An Information Visualization Feature Model for Supporting the Selection of Software Visualizations

Renan Vasconcelos Marcelo Schots Cláudia Werner

Systems Engineering and Computer Science Program – COPPE/UFRJ
PO Box 68,511 – CEP 21945-970 – Rio de Janeiro, RJ, Brazil
{renanrv,schots,werner}@cos.ufrj.br

ABSTRACT

Software development comprises the execution of a variety of tasks, such as bug discovery, finding reusable assets, dependency analysis etc. A better understanding of the task at hand and its surroundings can improve the development performance in general. Software visualizations can support such understanding by addressing different issues according to the necessity of stakeholders. However, knowing which visualizations better fit a given task in progress is not a trivial skill. In this sense, a feature model, intended for organizing the knowledge of a given domain and allowing the reuse of components, can support the identification, categorization and selection of information visualization elements. This work presents an ongoing domain analysis performed for building an information visualization feature model, whose goal is to support the process of choosing and building proper, suitable software visualizations.

Categories and Subject Descriptors

D.2.6 [**Software Engineering**]: Programming Environments – *graphical environments*; D.2.13 [**Software Engineering**]: Reusable Software – *reusable libraries*.

General Terms

Management, Measurement, Experimentation, Human Factors, Verification.

Keywords

Software visualization, Feature model, Awareness.

1. INTRODUCTION AND MOTIVATION

In order to improve software development, it is wise to identify the different characteristics that belong to the scenario at hand and, if possible, watch their behavior dynamically. The analysis of this information can provide means to manage a development scenario and avoid or mitigate potential problems. Despite the involvement of each stakeholder (e.g., a project manager or a programmer), awareness is a key factor to a structured development process [12].

Visualization metaphors and techniques represent alternative means to improve awareness. Information visualization can have an impact on assisting software development activities that involve human reasoning. The use of abstractions can help developers understand different aspects of the development process, especially when the amount of data to analyze becomes large. For instance, class diagrams provide an abstraction to understand software architectures and share a common idea among developers.

Although there are several visualization elements[1] available, it is not a simple task to choose a visualization that will meet all the expectations and represent everything needed. Since the number of visualization alternatives keeps growing, it is important to adopt some sort of mechanism for selecting the most suitable ones, i.e., making an appropriate choice. In this sense, by selecting only the necessary features, visualizations can be composed with less effort.

Feature models are a useful way to represent domain knowledge in terms of the elements (features) and their relationships and constraints, facilitating the understanding. An advantage of this approach lies on the acceptance of features as an effective "media" supporting communication among stakeholders [3]. Considering each visualization and interaction element as a single feature, domain engineering applied to information visualization can be a way to select or build a view according to awareness requirements.

In this sense, this paper presents a domain analysis that has been carried out in order to identify different characteristics in a visualization, organizing them into a feature model [7] for easing their selection and organization. A visualization feature model can favor building different views that address the same issue but are intended for different stakeholders with their specific analysis perspectives. For instance, a project manager may prefer a high-level analysis, with an overview of the scenario as a whole, while a fine-grained visualization may better fit a developer's necessity, highlighting details. This is handled by CAVE (Context-Aware Visualization Engine), a tool under development providing context-awareness for visualizations of software reuse scenarios [13].

The remainder of this paper is structured as follows: Section 2 presents the main approach with an excerpt of the visualization feature model, Section 3 summarizes the related works, and Section 4 concludes the paper with the final remarks.

2. VISUALIZATION FEATURE MODEL

2.1 Domain Analysis

Since the domain addressed in this study refers to information visualization, a domain analysis was performed in order to better

[1] A visualization element is interpreted in this study as a concept that can be applied in the context of a visualization, and can be represented by visualization paradigms and techniques.

ICPC'14, June 2–3, 2014, Hyderabad, India
Copyright 2014 ACM 978-1-4503-2879-1/14/06...$15.00
http://dx.doi.org/10.1145/2597008.2597796

understand its extension. Common characteristics of visualizations are identified in order to proceed with the feature-oriented analysis.

The adopted methodology for feature analysis was based on two steps: (i) an informal literature review for identifying a set of visualization and interaction elements; and (ii) a *quasi*-systematic literature review [8] that is used in the context of this study for confirming the use of the already classified elements and for complementing the model with new candidates. Although the object of investigation of (ii) was restricted to software visualization approaches that have been proposed to support software reuse, such a literature review was able to gather some initial knowledge from software engineering researches. The adopted data extraction methodology was based on the dimensions of software visualization (discussed in [8]): Task, Audience, Target, Representation, Medium, Requirements and Evidences.

By analyzing the relevance of visualization elements, the results obtained from the *quasi*-systematic literature review point out some interesting findings. For instance, from 34 approaches selected in 36 publications, 6 visualization and interaction elements were mentioned simultaneously in more than 10 approaches, namely: *Selection*, *Navigation*, *Drill-Down*, *Clustering*, *Highlighting* and *Labeling*. More details can be found in [8]. Among others, such candidate elements were selected as externally visible characteristics of the visualization domain. A total of 53 elements were extracted from the feature analysis process.

2.2 Feature Model

Aiming at structuring the feature model, a notation is necessary to represent the different types of elements and support the domain analysis process. To this end, the Odyssey-FEX notation [1] was used, due to the researchers' previous knowledge on its syntax and to its wide-scope representation model, comprising characteristics such as category, variability and optionality.

For better structuring the model, some categories were defined to group similar elements. Although all the elements were identified based on works that present visualizations, some were strictly applied regarding interaction functionalities (*Interaction* category). Another group of visualization elements was interpreted as an alternative for presenting different visualizations (*Presentation* category). Finally, the third proposed group was related only to changing the exhibition mode (*Information Visualization* category). These categories led to the creation of three corresponding high-level, conceptual features.

The visualization feature model is a work-in-progress[2] and new features can be identified or its organization can be updated. Only an excerpt of the model is addressed in the following subsections, due to space restrictions.

2.2.1 Interaction Features

Related to user actions on a view, interaction features map existent interaction elements and categories. Some of these features and their relationships are shown in Figure 1.

[2] The most recent version of the visualization feature model, as well as each model element and the composition rules, can be found at http://www.cos.ufrj.br/~schots/featuremodel.html

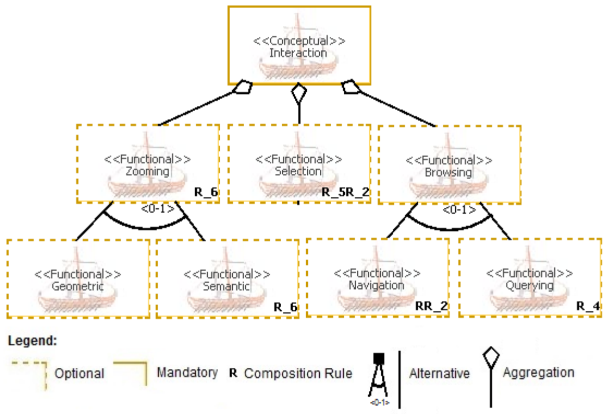
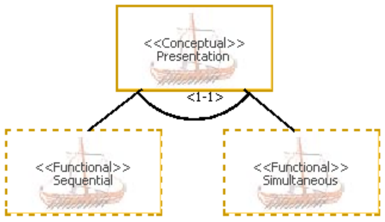


Figure 1. Some of the Interaction features

Although the application of those elements can be automatized, in general, its use can be controlled by the viewer. For instance, selecting the *Zooming* feature with its *Semantic* variant for a composed visualization, a user can zoom in and out, revealing different visual representations and details to information items [2]. Also, through the *Browsing* feature with its *Querying* variant, the user can apply queries, in order to organize or restrict the amount of data displayed in a view.

2.2.2 Presentation Features

Regarding the presentation of multiple visualizations, presentation features map the possible ways of showing different views. These features are displayed in Figure 2.



<<Conceptual>> Presentation
<1-1>
<<Functional>> Sequential
<<Functional>> Simultaneous

Figure 2. Presentation features

Usually, the application of such features on a view is totally controlled by the software visualization tool, i.e., a decision made by the visualization developer. This is an important difference to a normal interaction element.

In terms of the presentation mode, by selecting the *Sequential* feature, a visualization displays different views in a sequential order, each view at a time. With the *Simultaneous* feature, multiple views can be presented at the same time, similarly to a dashboard.

2.2.3 Information Visualization Features

Regarding general exhibition elements, information visualization features map the visual methods for changing the views. Some of these features are displayed in Figure 3.

In order to customize the visual aspects of an individual visualization, the information visualization features must be selected. For instance, the *Details on Demand* feature with its *Drill-Down* variant reveal details, following a hierarchical structure, according to the user needs [10]. On the other side, the *Clustering* feature aims at splitting a large data set into subgroups based on certain similarity measures to ease the data analysis [3].

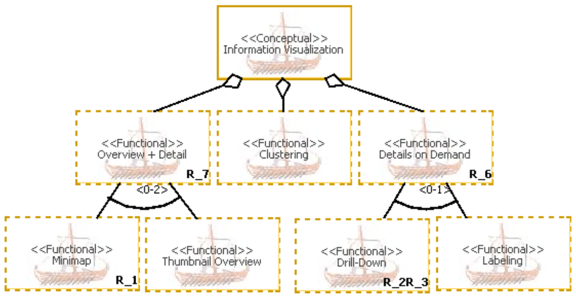


Figure 3. Some of the Information Visualization features

It is noteworthy that the selection of each feature may impose the selection of other features. Thus, besides the features and their relationships, composition rules supplement the model with mutual dependency and exclusion relationships [7].

2.2.4 Composition Rules

With different features in the model mapping to a visualization context, the application of a single element may require or exclude the use of another visualization or interaction element. Thus, composition rules may be applicable to the visualization concepts, constraining the selection from optional or alternative features [7]. A composition rule between features is specified as follows:

R_X - <*Visualization Feature*> **requires** **<***Visualization Feature***>**

R_Y - <*Visualization Feature*> **excludes** **<***Visualization Feature***>**

Relating 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 the *Drill-Down* or the *Navigation* feature is chosen. 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 [10]. Similarly, the application of *Navigation* needs a method to select 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 [2]. 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 belong to the feature model and can also be updated. The explicit relationships between features shown by the rules are extracted from the literature.

2.3 Selection of Visualization Features

After all the elements have been mapped and organized in the feature model, the next phase is the selection of features in order to build a visualization. A wizard under development supports the manual selection. In an offshoot of this work [13], this process is intended to be automatized, i.e., the selection of features will be done according to context definitions and the corresponding implemented visualization elements will be selected accordingly. From a mapped context, structured as a context-aware feature model [6], it is possible to propose context rules associating the occurrence of a situation to a feature set of visualizations. Thus, a visualization that meets the requirements proposed by the selected features can be chosen. A mapping rule must follow the model:

<*Context Information*> **implies** <*Visualization Feature*>

Thus, multiple context-aware visualization mapping rules can be established in order to provide the composition of different views according to a set of context information. For instance, given the need for tasks involving software evolution, like the comparison of code versions in a development scenario, some elements can be specified through the following rule:

(*Compare Code Versions*) **implies**

((*Overview + Detail*) AND (*Browsing*))

Mapping rules can elicit the visualization requirements for representing specific context information. Such requirements can use any feature from the feature model, expressing the interaction, presentation and information visualization needs for a view. Given a software reuse task, existing context mapping rules will select the appropriate features and a visualization will be built from those features.

An illustrating scenario of the use of the feature model is described as follows. A developer needs to understand the package structure of the source code of a given reusable asset, aiming at its adoption in a software project. Based on some defined context rules, a hierarchical visualization featuring semantic zoom techniques and details on demand (with an overview of the asset classes) can be selected for supporting his/her decision. If no context rule is applicable, he/she can manually provide (through a wizard) information that supports the choice of the visualization. Figure 4 illustrates this process.

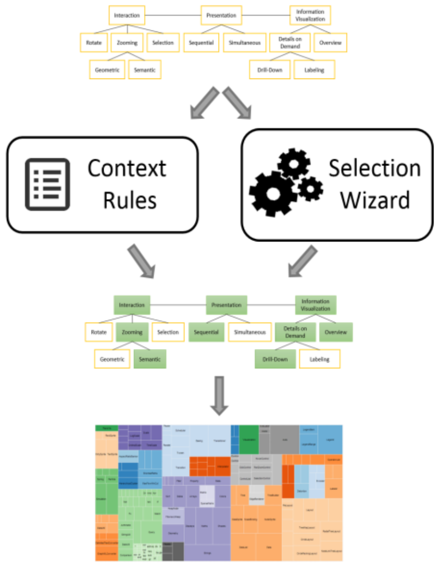


Figure 4. Features composition to create a view

3. RELATED WORK

Some previous studies, recognizing the importance and utility of information visualization, propose means to structure a body of knowledge in terms of techniques and algorithms. Although these works are not feature model proposals, they seek to comprise a set of techniques related to visualizations.

The Data State Model (DSM) approach [4] shows the similarities in terms of operation steps that can be reused. Its goal is to support implementers 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. However, neither the procedure for relevance analysis nor the concept of relevance are described.

Another approach presents a high-level visualization taxonomy [11], classifying algorithms instead of data. The taxonomy is considered as flexible because it is based on assumptions that algorithms make about the data to be visualized. 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.

These related works propose different taxonomies for the visualization domain. Despite presenting different modes of classifying techniques, none of these studies focus on methods to compose visualizations. Although several studies mention the application of visualization elements in software tools, no work categorizing a large set of elements as feature models was found in order to realize a proper comparison. Furthermore, unlike this work, no analysis against a more comprehensive study was found in these related works. In this sense, the conducted verification within existent studies through the *quasi*-systematic review proved to be an important indicator in terms of confirming and evolving the items from the feature model.

4. FINAL REMARKS

The visualization domain is known for its variety, with multiple popular visualizations adopted for different goals. At the same time, there is a need for mechanisms to analyze data in terms of software development. Visual abstractions are crucial to help understand the generated data from these software processes and products. In this sense, the visualization feature model intends to provide a support approach to select the most appropriate view and build it. Such customized composed visualizations are expected to increase awareness in software development scenarios [9].

Despite the model composition process being 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 being developed, it is expected to constantly expand this model based on findings of new studies.

The evaluation methodology will be based on two steps: (i) validating the feature model syntax and semantics by using a feature model checklist [5]; and (ii) carrying out a peer review method. In order to check both the visualization aspects and the feature model notation, expert professionals from the two areas will participate in this review. These professionals are already selected and the review itself will be executed next.

The presented approach will be incorporated in the APPRAiSER environment [9] through CAVE, a work in progress that will tackle on context-aware visualizations [13]. It aims at analyzing software reuse characteristics to extract the main relevant information for a context model. The execution of this software reuse awareness engine will show dynamic visualizations adapted to a given scenario that should be addressed.

5. ACKNOWLEDGMENTS

The authors would like to thank CAPES, CNPq and FAPERJ for the financial support.

6. REFERENCES

[1] Blois, A. P. T. B., de Oliveira, R. F., Maia, N., Werner, C., and Becker, K. 2006. Variability modeling in a component-based domain engineering process. *Reuse of Off-the-Shelf Components*, 395-398. Springer Berlin Heidelberg.

[2] Buering, T., Gerken, J., and 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, 12, 5, 829-836.

[3] Chen, C. 2006, Information Visualization: Beyond the Horizon. 2 ed. Springer.

[4] Chi, E. H. H. 2000. A taxonomy of visualization techniques using the data state reference model. In IEEE Symposium on Information Visualization (InfoVis), 69-75.

[5] De Mello, R. M., Teixeira, E. N., Schots, M., Werner, C. M. L., and Travassos, G. H. 2014. Verification of Software Product Line Artefacts: A Checklist to Support Feature Model Inspections. J.UCS (accepted for publication).

[6] Fernandes, P., Werner, C., and Teixeira, E. 2011. An Approach for Feature Modeling of Context-Aware Software Product Line. J. UCS, 17, 5, 807-829.

[7] Lee, K., Kang, K. C., and Lee, J. 2002. Concepts and guidelines of feature modeling for product line software engineering. In *Software Reuse: Methods, Techniques, and Tools*, 62-77. Springer Berlin Heidelberg.

[8] Schots, M., Vasconcelos, R. and Werner, C. 2014. *A Quasi-Systematic Review on Software Visualization Approaches for Software Reuse*. Technical Report. Federal University of Rio de Janeiro.

[9] Schots, M. 2014. On the Use of Visualization for Supporting Software Reuse. In 36th International Conference on Software Engineering (ICSE), Doctoral Symposium, 694-697.

[10] Shi, K., Irani, P., and Li, B. 2005, An Evaluation of Content Browsing Techniques for Hierarchical Space-Filling Visualizations. In IEEE Symposium on Information Visualization (InfoVis), 81-88.

[11] Tory, M., and Moller, T. 2004. Rethinking visualization: A high-level taxonomy. In IEEE Symposium on Information Visualization (InfoVis), 151-158.

[12] Treude, C., and Storey, M. 2010. Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds. In 32nd International Conference on Software Engineering, 365-374.

[13] Vasconcelos, R. R., Schots, M., and Werner, C. 2013. Recommendations for Context-Aware Visualizations in Software Development. In 10th Workshop on Modern Software Maintenance, 41-48.