User Interfaces for and by Domain-Specific Visual Modeling

In general, coding graphical user interfaces (GUIs) can be tedious and error-prone. In addition, the cost of maintaining this rarely documented code is usually exorbitant. We propose to investigate Model-Based development of User Interfaces (MBUI), focused on domain-specific modeling environments. Most current MBUI prototypes do not achieve a running user interface [2]. In my research, I propose to improve the quality of GUI development by providing formalisms and tools for the specification of GUI structure and behavior with a focus on GUIs for domain-specific modeling. This will include analysis, simulation and code synthesis components.

Domain- and formalism-specific modeling have the potential to greatly improve productivity [1]. They are able to exploit features inherent to a specific domain or formalism. This will for example enable specific analysis techniques or the synthesis of efficient code. In particular, a domain-specific visual modeling environment maximally constrains users, allowing them, by construction, to only build syntactically and (for as far as this can be statically checked) semantically correct models. Furthermore, the domain-specific, visual syntax used matches the users' mental model of the problem domain.

UML is currently the standard modeling language for modeling object-oriented design [2]. UML does however not provide enough support for modeling UIs [2]. Several formalisms have been proposed to facilitate the specification of UIs. Algebraic specification, (graph) grammars, task description languages, transition diagrams with or without extensions, and rule-based systems have been suggested [4]. None of these was widely adopted. This is mainly because of two factors: (1) writing these specifications is not easy (the formalisms are not domain-specific enough) and (2) these specifications are not executable. Furthermore, description of domain-specific semantic as well as visual constraints on the one hand and of reactive behavior of the GUI on the other hand needs to be possible in a unified manner.

The starting point for my research will be AToM^3, which is A Tool for Multi-formalism and Meta-Modeling [5] that is being developed at the Modeling, Simulation and Design Lab of the School of Computer Science at McGill University. AToM^3 will also be one of my test cases.

This project will change the way programmers develop GUIs, in particular for domain-specific modeling. Besides offering them a GUI-domain specific language, which they understand and can easily work with, it will offer them analysis, simulation, and code synthesis capabilities. This new approach will minimize the cost of software development and maximize maintainability.

References:

[1] J. de Lara and H. Vangheluwe. Defining visual notations and their manipulation through meta-modelling and graph transformation. Journal of Visual Languages and Computing, Pages 309-330, June - August 2004. Special Issue on Domain-Specific Modeling with Visual Languages.

[2] P. P. de Silva, T. Griffiths, and N. W. Paton. Generating User Interface Code in a Model Based User Interface Development Environment. In Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI2000), V. Gesu, S. Levialdi and L. Tarantino (Eds.). Palermo, Italy, Pages 1555-160, ACM Press, May 2000.

[3] P. P. da Silva and N. W. Paton. User Interface Modelling with UML. In Information Modelling and Knowledge Bases XII (10th European-Japanese Conference on Information Modelling and Knowledge Representation, Saariselka, Finland, May 2000), H. Kangassalo, H. Jaakkola, E. Kawaguchi (Eds), Pages 203-217, IOS Press, Amsterdam, 2001.

[4] D. A. Carr. Toward More Understandable User Interface Specifications. Design, Specification, and Verification of Interactive Systems '96, F. Bodart & J. Vanderdonckt eds., Springer-Verlag, Vienna, 1996, Pages 141-161.

[5] J. de Lara and H. Vangheluwe. AToM^3: A Tool for Multi-Formalism Modelling and Meta-Modelling. Lecture Notes in Computer Science 2306, Pages 174-188. Also presented at Fundamental Approaches to Software Engineering - FASE'02, in European Joint Conferences on Theory And Practice of Software - ETAPS'02 . Grenoble. France, April 2002.