Refounding Software Engineering: The Semat Initiative

Mira Kajko-Mattsson, Ivar Jacobson, Brian Elvesæter, Michael Goedicke

Quick-and-dirty version for the Introduction to Management Engineering special lecture

See http://task3.cc/1328 for context and references
The Kernel

- Captures the essence of software engineering
- Forms a map of the software engineering context
- Constitutes a basis for evaluating on-going work
Organizing the Kernel

- Three areas of concern
  - Alphas
  - ....
What is an alpha?

- An essential element of the software engineering endeavor that is relevant to an assessment of the progress and health of the endeavor.
- Alpha is an acronym for an Abstract-Level Progress Health Attribute.
The Kernel Alphas

- Opportunity < provides to Stakeholders
- Stakeholders < helps to address Opportunity
- Opportunity focuses on Requirements
- Requirements < demand Software System
- Software System < fulfils Requirements
- Requirements scopes and constrains Work
- Work < performs and plans Team
- Team < applies Way of Working
- Way of Working < guides Work

- Set up to address Opportunity
- Opportunity focuses on Stakeholders
- Stakeholders use and consume Software System
- Software System supports Stakeholders
- Stakeholders supports Opportunity

- Customer
- Solution
- Endeavor
The Alpha structure

Checklist
- State
  - xxxxxxxxxxxxxxxxx
  - xxxxxxxxxx
  - xxxxxxxxxx
- State
  - xxxxxxxxxxxxxxxxx
  - xxxxxxxxxx
  - xxxxxxxxxx
- State
  - xxxxxxxxxxxxxxxxx
  - xxxxxxxxxx
  - xxxxxxxxxx
- State
  - xxxxxxxxxxxxxxxxx
  - xxxxxxxxxx
  - xxxxxxxxxx
...
A system made up of software, hardware, and data that provides its primary value by the execution of the software.

A software system can be part of a larger software, hardware, business or social solution.
An architecture has been selected that addresses the key technical risks and any applicable organizational constraints.

An executable version of the system is available that demonstrates the architecture is fit for purpose and supports functional and non-functional requirements.

The system is usable and demonstrates all of the quality characteristics required of an operational system.

The system (as a whole) has been accepted for deployment in a live environment.

The system is in use in a live environment.

The system is no longer supported.
Checklist for Software System

- The criteria to be used when selecting the architecture have been agreed on.
- Hardware platforms have been identified.
- Programming languages and technologies to be used have been selected.
- System boundary is known.
- Significant decisions about the organization of the system have been made.
- Buy, build and reuse decisions have been made.
Using the Kernel in practice
Using the Kernel in practice

Opportunity

Requirements

Work

Achieved  In Progress  Not yet achieved
Agenda

Semat Presentation

- The Semat Initiative
  - Ivar Jacobson
- The Semat Kernel
  - Mira Kajko-Mattsson
- The Language
  - Michael Goedicke
- Evaluation of Semat
  - Brian Elvesæter
- The Value of Semat
  - Ivar Jacobson

Semat Panel

- Participants
  - The Semat members
  - Bertrand Meyer
  - Barry Boehm
- Subject:
  - Do we need Semat?
Motivation

• There are many languages describing software development processes – why a new one?
• Some reasons:
  – Support of a Kernel
  – Support of Dynamic Semantics
  – Focus on Practitioners
A Quick Example

Scrum

Requirements

Work

Team

Software System
A Quick Example

- Work
- Sprint
- Sprint Backlog

Support Team
- Sprint Planning Meeting
- Daily Scrum
- Sprint Review
- Sprint Retrospective
List of Practices

- Iterative Development
- Scrum
- User Story
- Use Case
- Test Driven Development
- Concurrent Testing
- Architecture Essentials
- Prince2 Risk Management
Scrum

- Scrum team (roles)
  - Product Owner
  - Development Team (of developers)
  - Scrum Master
- Scrum artifacts
  - Product Backlog
  - Sprint Backlog
  - Increment
- Scrum events
  - The Sprint
  - Sprint Planning Meeting
  - Daily Scrum
  - Sprint Review
  - Sprint Retrospective
- Important note
  - Scrum’s roles, artifacts, events, and rules are immutable and although implementing only parts of Scrum is possible, the result is not Scrum.
- Source
Extending the Work Alpha

- The Work alpha covers the whole duration of a development project that may consist of a number of sprints.
- The Sprint sub-alpha has its own state graph.
- Scrum-specific rules and guidelines can be added as checkpoints for each of the Sprint states.
Enactment using Alpha State Cards

State cards can be used for reading and understanding the practice, and to progress the states of the Sprint according to the checklist defined.

Here we show the state card for the Sprint alpha in the Planned state. This requires that all checkpoints are ticked off.
Adding Work Products to the Alphas

- The Sprint Backlog is associated with the Sprint sub-alpha.
- The Product Backlog and Sprint Backlog are associated with the Requirements alpha.
- The Increment is associated with the Software System alpha.
The Scrum Activities define one or more kinds of work items and gives specific guidance on how to perform these.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Alpha Input</th>
<th>Alpha Output</th>
<th>Completion Criterion</th>
<th>Work Product Input</th>
<th>Work Product Output</th>
<th>Work Product Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meeting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daily Scrum</td>
<td>Sprint, Team</td>
<td>Sprint</td>
<td>Sprint.Under Control</td>
<td>Sprint Backlog, (Sprint Goal)</td>
<td>Sprint Backlog</td>
<td></td>
</tr>
<tr>
<td>Sprint Retrospective</td>
<td>Sprint, Team, Way-of-Working</td>
<td>Sprint,</td>
<td>Sprint.Closed</td>
<td>-</td>
<td></td>
<td>(Improvement Plan)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Way-of-Working</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>