Multi-dimensional Navigation Spaces for Software Evolution

Project N° MMI-1976 supported by the Hasler Foundation
« Man-Machine Interaction »

Final report
April 2008

Philippe Dugerdil & Sazzadul Alam
HEG, Univ. of Applied Sciences, Geneva

Harald Gall & Sandro Boccuzzo
University of Zurich

Michele Lanza & Richard Wettel
University of Lugano
Executive Summary

Software is an intangible artifact that does not have any “natural” visual representation. While many software visualization techniques have been proposed in the literature, they are often difficult to interpret. In fact, the user of such representations is confronted with an artificial world that contains and represents intangible objects. The cognitive mapping is the major research challenge we tackled in our research project.

The goal of this project was to investigate the best visual metaphors (i.e. analogies) between “natural” objects and software objects so that we can exploit the cognitive understanding of the user. The difficulty of this approach is that the common sense expectations about the displayed world should also apply to the world of software objects.

To solve this “common sense representation problem” for software objects our project addressed both the small-scale (i.e. the level of individual objects) and the large-scale (i.e. the level of groups of objects). We investigated the best visual metaphors between “natural” objects and software objects so that when displaying a “natural object” one can follow the cognitive mapping intuitively. After many experiments we settled on the overall metaphor that our tool follows: that of a “city”: At the small scale we include different houses and their shapes as visual objects to cover size, structure and history. At the large-scale level we arrange the different types of houses in districts and include their history in diverse layouts. The user then is able to use the EvoSpaces virtual software city to navigate and explore all kinds of aspects of a city and its houses: size, age, historical evolution, changes, growth, restructurings, and evolution patterns such as code smells or architectural decay.

For that we have developed a software environment named EvoSpaces as a plug-in to Eclipse so that visual metaphors can quickly be implemented in an easily navigable virtual space. Due to the large amount of information we complemented the flat 2D world with full-fledged immersive 3D representation. In this virtual software city, the dimensions and appearance of the buildings can be set according to software metrics. The user of the EvoSpaces environment can then explore a given software system by navigating through the corresponding virtual software city.

The results of this project can be summarized as follows:

- Visual metaphors for software objects and objects groups;
- Mappings between software metrics and features of visual objects;
- Explorative ways to display large quantities of software objects;
- Time travel along many releases of software structures;
- Navigation techniques in this visual world.

In terms of research management, we decided that the Geneva, Lugano and Zurich groups investigate alternatives in their particular area of expertise and provide prototypes for discussion and evaluation to the whole team. This had the advantage that the project could benefit from a large set of possible solutions and adopt the best ones. The Geneva group could then integrate the best out of the pool of ideas for the common EvoSpaces platform. This allowed the PhD students to effectively pursue their line of research and our project team to select the best results for the platform.

The remainder of this report first presents the metaphors we developed in EvoSpaces and the way software metrics can be represented through this metaphor. Then we describe the technical environment and the tools we have developed to tune the software representation to meet the needs of a maintenance engineer. Next we present the general architecture of the EvoSpaces system, which is implemented as Eclipse plug-in. Then we give a presentation of a usage scenario in a maintenance situation. In particular we show how the information about a software system can be analyzed and how the EvoSpaces tool can be used. We conclude with the presentation of the virtual cities associated to different software systems.
**Goals of the EvoSpaces Project**

This project aimed at investigating 3D visual representations for software understanding, complexity management and evolution. The central idea is to represent software artifacts and structure as intuitive, cognitive and therefore familiar 3D scenes, then to exploit common sense reasoning to navigate such an environment. Therefore, the research tracks, structured as work packages, were:

**WP 1: Visualization models** – finding metaphors and visual artifacts to represent software elements and their relations. The analogy represented by the metaphor should be easily and accurately interpretable by the user of the system.

**WP 2: Interaction and navigation models** – finding ways to navigate the 3D space to find information and investigate the represented software elements. In fact, the complexity of software systems comes from their scale, the diversity of their components as well as the entangled nature of the interactions of these components. Therefore, the navigation modes should help the user to selectively explore this information space to find relevant information to understand the system.

**WP 3: Software models** – to navigate in a software system at different levels of details, the fundamental components of the system as well as their relationships at the required level of detail should be known and modeled. In other words, the system should be decomposed and its parts stored in a repository for the visual modeling tool to obtain the required information. At the same time, some measurements on the system (software metrics) should be computed and stored in the repository. These will be used for visualization as well. Therefore the software model and the repository structure should be carefully designed.

**WP 4: Prototyping and Integration** – the result of our research should be “instantiated” as a working prototype to explore the applicability of our findings.

In terms of research management each of the three research groups had specific tasks:

- The Geneva team concentrated on the large-scale architecture of our system, the navigation in the 3D space, the representation of the dynamics of the system and the implementation techniques used to add new visual metaphors. It also worked on the techniques used to speed up the display of visual scenes.

- The Zurich team focused on small-scale visual representations, i.e. individual software objects. It explored ways to map software metrics to the dimensions of complex visual objects and their cognitive visual interpretation. Especially they concentrated on the concept of a metrics mixer to select and tune a set of metrics so that a well-designed software object would be represented as well-shaped (i.e. easily interpretable as such) visual object.

- The Lugano team explored large-scale visual representations of software objects. They investigated and developed the city metaphor and presented ways to normalize object groups for single versions and multiple versions to allow evolutionary analyses. In particular, effective layout mechanisms were devised to enable exploring change and growth scenarios with playing and replaying “periods” of the software’s evolution.

This project has been a scientific success, as attested by the encouraging feedback we received when presenting our results to international conferences and workshops. Several papers have been accepted and presented at international scientific conferences and workshops; we include them in the appendix.

In the following we present our achievements and deliverables following the work package structure of our project plan.
**WP 1: Visualization Models**

The goal was to identify key properties of software objects to be represented as well as the mode of representation. Such ways to represent and navigate through this information consisted of the construction of a global mental and cognitive picture. Moreover, for a software representation to be effective it should follow some “natural” paradigm. For that we devised and examined some visualization models on different levels of abstraction.

**Achievements:**

- **Software as City**: We adopted the city metaphor to represent the structure of software on small and large scale.
- **Cognitive glyphs**: Cognitive glyphs such as houses, tables, sticks and many more have been investigated and selected ones integrated into the EvoSpaces tool.
- **Metric Clusters** have been devised that represent effective arrangements of cognitive houses. Sets of metrics are mapped to the layout of the glyphs. This allows us to represent well-shaped and miss-shaped houses in a Software City.
- **Historical buildings and districts** are represented in a 3D view and effective layouts for that have been developed, each focusing on particular historical phases of a system.
- **Evolutionary views** have been integrated that allow us to navigate along the time axis and discover change, growth or restructuring phases of a software system.
- **Dynamics** such as execution traces have been integrated as day and night views enabling an effective animation of the execution architecture.

**Deliverables:**

- The EvoSpaces prototype demonstrates our common sense representation system.
- The Software City metaphor for visualization of software systems [WL07b]
- Cognitive glyphs and metric clusters [BG07a]

**WP2: Interaction and Navigation Models**

The goal of this WP was to identify the interaction and navigation mode to explore the many representations we were going to build. The success criterion for this project was whether we are able to construct a mental picture of the software and its evolution from the many partial representations. Due to the quantity of information, this mental picture would be interactively built by selecting and pulling information through semantic links and not pushed by the system.

First, the models should allow the navigation to be horizontal (between software elements at the same conceptual level) as well as vertical (from a conceptual level to another one for the same software element). While zooming in into a given element it is important to stay in the same representation.
paradigm. Second, the navigation technique in the software model should correspond to navigation in the domain concept model. A mapping had to be defined as well as the constraints that the domain model brings to the software representation. Moreover, the interface between the different levels should be formally defined so that a user should experience continuity in the information provided.

The main surprise in this part of our research, however, was the absence of any effective interaction and navigation models we could borrow from the civil architecture domain. In fact, even if architects use sophisticated techniques to model buildings in 3D, no specific navigation techniques have been designed. It seems that architects have not taken advantage of the interactive nature of the digital media to invent new ways of working. Navigation stays as close as possible to what it would be in the real world. The paradigm seems to be: the closer to the real world the better.

Achievements:

- **Vertical navigation model based on the software containment hierarchy**: Following the hierarchical structure of large software systems (i.e., packages, sub-packages, classes, systems, subsystems, modules, directories, libraries, etc.) we devised a vertical navigation and exploration model: allowing the user to move from packages to sub-packages, etc.
- **Horizontal navigation model in the software city**: The software city visualization metaphor allowed us to introduce an effective horizontal navigation model: navigating among the buildings representing classes or modules became natural. In this model, containment is used to distribute the buildings in the 3D space as the city districts. But not much information on containment is displayed except the limits of the district on the “ground”. Therefore, vertical navigation is limited to the investigation of the inside of the buildings to see the contained procedures, functions, variables and macros.
- **Natural exploration following horizontal and vertical relationships**: In our navigation model it is quite natural to follow the relationships among the classes to navigate in the city: from streets into buildings, onto different floors, back to other related buildings or even districts of the software city.
- **Multi-level caller-callee navigation**: The same navigation mode is possible at the level of procedures and functions: it is possible to move from the calling and called functions easily.
- **Proportional dimensioning principle**: Since the shape and dimension of the buildings are set according to metrics, the representation of the software under investigation allows the user to visually identify objects of interest for further exploration.
- **Run-time interaction mode**: we investigated the *representation* of the run time interaction between the displayed elements. This representation displays a pre-recorded execution trace of the system. However due to the quantity of information to display, an execution trace can contain on the order of $10^5$ to $10^6$ events, an information “compression” technique has been designed. In this representation, a vertical navigation technique has been added in the “time” space: from large intervals in time considered as a whole to single events contained in these intervals.
Deliverables:

- The EvoSpaces prototype demonstrates our common sense representation system.
- A software habitability model [WL07a]
- A 3D navigation model for software architectures [AD07a]
- A visualization model for execution traces [DA08]

WP 3: Software Models

The goal of this WP was to prepare the “raw evolution data” of large-scale software systems and produce models of them from the point of view of the structure, architecture, design, etc. First, the system should be explained through a presentation of each of the views and navigation techniques, together with the technology to acquire the information from the software system. Then a set of software analysis scenarios must be proposed together with the typical usage of our system to make optimal use of the EvoSpaces tool.

In a previous project we have devised meta-models for version control, bug tracking and software evolution. For EvoSpaces, these models are essential ingredients for software analysis and software architecture exploration and navigation. Change types, change couplings, or code smells are mapped to dimensions of visual glyphs with our CocoViz metric mixer. It enables metric clusters to be defined and represent certain architectural or code smells. All these evolution data are stored in a relational database that contains release history information over many releases of a software system. This so called Release History Database (RHDB) provides the necessary abstractions and the data model that is fed into the EvoSpaces tool. Such a RHDB can cover multiple projects and represents a rich resource for assessing the evolution history of systems with EvoSpaces.

Achievements:

- Metric Clusters representing code smells such as god classes, data containers, etc. This includes configurability for new or customized smells to be defined by the user.
- Metric Mixer that maps metric clusters to visual cognitive glyphs such as houses for the software city.
- Release history data model for a software system to be investigated as input for the EvoSpaces tool. This is based on the previously developed data models for versioning and bug tracking data.
- Software evolution analysis scenarios and data sources used. Major approaches in software evolution analysis and visualization have been presented in an OOPSLA tutorial [GL07].

Deliverables:

- The CocoViz tool for assembling metric clusters, normalizing metrics for cognitive glyphs and facilitating a cognitive visual understanding.
- An integration of the RHDB data into the EvoSpaces tool.
WP 4: Prototyping and Integration

The goal of this WP was to constitute an overall parallel track that involves the development of a prototype system to integrate the previously mentioned models. It should integrate the work of all three teams into a single consistent prototype.

This prototype system was built in parallel to the other work packages as the work progressed. To make the integration easier, standard architecture and component interfacing techniques were documented and enforced. This WP was one of coordination: it assured that the developments made by all the teams were consistent with the overall goal and that the contribution of each member of the project went into the right direction.

This architecture lasted for the entire phase of the project. The internal architecture of each layer let us easily extend the visual features of EvoSpaces to account for new artifacts, navigation modes and metrics. Moreover, being a plug-in of Eclipse, the system can be interfaced with other tools running under Eclipse.

The architecture was revised after a first implementation that focused on data from the case study of the Mozilla web browser. It has been extended to cover large C++ and Java systems. This work is still ongoing.

The prototype system has been developed all along the project, integrating new ideas that were devised by the respective research teams. This let us continuously monitor the integration of all the features as a coherent whole. And it let us experiment on ideas and just merge effective and promising solutions to the main trunk of the EvoSpaces tool.

Using metrics to filter the information to display    The metrics mixer to set the shape of buildings
Display of the metrics information on the selected software object

Achievements:

- Tool paper and demo of the Code City tool showing the software city metaphor [WL08].
- Tool paper and demo of the CocoViz tool showing visual cognitive glyphs [BG07b].
- Tool paper and demo of the EvoSpaces tool environment [AD07b] including the software city and visual cognitive glyphs aspects.

Deliverables:

- A guided tool demo video of how to use the EvoSpaces tool that can be downloaded at http://www.inf.unisi.ch/projects/evospaces/tool/.
- The EvoSpaces tool to efficiently visualize large software systems using the software city metaphor and visual cognitive glyphs on different levels of abstraction.

Conclusions and Future Work

With software systems becoming increasingly complex, software evolution has to be addressed in a systematic way by providing adequate means for dealing with this complexity. Multi-dimensional, multi-view visualization adopting techniques from the areas of architecting as well as multimedia and visualization will be one major technology for software engineers to get a grip on software complexity for development, maintenance and evolution of successful software systems.

Our project aimed at exploiting multi-dimensional navigation spaces to efficiently visualize evolving software systems to ease analysis and maintenance. One of the outcomes of the project is the EvoSpaces prototype implemented as plug-in to Eclipse, in which large software systems can be visualized in 3D and navigated.

This virtual world has been named “Software City” because we used the city metaphor to represent the virtual software entities. In a “Software City”, the classes or files are represented as buildings displayed in districts representing the containing packages or directories. Then, the methods and procedures are represented as stickmen when the inside of the buildings is displayed. The size and dimension of the buildings are set according to the value of user-selected metrics. The city gets a different shape depending on the selected metric. We can also easily filter information based on given metric values. Finally, the relationships between the classes or files are represented as solid pipes with an animation to show the flow of information. In this virtual world, not only the static relationships, but also the dynamic behavior can be visually represented thanks to a technique to display the
execution trace. The prototype has been architected so that extensions can easily be built and integrated.

As for the scientific dissemination, our approaches and prototypes have been presented in specialized international conferences and have attracted the interest of the research community (see our Publications section).

For future work, we will extend our research to a multi-user environment where people collaborate to solve problems. This implies to develop research in two directions: (1) collaborative maintenance of software systems; and (2) metaphors and models to ease teamwork on the same system. This line of research will be continued in the EvoSpaces II project that was kindly accepted by the Hasler Foundation.

On behalf of the Evospaces team,

Prof Dr Philippe Dugerdil
Project leader
**Publications**

In the following we list all project publications, which are available for download from the EvoSpaces project web site [http://www.inf.unisi.ch/projects/evospaces/publications/](http://www.inf.unisi.ch/projects/evospaces/publications/). Please note that all publications have been peer reviewed and were presented at international software engineering venues.

**International Conferences and Symposia**


**International Workshops**


**Tool Demos**


**Tutorials**
