

CrocoCosmos – Analysis and Visualization of Hierarchically Clustered Graphs

Andreas Noack, André Preußner, Claus Lewerentz
Brandenburg University of Technology at Cottbus

CrocoCosmos analyzes and visualizes *abstractions* of graphs, because large irregular graphs are generally incomprehensible without abstraction. It enables the user to interactively control abstraction by filtering vertices and edges, and by aggregating vertices and edges along a hierarchical clustering.

The main application of CrocoCosmos is the comprehension, evaluation, and reduction of dependencies in large software systems [1,3,4]. However, it is not restricted to this domain, and has also provided new insights e.g. into hyperlink structures, bibliographical networks, and social networks [2,3].

Graph Model. The input of CrocoCosmos is a hierarchically clustered, attributed graph. A hierarchically clustered graph consists of a (plain) graph called underlying graph, and a tree called cluster tree, such that the leaves of the cluster tree are exactly the vertices of the underlying graph. For example, the underlying graph may model the classes of an object-oriented program and their inheritance relationships, and the cluster tree may correspond to the containment hierarchy of packages and classes of the program. Numerical attributes store additional properties of the vertices and edges, for example the number of statements in each class.

Navigation. CrocoCosmos analyzes and visualizes abstractions of the underlying graph. A systematic set of navigation operations enables the user to interactively control abstraction. Firstly, the vertices and edges of the underlying graph can be aggregated along the cluster tree. For example, the user can start with a package-level visualization of a software system, and then selectively zoom into packages of interest while keeping the global context. Secondly, arbitrary vertices and edges of the underlying graph can be filtered out, which allows, for example, to focus on the relationships between a few packages without being distracted by irrelevant information. The measurement values, layouts, and visualizations are automatically kept consistent with the current abstraction.

Layout. CrocoCosmos provides a variety of layout styles (see [4] for details), because different types of inferences about graphs require different types of layouts to support them. A task-oriented organization of the layout styles largely avoids tedious trial-and-error tuning of layout parameters. A distinguishing feature is the well-defined correspondence between properties of the layouts and properties of the graph, which enables the user to draw valid inferences about the graph from the layouts. As an example, supported inferences about software systems range from local properties like the coupling of single classes to global properties like the quality and possible improvements of the package hierarchy. The layouts are formalized as energy models and efficiently computed using the Barnes-Hut force calculation algorithm.

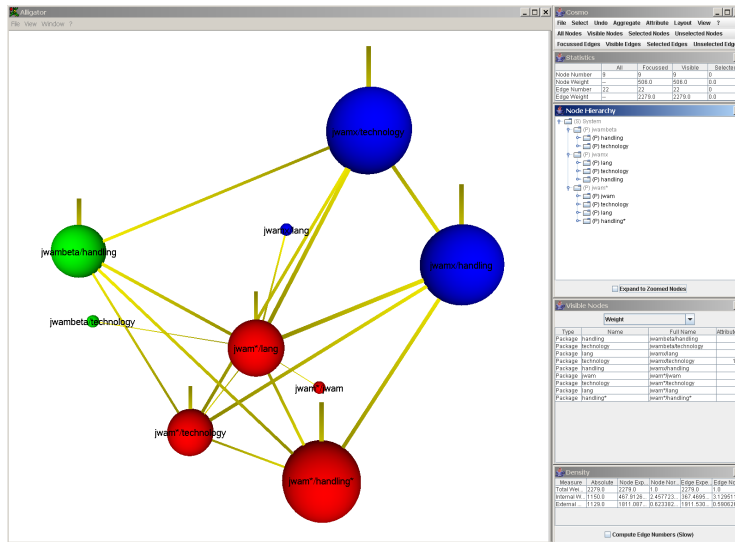


Fig. 1. Various views of a model of a software system in CrocoCosmos.

Measurement. CrocoCosmos not only accepts vertex and edge attributes as input, but also computes various attributes itself, including the degrees of vertices, the number of aggregated vertices or edges of the underlying graph, and the clustering indices discussed in [2]. When applied to models of software systems, several of these measures coincide with standard software metrics, and are useful for assessing design quality and finding design problems.

Visualization. CrocoCosmos provides various complementary views of (abstractions of) graphs, including a three-dimensional box-and-line visualization, an adjacency matrix visualization, a tree widget showing the cluster tree, and lists of vertices and edges for the detailed examination of attribute values. The mapping of vertex and edge attributes to visual attributes in the graphical views can be controlled by the user. For example, the volume of the sphere representing a software package can reflect its number of statements, its number of classes, or the number of packages to which it is related.

References

1. Dirk Beyer and Andreas Noack. Clustering software artifacts based on frequent common changes. In *Proc. IWPC 2005*, pages 259–268. IEEE, 2005.
2. Andreas Noack. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007.
3. Andreas Noack. *Unified Quality Measures for Clusterings, Layouts, and Orderings of Graphs, and Their Application as Software Design Criteria*. PhD thesis, Brandenburg University of Technology, 2007.
4. Andreas Noack and Claus Lewerentz. A space of layout styles for hierarchical graph models of software systems. In *Proc. SoftVis 2005*, pages 155–164. ACM, 2005.