Model Driven Development

- Reuse of IP
- Full Application Code Generation
- Concurrent Design Collaboration
- Model Driven Development
- Design For Testability (DFT)
- Real-Time Framework
- Dynamic Model-Code Associativity
Rhapsody leverages all structural and behavioral model views to produce an executable application
- Structure models
- State charts: event driven behavior
- Activity graphs: algorithms and process flows
- Components and artifacts

Rhapsody generates very clean, readable code, easily debugged through any commercial IDE
- Integrated “white-box” Code (C, C++, Java, Ada, IDL) generation
- High productivity; low cost of maintenance

Rhapsody generates all application construction artifacts to provide an integrated build environment

Comprehensive code generation technologies
- OO based and / or functional based
- Stereotype based
- Rules based
I-Logix offers full code generation for specific needs/environments, such as:
- GNAT
- Greenhills
- Aonix

End-Users can generate their own variants of code:
- Offers Full control over the generated code
- Addresses the various code standards and needs of Mil/Aero
- End user experts can modify the rules to address particular needs
  - e.g. Code indentation, file naming conventions, case vs. if statements
- New rules can be used by all team members
  - As simple as a selection from a pull-down menu
State of the art code generation technology

- “Hard-coded” code generation
- Template based code generation
- Stereotype based code generation
- Rules based code generation

Execution environment includes:

- On the fly code generation from a UML model
- WYSIWYG editor to ease rules editing
- Rules, macros and scripts to control dynamic content
- Rules tracer to analyze and debug rules execution
- Built-in rules diff&merge

Low flexibility

High flexibility
You can’t test what you can’t execute!

- Rapid execution at the design level on host or even target
- The best way to avoid having bugs is to not introduce them to the system
DFT: Test Conductor™

Sequence Diagrams

Stimulate & Monitor the Model

Test Results

Test Configuration

Finding & Correcting Errors
**Dynamic Model Code Associativity**

*Rhapsody works the way you do*

- Design, Code and Documentation are always kept in sync
- Freedom to work at code level or design level
- Change one view, the others *change automatically*
- Critical for real-time embedded software development
Rhapsody provides an executable real-time framework

- Most applications are over 50% “housekeeping code” which is redeveloped every time you create a system
- A framework is a partially completed application
  - you customize and specialize for your application
- A real-time framework is an
  - integrated set of design patterns
  - optimized for embedded applications
Concurrent Design Collaboration

- Small and Large Scale Development
- Tight integration with configuration management
- Partial loading
- Visual Differencing and Merging
Reuse of IP: Import Legacy Ada code

Import Ada83/95 legacy code into Rhapsody
Extending Rhapsody’s existing RM interface capabilities with the introduction of the Rhapsody® Gateway (RG):

- Allows Rhapsody users to seamlessly work with Rhapsody and classic 3rd Party Requirements Management tools and other common requirements authoring tools.
- Enables traceability analysis and reporting from within Rhapsody.
- Complete and constant synchronization between UML and Req Tool Views

- Teelogic DOORS
- UGS Slate
- Microsoft Word/ Excel
- XML
- Adobe PDF Acrobat Files
- Generic Text files
- Code
- IBM Requisite Pro
- UGS TeamCenter Requirements (Future)
- Serena Software RTM (Future)
4. Package : RequirementsPkg

4.1 Object Model Diagram : Requirements Taxonomy

5. Gateway Traceability Matrix

5.1 UML Model covers RequirementsPkg

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Owner</th>
<th>Status</th>
<th>Andhem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req1</td>
<td>The wheat needs to be processed</td>
<td>Req1</td>
<td>Complete</td>
<td>Req1</td>
</tr>
<tr>
<td>Req2</td>
<td>The machine needs to be cleaned</td>
<td>Req2</td>
<td>Complete</td>
<td>Req2</td>
</tr>
</tbody>
</table>

5.2 TestConductor Tests covers UML Model

<table>
<thead>
<tr>
<th>Test</th>
<th>Coverage</th>
<th>Start Time</th>
<th>End Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test1</td>
<td>100%</td>
<td>01/01/2022</td>
<td>01/02/2022</td>
</tr>
<tr>
<td>Test2</td>
<td>90%</td>
<td>01/03/2022</td>
<td>01/04/2022</td>
</tr>
<tr>
<td>Test3</td>
<td>80%</td>
<td>01/05/2022</td>
<td>01/06/2022</td>
</tr>
</tbody>
</table>