others"	Nodes (depict producers that collaborate with the selected producer) AND Arrows (point to the owner of the asset project)	• Who are the producers with whom this producer contributes/collaborates (in assets development)? • Which producers contribute/collaborate in the development of assets developed by this producer?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
D-PCollab-COM	Metadata Exploration	Developers	--	Co-Occurrence Matrix	Filter by producer (and kind of collaboration?)	Matrix cell content matching row and column represents producers that collaborate somehow with other producers, according to the defined filter	• Which producers contribute/collaborate with which producers in assets development?
P-BC-Ov	Metadata Exploration	Projects	--	Bubble Chart	--	Bubble (depicts the projects)	No question associated
P-BC-HMF	Metadata Exploration	Projects	--	Bubble Chart	Filter by projects with reusable assets OR Filter by projects with reusable assets in their releases	Highlight filter (showing which projects contain reusable assets) OR Mitigation (hiding the ones who do not match the filtering criteria) (could use color too)	• Which projects have ever had an asset included (i.e., contain at least one reusable asset in their development history)? • Which projects contain reusable assets among their releases?
P-BC-S	Metadata Exploration	Projects	--	Bubble Chart	Automatically filter by projects that contain reusable assets	Size (indicates number of reusable assets)	• In which projects assets are most often reused (i.e., which projects contain the largest number of assets)?
P-RA-BC	Metadata Exploration	Projects	Reused Assets	Bubble Chart	Each asset can be exploded to depict its versions	Bubble (depicts assets reused in this project)	• Which assets were reused in this project? • Which versions of this asset were reused in this project?
P-RA-BC-S	Metadata Exploration	Projects	Reused Assets	Bubble Chart	--	Size (represents the number of distinct versions of an asset in this project)	• Which assets are most often reused in this project (i.e., have the largest number of distinct versions included in this project)?
P-RA-BC-LC	Metadata Exploration	Projects	Reused Assets	Bubble Chart	Select asset of interest	Line path (each line depicts an asset version)	• How often is this asset reused in this project (i.e., are there distinct versions of this asset reused in this project)?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
P-RA-BC-HMF	History (from Metadata Exploration)	Projects	Reused Assets	Bubble Chart	Filter by "Assets that are among releases of this project"	Highlight filter (showing which assets are among releases of this project) OR Mitigation (hiding the ones who do not match the filtering criteria) (could use color too)	• Which [versions of] assets are among the releases of this project?
P-RA-BC-DHG-HMF	History (from Metadata Exploration)	Projects	Reused Assets	VCS Development History Graph	Select project of interest, then choose "Permanence of this asset in this project over time" option	Highlight filter (showing the versions since the moment the asset was included in this project until the moment it was removed, if so) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• For how long (over time) does this project contain this asset?
P-RA-C-BC	Metadata Exploration	Projects	Reused Assets	Asset's Bubble Chart, drilling down to Consumers Bubble Chart	Select asset [version] of interest	Bubble (depicts consumers who reused the asset [version] in the project)	• Which consumers reused this asset [version] in this project?
P-RA-C-BC-DHG-HMF	History (from Metadata Exploration)	Projects	Reused Assets	Asset's Bubble Chart, drilling down to Consumer's Bubble Chart, drilling down to VCS Development History Graph	Select asset of interest, then select consumer of interest	Highlight filter (showing the versions since the moment the asset was included in this project by this consumer until the moment it was removed, if so) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• For how long (over time) does this project contain this asset included by this consumer?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
P-RA-BC-DHG-AIContrib	History (from Metadata Exploration)	Projects	Reused Assets	Bubble Chart, drilling down to VCS Development History Graph	Filter by assets included and removed afterwards AND Select asset [version] of interest	Bubble (each bubble represents an asset in these conditions) AND Asset icon with + or - visible signs (depicts a VCS project version in which the asset was included/excluded)	• Which [versions of] assets were included at some point of the development life cycle of this project but were removed afterwards?
P-C-BC	Metadata Exploration	Projects	Consumers who reused assets in this project	Bubble Chart	Each asset can be exploded to depict its versions	Bubble (depicts consumers who reused assets in this project)	• Which consumers reused [versions of] assets in this project?
P-C-BC-S	Metadata Exploration	Projects	Consumers who reused assets in this project	Bubble Chart	--	Size (indicates number of reuses in this project)	• Who are the main consumers in this project (i.e., the consumers who most often included different [versions of] assets in this project)?
P-C-BC-HMF	Metadata Exploration	Projects	Consumers who reused assets in this project	Bubble Chart	Filter by "Show only project releases"	Highlight filter (showing only project releases) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• Which consumers included assets that are among the releases of this project?
P-C-A-BC	Metadata Exploration	Projects	Consumers who reused assets in this project	Consumer's Bubble Chart, drilling down to Assets Bubble Chart	Select consumer of interest	Bubble (depicts [versions of] assets reused by this consumer in this project)	• Which [versions of] assets did this consumer reuse in this project?
P-C-A-BC-HMF	Metadata Exploration	Projects	Consumers who reused assets in this project	Consumer's Bubble Chart, drilling down to Assets Bubble Chart	Select consumer of interest AND Filter by "Show only project releases"	Highlight filter (showing only project releases) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• Which [versions of] assets included by this consumer are among the releases of this project?

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
P-C-BC-DHG-AIContrib	History (from Metadata Exploration)	Projects	Consumers who reused assets in this project	Bubble Chart, drilling down to VCS Development History Graph	Filter consumers who included assets that were removed afterwards AND Select consumer of interest	Bubble (each bubble represents a consumer in these conditions) AND Asset icon with + or - visible signs (depicts a VCS project version in which the asset was included/excluded)	• Which consumers included, at some point of the development life cycle of this project, [versions of] assets that were removed afterwards?
P-C-BC-RCC	History (from Metadata Exploration)	Projects	Consumers who reused assets in this project	Range Column Chart	Select project of interest, then choose "Permanence of assets in this project over time (included by this consumer)" option	Each line represents an asset in a given project; Length and position of the column-bars in the x-axis represent the time during which the project contains the asset	• For how long (over time) does this project contain assets included by this consumer?
P-C-AC-BC-RCC-DHG-HMF	History (from Metadata Exploration)	Projects	Consumers who reused assets in this project	Consumer's Bubble Chart, drilling down to Asset's Range Column Chart, drilling down to VCS Development History Graph	Select consumer of interest, then choose "Permanence of assets in this project over time (included by this consumer)" option, then select asset of interest from the Range Column Chart	Highlight filter (showing the versions since the moment the asset was included in this project by this consumer until the moment it was removed, if so) OR Mitigation (hiding the ones who do not match this filtering criterion) (could use color too)	• For how long (over time) does this project contain this asset included by this consumer?
P-R-RHG-Pos	History (from Metadata Exploration)	Projects	Releases	VCS Release History Graph	--	Position of elements in the x-axis (which represents time)	No question associated

Visualization ID	Perspective	Core Element	Contextual Element	Visualization metaphor	Interaction	Data and Visualization attribute	Questions it answers
P-R-RHG-SI	History (from Metadata Exploration)	Projects	Releases	VCS Release History Graph	--	Star Icon (depicts a VCS project version that was released)	No question associated
P-Corr-BC	Metadata Exploration	Projects	Correlations with Other Projects	Bubble Chart	--	Bubble (depicts projects that contain the same asset(s) of the selected project)	• Which projects contain the same [version(s) of] asset(s) reused in this project?
P-Corr-COM	Metadata Exploration	Projects	--	Co-Occurrence Matrix	Filter by project	Matrix cell content matching row and column represents projects that contain the same asset(s) of another project, according to the defined filter	• Which projects contain the same [version(s) of] asset(s) reused in another project?

APPENDIX B – QUESTIONNAIRE FOR EVALUATING THE VISUALIZATION FEATURE MODEL

B.1 Part 1/5: Characterization

Academic degree:
○ Ph.D. Degree
○ Ph.D. Student
○ Master Degree
○ Master Student
○ Bachelor Degree
○ Undergraduate Student

Please fill out your level of experience with INFORMATION VISUALIZATION.
Please check all the options that apply.
□ None (if you choose this option, please do not choose any other one)
□ I have a superficial knowledge about this topic
□ I have a good knowledge about this topic
□ I studied this topic in a course/discipline
□ I studied this topic by reading one or more books
□ I used my knowledge about this topic in the context of a course in practice
□ I used my knowledge about this topic in personal projects
□ I used my knowledge about this topic in industry projects

Please fill out your level of experience with COMPUTER-HUMAN INTERACTION (CHI).
Please check all the options that apply.
□ None (if you choose this option, please do not choose any other one)
□ I have a superficial knowledge about this topic
□ I have a good knowledge about this topic
□ I studied this topic in a course/discipline
□ I studied this topic by reading one or more books
□ I used my knowledge about this topic in the context of a course in practice
□ I used my knowledge about this topic in personal projects
□ I used my knowledge about this topic in industry projects

Please fill out your level of experience with FEATURE MODELING.
Please check all the options that apply.
□ None (if you choose this option, please do not choose any other one)
□ I have a superficial knowledge about this topic
□ I have a good knowledge about this topic
□ I studied this topic in a course/discipline
□ I studied this topic by reading one or more books
□ I used my knowledge about this topic in the context of a course in practice
□ I used my knowledge about this topic in personal projects
□ I used my knowledge about this topic in industry projects

B.2 Part 2/5: Overview of the Visualization Feature Model

The following figure shows the current version of the feature model, depicting its elements and their relationships. The notation used is described in the legend. For a better visualization of the

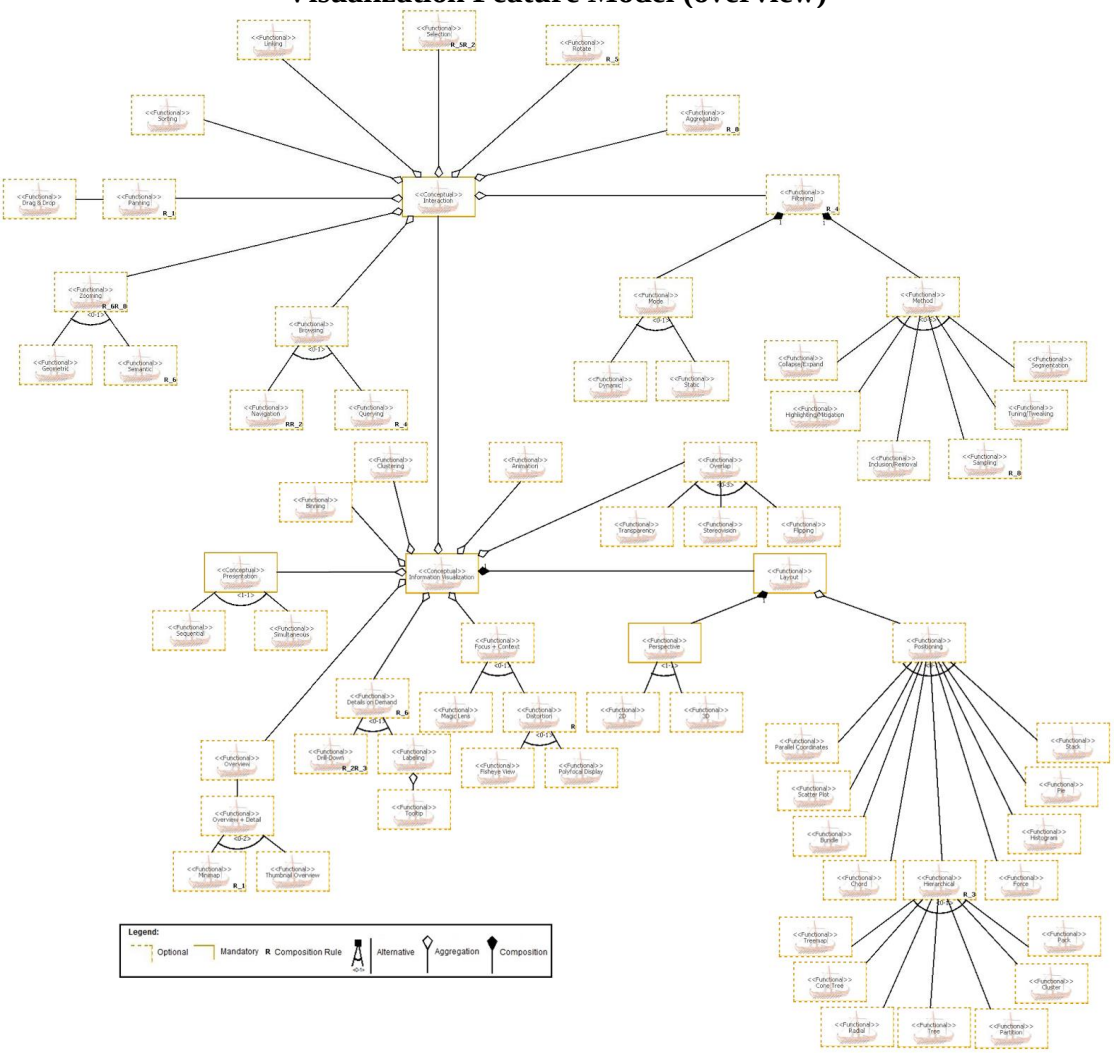
feature model as a whole, please visit http://www.cos.ufrj.br/~schots/papers/featuremodel.png and save the image or use the zoom-in feature in your web browser.

Please notice that we are not evaluating the completeness of the visualization feature model. This model is expected to evolve along time. The focus of this evaluation is to obtain feedback regarding the classification or the currently identified features and obtain suggestions of additional features in a particular category/group.

Any comments regarding the evaluation or the notation used should be kept for the comments section, at the end of this form. For now, we are only interested in answering the questions that will help us evaluate the model.

Thank you.

Visualization Feature Model (overview)[70]



Please choose one of the following groups of features to evaluate:

After selecting the group and start filling out the form, please do not press the "back" button in your web browser for evaluating another group, otherwise all your previous responses will not be saved. Thank you.

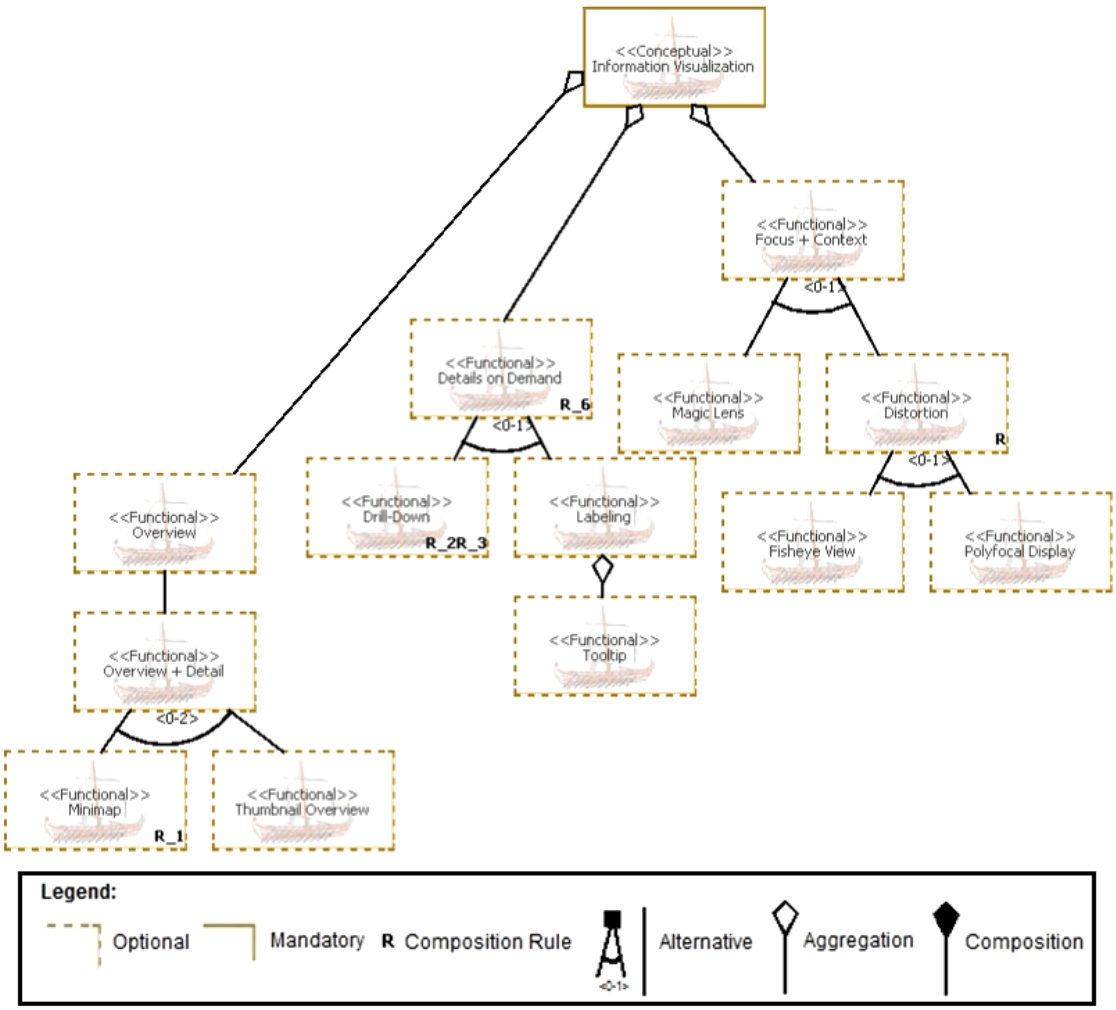
[70] This figure may not be legible in this page. Please refer to Figure 4.16 and Figure 4.17 in this thesis, which represent a larger version the same model, or refer to an electronic version (available at http://www.cos.ufrj.br/~schots/papers/featuremodel.png) for a better visualization.

○ Information Visualization – Focus + Context, Overview + Detail, and Details on Demand features
○ Information Visualization – Hierarchical Layout and Perspective features
○ Information Visualization – Layout features
○ Information Visualization – Other features
○ Interaction – Filtering features
○ Interaction – Panning, Browsing, and Zooming features
○ Interaction – Other features

B.3 Part 3/5: Evaluation of the Visualization Feature Model

Information Visualization – Focus + Context, Overview + Detail, and Details on Demand features

This is an excerpt of the Visualization Feature Model. Please read carefully the description of the features in order to answer the questions stated in this form. Also, please check if the relationships between elements in the model are represented correctly (taking into account the notation used, explained by the legend in the model).



The description of each feature is presented to allow a proper understanding of their characteristics and constraints of use (i.e., its use may require or exclude the use of another feature).

While reading this form, please keep in mind that the following defects are under evaluation [de Mello et al. 2014]:

- Omission: Some information from the domain was not properly included in the feature model.
- Incorrect Fact: Some information or behavior in the feature model contradicts its domain specification.
- Inconsistency: Some feature model element is not consistent with another element from the same feature model.
- Ambiguity: Some Information from the feature model is not clear, allowing multiple interpretations for the specified domain.
- Extraneous Information: Some information in the feature model is outside the domain scope.

IMPORTANT OBSERVATION: The questions for evaluating incorrect facts are listed throughout the descriptions, while the questions for the remaining defects are listed at the end of this page. Thus, if any ambiguity, inconsistency, omission, or extraneous information is identified at any time while you are reading the form, we recommend that you instantly move to the end of the page and indicate the problems identified, in order to avoid any forgetfulness.

Thank you.

Reference: [de Mello et al. 2014] de Mello, R. M., Teixeira, E. N., Schots, M., Werner, C. M. L., Travassos, G. H. (2014). "Verification of Software Product Line Artefacts: A Checklist to Support Feature Model Inspections". Journal of Universal Computer Science, v. 20, n. 5, pp. 720-745.

A.3.1. Minimap[71]

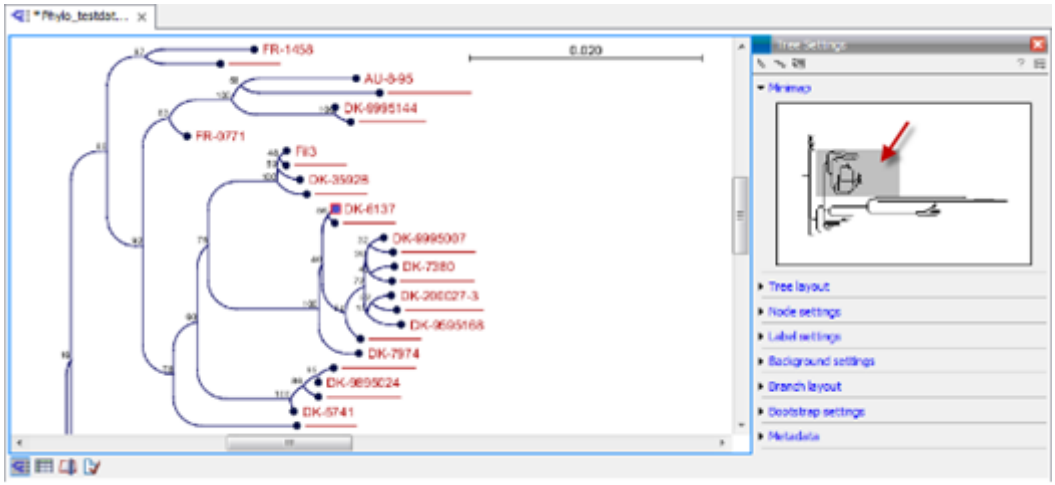
Definition: A minimap (also known as miniview) comprises a scaled down version (overview) of the data with an indication of the current viewport (detail), partially overlaid on top of the viewport in order to provide user orientation [Roto et al. 2006]. The viewport is the main view area, responsible for the details in the visualization [Oliveira 2011]. If there are constraints in the display dimensions, a minimap can be adopted while scrolling a visualization to present an overview of the content to the user [Roto et al. 2006]. Due to the reduced scale, a minimap can present a slightly modified version of the original content, emphasizing items whose identification in the minimap context is relevant [Oliveira 2011]. It is common to use geometric shapes (usually rectangles) to indicate the part of the overview where the current location of the detailed view is [Roto et al. 2006].

Constraints: Composition Rule R_1 – (Minimap) requires (Panning) [Roto et al. 2006].

References:

- [Roto et al. 2006] Roto, V., Popescu, A., Koivisto, A., Vartiainen, E. (2006). "Minimap: a Web Page Visualization Method for Mobile Phones". In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2006), Montreal, Canada, pp. 35-44, April.
- [Oliveira 2011] Oliveira, M. S. (2011). "PREViA: An Approach for Visualizing the Evolution of Software Models" [PREViA: Uma Abordagem para a Visualização da Evolução de Modelos de Software] (in Portuguese), M.Sc. Thesis, COPPE/UFRJ, Rio de Janeiro, Brazil, March.

[71] This is just an example of the evaluated features. The full set can be found in [Schots et al. 2015].



Minimap (source: http://clcsupport.com/phylogenymodule/current/Minimap.html)

A.3.1-Q1) Is the textual description of this feature correct and complete enough to understand its meaning?
If the feature is not well described textually, please answer "No" and provide more information about your opinion in the last question (Q4) corresponding to this category. Optionally, feel free to recommend publications that can help describing it, if you want to.
○ Yes
○ I don't know / I am not sure
○ No

A.3.1-Q2) Is the figure used to represent this feature clear enough to understand its usage?
If the feature has a figure that does not clearly allow understanding its usage, please answer "No" and provide more information about your opinion in the last question (Q4) corresponding to this category. Optionally, feel free to recommend publications or links to figures that can help describing it, if you want to.
○ Yes
○ I don't know / I am not sure
○ No

A.3.1-Q3) Are the constraints associated to this feature correctly identified and described?
If you think that (i) a constraint is incorrect or (ii) a constraint exists but was not described, please answer "No" and explain your answer in the last question (Q4) corresponding to this category. Optionally, feel free to recommend publications that can help describing it, if you want to.
○ Yes
○ I don't know / I am not sure
○ No

A.3.1-Q4) For each negative answer to the questions related to this feature, please provide more information according to the instructions given in such questions.
If no problems were identified (i.e., if there was no negative answer), please write "N/A".

B.4 Part 4/5: Complementary Check for Clarity and Correctness

In the model, are the relationships between the features identified and described correctly?
If you think that (i) a relationship is incorrect or (ii) a relationship exists but was not identified or described correctly, please answer "No" and explain your answer in the following question.

	Yes	I don't know / I am not sure	No
A.1. Focus + Context	○	○	○
A.1.1. Magic Lens	○	○	○
A.1.2. Distortion	○	○	○
A.1.2.1. Fisheye View	○	○	○
A.1.2.2. Polyfocal Display	○	○	○
A.2. Overview	○	○	○
A.3. Overview + Detail	○	○	○
A.3.1. Minimap	○	○	○
A.3.2. Thumbnail Overview	○	○	○
A.8. Details on Demand	○	○	○
A.8.1. Drill-Down / Roll-Up	○	○	○
A.8.2. Labeling	○	○	○
A.8.2.1. Tooltip	○	○	○

For each of the Focus + Context, Overview + Detail, and Details on Demand features whose relationship(s) are not described correctly or are missing, please provide more information about your opinion. Optionally, feel free to recommend publications that can help describing it, if you want to.
If no problems were identified with the relationships, please write "N/A".

_____

Check for Ambiguity

Are there different Focus + Context, Overview + Detail, and Details on Demand features in the model representing the same domain concept? If so, please indicate (i) the features to which this problem applies, and (ii) the reasons why you consider them ambiguous. Optionally, you can suggest ways to remove the identified ambiguities.
If no ambiguity was identified, please write "N/A".

_____

Check for Inconsistency

Are there Focus + Context, Overview + Detail, and Details on Demand features in the model contradicting other features? If so, please indicate (i) the features to which this problem applies, and (ii) the reasons why you think there are contradictions. Optionally, you can suggest ways to remove the identified contradictions.
If no inconsistency was identified, please write "N/A".

_____

Check for Omission

Is there any domain concept related to the Focus + Context, Overview + Detail, and Details on Demand features that has been omitted from the model? If so, please indicate (i) the name of this concept, (ii) a brief description of it, and (iii) where it should be located (e.g., as a variant of feature A or as a child of feature B). Optionally, you can state the reason why you think it should be included in the feature model, or recommend publications that present and/or describe the missing feature.
If no omission was identified, please write "N/A".

_____

Check for Extraneous Information

Is there any feature among the Focus + Context, Overview + Detail, and Details on Demand features that, regardless of being described correctly or not, seem to be out of the domain scope? If so, please indicate (i) which are these features, and (ii) the reasons why you consider it out of scope.
If no extraneous information was identified, please write "N/A".

_____

B.5 Part 5/5: Follow-Up

General comments
If you already evaluated another category, there is no need to answer this again – but please, inform this as the answer to the first question. Thank you.

In which scenarios do you think the feature model can be used?
For answering this question, please take into account the idea of the feature model, not only its current version.

_____

Do you have suggestions for improving the description of the feature model? If so, please state them.

_____

Do you have suggestions for improving the research and evolution of the feature model? If so, please state them.

_____

If you have any additional comments, please state them.

_____

APPENDIX C – INSTRUMENTS USED IN THE ZOOMING BROWSER EVALUATION

C.1 Characterization Questionnaire

C.1.1 Part 1/3: Characterizing the participant's background

Academic level background
○ Undergraduate course (ongoing)
○ Undergraduate course (finished)
○ Specialization course (ongoing)
○ Specialization course (finished)
○ Master course (ongoing)
○ Master course (finished)
○ Ph.D. course (ongoing)
○ Ph.D. course (finished)

What is your experience with object-oriented (OO) software development?
Please check all the items that apply
□ I have read materials about OO development
□ I have attended an OO development course
□ I have never developed OO software
□ I have developed OO software for personal use
□ I have developed OO software in the context of a course
□ I have developed OO software as part of a team in industry

Please detail your answer. Include the number of months or number of years of relevant experience in software development.
For instance, "I worked for two years and three months as a programmer in industry"
_____

What is your current occupation?
_____

In what organization you currently work (or what was the last software development organization in which you worked), and for how long (years / months)?
Information about the organization will be kept confidential and is only accounted for characterization purposes of the organization's profile and operational domain
_____

Please indicate your level of familiarity with respect to the following items, based on the presented scale:

	I have no expertise in this topic	I believe I have a basic expertise level in this topic	I believe I have an intermediate expertise level in this topic	I believe I have an advanced expertise level in this topic
Software development (programming)	○	○	○	○
Software reuse	○	○	○	○

	I have no expertise in this topic	I believe I have a basic expertise level in this topic	I believe I have an intermediate expertise level in this topic	I believe I have an advanced expertise level in this topic
Version control systems	○	○	○	○
Issue tracking or task management systems	○	○	○	○
Software project management	○	○	○	○
Software visualization	○	○	○	○
English fluency (reading)	○	○	○	○

What is your knowledge about the MPS.BR maturity model (or a correlated standard/model that encompasses software reuse processes)?
○ I have no knowledge about any maturity model involving reuse processes
○ I have heard or have read about it
○ I have studied about it in the context of a course
○ I have done some work involving this subject
○ I have participated in an implementation/assessment of this model, but at a level that did not involve reuse processes
○ I have participated in an implementation/assessment of this model at a level that involved reuse processes, but I did not worked directly in these processes
○ I have participated in an implementation/assessment of this model at a level that involved reuse processes, and I worked directly in the execution of these processes

C.1.2 Part 2/3: Characterizing the organization

All data will be made anonymous, so that it cannot be possible to identify neither the participant nor the organization. Besides, no one other than the researcher responsible for this work will have access to these data under any circumstances.

Do you think the reuse initiatives (if any) in the organization you work for are effective?
○ There are no reuse initiatives in my organization
○ There are reuse initiatives in my organization, and I think they are not effective
○ There are reuse initiatives in my organization, and I think they are partially effective
○ There are reuse initiatives in my organization, and I think they are effective

Regardless of being effective, what are the problems or drawbacks of these initiatives?
If you think there are no problems or drawbacks, please write "N/A".

_____

What would you change? Which improvements would you make?
If you think there is nothing to change, please write "N/A".

_____

Could you point out success factors that make (or would make) the initiatives more effective, even if they are not already fully realized?
If you cannot identify any success factor, please write "N/A".

_____

If your organization has a reuse repository, could you explain what the reuse repository is (e.g., a tool, a shared folder, a database)?
If the organization does not have a reuse repository, please write "N/A".

_____

If your organization has a reuse repository, how are the reusable assets stored? How are they retrieved?
If the organization does not have a reuse repository, please write "N/A".

_____

How often do you think the organization members attempt to reuse existing assets? Why?

_____

C.1.3 Part 3/3: Characterizing the participant in the software development context

What is your knowledge about the role of a reuse manager?
○ I am unaware of that role
○ I have heard or have read about it
○ I am aware of the assignments of this role
○ I worked with someone who played this role in at least one organization
○ I already played that role in at least one organization

If you already performed this role in an organization, please describe what your responsibilities were.
In case you never performed the reuse manager role, please write "N/A".

_____

What sources do you use for obtaining reusable assets?

_____

Which steps do you perform when you need or intend to reuse an asset?

_____

Do you have any difficulty in performing these steps? If so, please describe them.

_____

What information do you actually take into account for deciding whether or not to reuse an asset?

_____

Is there any additional information that you think is relevant for deciding whether or not to reuse an asset?

_____

C.2 On the Relevance of Information/Metadata for Software Reuse

By answering the following question, keep in mind your current beliefs about each information. The purpose of this question is not to find out if this information may be relevant, but if actually they are at the moment for the respondent.

Among the following information, give a scale on how relevant you think they are taking a reuse decision (e.g., for deciding whether to reuse an asset).
By reusable asset, please consider any kind of artifact that can be reused in software development, especially the ones with which you are used to.

	Totally relevant	Quite relevant	Somewhat irrelevant	Totally irrelevant
Organization that developed the asset	○	○	○	○
Asset producers (developers who created or contributed to the asset development)	○	○	○	○
Asset consumers (developers who reused the asset)	○	○	○	○
Asset producers' contact information	○	○	○	○
Asset consumers' contact information	○	○	○	○
Number of reuse occurrences of the asset	○	○	○	○
Asset development history (commit history)	○	○	○	○
Asset release history	○	○	○	○
Asset issues (bugs, feature requests etc.)	○	○	○	○
Asset dependencies on other assets	○	○	○	○
Other assets that depend on the asset	○	○	○	○
Projects in which the asset was reused	○	○	○	○
Asset license	○	○	○	○

C.3 Descriptions of Tasks

The participants who performed the role of reuse manager received the scenario (and associated questions) described in Section C.3.1, while the participants who performed the software developer role received the scenario (and associated questions) described in Section C.3.2.

C.3.1 Reuse manager

1) You were hired as the **reuse manager** of Melhorandus and you are responsible for **checking if the ongoing reuse initiatives are being effective** and **communicating it to the top management**. To this end, you have to answer the following questions:

ID	Question	How to answer
RM1a	What information would you consider for performing this task?	N/A
RM1b	Which assets have ever been reused?	Metadata Exploration
RM1c	Which assets are most often reused?	Dashboard / Metadata Exploration
RM1d	Considering only the asset most often reused, how often is it reused over time?	Dashboard
RM1e	How many and which assets were reused in the projects "FeatSelect", "Travel4All", and "ZombieBattle3"?	Reuse Map
RM1f	Who are the 3 consumers who reuse assets more often?	Dashboard / Metadata Exploration
RM1g	Among the most reused assets, which of them were reused by the 3 most active asset producers?	Dashboard
RM1h	Which assets were reused by which consumers in the 3 projects that contain more assets?	Dashboard + Reuse Map
RM1i	What would you do in this regard?	N/A

2) The asset JUnit was identified as a candidate for entering the reuse repository. In order to approve it, you need to **assess some of its properties** by answering the following questions:

ID	Question	How to answer
RM2a	What information would you consider for performing this task?	N/A
RM2b	How active is the release history of this asset?	History View (Releases)
RM2c	Which producers contributed to the development of this asset?	Metadata Exploration
RM2d	Among the reported bugs, improvement suggestions, or feature requests related to this asset, are most of them open or fixed?	Metadata Exploration
RM2e	How often do producers of this asset fix reported bugs?	Metadata Exploration
RM2f	How long (in average) does it take for producers of this asset to fix reported bugs?	Metadata Exploration
RM2g	What would you do in this regard?	N/A

3) After all, it was decided that JUnit should be included in the repository. Its newest version was developed in order to fix a severe bug detected in the previous version. In order to **attest that all the organization projects that run the previous version were upgraded to the newest one**, you have to provide answers to the following questions:

ID	Question	How to answer
RM3a	What information would you consider for performing this task?	N/A
RM3b	What is the latest version of the asset?	History View (Releases)
RM3c	What is the version immediately before the latest one?	History View (Releases)
RM3d	When was the latest version of this asset released?	History View (Releases)
RM3e	Is version 4.12-beta-3 still being reused by some project? How many? (please assume that the other projects that reused previous versions are no longer being maintained).	Metadata Exploration
RM3f	What would you do in this regard?	N/A

C.3.2 Software developer

1) You were hired as a **developer** of Melhorandus and you are responsible to include unit testing functionalities to the XHealth project. Some organization members (*felixge* and *jasondavies*) suggested JUnit. In order to **decide whether it is worth or not to reuse it in XHealth**, you have to provide answers to the following questions:

ID	Question	How to answer
SD1a	What information would you consider for performing this task?	N/A
SD1b	Was this asset already reused in the organization? If so:	Dashboard / Metadata Exploration
SD1c	Which versions of this asset were reused?	Metadata Exploration
SD1d	What are the 3 most often reused versions of this asset?	Metadata Exploration
SD1e	Which consumers reused this asset in which projects?	Metadata Exploration
SD1f	How active is the release history of this asset?	History View (Releases)
SD1g	Among the reported bugs, improvement suggestions, or feature requests related to this asset, are most of them open or fixed?	Metadata Exploration
SD1h	How often do producers of this asset fix reported bugs?	Metadata Exploration

SD1i	How long (in average) does it take for producers of this asset to fix reported bugs?	Metadata Exploration
SD1j	How often do producers of this asset implement improvement suggestions or feature requests?	Metadata Exploration
SD1k	What would you do in this regard?	N/A

2) The project team required you to incorporate JUnit into the XHealth project (regardless of your previous answer), but you do not know where to start from. You have to **overcome this difficulty and reuse it in the XHealth project**.

ID	Question	How to answer
SD2a	What information would you consider for performing this task?	N/A
SD2b	Which producers contributed to the development of this asset?	Metadata Exploration
SD2c	Who are the 3 main producers of this asset (i.e., the producers who most contributed to the development of this asset)?	Metadata Exploration
SD2d	Was this asset already reused in the organization? If so:	Dashboard / Metadata Exploration
SD2e	Which consumers reused this asset?	Metadata Exploration
SD2f	Who are the 3 main "reusers" of this asset (i.e., the consumers who reused this asset more often)?	Metadata Exploration
SD2g	What would you do in this regard?	N/A

C.4 Follow-Up Questionnaire

This is the last stage of the study. Please provide information that can help improving the approach. Feel free to suggest, in the corresponding field, future work or developments that you consider relevant.

Based on your experience with Zooming Browser, please classify your perception of the following aspects according to the provided scale:

	Very high	High	Medium	Low	Very low
Perceived usefulness of the presented information	○	○	○	○	○
Perceived usefulness of the presented visualizations	○	○	○	○	○
Perceived usefulness of the employed interaction resources	○	○	○	○	○
Perceived efficiency of the tool (related to your expectations for solving similar problems)	○	○	○	○	○
Perceived easiness in performing the study tasks	○	○	○	○	○

Do you think the executed tasks match the day-to-day reality of the role you performed? Please, provide some feedback in this regard.

_____

What are the perceived difficulties identified in performing the tasks?

_____

What are the benefits of the tool, if any?

_____

What are the drawbacks of the tool, if any?

_____

In which aspects the tool can be improved?

_____

Regarding the relevance of reuse-related information, would you change any of the answers you gave after using the tool? If so, which ones and why?

_____

Is there anything that has not been asked and you would like to say?
If there is nothing else to say, please write "N/A".

_____