VARDA Rule Design and Visualization Tool-Chain[⋆]

Grzegorz J. Nalepa and Igor Wojnicki

Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
`gjn@agh.edu.pl, wojnicki@agh.edu.pl`

Abstract. A prototype design tool-chain (VARDA) for the ARD hierarchical rule design method is presented in the paper. It is implemented in the Unix environment using Prolog and Graphviz for design visualization.

1 Rule Design with ARD+

The HeKatE project (`hekate.ia.agh.edu.pl`) aims at providing design methods and tools that support a rule-based systems design process. Currently HeKatE supports a *conceptual design* with the ARD+ method (Attribute Relationships Diagrams). The main, so-called *logical design*, is carried out with the use of the XTT method (eXtended Tabular Trees) [1]. This paper is dedicated to the presentation of VARDA (*Visual ARD Rapid Development Alloy*), a rapid prototyping environment for the ARD+ method.

The main objective of ARD+ is to capture relationships between *attributes* in terms of *Attributive Logic* [2]. *Attributes* denote certain system *property.* ARD+ captures *functional dependencies* among these *properties.* Such dependencies form a directed graph with nodes being properties. There are two kinds of attributes: a *conceptual attribute* is an attribute describing some general, abstract aspect of the system to be specified and refined; a *physical attribute* is an attribute describing a well-defined, atomic aspect of the system. There are two transformations allowed during the ARD+ design process. *Finalization* transforms a simple property described by a conceptual attribute into a property described by one or more conceptual or physical attributes. It introduces a more specific knowledge about the given property. *Split* transforms a complex property (source property) into a number of properties (resulting properties) and defines functional dependencies among them. Attributes are unique, the same attribute cannot describe more than a single property. The evolution of the hierarchical ARD+ design is captured within the *Transformation Process History.* The TPH forms a tree that allows for recreation of any stage of the design.

[⋆] The paper is supported by the Hekate Project funded from 2007–2009 resources for science as a research project.

2 VARDA Tool

VARDA has a multi-layer architecture that consists of: knowledge base to represent the design, low-level primitives (adding and removing attributes, properties and dependencies), transformations (finalization and split including defining dependencies and automatic TPH creation), low-level visualization primitives (generating data for the visualization tool-chain, so-called DOT data), high-level visualization primitives (drawing actual dependency graph among properties and the TPH), and interoperability primitives (using XML). VARDA is implemented in Prolog, it has currently over 900 lines of Prolog code. Attributes, properties, dependencies and TPH are represented as Prolog facts.

A proper visualization of the current design state is the key element. It allows to browse the design more swiftly and identify gaps, misconceptions or mistakes more easily. A tool-chain of well proved tools is assembled to provide actual visualization of the ARD+ and TPH graphs. The tool-chain is based on three components: SWI-Prolog (`www.swi-prolog.org`), GraphViz (`www.graphviz.org`), and ImageMagick (`www.imagemagick.org`). The detailed interaction between the tool-chain components is given in Fig. 1. There are two scenarios the tool-chain is used, generating diagrams for an already designed system described in Prolog, and on-line during the design process.

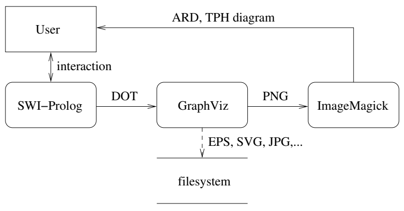

Fig. 1. Visualization tool-chain

3 Future Work

VARDA is currently being extended with a native, interactive, full-fledged shell, supporting the ARD+ design process based on a text-oriented environment with heavy hinting. The next step is developing a visual design environment for ARD+ based on VARDA using GUI toolkits in Java and Prolog XPCE.

References

1. Nalepa, G.J., Ligęza, A.: A graphical tabular model for rule-based logic programming and verification. Systems Science **31**(2) (2005) 89–95
2. Ligęza, A.: Logical Foundations for Rule-Based Systems. Springer-Verlag, Berlin, Heidelberg (2006)