On Mapping Goals and Visualizations: Towards Identifying and Addressing Information Needs

Marcelo Schots, Cláudia Werner

Systems Engineering and Computing Program (PESC)
Federal University of Rio de Janeiro (COPPE/UFRJ) – Rio de Janeiro, RJ – Brazil

{schots,werner}@cos.ufrj.br

Abstract. *The design of visual metaphors and the development of visualization tools have a key difference: while the former is a creative process to produce novel and generic approaches, the latter is a goal-oriented process. Ensuring that a tool properly maps the data required to achieve its goals into visual attributes is not trivial, since there are several abstraction gaps to be addressed. The mapping structure presented in this paper aims to provide developers of visualization tools with a focused and cautious decision making. Its use in the design of Zooming Browser, a tool for performing software reuse analytics, enabled to check if the tool could help answering the established questions before evaluating it with the intended audience (i.e., stakeholders).*

1. Introduction and Motivation

The design of visualizations is a process that usually requires insights and/or creativity, and can occur in several ways. Information visualization designers may create concepts from the real world to represent abstract entities, or (more usually) they may try to combine knowledge on existing paradigms, strategies, and techniques to produce novel approaches. The produced visual metaphors are usually general, enabling their use for several purposes (as shown in [Abuthawabeh et al. 2013]) in diverse fields of interest.

On the other hand, the development of visualization tools goes the opposite way. Developers[1] 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 be met. Thus, not only there is the need to represent data using proper abstractions; it must be made in such a way that it can help somebody (audience) to do something (tasks). If this premise is neglected, the tool will be either useless or not fully meet the needs for which it is created, leading users to resort to other sources of information or additional tools.

It is important 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. However, this is not trivial, since there are several abstraction gaps to be addressed. The mapping of goals and visualizations cannot be made instantly, as this brings a considerable risk of overlooking important intermediary decisions; it must be decomposed into stages and performed carefully.

[1] In this paper, *developers* include roles who 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.

This necessity became more evident during the design of Zooming Browser, a visualization tool for performing software reuse analytics [Schots 2014]. Its goals were based on findings from our previous studies, which pointed out limitations on existing visualization approaches for supporting software reuse [Schots & Werner 2014a] and needs identified in the software industry [Schots & Werner 2014b]. We wanted to assure that Zooming Browser visualizations would be actually helpful in answering some reuse-related questions, accomplishing the established goals. To this end, we realized that we needed 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. This resulted in a mapping structure created to guide this process.

In this paper, we present a staged process for mapping *user* (or *organization*) *goals* to the *visualizations* that can help achieving such goals. 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, we want to encourage developers to perform a more focused and cautious decision making (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. For illustrating how each stage can be carried out, we present some examples of the mapping performed during the development of Zooming Browser.

2. Related Work

Before elaborating the mapping structure, we analyzed related work to find out whether existing solutions would meet our 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, we 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.

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.

3. The Zooming Browser Tool

Nowadays, developers write less code and consume more reusable code. Software reuse has become present in the daily routine of software developers, yet mostly in an ad-hoc or pragmatic way. However, 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 tool support to this purpose.

The Zooming Browser tool aims at providing reuse awareness to support reuse managers[2] 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). It is part of APPRAiSER, an environment for visually supporting software reuse tasks with awareness resources [Schots 2014]. Zooming Browser is composed by three core elements (assets, developers, and projects), and the reuse questions are based on the relationships between these elements, as shown in Figure 1 (detailed in [Schots 2014]). 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 the mapping structure.

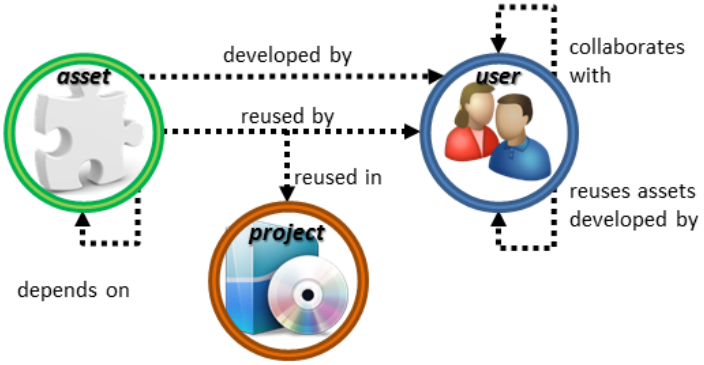

Figure 1. Core elements of Zooming Browser [Schots 2014]

4. The Mapping Structure

The structure for mapping goals and visualizations is presented in Figure 2. All the relationships are many-to-many, except between data and visual attribute, which is one-to-one to avoid user confusion due to ambiguity. During the mapping process, 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 is indicated when there are some goals and visualizations in mind, but some aspects in the middle-part of the mapping are not clear yet and require further reflection and analysis. This was the case of Zooming Browser.

---

[2] 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.

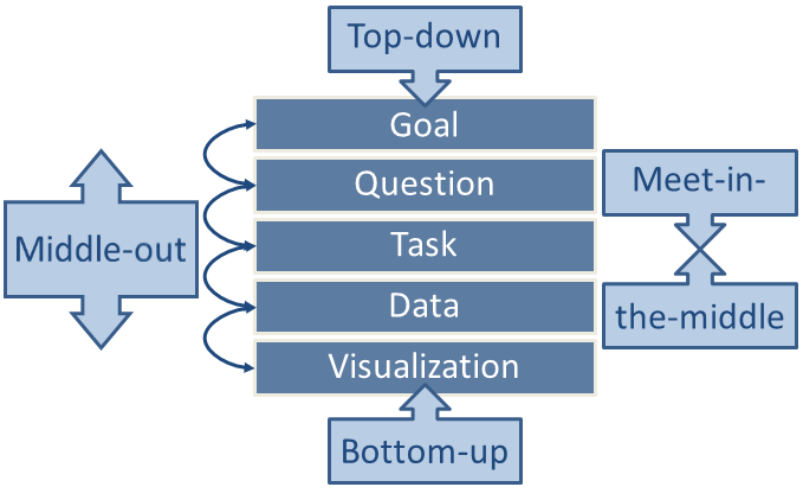

Figure 2. Mapping structure between goals and visualizations

The next subsections present a description of each part of the mapping, with an example of what has been made in the context of Zooming Browser[3], the driver of this work. For didactic purposes, the mapping is described in terms of the top-down strategy.

4.1. 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. The GQM format is not mandatory, but is advisable.

For illustration purposes, we present some asset-centric questions[4] derived from 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*) [Schots 2014]. They are listed as follows. It is advised to keep record of the rationale that relates each question to the goals (this also 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.

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], how many of them are fixed/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?*

[3] The full mapping is not presented in this paper; the parts shown are only for illustration purposes. Zooming Browser is used as example to avoid the risk of misinterpreting a system developed by others.

[4] There are also other questions (including project-centric and developer-centric questions) that are related to this goal, but they are not presented in this paper.

4.2. 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*[5] 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 their relevance to the current state-of-the-practice [Novais et al. 2014]. In the Zooming Browser design, the following tasks are associated with the following questions:

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

4.3. 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 as follows. They are extracted from reuse repositories, version control repositories, and issue tracking/task manager systems. Links to original sources or other representations (*e.g.*, HTML websites) are also stored for allowing drill-down to additional information.

Project history information *(both from assets' projects and other projects in which assets were reused): project name (related to* ***Ta****,* ***Tb****), commit author (related to* ***Tb****,* ***Td****), commit date (related to* ***Tb****,* ***Td****), added files* (related to* ***Ta****), removed files* (related to* ***Ta****)*
Reuse repository information*: asset name (related to* ***Ta****,* ***Tc****), asset versions (related to* ***Ta****,* ***Tc****), asset dependencies** (related to* ***Tc****)*
Issue tracking information*: issue status (related to* ***Td****), issue type (related to* ***Td****), issue creation date (related to* ***Tb****,* ***Td****), issue close date (related to* ***Tb****,* ***Td****), issue assignee (related to* ***Tb****,* ***Td****), issue resolver (related to* ***Tb****,* ***Td****)*
** Filtering is applied in order to retrieve only commits of assets. ** There are different kinds of software dependencies; the current scope is limited to explicit dependencies (e.g., described in a build configuration file).*

4.4. 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. Our studies showed that the mapping between data and visualizations is barely

[5] Interaction tasks with the visualizations (such as filtering, browsing, drill-down etc.) will be handled separately in an upcoming stage.

described in publications, so users have to "guess" it, which can be risky and lead to wrong interpretations of data [Schots & Werner 2014a]. 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 will 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[6]. 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, there may be 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* perspective (from Zooming Browser's *asset-centric* view) is depicted in Figure 3, while Table 1 shows how some of the aforementioned data were mapped to visual attributes of this visualization. Although it is not explicit as a separate stage in the mapping, this stage also requires the choice of interaction resources to be employed, so as to allow users of the visualization tools to explore the data and perform their tasks. In Figure 3, for illustration purposes, issues can be filtered by type through a filtering mechanism.

5. Final Remarks

The mapping structure presented in this paper (and the example of its concrete application) aims at supporting developers of visualization tools in better planning the resources to be used towards the usefulness of these tools, before evaluating them with the intended audience. The mapping stages point out aspects that should be considered

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

when building visualization tools, aiming to make a proper use of the visual attributes of the chosen visualizations, based on the data to represent. This mapping is purposely "open" so that each stage can be accomplished by stakeholders in the most convenient way to them. We hope that this can serve as an initial guidance to future developments of visualization tools, emphasizing the importance of performing each stage carefully.

Figure 3. Issues perspective (draft/sketch)

Table 1. Mapping between data and visual attributes in the issues perspective

Visualization	Visual Attribute	Data	Value	Description
adapted bubble layout	geometric shape	issue	• circle	A circle represents an issue.
	size	issue lifetime	• for closed issues: (*issue close date* – *issue creation date*) • for open issues: (system current 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.
	color	issue status	• green, for closed issues • red, for open issues (*issue type* = bug) • yellow, for open issues (*issue type* = feature request or suggestion)	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.
	icon	issue type	• exclamation mark (*issue type* = bug) • lamp bulb (*issue type* = feature request or suggestion)	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.

The use of this mapping provided more confidence for implementing Zooming Browser, since we were able to check whether the tool could help answering the established questions before evaluating it with its intended stakeholders. We believe that the effective evaluation of the performed mapping in the context of Zooming Browser may be done by the implementation and use of the tool. This will provide information on what aspects are lacking or could be mapped differently. The evaluation of the tool with its stakeholders is one of the next steps of the research.

We intend to continue this research in order to deepen our understanding with respect to the usefulness of this mapping. Some open questions include: (i) Does this

mapping structure actually support developers of visualization tools in better planning the visualization metaphors and resources to be used? (ii) How can we facilitate each stage of this mapping, *i.e.*, what additional support can be provided for easing such stages (especially the last one)? (iii) Should we build tools to help performing this mapping? We want to investigate with the visualization community answers to these questions and others that may emerge through other applications of the mapping.

Acknowledgment

The authors would like to thank CNPq and FAPERJ, for the financial support for this research, and Natália Schots, for providing useful feedback on this paper.

References

Abuthawabeh, A., Beck, F., Zeckzer, D., Diehl, S. (2013). "Finding structures in multi-type code couplings with node-link and matrix visualizations". In: 1st IEEE Working Conference on Software Visualization (VISSOFT 2013), pp. 1-10, September.

Basili, V., Caldiera, G., Rombach, H. (1994). "Goal Question Metric Paradigm". Encyclopedia of Software Engineering, v. 1, John J. Marciniak, Ed. John Wiley & Sons, pp. 528-532.

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.

Carneiro, G. F., Sant'Anna, C., Mendonca, M. (2010). "On the Design of a Multi-Perspective Visualization Environment to Enhance Software Comprehension Activities". In: 7th Workshop on Modern Software Maintenance (WMSWM), pp. 61-68, June.

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), pp.688-693, March.

Lanza, M., Marinescu, R. (2006). Object-Oriented Metrics in Practice. Springer-Verlag Berlin Heidelberg New York, 1st ed.

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), pp. 14-21, September.

Schots, M. (2014). "On the use of visualization for supporting software reuse". In: 36th International Conference on Software Engineering (ICSE 2014), Doctoral Symposium, pp. 694-697, June.

Schots, M., Werner, C. (2014a). "Using a Task-Oriented Framework to Characterize Visualization Approaches". In: 2nd IEEE Working Conference on Software Visualization (VISSOFT 2014), pp. 70-74, September.

Schots, M., Werner, C. (2014b). "Characterizing the Implementation of Software Reuse Processes in Brazilian Organizations", Technical Report ES-749/14, COPPE/UFRJ, Rio de Janeiro, Brazil.