Managing Software Components with Teamcenter Enterprise and UML-based Model-Driven Development

Jim McElroy
I-Logix, Inc.
Jimm@ilogix.com
978-645-3026
Our Goal

Present a practical cohesive methodology and infrastructure for both developing and managing “Software Parts” with the mechatronic systems they control
Primary Issues in Today’s Global Software Development Market

- Complexity
- Communication and Collaboration
- Productivity
- Time-to-market
- Quality
- Safety, Security, Reliability
Strategies and Technologies to Solve these Global Issues

- Model-Driven Development
  - Abstraction
  - Automation (Code generation, testing, documentation)
  - Reuse (through models and CBD)
  - Communication and collaboration

- Software Configuration Management

- Product Life-cycle Management
  - Functional Decomposition
  - Traceability of Requirements, into and throughout the Physical Design
  - Linkage and Navigation to/from the BOM
A design and development methodology which uses models as the basis for analyzing requirements, developing the design, implementing, testing, and deploying the application
Modeling Requires a Language

- **UML 2.0 – Standards Compliance**
  - Architecture
  - Behavior
  - Collaboration

- **SysML**
  - A specialized profile of UML 2.0 for systems engineering. Both reuses and extends UML 2.0.

- **DoDADF**
  - An architectural framework providing a common communication mechanism for operational, system, and technical architectural views.
UML 2.0 Diagrams
MDD by Example
Handset Protocol Stack

Place Call

Supplementary Service

Receive Call

Provide Status

Represents Handset User Interface

The mobile shall be able to receive short messages while the mobile is registered.

System Network access through air interface

Network
Subsystem view showing realization of flows between blocks through links and ports.

Click to show collaboration.
MDD – Architecture Level 2
MDD – Behavior Example
MDD – Collaboration Example

This is a Subsystem level scenario which describes how the subsystem interact with a successful request to place a call.
Benefits of Modeling

- Modeling gives us the ability to visualize the system clearly
- Simplify the problem through abstraction
-Executable models further enhance the visualization, understanding, clarification of intended functionality and behavior
Key Differentiators for MDD

- **Modeling (UML 2.0 PLUS)**
  - Benefit: Designing systems at a higher level of abstraction in order to easily deal with complexity

- **Code Generation**
  - Benefit: Get to the final product quicker. Enables designers to work at a higher level of abstraction to deal with system complexity

- **Model/Code Associativity**
  - Benefit: Freedom to work at the model level or source code level, and ensure views of the system are always synchronized.

- **Real Time Framework**
  - Benefit: Allows you to create a deployable application, with the generated code, onto a real time operating system

- **Reverse Engineering**
  - Benefit: Allows you to reuse your IP, as well as coexist with ongoing hand coding activities

- **Design for Testability**
  - Benefit: Automate the testing process through using the *design requirements* to validate and to completely cover all system scenarios
Crossing the PLM Chasm

- Historically, versioning and configuration management have been major bottlenecks to system development
- Hardware and physical elements follow their own life-cycles separate from the software
- Need to connect the software development life-cycle and the artifacts that are produced with the actual mechatronic systems they control
- Save development time and energy while at the same time improving system quality
Integrating MDD with PLM

- Software Component Development and Management with MDD
- Elevate Components to “Software Parts”
- Managing and Controlling “Software Parts” within the PLM System and the Vault
A component is a nontrivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interfaces.
Component Life-cycle and Management

<table>
<thead>
<tr>
<th>Required Capabilities</th>
<th>Utility Capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>· Creation</td>
<td>· Storing</td>
</tr>
<tr>
<td>· Validate</td>
<td>· Finding</td>
</tr>
<tr>
<td>· Modification</td>
<td>· Downloading</td>
</tr>
<tr>
<td>· Documenting</td>
<td>· Testing</td>
</tr>
<tr>
<td>· Publishing</td>
<td>· Submit and monitor</td>
</tr>
<tr>
<td></td>
<td>· Defect and</td>
</tr>
<tr>
<td></td>
<td>· Enhancement</td>
</tr>
<tr>
<td></td>
<td>· Requests</td>
</tr>
<tr>
<td></td>
<td>· Metrics collection</td>
</tr>
</tbody>
</table>
The Software Part

Software Part

Meta Data
Author, Date of Creation, Type
Links to Defects/Enhs

PLM Links

Design Documentation
UML Design Files
Links to Reqs Files

Test Documentation
Test Plans and Procedures
Test Results

Component
Include Files
Source Files

Binary Files
Libraries
Executables
Teamcenter - Management and Distribution of the Component

- Storing the component for reuse
- Determining system-wide dependencies
- Publishing the component
- Finding components in the gallery
- Keeping metrics and data pertaining to component
- Software Part becomes a part within the Teamcenter Assembly
  - Single-source configuration control
MDD, CM, and Vault Workflow

MDD

Artifacts
Models
Code
Documentation

Artifacts

Milestone – Software Part
Release Mgr - As required

Teamcenter

Software Config Mgt

Developer - Daily/Weekly
Integrated Approach

- MDD will enable the creation, validation, and executable visualization of Software Components
- Teamcenter will provide the infrastructure for elevating Software Components to Software Parts and then managing and controlling those parts within the assembly
- Synchronizes and couples the “Software Parts” with the mechatronic parts they control improving workflow, productivity, and quality
For a cellular phone this might be a collection of components such as calendar, voice recognition, gaming package etc.

The voice recognition package may include a design package, specification package etc.

And for deployment purpose we may keep a specific configuration assembly.

The voice recognition component may have a test harness.

The voice recognition package may be built for various OS and frameworks.
Pulling all the Value Together

- Synchronized MDD, SCM, and PLM
- Single source for configuration control ensures seamless workflow and productivity gains
- Made software developers more productive, release managers more effective and efficient
- Strengthened our joint position in developing highly scalable software solutions