EAM: Ecosystemability Assessment Method

Imed Hammouda and Eric Knauss
{imed.hammouda, eric.knauss}@cse.gu.se

Participating companies: Axis and VCC
Motivation: Ecosystem Questions

- What technical/organizational/business aspects can be assessed in a software ecosystem?
- How to align technical/organizational/business decisions with ecosystem business strategies?
- How to engage ecosystem actors in the assessment and decision making processes?
- How to plan and perform the assessment process?
- What criteria should be used in the assessment process?
- How to use the assessment results?
EAM: Conceptual Flow

- **Business drivers**
- **Ecosystem characteristics**
- **Ecosystem Scope**
- **Ecosystem strategy**
- **Ecosystem Element**
- **Relevant Actors & Relationships**
- **Fitness dimensions**
- **Ecosystem Personas**

**Prioritize**

**Select**

**Assess**

- **Tradeoffs**
- **Sensitivity points**
- **Non-risk**
- **Risks**

**Impacts**

**Distilled into**

- **Business scenarios (Goals)**
- **Risk themes**
Case I: Axis Communications

API-DESIGN FOR THE ECOSYSTEM
EAM: Conceptual Flow - APIs

- Business drivers
- Ecosystem characteristics
- Ecosystem scope

- Ecosystem strategy
- Platform
- Relevant actors & relationships

- Application features
- Technical & cognitive dimensions
- Application developer personas

- API assessment
  - Decide if movement occurred
  - Detect tampering
  - Count object(s) crossing a line
  - Take a snapshots at specified frequency
  - ...
  - Abstraction Level
  - Work step unit
  - API latency
  - Security
  - ...

- Risk themes
- Sensitivity points
- Non-risk
- Risks

Impacts distilled into:

- Client application developer
- Analytics application developer
- Camera feature external developer
- Camera feature internal developer
- ...

- Tradeoffs
- Sensitivity points
- Non-risk
- Risks
**API Fitness Dimensions**

**Abstraction Level** fitness dimension describes the levels of abstraction exposed by the API. The **Abstraction Level** ranges from **low** (many API components are needed for implementing a particular developer goal) to **high** (only one API component is needed for implementing a particular developer goal).
Fitness Dimensions – Priority (API Architects)

- Error Checking and Responsiveness - Priority
- Compatibility - Priority
- Consistency - Priority
- Abstraction Level - Priority
- Model Complexity - Priority
- Atomic Setting - Priority
- Security - Priority
- Role Expressiveness - Priority
- Learning Style - Priority
- Working Framework - Priority
- Domain Correspondence - Priority
- API Latency - Priority
- Functional Completeness - Priority
- Easiness to use - Priority
- Testability - Priority
- API Elaboration - Priority
- Premature Commitment - Priority
- Penetrability - Priority
- Work Step Unit - Priority
- Progressive Evaluation - Priority
- API Evolvability - Priority
- Synchrony - Priority
- Viscosity - Priority
Fitness Dimensions – Priority (API Architects vs Users)

- Error Checking and Responsiveness - Priority
- Compatibility - Priority
- Consistency - Priority
- Model Complexity - Priority
- Abstraction Level - Priority
- Atomic Setting - Priority
- Security - Priority
- Role Expressiveness - Priority
- Learning Style - Priority
- Working Framework - Priority
- Domain Correspondence - Priority
- Functional Completeness - Priority
- API Latency - Priority
- Easiness to use - Priority
- Testability - Priority
- API Elaboration - Priority
- Premature Commitment - Priority
- Penetrability - Priority
- Progressive Evaluation - Priority
- API Evolvability - Priority
- Synchrony - Priority
- Viscosity - Priority
- Work Step Unit - Priority

[Bar chart showing the priority levels for architects vs users for each fitness dimension]
Fitness Dimensions - Classification

- Learning Style - Technical/Cognitive
- Consistency - Technical/Cognitive
- Role Expressiveness - Technical/Cognitive
- Domain Correspondence - Technical/Cognitive
- Abstraction Level - Technical/Cognitive
- Premature Commitment - Technical/Cognitive
- Work Step Unit - Technical/Cognitive
- Easiness to use - Technical/Cognitive
- Penetrability - Technical/Cognitive
- Model Complexity - Technical/Cognitive
- Functional Completeness - Technical/Cognitive
- Progressive Evaluation - Technical/Cognitive
- API Elaboration - Technical/Cognitive
- Working Framework - Technical/Cognitive
- Viscosity - Technical/Cognitive
- Error Checking and Responsiveness - Technical/Cognitive
- Atomic Setting - Technical/Cognitive
- Testability - Technical/Cognitive
- Compatibility - Technical/Cognitive
- Security - Technical/Cognitive
- API Latency - Technical/Cognitive
- API Evolvability - Technical/Cognitive
- Synchrony - Technical/Cognitive

Technical | Neutral | Cognitive
Fitness Dimensions - Tradeoffs

The image shows a matrix diagram illustrating the tradeoffs between various fitness dimensions. The matrix compares different dimensions such as Compatibility, Consistency, Error Checking and Responsiveness, Abstraction Level, Model Complexity, Atomic Setting, Security, Role Expressiveness, Learning Style, Working Framework, Domain Correspondence, API Latency, Functional Completeness, Ease of Use, Testability, API Elaboration, Premature Commitment, Penetrability, Work Step Unit, Progressive Evaluation, API Evolvability, Synchrony, and Viscosity (i.e. opposite to flexibility). The values in the matrix indicate the degree of tradeoff, with higher values suggesting a greater tradeoff effect.
Platform API Fitness

- Abstraction level
- Role expressiveness
- Flexibility (was API Viscosity)
- API Elaboration
- Progressive evaluation
- Work step unit
- Domain correspondence
- Learning style
- Working framework
- Premature commitment
- Penetrability
- Consistency
# EAM Agenda

<table>
<thead>
<tr>
<th>Day 1</th>
<th>Day 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:00 Introduction of EAM (1)</td>
<td>08:30 Fix fitness dimensions and discuss their significance (5)</td>
</tr>
<tr>
<td>13:30 Customer presents the ecosystem (2)</td>
<td>09:30 Analyze ecosystem model (6)</td>
</tr>
<tr>
<td>14:00 Customer presents the ecosystem elements and their interfaces (3)</td>
<td>10:00 Break</td>
</tr>
<tr>
<td>14:30 Break</td>
<td>10:15 Analyze ecosystem model (6)</td>
</tr>
<tr>
<td>14:45 Identify ecosystem model: personas/scenarios (4)</td>
<td>11:15 Prepare and present assessment results (7)</td>
</tr>
<tr>
<td>17:00 Closing of day 1</td>
<td>12:00 Identify action plans</td>
</tr>
<tr>
<td>18:30 Evening programme</td>
<td>12:30 Closing of day 2</td>
</tr>
</tbody>
</table>
Case II: Volvo Cars Group

TOWARDS AUTOMOTIVE ECOSYSTEMABILITY MEASUREMENT
EAM: Conceptual Flow

1. **Business drivers**
   - Flexibility; development cost; re-use, time-to-market

2. **Ecosystem strategy**
   - SW; Code; HW; Composition; HIL Test; plant model

3. **Shared Tools**
   - Standardization; interoperability

4. **Ecosystem Scope**
   - ECU development using AUTOSAR

5. **Development Scenarios**
   - Continuous integration of HW/SW development; Virtual verification; collaborative HIL

6. **Sharing Criteria**
   - Freq. of integration, feedback; IP protection; complexity

7. **Development Group Personas**
   - SW/HW developers, testers, integrators

8. **Prioritize**
   - Select
   - Assess

9. **Tradeoffs**
10. **Sensitivity points**
11. **Non-risk**
12. **Risks**

**Impacts**

**Distilled into**

**Risk themes**
Zooming in...

- Automotive Ecosystem
- AUTOSAR Ecosystem
- AUTOSAR Component Development Ecosystem
- Sharing Requirements and Related Knowledge in the Automotive Ecosystem
- Shared tooling ecosystem
AUTOSAR Component Development

- The Software Supplier network

Strange mix of commodity and innovation

Main idea: By standard AUTOSAR components → Commodity
- But: 10-20% non-standard requirements → Innovation, no good processes on ecosystem level today
## Requirements flow in the automotive ecosystem

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Practices</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balancing under-specification and over-specification of requirements</td>
<td>• Personal network</td>
<td>• Networking infrastructure for communication and feedback</td>
</tr>
<tr>
<td>Synchronizing cross-function and cross-system requirements</td>
<td>• Personal network</td>
<td>• Continuous integration and deployment</td>
</tr>
<tr>
<td>Friction for changing requirements increases over time</td>
<td>• Workarounds</td>
<td>• Defer commitment</td>
</tr>
<tr>
<td>Avoid premature commitment</td>
<td>• Workarounds</td>
<td>• Guided and Just-in-time RE Rationale as by product</td>
</tr>
<tr>
<td>Slow feedback cycle on requirements structure does not fully support (e.g. need to decide too early)</td>
<td>• Oral communication</td>
<td>• Continuous verification / agile Virtual verification on early</td>
</tr>
<tr>
<td>Balancing the need for oral communication and thorough documentation of requirements</td>
<td>• Personal network</td>
<td>• On demand / Just-in-time RE Rationale as by product</td>
</tr>
<tr>
<td>Find the right person for getting or giving feedback or information</td>
<td>• Personal network</td>
<td>• Facilitate networking within company and supply-chain</td>
</tr>
<tr>
<td>Sufficiently fast supplier interaction</td>
<td>• Expert seeking</td>
<td>• New business cases / opportunities for suppliers</td>
</tr>
</tbody>
</table>

---

Ulf Eliasson, Rogardt Heldal, Eric Knauss, Patrizio Pelliccione. Experiences with Transitioning to Lean Requirements Engineering at Volvo Car Group. Submitted to IEEE Requirements Engineering Conference 2015, Ottawa, CA
Shared tools = Shared Knowledge

- **API scripts** – This will help to have a common base good to have a common base. Today we have a lot tools that are doing different/similar things at different units.
- **Testing environments**, not test cases but some environment for making system tests
- **VCC tools** for fixing RXML files that will be good because it is quite common
- **Analysis of data & software configuration tool** for tracking what software components are in the delivery
- **Model scripting** - Automation builds and analysis of models script from math works. Not much in public
- **Patches** from the web for internal analysis of Simulink models where we run queries on Simulink models & get something out of them like for processing
- …

Opportunities of cross-organization SSW development.

- Promotes continuous script growth and usage e.g. VCC tools. When the tool are compliant in one department / organization other department can still use it.
- Reduced maintenance due to collective efforts
- “We share with suppliers to save their time but no joint development i.e. we share supportive scripts verse Source code “
- Scripts may be used as a basis for other scripts
- Facilitates exchange of ideas and working principles e.g. joint knowledge about operating systems.
- Reuse by browsing some big repo to see what is existing elsewhere.
- “More benefit for our group and for VCC. It will reduce time and effort and improve quality since it has already been done and tested several time.”
- Sharing with other OEMs is Win-win situation
- “Though it important to have autonomous groups that are doing scripting on unit level to make it more specialized a community could be an avenue to share and get hints.”

Challenges and Way Forward

- “We don’t have same tools here at Volvo, different units select different tools to use.” – Also: Different platforms

- Proposals
  - Better understand what types of scripts exist and which should be shared for what reasons (commodity / innovation)
  - Identify (simple) business case for scripts – when to develop them more formal
  - Explore how to create communities of practice around types of scripts

Balance diversity and compliance between units/groups within and without VCC
Summary

SPRINT 8 RESULTS
Results of Sprint 8 (wrt Sprint Goals)

• Connect with Strategic Decision Making project
  — Defined Interface to Ecosystem Strategies in Assessment Framework
• Workshops and Interviews with Teams
  — Axis
    • Assessment Workshop with Technical Leaders
    • Follow-up Survey on Fitness Dimensions
    • Study/Interviews on API Evolution and Decisions
  — VCC
    • Secondary Software Ecosystem: 19 Interviews throughout VCC and 2 suppliers
• AUTOSAR Component Development Ecosystem: 2 Workshops, 14 Interviews (VCC, Tier 1, AUTOSAR Tier 2)
• Refining Ecosystemability Framework
  — Defined Conceptual Flow
  — Defined Ecosystemability Workshops
• Outreach
  — Apply EAM to eCommerce Case study
  — Triangulate findings with GM
Results of Sprint 8 (Dissemination)

• Hammouda, I.; Knauss, E. & Costantini, L.: **Continuous API-Design for Software Ecosystems.** In: *Proceedings of 2nd International Workshop on Rapid and Continuous Software Engeering (RCoSE ’15 @ ICSE)*, Florence, Italy, 2015


• Mozhan Soltani, Eric Knauss: **Challenges of Requirements Engineering in AUTOSAR Ecosystems.** In *Proceedings of 23rd IEEE International Requirements Engineering Conference*, Ottawa, Canada, 2015

• Upcoming:
  Bringing 2 International Conferences to Gothenburg
  — REFSQ: **Requirements**: Foundation for Software Quality, March 2016, Gothenburg
  — OSS: **Open Source** Software, April 2016, Gothenburg
Results of Sprint 8 (Tangible Results)

• API Design for the (internal) ecosystem
  – Guidelines for **API Design** (Tradeoffs, Risks)
  – Towards a **Theory of API Evolution**

• Designing Interactions and Interfaces for Cross-Organizational Continuous Deployment
  – **Secondary SW**: The role of scripts in managing integration and verification know-how for the ecosystem
  – **AUTOSAR**: The role of standards and tools on collaborative software development in software ecosystems
  – **Requirements Flow**: Need to increase inter-organizational feedback cycle speed

• **Assessment Method**
Guidelines for Interfaces and Interactions in the Software Ecosystem

SPRINT 9 GOAL
Putting things together

Platform provider

OEM

Customer

Tool provider

Tier 2 supplier

Basic SW

Tier 1 supplier

HW

Integrated tool chain

OEM
Putting things together

Which **Personas** exist and what are their needs and preferences?
- Measure API usage
- Map reqts to product

What is the **impact** of a change on the ecosystem?
Include 3\textsuperscript{rd} party extensions in testing?

How to enable innovation/commodity?
- SLA
- COTS

Should Service Providers deliver or deploy? How often/frequent?

**API-Design**
**Shared Tools**
**Transparency**
**High Frequency of Updates / Feedback**

Common Ecosystem Considerations
Guidelines for Balancing Ecosystemability Tradeoffs

Sprint 9


Bringing 2 International Conferences to Gothenburg

- REFSQ: Requirements: Foundation for Software Quality, March 2016, Gothenburg
- OSS: Open Source Software, April 2016, Gothenburg

Thanks!

- Imed Hammouda & Eric Knauss
  Imed.hammouda@cse.gu.se / eric.knauss@cse.gu.se

Ecosystemability Assessment Method