IBM ILOG Multi-platform Graph Layout Technology

Georg Sander

IBM ILOG Visualization Group
georg.sander@de.ibm.com

Introduction

Visualization components from IBM provide a comprehensive set of graphics products for creating highly graphical, interactive displays, including diagrams, gantt charts, maps, business dashboards, business charts, telecom displays, SCADA/HMI Screens and many more. User interface developers reduce development risks and implementation time when deploying the IBM visualization technology. The components are available for many different platforms: various C++ platforms, Java Swing applications, Java Eclipse plugins, thin client AJAX applications, for the Microsoft .NET and the Adobe Flex platform.

Graph layout and label layout are key functionalities of many our components. It is part of the algorithmic core of the components that gets enriched by a graphical display layer (classes that actually draw elements on the screen). Graph layout needs the highest graph-theory expertise and is optimized the most. In order to leverage the platform infrastructure in the best way, the graphical display layer was specially developed for the different platforms (Swing, Microsoft .NET, Adobe Flex and various C++ and AJAX platforms). However, for the nonvisual and mathematical part, this is not needed and would be a waste of manpower, since the pure mathematics is always the same, no matter whether you are in C++, Java, C# or ActionScript. On the other hand, IBM ILOG follows a strict *uniform platform strategy* that avoids hybrid technologies (for instance no C++ inside Java). Even the reference documentation must use the native tool (for instance, no Javadoc inside C#). Last but not least, the support engineers must be able to work with the code for debugging or consulting assignments even when they are specialists of only one single platform.

Graph Layout Translation Technology

How to provide graph layout technology in C++, Java, C# and ActionScript without the need for huge development teams that redevelop the same algorithms over and over again? IBM takes an automatic translation approach from a single common source. The algorithms are implemented in a central repository and are translated by a set of tools fully automatically to the different target languages. The advantage: the graph layout team can concentrate on improving the layout

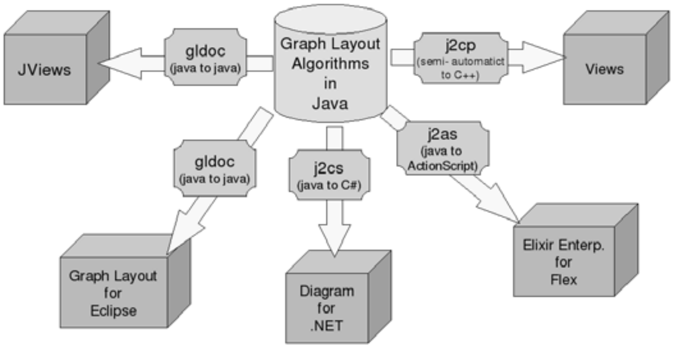

Fig. 1. Graph Layout Translations at IBM ILOG

algorithms instead of spending time on platform coding, and every improvement in the layout algorithms is immediately available on all platforms without effort.

When Microsoft introduced the .NET platform, some simple tools were also introduced to convert Java to C#. Most of these tools were one-shot translations that required manual adaptations of the translation results. As disadvantage, all manual adaptations are lost whenever the common source needs to be retranslated (for instance whenever an incremental improvement was added to a graph layout algorithm). Therefore, IBM ILOG has developed its own tool set that does not require any manual adaptations of the result of the translation. The development is done entirely on the common source that is fed into the translator tools. Some other Microsoft tools converted Java bytecode to C# bytecode, with the disadvantage that the translated code is not human readable and hence unsuitable for the IBM support engineers, and that the translation does not provide any API documentation in the same step. The IBM tool set however translates source to source and converts all the code comments into the canonical format expected by the targeted platform. This allows to produce a reference manual by using the native reference tool on the translated source code.

We develop the graph layout algorithms in a simplified Java based language, a subset of Java enriched by an annotation formalism. Annotations specify which Java methods should become C# properties, or how enumerations are mapped in ActionScript. Since these annotations are comments for Java, it allows for quick turnaround time when testing new algorithms, as the original code can be compiled with Java. However, the production code contains only the result of the source to source translations, even for the Java products (a Java to Java translator converts GUI related code from the common source alternatively to Swing for IBM ILOG JViews Diagrammer, or to SWT/GEF for IBM ILOG JViews Graph Layout for Eclipse). Other tools translate the common source to C# for IBM ILOG Diagram for .NET, to ActionScript for IBM ILOG Elixir Enterprise for Adobe Flex, or to C++[1] for IBM ILOG Views.

[1] In the current state, all translations are fully automatic except C++, which requires still small manual adaptations for the memory management.