CodeCity: 3D Visualization of Large-Scale Software

Richard Wettel
REVEAL @ Faculty of Informatics,
University of Lugano, Switzerland
richard.wettel@lu.unisi.ch

Michele Lanza
REVEAL @ Faculty of Informatics,
University of Lugano, Switzerland
michele.lanza@unisi.ch

ABSTRACT

CODECITY is a language-independent interactive 3D visualization tool for the analysis of large software systems. Based on a city metaphor, it depicts classes as buildings and packages as districts of a "software city". By offering consistent locality and solid orientation points we keep the viewer oriented during the exploration of a city. We applied our tool on several large-scale industrial systems.

Categories and Subject Descriptors

D.2.7 [**Distribution, Maintenance and Enhancement**]: Restructuring, Reengineering

General Terms

Languages, Design, Human Factors

1. INTRODUCTION

CodeCity is a language-independent interactive 3D visualization tool, developed to support the analysis of large-scale object-oriented software systems. CodeCity revolves around a 3D *city* metaphor [7, 8], i.e., it represents systems as cities, where classes are depicted as buildings, and packages as the districts of the city. One reason that led us to using the city metaphor is that a city, with its downtown area and its suburbs is a familiar notion with a clear concept of orientation. Also, a city, especially a large one, is an intrinsically complex construct which can only be incrementally explored, the same way that the understanding of a complex system increases step by step. Using an all too simple visual metaphor does not do justice to the complexity of today's software systems, and leads to incorrect oversimplifications: Software is complex, there is no way around this. Lastly, classes are the cornerstone of the object-oriented paradigm, and together with the packages they reside in, the primary orientation point for developers. We do not explicitly represent the class internals, because for a large-scale understanding it is not necessary. Apart from over-plotting problems, it is also contrary to the way one explores a city: One does not start by looking into particular houses.

To depict classes as buildings we extend the polymetric view approach [4] into 3D, and therefore map software metrics on all three dimensions of the buildings. In the following examples, we map the number of attributes metric for the classes on both the width and length of the buildings, and the number of methods metric on their height. Tall buildings thus represent classes with high functionality, while buildings with a large base represent classes which encode a high amount of state. As a linear mapping of a metric value is not always appropriate, CodeCity supports different mapping mechanisms, described in [8]. Finally, to depict packages as city districts, we use a containment-based layout inspired by rectangle-packing algorithms and map the nesting level of a package on the color of the districts *i.e.,* the higher the nesting level, the darker the color hue. We can also represent the nested packages as stacked platforms, thus placing the buildings at different altitudes.

ICSE'08, May 10–18, 2008, Leipzig, Germany.
ACM 978-1-60558-079-1/08/05.

2. CODE CITY

CodeCity supports reverse-engineering activities by representing the system under investigation as a 3D interactive urban environment, in which we can freely move. The city provides an overview of the system and by navigating around it we can investigate its structural organization. As with a real city, we get incrementally familiar with the environment to avoid getting overwhelmed by the amounts of information such a system carries. One of Codecity's prominent features is that, due to the fact that the user can immerse himself in the visualizations he can contextualize the presented information. Throughout our experiments with CodeCity, we applied our approach on several open-source software systems, ranging from small and medium-sized systems (*e.g.,* JHotDraw, with 30 kLOC and 1,000 classes) to fairly large systems (*e.g.,* Azureus, with 275 kLOC and about 5,000 classes). In the following, we describe the main features of CodeCity that provide support in program comprehension.

City overview. In Figure 1 we see an example of City overview representing ArgoUML, a 140 kLOC Java system for handling UML diagrams. The visualization allows us to easily spot some patterns such as the two massive buildings (potential god classes [6]), some antenna-shaped constructs, a number of classes looking like parking lots, and a large number of small houses. The visualization is interactive and navigable using the keyboard, *i.e.,* it is easy to zoom in on details of the city or to focus on one specific district by spawning separate windows. The overall goal of the city overview is to give a first impression of both the magnitude of the system and the structural distribution of its intelligence.

3. IMPLEMENTATION

CodeCity is written in Smalltalk and built on top of Moose [3], which provides an implementation of the FAMIX [2] language-independent meta-model. Since we work with a model and not directly with the source code, we first need to obtain the former. For Java and C++ systems we use iPlasma [5] to parse the code and export the model to an interchange format readable by Moose, while for building the models of Smalltalk systems we use Moose itself. Based on the model of the system, we then create interactive visualizations in CodeCity, using Jun [1] for OpenGL rendering.

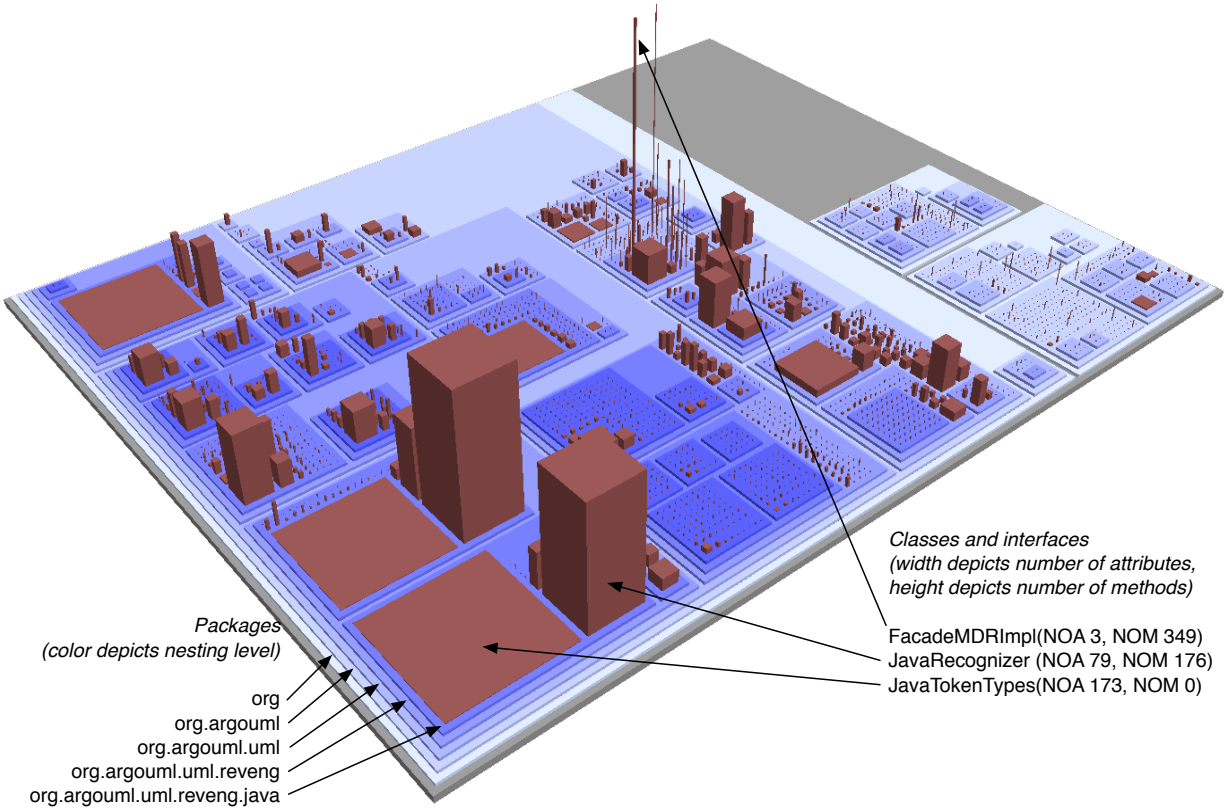

Figure 1: An overview of the city of ArgoUML v.0.24

CodeCity (see Figure 2) provides means to define view configurations, by letting the user specify which model elements to visualize, the figure types for each element type, the mapping set between software metrics and visual properties of the figures, layouts, etc. The created visualizations are interactive, allowing the user on the one hand to navigate the urban environment (*e.g.,* orbiting, rotation, panning, moving back and forth) and on the other hand to interact with the entities by using a query mechanism or manual inspection.

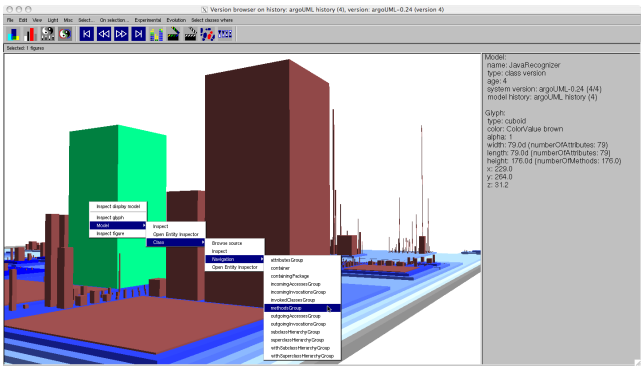

Figure 2: CodeCity in action

Tool Availability. CodeCity runs on every major platform and will soon be freely available for download at:
`http://www.inf.unisi.ch/phd/wettel/codecity.html`

4. REFERENCES

[1] A. Aoki, K. Hayashi, K. Kishida, K. Nakakoji, Y. Nishinaka, B. Reeves, A. Takashima, and Y. Yamamoto. A case study of the evolution of jun: an object-oriented open-source 3d multimedia library. In *Proceedings of International Conference on Software Engineering (ICSE)*, 2001.

[2] S. Demeyer, S. Tichelaar, and S. Ducasse. FAMIX 2.1 — The FAMOOS Information Exchange Model. Technical report, University of Bern, 2001.

[3] S. Ducasse, T. Gîrba, and O. Nierstrasz. Moose: an agile reengineering environment. In *Proceedings of ESEC/FSE 2005*, pages 99–102, Sept. 2005. Tool demo.

[4] M. Lanza and S. Ducasse. Polymetric views — a lightweight visual approach to reverse engineering. *Transactions on Software Engineering (TSE)*, 29(9):782–795, Sept. 2003.

[5] C. Marinescu, R. Marinescu, P. F. Mihancea, D. Ratiu, and R. Wettel. iplasma: An integrated platform for quality assessment of object-oriented design. In *ICSM (Industrial and Tool Volume)*, pages 77–80, 2005.

[6] A. Riel. *Object-Oriented Design Heuristics*. Addison Wesley, Boston MA, 1996.

[7] R. Wettel and M. Lanza. Program comprehension through software habitability. In *Proceedings of ICPC 2007 (15th International Conference on Program Comprehension)*, pages 231–240, 2007.

[8] R. Wettel and M. Lanza. Visualizing software systems as cities. In *Proceedings of VISSOFT 2007 (4th IEEE International Workshop on Visualizing Software For Understanding and Analysis)*, pages 92–99, 2007.