Towards Enabling Service Interoperability by Graph-based Meta-Modeling

Andreas Winter

Institute for Software Technology
University of Koblenz
D-56070 Koblenz
Universitätsstraße 1
www.uni-koblenz.de/~winter/
mailto:winter@uni-koblenz.de

Abstract

Sharing functionality of various services in collaborative tool environments requires to describe, transform, *and* exchange *data by using generic and adaptable techniques. This paper sketches a graph-based meta-modeling approach to enable service interoperability in repository based systems.*

Service Interoperability

Interoperability is the challenge of enabling software tools from different suppliers to work together. Aspects of tool interoperability deal with connecting different platforms, integrating tools by common user interfaces, exchanging data between tools, or sharing control by encapsulated libraries/APIs or predefined communication protocols [Wa89, Ea91, Si00].

Current work on tool interoperability focuses on coupling tools based upon *services* offered by single components (e. g. [CJ05]). *Service interoperability* addresses tool interoperability by means of exchanging well defined and self-contained functionality. Instead of combining certain tools, service interoperability focuses on offering and applying required *functionality*, independent of specific tools.

In distributed service-oriented, collaborative tool environments, services have to be specified at least by their *address*, showing where to find a service, by their *interfaces*, describing how to request a service, by their *business protocols*, describing the order in which subtasks have to be invoked, and by their *semantics*, specifying the functionality of a service in human readable form or even better in a form which is interpretable by further clients (cf. e. g. [ACKM04]).

Tools applied to software development and software reengineering found on *repository-based architectures*. Information on software systems are usually stored in complex graph-based data structures [HWS00]. Various services providing functionality for analyzing or restructuring software systems (i. e. services offering parsers, architectural analysis, data- and control flow analysis, slicing, software measurement, software visualization, ...) use specialized data structures optimized for efficient calculation. Coupling these services requires to map the data used in interoperable services. This mapping has to be encapsulated in appropriate filters within the components interface.

Meta-Model-based Data-Exchange

Interoperable components have to agree upon exchanged data structures and supply means to accommodate clients data to the data required by service provider, and vice versa. Thus, service interoperability in repository-based environments (which are not only restricted to software engineering purposes) necessitate powerfull and adaptable mechanisms to *define*, *transform*, and *exchange* data.

Meta-Modeling provides such a mechanism. Meta-models define data structures of data to be exchanged. While specifying the data used by service requester and service provider, they offer the formal base to specify transformations to convert these data. Combined with a notation for packing data and data structures, meta-models also define an exchange notation.

GXL Graph-eXchange Language

The GXL Graph-eXchange Language [HWS00, Wi02] was developed to provide an easy to use, general purpose mechanism for packaging and exchanging graph-based data

with its corresponding schema or structure information. GXL originated in discussions held at various international reengineering conferences and was ratified as a standard exchange format in reengineering at Dagstuhl Seminar 01041 “Interoperability of Reengineering Tools” in January 2001 [Dag01]. In the years since GXL ratification, GXL was applied e. g. to approaches in software reengineering, graph transformation, graph visualization, biochemical analysis, and business process modeling.

GXL is an XML based configurable exchange format for graphs. It can be adapted to a broad variety of different types of graphs and hypergraphs. The adaptability of GXL is based on *meta-models* specifying the required graph structure. Thus, GXL representations of graphs typically consist of two parts: in a first part, the actual *graph* is specified, and, in a second (optional) part the graph structure is defined in a *graph schema*.

GXL makes use of the UML unified modeling language [RJB99] and the XML Extensible Markup Language [W3C04]. For visualization purposes, meta-models defining graph structures are depicted by UML class diagrams. Instance data, as well as representations of meta-models, are exchanged by appropriate XML documents. In contrast to other XML-based approaches like XMI [OMG02], GXL is kept very simple to avoid unneeded overhead and uses an elaborated meta-(meta-)modeling approach, which leads to use *one* common exchange language (specified by the GXL DTD or XML-Schema) to represent both, instance and schema data.

To support service interoperability, GXL-meta-models will found the base for specifying transformations between data. These transformations specify *filter-components* to transfer data from client components to server components and vice versa.

While exchanging data with GXL is an established support in software reengineering and neighboring disciplines (cf. [GXL04]), the extension of GXL to provide means for specifying data transformation is ongoing work.

This Talk

This talk will introduce a meta-modeling based approach to service interoperability by using the generic GXL-approach. The approach will be exemplified by collaborative software reengineering components.

References

[ACKM04] Alonso, G., Casati, F., Kuno, H., und Machiraju, V.: *Web Services, Concepts, Architectures and Applications*. Springer. Berlin. 2004.

[CJ05] Cordy, J. und Jin, D.: Factbase Filtering Issues in an Ontology-Based Reverse Engineering Tool Integration System. In: *J.-M. Favre, M. Godfrey, A. Winter (eds.): Integrating Reverse Engineering and Model Driven Engineering, 2nd International Workshop on Meta-Models and Schemas for Reverse Engineering. Electronic Notes in Theoretical Computer Science*. to appear. 2005.

[Dag01] J. Ebert, K. Kontogiannis, J. Mylopoulos: Interoperability of Reverse Engineering Tools. `http://www.dagstuhl.de/DATA/Reports/01041/`. 2001.

[Ea91] Earl, A.: A Reference Model for Computer Assisted Software Engineering Environments Frameworks. *Softwaretechnik-Trends*. 11(2):15–48. Mai 1991.

[GXL04] GXL: Graph Exchange Language. `http://www.gupro.de/GXL/tools/tools.html`. 2004.

[HWS00] Holt, R. C., Winter, A., und Schürr, A.: GXL: Toward a Standard Exchange Format. In: *7th Working Conference on Reverse Engineering*. IEEE Computer Society. Los Alamitos. pp. 162–171. 2000.

[OMG02] OMG XML Metadata Interchange (XMI) Specification. `http://www.w3.org/TR/2000/REC-xml-20001006.pdf`. January 2002.

[RJB99] Rumbaugh, J., Jacobson, I., und Booch, G.: *The Unified Modeling Language Reference Manual*. Addison Wesley. Reading. 1999.

[Si00] Sim, S. E.: Next Generation Data Interchange, Tool-to-Tool Application Program Interfaces. In: *7th Working Conference on Reverse Engineering*. IEEE Computer Society. Los Alamitos. pp. 278–280. 2000.

[W3C04] Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation. W3C XML Working Group. `http://www.w3.org/TR/2004/REC-xml-20040204`. 4 February 04 February 2004.

[Wa89] Wasserman, A. I.: Tool Integration in Software Engineering Environments. In: *International Workshop on Software Engineering Environments (SEE), Chinon, France*. pp. 137–149. 1989.

[Wi02] Winter, A.: Exchanging Graphs with GXL. In: Mutzel, P., Jünger, M., und Leipert, S. (Eds.): *Graph Drawing, 9th International Symposium, GD 2001 Vienna, Austria, September 23-26, 2001. Revised Papers*. LNCS 2265. Springer. Berlin. pp. 485–500. 2002.