HiPeS
High Performance Software

Controlling the front-end

Intentional GUI Modelling with StUIML

Jim van Dam, jim.van.dam@hipes.nl
HiPeS, Amsterdam
Netherlands

Adding a GUI front-end
Adding a front-end to back-end functionality is not as easy as it seems. It is not simple for new applications created from scratch, and it's even harder for a legacy system, interactive processes resulting from BPM[1] efforts or a collection of services developed in the spirit of SOA[2]. It is not uncommon for development teams to spend man-years on a limited set of screens and to spend a significant percentage of the maintenance budget on adapting to changing UI requirements and supporting new presentation technologies.
And the problem does not go away after the first delivery of the GUI. Front-ends today are developed from a technology perspective and the underlying intention of the UI is not captured explicitly. When the required presentation technology changes, the front-end has to be rebuild from scratch.

What's wrong with the current software development practice for the presentation layer?
Developing the presentation layer still requires hard manual labour. While Object Oriented models offers extensive modelling expression for the business layer, there is no equivalent for the presentation layer. The MDA and software factory approach, promising as they may seem, so far have not much improved the UI development practice. Maintaining a presentation layer with a rapidly changing UI technology platform and no future proof standard is a challenging task.
The Implementation techniques for the presentation layer have changed frequently in the last few years and are likely to change again given the array of competing presentation technologies: DHTML/javascript (Ajax), Swing, SWT/JFace, JSP, JSF, ASP.Net, Winforms, WebForms, InfoPath, XAML, XForms, XUL, RUIB, XSWT, Canoo, Laszlo, Flash and Flex. This list does not contain any obscure niche implementation. Every one of these technologies is either used in hundreds and some even thousands of real-life applications or it is presented as the new presentation layer technology by one of the major players like Microsoft, Sun, IBM, Oracle and Adobe or organisations like W3C and OMG.

What can StUIML do to help?
StUIML (Standard User Interface Modelling Language) is to the presentation layer what BPEL and BPML are for business processes: a modelling language that, in a well defined context, is precise enough to generate code from. Using StUIML the intention of a UI can be specified independent of the presentation technology.
After modelling the UI and capturing the intentions of all UI elements, a fully functional front-end is generated for the desired technology platform.

[1] Business Process Modeling
[2] Service Oriented Architecture

HiPeS
High Performance Software

What are the benefits of using StUIML?
Capturing the UI intentions in StUIML has the following benefits:

1. StUIML is platform independent so changes in presentation technology only affect the StUIML generator: based on a single StUIML model, front-ends can be generated for multiple platforms based on different mappings. Current UI development efforts are not wasted if you have to switch to a new UI technology platform[3].
2. Separate mapping of StUIML elements to technology platform elements ensures standardization of the UI.
3. The StUIML models reference OO domain, component and service models and these references are automatically validated. Any change in the OO models causing invalid StUIML elements is immediately flagged. The impact of changes is clear, because of the automated propagation.
4. Developing a UI has become a modelling activity instead of a technology focused effort. Only a few people need to have the lower level technical expertise. Quick feedback during initial prototyping and all UI development is the result.
5. Developing StUIML models and generating a fully functional front-end saves at least 60% of development time and even more during maintenance.

The end result is higher productivity, less maintenance costs and increased flexibility for multi-channelling support when developing front-ends.

What is StUIML exactly?
StUIML itself is a modelling language, but the StUIML environment contains editors and code generators. StUIML and its environment can run as a stand-alone application on any platform. Using StUIML, developers and analysts can specify intentions of UIs and UI platform specialists can define the mapping between these intentions and technical implementations in Java Swing, JSF or WPF[4] for example.

Multi Channelling
Applications offered to end-users frequently have to be used by internal administrative employees as well. Most projects take the web based approach for end users which forces internal employees to use a browser for administrative work. Despite advances in web technology they are no match for integrated rich clients when it comes to supporting call centres or office work.
Developing completely different types of applications for different target groups is not an option when doing manual UI development as the costs would be multiplied. Although the different applications have the same intentions, they cannot be merely styled using CSS for instance.

When using StUIML the intentions can be specified and generated to different targets, supporting complete flexibility in choosing a technology platform, specialized flows, styling and look and feel.

[3] WPF on Windows Vista, web clients, PDA's and general devices
[4] Windows Presentation Foundation

Three common scenarios using existing back-ends

Developing front-ends is part of most software efforts when *new* applications have to be build. There are three common scenarios, however, for developing front-ends when it concerns *existing* back-ends:

1. *Legacy systems*
   A large number of business critical COBOL programs are still running strong. Sometimes all that needs to be done to prolong the lifetime of this software with a character based interface is to integrate the UI in the rest of desktop environment of end-users. Merely screen scraping is not good enough anymore: the UI should seamlessly integrate with the rest of the desktop interface and the new UI is not a one-to-one translation of the old screens.
2. *SOA*
   SOA based software can support multiple clients. In fact one of the arguments to choose a SOA is to clearly separate the back-end functionality from the clients and enable supporting a number of different types of clients with a single set of services. With SOA this separation is obvious, but the clients still depend on the service interfaces and these are not always as frozen as the theory prescribes.
3. *BPM*
   Modelling business processes and managing the dynamics of these processes has received renewed attention during the last few years. Tooling for BPM has improved considerably, but developing UIs for the interactive processes remains an activity requiring low level technical expertise. The business and process analysts and designers are able to model the processes and the required dynamics, but they are forced to leave the UI specification and implementation to technical UI platform specialists.

StUIML is not only beneficial for UI development in a green field operation, but also has important advantages in the scenarios described above.

Who uses StUIML today?

The predecessor of StUIML has been used for more then 10 years in over 40 business critical banking and insurance applications. The presentation layer generated by the environment changed from Windows 3.1 in 1995 to Ajax implementations without the need to change the UI models.
A recent version of StUIML is used in a large[5] administrative system where the UI models are generated into Java Swing.
A large player on the MDA tools market has confirmed to be planning to implement a ‘one-to-one mapping’ of StUIML in future releases.

[5] 700.000 lines of code