Motion Control Adopts Advanced Software Development Process For Production of Elevator Control Units Using I-Logix' Rhapsody

We step inside, push a button and travel to the correct floor. Most of us use elevators daily without a thought. We never consider the technology behind their operation. But, each push of the button sends the elevator controller processing millions of lines of code. Preprogrammed software tells the elevator in what order to stop at floors, when to open or close the door and if there is a safety-critical issue. The elevator control and monitoring systems are essential in the smooth and safe operation of each elevator.

Motion Control Engineering Inc., with approximately $80 million in sales per year, is the largest independent manufacturer of elevator control and monitoring systems in the world. Their goal is to build high quality, non-proprietary "universally maintainable" equipment. Motion Control Engineering (MCE) is continually working to bring together the right people and technology to improve elevator performance while ever-simplifying installation, maintenance and operation.

In early 2000, MCE began development of their next generation elevator controller. Key to this development was the adoption of a more advanced software development process, one in which they could continually add advanced features while developing and maintaining a set process. Their decision was to focus on a Unified Modeling Language™ (UML™)-based object-oriented approach. Once they decided to use the VxWorks RTOS, the next step was to select a software development tool. After evaluating several tools, including Project Technology Inc. BridgePoint, Rational Rose and RTI ControlShell, MCE selected the Rhapsody® in C++ UML-based visual application development platform.

By April of 2000 MCE's software engineers began the redesign of their elevator computer control logic. This was Motion Controls first project using object-oriented design, the UML and a visual application development platform. They brought in an I-Logix expert to provide in-house training on all three.

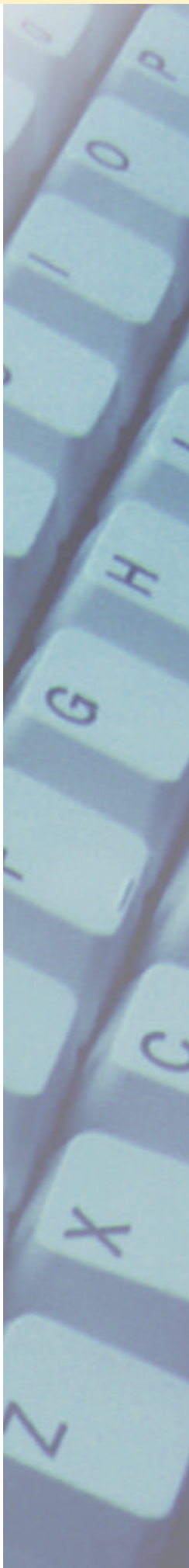
"Our software developers were proficient in manually writing code in C and Assembly, but no one had experience with a visual application development platform or the Unified Modeling Language," said Gavin Arthurs of MCE. "I-Logix training allowed us to first learn the fundamentals of object-oriented design, then learn the "ins and outs" of the Rhapsody tool, while tying them together. This on-site training was beneficial to ramping up our team."

The first step for the development of the elevator control was to define the basic requirements. MCE software developers captured their requirements using Rhapsody Use Cases during the initial development stages. They then used UML Sequence Diagrams during the architectural design phase to develop core classes of the model and the relationship between those classes. Once the requirements were defined, MCE moved into the use of UML Statecharts for developing the control logic. During elevator use, control logic takes into consideration user input (push of a button) and the current state of the elevator (travelling direction), makes a decision based upon both, and then triggers an action (such as stop or change direction). Through Rhapsody, "current user inputs" were modeled as events in UML Statecharts, and the "current state" as states, including concurrent states. From the Statechart, code was automatically generated and could later be used to run animation and debug the design. During animation, the developers debugged their design at the model level. They were able to inject an event and view the resulting effect on the Statechart graphically. This allowed them to watch how an event triggers an action to ensure the control logic was behaving as desired.

"Developing our application in a visual manner is key because an elevator is primarily built using module based development," said Arthurs. "The Statecharts, and ability to animate Statecharts, has been very useful in developing the logic that runs the elevator, especially when people collaborate. It is much easier to understand what is going on when you see Statecharts running rather than looking at code."

A major benefit MCE experienced with Rhapsody was the ability to have multiple software engineers working independently, without fear of losing data. During the initial stages each engineer was able to focus on their assigned area and then merge the model together when it was critical to synchronize the project. The pieces of the design were combined initially using the "Add to Model" feature in Rhapsody. Basically, developers would work independently for a time, would merge the models periodically, and repeat. This method worked well when there was a lot of parallel development going on.

As the model evolved and the official release was



approaching, the engineers began using an official Configuration Management tool in conjunction with Rhapsody. Motion Control was then able to version control their design. As a team, they were able to bring together all of the elements, get the project running, validate that it was operating as defined, and move onto the next step in an iterative fashion. Rhapsody greatly facilitated collaboration in these ways. Also, Rhapsody's hierarchical repository eased the pain associated with storing elements hierarchically in the file system, and enabled potential component reuse for the future.

"The package elements and the component / configuration structures are useful in managing the "host" and "target" builds within the same model," said Arthurs.

Rhapsody's real-time framework has built in features such as communication between classes and event structures. As mentioned previously, before Rhapsody was introduced, the team at Motion Control wrote all of their code manually in Assembly or C. This code targeted five or six processors, many without the benefit of an operating system.

"Consequently, exchanging data between processes was done at a very low level and driven heavily by the hardware. Most of our problems were in data latency and synchronization," said Arthurs. Motion Control was able to benefit from many of the Rhapsody framework classes, such as OMReactive, OMEvent, etc., to accomplish common OS functions, including asynchronous communications between objects. Rhapsody was able to help the team raise the level of design abstraction as well as save time and effort for the development team.

Throughout the entire development process, MCE was able to automatically generate code for their application. The code was used to test the application at various stages to ensure that specifications were met. They were also able to debug their applications as the code was running on their PC, and this was automatically reflected back in the model, then they were able to immediately deploy it on VxWorks. Design-level debugging was conducted on both the host and the target. This was a huge benefit, as it allowed the engineers to concentrate on their design instead of "housekeeping" code. The Rhapsody framework, specifically designed for embedded real-time systems, ensured that the real-time constructs such as timers, semaphores and threads were automatically generated for the specific operating system being targeted, whether that be Windows or VxWorks, MCE chose to work mostly at the design level, and used all of the Rhapsody generated code for their application.

"The greatest benefit we have experienced using Rhapsody is the ability to run the logic on the host, targeted for our PC, then test it and debug it logically. Previously we could not do that. Most of our testing had to be done directly on the hardware because we could not emulate it on the PC. We were very tied to actually having to keep the hardware with us. Rhapsody has allowed us, from a logical point of view, to debug a lot earlier without any hardware," said Arthurs. "This has been huge, we now can focus on getting the application to connect to the hardware so that the information coming up is correct. The logic is already debugged, and that has been very helpful. It is an order of magnitude in time and cost savings."

Through the use of Rhapsody, MCE was able to develop their elevator control using a visual, UML-based application development platform. They benefited from the ability to visualize a logic-heavy application during the development phase, improve communication and collaboration among their engineers, ability to produce higher quality code which allowed them to debug and test their application up front, and the ability to deploy that code on both the host and target platforms. MCE is currently shipping to beta sites. With Rhapsody, they were able to take what was developed over 15 years, put it into one product and deploy on a new platform in less than two years.

Motion Control is committed to continued development with Rhapsody. "When we selected Rhapsody, we selected the way we planned to develop software for the foreseeable future," said Arthurs.

I-Logix

I-Logix Inc.
3 Riverside Drive
Andover, MA 01810
Tel: 978-682-2100
Toll Free: 888-845-6449
Fax: 978-682-5995
E-mail: info@ilogix.com
http//www.ilogix.com

European Headquarters
I-Logix UK Ltd.
1 Cornbrash Park
Bumpers Way
Chippenham
Wiltshire SN14 6RA
England
Tel: +44 1249 467-600
Fax: +44 1249 467-610
E-mail: info_euro@ilogix.com