Harmony/ESW
An Agile Real-time Development Process

Jeff Vodov
Objectives

• Provide a high-level overview of Harmony
  – Challenges of SE and SW Development
  – Harmony Best Practices
  – Rational / Telelogic Process Roadmap

• Provide an overview of Harmony/ESW
  – Understand the core principles, roles, work products, tasks, best practices and lifecycle of Harmony/ESW
  – Deployment of Harmony/ESW
    • Process Training
    • Interactive Process Handbook
The Importance of Process & Best Practices

“…The quality of a product is largely determined by the quality of the process that is used to develop and maintain it …”

*based on*: Shewhart, Juran, Deming and Humphrey
Combination of Technology and Process Results in 10x Business Value

![Bar chart showing Business Value](chart)

Source: London School of Economics – McKinsey Survey
Challenge of Leveraging Best Practices

- Different sources in different formats
- Difficult to customize and maintain
- Large volumes of information
- Difficult for practitioners to find relevant information
Telelogic Harmony

• Library of re-useable Best Practices
  - Standard meta-model (OMG SPEM 2.0)
  - Standard tool (EPF Composer/RMC)

• Pre-configured processes include:
  - Harmony/ITSW
    • IT Software
  - Harmony/ESW
    • Embedded Software
  - Harmony/SE
    • Systems Engineering
Rational / Telelogic Process Roadmap

<table>
<thead>
<tr>
<th>2Q08</th>
<th>3Q08</th>
<th>4Q08</th>
<th>1Q09</th>
<th>2Q09</th>
<th>3Q09</th>
<th>4Q09</th>
<th>1Q10</th>
</tr>
</thead>
<tbody>
<tr>
<td>APR</td>
<td>MAY</td>
<td>JUN</td>
<td>JUL</td>
<td>FEB</td>
<td>MAR</td>
<td>MAY</td>
<td>JUN</td>
</tr>
<tr>
<td>AUG</td>
<td>SEP</td>
<td>OCT</td>
<td>NOV</td>
<td>DEC</td>
<td>JAN</td>
<td>FEB</td>
<td>MAR</td>
</tr>
</tbody>
</table>

Rational

Telelogic Harmony

ITSW

- ESW Content
- SE Content
- Change Control Practice

Process Guidance for Telelogic Tools

- Requirements Definition
- Requirements Management
- Continuous Integration Practices

RMC 7.5 Practices and Measurements

RMC 7.5.0.1 APARs

RMC 7.5.0.2 APARs

RMC 7.5.0.3 APARs

RMC 8.0 Jazz Based (Tentative)
The Telelogic Process and Product Portfolio

Telelogic Harmony

- Telelogic System Architect
  - EA and Business Process
- Telelogic Focal Point
  - Product Portfolio Management
- Telelogic Dashboard
  - Measurement
- Telelogic Integrations
  - Production

Telelogic DOORS
- Requirements

Telelogic Tau & Rhapsody
- Analysis & Design

Telelogic Change & Synergy
- Configuration & Change Mgmt

Telelogic Tau, Rhapsody & Eclipse & .Net Integrations
- Implementation

Telelogic Tester, Test Conductor and Test Generator
- Test

Telelogic Synergy
- Release Mgmt
Welcome to Harmony/ESW

Harmony/ESW is a member of the The Telelogic Harmony Library of Best Practices specifically for Embedded Software development.

Harmony/ESW connects the dots between people, process, tools and best practices to provide a complete solution for embedded software development teams.
What is Harmony/ESW?

• The Harmony/ESW Process is generally applicable to software and systems development, but is optimized for the development of software-intensive real-time and embedded systems.

• Harmony/ESW is directly derived from the Rapid Object-oriented Process for Embedded Systems (ROPES), authored by Dr. Bruce Powel Douglass.
What is Harmony/ESW?

• Harmony/ESW is:
  – **Agile**: embodies the use of key concepts and guiding principles of agile development
  – **Efficient**: emphasis is on tasks and work products that add significant to the development
  – **High quality**: stresses on continual validation of correctness and completeness throughout development
  – **Requirements-driven**: development tasks concentrate on identifying and meeting stakeholders needs
  – **Architecture-centric**: strong concepts of key architectural views
  – **Scalable**: process is defined so that aspects are included only when needed
Core Principles

• The core principles of Harmony/ESW capture the overall intent and philosophy behind the Harmony Process

• Your Primary Goal is to Develop Software or Systems
• Principle of Continual Feedback
• Key Views of Architecture
• Plan, Track, Adapt
• Leading Cause of Project Failure is ... Ignoring Risk
• Modeling is next to godliness
Primary Goal: Some key practices

• 80% of developer’s time should be spent on activities supporting the development of high-quality software
  – Even if you’re signing off documents faster than planned, you’re not making real progress if the software doesn’t work

• Daily activities should focus on:
  – The nanocycle of development performs a rapid model-compile-debug-unit test workflow every few minutes
  – Source code is generated and tested every few minutes
  – Continuous integration builds the system at least daily to ensure compatibility of components and subsystems

• This is not to say writing documents and other tasks are inappropriate, just that they are not your primary goal
Continuous Feedback: Some key practices

“It ain’t right if it don’t run” – Law of Douglass

“Optimism is the enemy of realism” – Law of Douglass

“Optimism is a disease – feedback is the cure” – Kent Beck

- As we develop software over days, weeks, and months, we need assurance that we’re doing the right thing:
  - Constant debug and unit test every few minutes with the Harmony/ESW nanocycle
  - Test-driven development produces unit tests simultaneously with models and code
  - Constant elaboration and execution of the unit test suite
  - Continuous integration of components and subsystems removes integration problems early
  - Schedules are tracked frequently for deviations from plan with Bruce’s Evaluation and Review Technique (BERT) workflow
  - Software estimation accuracy is improved with the Effect Review for Nanocycle Iteration Estimation (ERNIE) workflow
  - Project issues and roadblocks are identified frequently and addressed at least every microcycle iteration (4-6 weeks)
Key Views of Architecture: Some key practices

• Harmony/ESW identifies 5 key views of Architecture (at right)
• Architecture focuses on overall design optimization concerns in each of the architectural views
• In each view, an optimization workflow is followed:
  – Identify the relevant design optimization criteria
  – Rank criteria in order of criticality
  – Identify design patterns and/or technologies that provide the important criteria at the expense of the lesser
  – Apply design patterns
  – Test
Plan, Track, and Adapt: Some key practices

• Dynamic planning ensures that plans always reflect “truth on the ground”
• Tracking is performed against goals (e.g. working functionality realized/validated) not against implementation (e.g. lines of code written)
• Plans are reviewed and updated at least once per microcycle in the Increment Review (Party) phase of the microcycle
• Scheduling is done via the BERT workflow
• 3 schedules are produced
  – Working schedule (based on weighted $e_{20}$, $e_{50}$, and $e_{80}$)
  – Customer schedule (based on weighted $e_{80}$)
  – Goal schedule (based on weighted $e_{20}$)
Risk Management: Some key practices

- Risks are actively identified, ranked, and addressed with scheduled risk mitigation activities
- Risks are tracked in the Risk Management Plan (Excel template is provided)
- Risks are tracked daily
- Risks are reevaluated at least once per microcycle in the Increment Review (Party) phase
Modeling: Some key practices

• High-quality semantic rich models are emphasized
• Models are continuously (and automatically) synchronized with the source code (“dynamic model-code associativity”)
• Model organization optimizes team collaboration while minimizing coupling between teams
• Constructed models
  – Analysis models (aka Platform Independent Models) produce correctly-executing models and source code of the necessary functionality
  – Design models (aka Platform Specific Models) products optimized models and source code, optimizing the weighted set of design criteria
• Each diagram is has a singular concept (mission) and includes all elements relevant to that mission and no elements not so relevant
Harmony/ESW is Agile and Model-Driven

• Harmony/ESW incorporates a number of agile practices…
  – Test-First Design
  – Continuous Integration
  – Agile Estimation
  – Microcycle planning and assessment

• …within the context of an iterative, incremental lifecycle (microcycle).
Governance Model – Balancing Agility and Discipline

• Harmony/ESW incorporates a three-tiered governance model to plan, execute, and monitor progress.

• These tiers correspond to personal, team and stakeholder concerns and each operates at a different time scale and level of detail.
Harmony/ESW Project Lifecycle

• Harmony/ESW uses an iterative, incremental lifecycle.
Prespiral Planning

- Prespiral Planning is meant to set up the information and teams for the project
Develop Stakeholder Requirements

• This activity is meant to define the vision and scope of the project and provide a set of customer-level requirements
Microcycle Lifecycle

- The software development takes place with a series of iterations, known as microcycles.
- Each microcycle produces a fully validated version of the model, source code and constructed system (“prototype”) that incorporates a subset of the use cases.
- Each prototype adds on to the functionality delivered in previous microcycle prototypes.
Prototype Definition

- Defines the mission and scope for the next microcycle including
  - Use cases realized
  - Architectural intent realized
  - Risk mitigation activities performed
  - Defects repaired
  - Target platforms supported
# Harmony/ESW Lifecycle WBS

<table>
<thead>
<tr>
<th>Breakdown Element</th>
<th>Steps</th>
<th>Index</th>
<th>Predecessors</th>
<th>Model Info</th>
<th>Type</th>
<th>Planned</th>
<th>Repeatable</th>
<th>Multiple Occurrences</th>
<th>Ongoing</th>
<th>Event Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define and deploy the development environment</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>Activity</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tailor Process</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>Task</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install Development Tools</td>
<td>3</td>
<td>2</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configure Development Tools</td>
<td>4</td>
<td>3</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initialize Development Tools</td>
<td>5</td>
<td>4</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Launch Development Environment</td>
<td>6</td>
<td>5</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preliminary Planning</td>
<td>7</td>
<td></td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Schedule</td>
<td>8</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Team Structure</td>
<td>9</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plan For Reuse</td>
<td>10</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plan For Risk Reduction</td>
<td>11</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify Logical Architecture</td>
<td>12</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perform Initial Safety and Reliability Analysis</td>
<td>13</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop Stakeholder Requirements</td>
<td>14</td>
<td></td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define Vision</td>
<td>15</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Find and Outline Stakeholder Requirements</td>
<td>16</td>
<td>15</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detail Stakeholder Requirements</td>
<td>17</td>
<td>16</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review Stakeholder Requirements</td>
<td>18</td>
<td>17</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control Project</td>
<td>19</td>
<td>7,14,1</td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update Risks</td>
<td>20</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update Schedule</td>
<td>21</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update Hazard Analysis</td>
<td>22</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refine and deploy the development environment</td>
<td>23</td>
<td></td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage Change</td>
<td>29</td>
<td>7,14,1</td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request Change</td>
<td>30</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review Change Request</td>
<td>31</td>
<td>30</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assign Change Request</td>
<td>32</td>
<td>31</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resolve Change Request</td>
<td>33</td>
<td>32</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify Change Request</td>
<td>34</td>
<td>33</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Close Change Request</td>
<td>35</td>
<td>34</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microcycle</td>
<td>36</td>
<td>7,14,1</td>
<td>Iteration</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype Definition</td>
<td>37</td>
<td></td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuous Integration</td>
<td>46</td>
<td>37</td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prepare for Validation Testing</td>
<td>50</td>
<td>37</td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object Analysis</td>
<td>54</td>
<td>37</td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architectural Design</td>
<td>63</td>
<td>54</td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mechanistic Design</td>
<td>74</td>
<td>63</td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detailed Design</td>
<td>80</td>
<td>74</td>
<td>Activity</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perform Model Review</td>
<td>87</td>
<td>80</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td>88</td>
<td></td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Increment Review (&quot;Party Phase&quot;)</td>
<td>91</td>
<td>88</td>
<td>Task</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Harmony Disciplines

• A discipline is a collection of tasks that are related to a major "area of concern" within the overall project.
• Within the lifecycle, tasks are performed concurrently across several disciplines.
• Separating tasks into distinct disciplines is simply an effective way to organize content that makes comprehension easier.
• Harmony/ESW defines the following Disciplines:
Project Management Discipline

**Discipline: Project Management Discipline**

The Management Discipline contains tasks focusing on the planning, execution, tracking, and evaluation of projects.

### Relationships

<table>
<thead>
<tr>
<th>Reference Workflows</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Project</td>
<td>Plan Iteration</td>
</tr>
<tr>
<td>Manage Change</td>
<td>Create Schedule</td>
</tr>
<tr>
<td>Prespiral Planning</td>
<td>Update Schedule</td>
</tr>
<tr>
<td></td>
<td>Create Team Structure</td>
</tr>
<tr>
<td></td>
<td>Plan For Reuse</td>
</tr>
<tr>
<td></td>
<td>Plan For Risk Reduction</td>
</tr>
<tr>
<td></td>
<td>Increment Review (&quot;Party Phase&quot;)</td>
</tr>
<tr>
<td></td>
<td>Update Hazard Analysis</td>
</tr>
<tr>
<td></td>
<td>Update Risks</td>
</tr>
</tbody>
</table>

### Main Description

The purpose of this discipline is to:

- Plan project work
- Track project work
- Allocate work tasks to personnel resources
- Create plans such as
  - Schedules
  - Risk Management Plans
  - Reuse Plans
- Assessing work tasks and work products in the context of the plans
# Harmony/ESW Roles

## Role Set: Roles

These are the roles defined in the current process.

## Relationships

<table>
<thead>
<tr>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any Role</td>
</tr>
<tr>
<td>Configuration Manager</td>
</tr>
<tr>
<td>Human Factors Engineer</td>
</tr>
<tr>
<td>IT Administrator</td>
</tr>
<tr>
<td>Moderator</td>
</tr>
<tr>
<td>Process Engineer</td>
</tr>
<tr>
<td>Project Manager</td>
</tr>
<tr>
<td>Quality Assurance Role</td>
</tr>
<tr>
<td>Reliability Czar</td>
</tr>
<tr>
<td>Request Implementor</td>
</tr>
<tr>
<td>Request Reviewer</td>
</tr>
<tr>
<td>Request Verifier</td>
</tr>
<tr>
<td>Requirements Engineer</td>
</tr>
<tr>
<td>Reviewer</td>
</tr>
<tr>
<td>Safety Czar</td>
</tr>
<tr>
<td>Scribe</td>
</tr>
<tr>
<td>Software Architect</td>
</tr>
<tr>
<td>Software Coder</td>
</tr>
<tr>
<td>Software Modeler</td>
</tr>
<tr>
<td>Stakeholder</td>
</tr>
<tr>
<td>Subject Matter Expert</td>
</tr>
<tr>
<td>System Engineer</td>
</tr>
<tr>
<td>Tester</td>
</tr>
<tr>
<td>Toolsmith</td>
</tr>
<tr>
<td>Use Case Analyst</td>
</tr>
</tbody>
</table>
Software Modeler Role

Role: Software Modeler

The Software Modeler uses modeling techniques to represent abstractions of the system under development for a specific purpose.

Role Sets: Roles

Relationships

Additionally Performs
- Detail Use Case
- Manage Integration Tests
- Validate Architecture
- Validate Optimized Class
- Validate Collaborative Model
- Increment Review ("Party Phase")

Modifies
- Source Code
- Prototype
- Test Results
- Test Plan
- Test Suite
- Traceability Record
- Work Items List
- Platform Independent Model
- Scenario
- Analysis Collaboration
- Model
- Baseline
- Platform Specific Model
- Change Request
- Change Set
A Typical Task Description

- Tasks typically have an associated concept, guideline and checklist.
- If one needs to perform a task
  - one reads the concept to understand the context,
  - reads the steps to determine what needs to be done,
  - reads the guideline to determine how to do it,
  - then reads the checklist to validate completion.
A Typical Artifact Description

- Typically artifacts have associated templates and checklists.
- The template provides additional guidance on completing the artifact and
- The checklist helps check the quality of the resulting artifact.
Deployment of Harmony/ESW services

• Interactive EPF-based Handbook
  – Project template that guides workers from concept to delivered system

• Process Training
  – The Harmony/ESW™ Process
    • 3-day hands-on course developed by Dr. Bruce Douglass
  – Introduction to EPF for Harmony/ESW
    • 1-day Introduction to Harmony and EPF
    • Concentrates on how Harmony/ESW can be customized
  – Introduction to EPF
  – Custom training based on client needs during project deployment
Harmony/ESW Website

Welcome to Harmony/ESW

Harmony/ESW is a member of The Telelogic Harmony Library of Best Practices specifically for Embedded Software development.

Harmony/ESW connects the dots between people, process, tools and best practices to provide a complete solution for embedded software development teams.

The Harmony/ESW Process is generally applicable to software and systems development, but is optimized for the development of software-intensive real-time and embedded systems. Harmony/ESW is directly derived from the Rapid Object-oriented Process for Embedded Systems (ROPES), authored by Dr. Bruce Powel Douglass (DOU99, DOU02, DOU04).

Harmony/ESW is:

- Agile: embodies the use of key concepts and guiding principles of agile development
- Efficient: emphasis is on tests and work products that add significant value to the development
- High quality: stresses on continual validation of correctness and completeness throughout development
- Requirements-driven: development tasks concentrate on identifying and meeting stakeholders needs
- Architecture-centric: strong concepts of key architectural views
- Scalable: process is defined so that aspects are included only when needed

In addition, Harmony supports specific technologies and needs common to real-time and embedded systems, including:

- Timeliness, schedulability, and performance
- Low-level (i.e. device level) development
- Early risk reduction
Summary

• Harmony/ESW is a coherent set of best practices meant to effectively realize a set of core principles

• Harmony/ESW is
  – Requirements-driven
  – Architecture-centric
  – Optimized for
    • Systems and Software Engineering projects
    • Real-time
    • Embedded
    • Hardware-software co-design
    • Scalability from small to large systems projects
Thanks for your attention & any questions?