Using WYSIWYG GUI Tools With UML

Ilya Lipkin
677th Aeronautical Systems Group

Martin Guldahl
309th Software Maintenance Group

This article will discuss the merging of Unified Modeling Language (UML) with “what you see is what you get” (WYSIWYG) graphical user interface (GUI) tools. The topics presented—and discussion of an example with benefits and hazards—will show that the merged solution can increase productivity and provide an improved rapid prototyping platform.

The UML language is complete enough to allow the creation of auto-generated code that implements the design. The code can be generated from the system description of the model through the use of diagrams and other model elements.

Overview of UML
The focus of UML is to model systems using object-oriented concepts and methodology. UML consists of a set of model elements that standardize the design description. These elements include a number of fundamental basic model elements and modeling concepts, in addition to views that allow designers to examine a design from different perspectives and diagrams to illustrate the relationships among model elements.

Several views—such as Use Case View, Logical View, Component View, Concurrency View, and Deployment View—create a complete description of the system design. Within each view, an organized set of diagrams and other model elements are visible. Diagrams include use case diagrams, class diagrams, object diagrams, sequence diagrams, collaboration diagrams, state-chart diagrams, activity diagrams, component diagrams, and deployment diagrams. Some key primitive model elements are states, transitions, signals, classes, class roles, attributes, and operations [1].

The UML language is complete enough to allow the creation of auto-generated code that implements the design. The code can be generated from the system description of the model through the use of diagrams and other model elements.

Overview of WYSIWYG GUI Tools
The WYSIWYG concept is a well-known technique that states that the end-product will look, act, and behave the same way as it does being designed on-screen from the developer to the end-customer look and
Using WYSIWYG GUI Tools With UML

77th AES Wing, 677th Aeronautical Systems Group, 2300 D Street, Bldg 32, Wright-Patterson AFB, OH, 45433-7249

Approved for public release; distribution unlimited

Security classification of:
- Report: unclassified
- Abstract: unclassified
- This page: unclassified

Number of pages: 5
feel. Currently, there are many tools and products on the market that make the development of the GUIs easier for embedded applications.

One of these tools is the Tilcon IDS. Although there are other tools—such as Altia Design and the Virtual Avionics Prototyping System, which are in many ways similar to Tilcon—the choice over other similar tools was determined using a CMMI Decision Analysis and Resolution (DAR) matrix [5].

The DAR was based on criteria such as customer service and technical support by the vendor, operating system neutrality, cost per use, development seat licenses, performance of the solution, ease of development, and training costs. Each was assigned a weight factor with respect to the importance for the project. In addition, a small prototype was implemented with some of these tools to serve as an input for the DAR. The winning solution was selected from the highest cumulative score.

The UML solution was designed with the idea of neutrality to the GUI development tool. Therefore, if the choice of Tilcon no longer becomes a best-fit selection, the impact to the UML back-end solution will be minimal when going with an alternative GUI.

The Tilcon IDS (see Figure 1) consists of three main components:

1. **The Tilcon Interface Builder.** The Tilcon Interface Builder is a WYSIWYG GUI design tool. An interface is created by drag-and-drop of high-level GUI objects like menus, buttons, text boxes, labels, etc. Each of these objects has properties associated with it. These properties, such as color, font, size, meter range, etc., can be tailored to meet a given requirement. As interface development is applied to each object so the application programming interface (API) can manipulate it, the resulting GUI design is saved in a .twd file. The Interface Builder does not require any programming skills to construct a GUI. Non-programmers such as graphic artists can use the Interface Builder to construct a GUI. It is highly recommended that a naming convention be followed so that programmers using the API can access the objects in a consistent way.

2. **The Tilcon Embedded Vector Engine (EVE).** The EVE is a platform-specific engine that renders the graphical interface. It reads the same file as the Interface Builder and ensures that the GUI is exactly the same as seen in the Interface Builder. The EVE runs as a separate process from the application and manages all user events (button presses, mouse clicks, etc.) and handles the screen display. This engine is available for many embedded operating systems, such as VxWorks.

3. **The Tilcon API.** An API is provided to connect the EVE to the application (see Figure 1). The API starts and stops the EVE, facilitates communication between the application and the GUI objects, and allows objects to be created, displayed, modified, or deleted. No low-level, platform-specific graphic calls are required; all of that work is handled by Tilcon [6].

**Merging of UML and a WYSIWYG Interface**

Now that the tools have been presented, it is time to discuss the benefits and comparison of UML and a WYSIWYG Interface. The Sequence of Event for the Traffic Light (SEI) [4] is shown in Table 1 and Figure 2. The benefits of using UML in conjunction with WYSIWYG tools include:

- **Ease of Use:** UML provides a standardized notation for describing system behavior, which is accessible to both technical and non-technical stakeholders.
- **Reusability:** UML models can be reused across different projects, reducing development time and costs.
- **Documentation:** UML models serve as a central repository for system documentation, making it easier to maintain and update the system.
- **Collaboration:** UML models facilitate collaboration among team members by providing a common language for discussing system requirements and design.

**Table 1: State Specification Template for the Sequence of Event for the Traffic Light (SEI) [4]**

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>System not running</td>
<td>Timer event set to 30 seconds</td>
</tr>
<tr>
<td>Green</td>
<td>On timer out event go to (goYellow) Yellow</td>
<td>Timer event set to 45 seconds</td>
</tr>
<tr>
<td>Yellow</td>
<td>On timer out event go to (goRed) Red</td>
<td>Timer event set to 10 seconds</td>
</tr>
<tr>
<td>Red</td>
<td>On timer out event go to (goGreen) Green</td>
<td>Timer event set to 30 seconds</td>
</tr>
</tbody>
</table>

**Figure 1: Tilcon IDS**

**Figure 2: Traffic Light Advanced Monitoring System States**
figure lists the object structure with the mentioned attributes were slightly adjusted for the mechanics of the speed gauge, pressure gauge, and traffic light animation can be used to demonstrate the effectiveness of this approach at the front end. Therefore, it is of interest to discuss how these objects were created in the Tilcon editor.

**Traffic Light**
The image for the traffic light in Figure 4 (created in Adobe Photoshop) was imported into the Tilcon Interface Builder and placed into a state object. One of many types available in the Interface Builder, this object type can display a different image depending on its state, which can be changed with messages sent to it through the API.

Using Adobe Photoshop with the Tilcon Interface Builder allows for an improved visual experience for the end customer, as graphics generated are generally more visually appealing at a fraction of the cost otherwise incurred if this was done in any other way (e.g., using C/C++).

In order to effectively identify the object for the UML application back end, the use of an API-unique ID such as “StateTrafficLight” needs to be assigned for the screen name. It is important to note: As more complex GUIs with hundreds of graphical objects are created in the Tilcon editor, a strict adherence to a naming convention will be required.

**Speed Gauge**
The speed gauge is a meter object that was created directly in the Interface Builder. This object is a standard development component that requires a minimum effort of customization. A meter object has many attributes—the range, alarm regions (green and red areas in the scale), tick marks, fonts, and colors—that were all entered in the Interface Builder. For this example, the previously mentioned attributes were slightly adjusted for the visual presentation.

**Speed Gauge Needle**
The needle for the speed gauge is a needle object that was also created from the standard Interface Builder object type. The needle selected is a predefined object type. Several predefined styles are also available or a custom style can be imported.

**Pressure Gauge**
Like the speed gauge, the pressure gauge is a standard object available in the Tilcon Interface Builder. It is an object of type “FillMeter” that represents meter position with a fill amount. The modifications for visual effects were primarily the adjustment of the visual width, font color, tick marks, and range.
Non-Programmers Collaboration
Using a WYSIWYG GUI design tool, it is possible to outsource the generation, prototyping, and user interaction analysis effort to usability experts, graphic artists, and other non-programmers. They no longer need to know anything about UML or any programming language. As well, a GUI object-naming convention should be followed for the project. This will allow programmers to access the objects in a consistent way [9].

Quick Prototyping
With WYSIWYG GUI tools, it is possible to adjust the user interfaces in a matter of minutes—even in the field—rather than hours or days with a comparable native programming solution. It also provides a way to easily evaluate different concepts and the integration with an event-driven GUI; to prototype solution is debugged in terms of visual actions and presentations to the operator. This allows for software developers to concentrate more on the UML part of the solution and spend more time enhancing functionality and quality.

Platform Neutrality
The development effort for the GUI is identical regardless of the deployment platform. Whether the target platform is Windows, Linux, VxWorks, or another operating system, the same solution is available and executes identically on the operating systems previously mentioned.

What Have We Gained?
The merged solution of a UML and WYSIWYG GUI development tool allows for several advanced flexibilities for the software creation effort. Most of those flexibilities are geared toward rapid development and prototyping. The separation between the front-end WYSIWYG GUI and a back-end UML provides the kind of platform development combination that can bring together technical and non-technical development efforts seamlessly.

What Have We Risked?
As one might expect, there are always drawbacks to any solution. Over the course of the project, several pitfalls of this merged tools approach have been identified. Here are some of the key issues:

Merging of Design Files
One of the features that has been sorely missed was the ability to merge GUI design files. Because the design resides in a binary file format, the source control tool could not merge them. Therefore, when multiple developers work on the GUI, developers have to be extra vigilant when checking in files to avoid overwriting each other’s changes.

Limited Development Language Support
Most of the WYSIWYG GUI tools have a predefined set of software languages that they support. The selection of the WYSIWYG GUI front-end might force the choice of a software language that is not in the best interest of the project, or same language translation must occur. For example, if someone wants to use Visual Basic .NET with a WYSIWYG GUI tool (such as Tilcon or Altia), they will find that it will not be supported and, therefore, be forced to reconsider going with C/C++. The implication of code generated from UML is that it forces a restriction to whatever language the UML tool generates and that this language must be compatible with the API supported by the WYSIWYG tool.

Conclusion
The example presented in this article shows that using UML for the back-end, run-time engine development and a WYSIWYG GUI builder tool for the front-end graphics development can result in overall gains in productivity and ease of prototyping. The event-driven nature of real-time UML facilitates straightforward integration with an event-driven GUI; to some extent, both solutions are platform-neutral. The example demonstrates the ease of these concepts and the integration and simplification of the problem.

One of the key benefits of this approach is that non-programmers can utilize the WYSIWYG GUI design tool to create the GUI. Requirements can be expressed in UML, and these descriptions can be used in the design and implementation of the system. As a result, the development of a complex system is simplified,
in turn minimizing risk, reducing development costs, and shortening schedules.

References

Notes
1. In the Rational Rose RealTime tool, active classes are called capsules, and the associated collaboration diagrams are called structure diagrams.
2. The Tilcon Interface Builder (see <www.tilcon.com/products/interface-development-suite/tilcon-interface-builder> to learn more) is not to be confused with the Interface Builder application for the Apple Mac OS X.

About the Authors
Ilya Lipkin is a project engineer for the 677th AESG/EN Global Hawk Simulations at Wright Patterson AFB. His current research interests include artificial intelligence, human knowledge capture and analysis, neural networks, fuzzy logic, user interface design, software engineering, UML, supply chain control, and customer relations management. Lipkin has a bachelor's degree in computer engineering, an MBA in operations management, and a master's degree in computer engineering. He is currently a doctoral student at the University of Toledo's College of Business Administration.

Martin Guldahl is an electrical engineer for the Common Aircraft Portable Reprogramming Equipment program, part of the 520th Software Maintenance Squadron, 309th Software Maintenance Group at Hill Air Force Base, Utah. He has more than 15 years of experience in a variety of industry and government positions. Guldahl has a bachelor's degree in electrical engineering and has taken graduate level computer science coursework from the University of Utah. His areas of interest include UML, C++, Verilog, and Perl.