

Overview of Instigate Application Framework

Instigate CJSC
www.instigatedesign.com

July 2009

Abstract

Instigate Application Framework is a framework for cost effective development of high quality desktop applications with focus on EDA applications. It is based on industry standard Model-Viewer-Controller architecture. Unlike most other frameworks based on the same architecture, Instigate Application Framework covers not only the Viewer component, but also the Controller component, allowing developers to fully concentrate on Model component only. Influenced by ideas of modern generic programming, Instigate Application Framework does not enforce any syntactic requirements on the Model component, thus, encouraging developers to use the data structures and design patterns they prefer.

1. INTRODUCTION

Instigate Application Framework offers a cross-platform desktop application development. It supports rapid application development based on ready-to-use design patterns. In particular the Model-Viewer-Controller architecture is enforced on the application.

Instigate Application Framework provides a number of nice looking high level GUI components for making up the Viewer component of the application, which is supposed to be sufficient for a typical desktop application. On the other hand, developers may extend the existing components as well as provide their own ones, whenever the provided functionality is insufficient. The GUI components of the Instigate Application Framework are based on Instigate Toolkit library which uses either GTK+ or Qt library for rendering widgets. It is also possible to combine both widget toolkits into a single executable and to choose the exact toolkit at start time via command line options. This makes applications, developed by Instigate Application Framework, available for a wide range of platforms without recoding.

The Controller component of applications is mostly generated by Instigate Application Framework from the Model description provided by developers. Unlike most existing frameworks in the market, Instigate Application Framework does not enforce its data structures and architecture on the Model component. In contrast, the Model component is developed completely independent from Instigate Application Framework, instead just a description is being provided. Thus, the interface of Instigate Application Framework is mostly declarative, rather than procedural or object oriented.

2. HISTORY

The five-year experience of desktop application development at Instigate has revealed a clear pattern in software development. First of all it is the Model-Viewer-Controller architecture with an embedded scripting language as a Controller component. The problems we encounter during the development include (but are not limited to) the following:

- A typical application contains hundreds of dialog windows and they are required to look similar.
- The embedded scripting language needs to be extended with hundreds or even thousands of commands and again for user convenience they are required to maintain the same course.
- Most applications require serialization / deserialization (also known as save / load) of objects and this needs to be fixed every time the structure of objects is modified.

At Instigate all these problems are usually solved in the spirit of generic programming. That is, a typical application contains a generic dialog builder, a generic command generator and a generic object exporter / importer. Instigate Application Framework emerged as an attempt to combine all those generic components inside a single library with a single interface and a single set of concepts. We have gone with the idea even further and designed high level interface for Instigate Application Framework. That is, instead of communicating with Framework in terms of GUI elements or scripting language details, users do that in terms of objects and properties.

After those goals were achieved and the first application was developed with Instigate Application Framework, we realized that we have enough semantics information to provide more high level components and services than it was planned at the beginning. For example, Model description is sufficient for construction of a tree viewer which displays the hierarchy of objects, or copying an object. On the other hand, as Framework covers also the Controller component, it is able to take care of undoing actions with Model. This led to reconsideration of Instigate Application Framework as a complete and valuable framework for desktop application development. Since then, Instigate Application Framework has been enhanced with a number of new components and services and has been used in most applications developed at Instigate.

3. ARCHITECTURE

Instigate Application Framework consists of two main components - GUI and UI.

3.1 GUI

The GUI component of Instigate Application Framework consist of a number of easy customizable high-level widgets methods for combining them to make up the Graphical User Interface of an application. It is built on top of Instigate Toolkit library which is a light-weight cross platform widgets toolkit that uses either GTK+ or Qt for rendering widgets. It is also possible to combine both widget toolkits in a single executable.

One of distinguishing features of Framework GUI library is automatic GUI completion: GUI library provides a service for evaluation of partially complete commands. Whenever mandatory options are missing in the partial command, a dialog window opens which asks for missing option. This is not only nice from the users point of view (as he / she does not need to remember all options of all commands or consult the help every time), but it makes the library extremely flexible. In fact, all actions that are performed by GUI components are partial commands that are being processed by automatic GUI completion engine. For example, to add a menu item one simply specifies a partial command, instead of a handler callback.

A part of widgets in Framework GUI library support Navigator-Viewer architecture. Navigators are widgets that display the content of a set of widgets and allow navigation through them. Viewers, in contrast, display a given object in a specific manner. Viewers and navigators may be combined together to form a user-friendly graphical interface. There are widgets, such as Netlist Viewer widget, that are both viewers and navigators.

3.2 UI

As it was already stated, Instigate Application Framework does not enforce any syntactic requirements on Model component of applications. That is, the Model component is designed completely independent from Instigate Application Framework and its description, referred here as Meta-model, is provided. UI component of Instigate Application Framework provides interface for implementing Meta-model and is responsible for generation of Controller component in the form of Command Line Interface. The Controller keeps the track of actions and supports undoing them. The Command Line Interface constructed by UI component is based on Instigate Scripting library which wraps Tcl and Python libraries. Thus, applications built by Instigate Application Framework have both Tcl and Python as embedded scripting language.

The Meta-model is the object-oriented description of the Model component of an application. Objects are statically typed and are described by their properties and methods. Based on Meta-model various services, such as object serialization or copying, is provided by UI. Nevertheless, there are still services and components for which the object-oriented description is not sufficient. For example, for the 'Netlist Viewer' GUI component, which displays an object in the form of netlist, more conceptual information is required. That is, to display an object in the form of netlist, the object must satisfy some requirements. Such requirements are grouped into concepts and the objects satisfying the requirements are said to be models of the concept. Thus, apart

from object-oriented description, Meta-model contains also specification of types as models of preexisting concepts.

4. EDA FRAMEWORK

Frameworks, generally speaking, aim to facilitate developers by allowing them to concentrate on meeting software requirements rather than on unimportant low level details needed to provide a working system. However, the problem with frameworks is that they add "code bloat" - a redundant and unnecessary code. For example, Instigate Application Framework aims to provide a lot of high level components and services to users, but a part of them (such as 'Netlist Viewer' or 'Verilog Parser') is targeted to specific industries (such as EDA). Of course, this introduces a "code bloat" for applications not using them. The problem is solved in form of add-ons to Instigate Application Framework. That is, instead of defining industry specific concepts and components directly, Framework provides mechanisms of extending it with them.

EDA Framework is an add-on to Instigate Application Framework which contains components and services that are specific to EDA industry. These components include:

- netlist concepts - module, instance, port, etc.,
- GUI components for viewing netlist graphically,
- parsers and exporters for widespread HDL languages - Verilog, EDIF and VHDL,
- utilities for manipulation with netlist - placers, routers, etc.

EDA Framework, in contrast to most other EDA oriented libraries, operates on native objects of Model component, via Meta-model provided to Instigate Application Framework. This frees users from the need of linking the functionality of libraries to their Model. Thus, benefits provided by EDA Framework come for at almost no R&D effort spent by users.

5. CONCLUSION

Instigate Application Framework is an attempt to reuse a huge piece of software in desktop application development. It pushes the idea of code reuse extremely further. The idea has proved to be working at Instigate. It is the time to present it to the public.