A new approach to combine Agile and EN 50128 for Railway software development
Agenda

01 Agile Overview

02 A new approach – Agile-R®

03 A case study at Intecs
A radical Life Cycle change

Waterfall
- Conception
- Analysis
- Design
- Development
- Integration
- Testing
- Deployment

Agile
- Conception
- Analysis
- Design
- Development
- Integration
- Testing
- Deployment

OR?
Agile Goals

- Reduce time to market and improve responsiveness to change
  - Shortening the time between development and bug fixing
  - Reducing regression risks
  - Avoiding large and late integration of software

- Better control and predictability of the development process
  - Progress is measured by the state of the product’s actually working and implemented functionalities rather than estimations and presentations

- Decrease the risk of producing unsatisfactory solutions
  - Strong involvement of the product owner
Agenda

1. Agile Overview
2. A new approach – Agile-R®
3. A case study at Intecs
Agile-R® is a Scrum based approach defined by Intecs Solutions to combine Agile and EN 50128 for Railway software development.

- **Agile-R®** is described in the dedicated guideline “Agile-R®: Agile and EN 50128 for Railway software development - how to embrace both”

- **Agile-R®** has been shared and discussed with external Independent Safety Assessors
The Agile-R® context

EN50126 Lifecycle

- Agile-R® is proposed for software development
Agile-R® vs EN 50128

- Some fundamental aspects arise from INTECS experience in Sw Engineering and ISA’s feedbacks
  - Agile defines **HOW** to manage software development projects, **it is not a new standard**
  - Agile does not impose specific work products
  - Agile is **not in contradiction with WHAT** is required by EN50128
  - Agile does not sacrifice quality (quality is usually better thanks to early detection of bugs and pair programming)
  - Few adaptations are required to best combine the two approaches and achieve the right **Balance of Agility and Discipline**
## Agile-R® High Level Overview – Building Blocks

<table>
<thead>
<tr>
<th>Phase</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>To coordinate the software development with all affected stakeholders</td>
</tr>
<tr>
<td></td>
<td>To elaborate all plans</td>
</tr>
<tr>
<td></td>
<td>To establish the Scrum team</td>
</tr>
<tr>
<td>Sprint 0</td>
<td>To provide solid foundations for all other sprints</td>
</tr>
<tr>
<td>Sprint 1</td>
<td>Development heartbeat</td>
</tr>
<tr>
<td></td>
<td>To validate the development</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>Development Sprint focused on Integration activities</td>
</tr>
<tr>
<td>Sprint K</td>
<td>To finalize work products before a Release</td>
</tr>
<tr>
<td>Release</td>
<td></td>
</tr>
</tbody>
</table>
Agile-R® High Level Overview

Planning

Sprint 0

Sprint 1
Sprint 2
Sprint 3

INTEGRATION
Sprint K
Sprint N
Sprint M

Release

Sprint 1
Sprint 2

EN-50128 updated work-products

Sprint 1
Sprint 2
Sprint 3

SPR
Updated work-products

Sprint 1
Sprint 2
Agile-R® Major Recommendations

- Engage the assessor from the very beginning and find an agreement on the approach and road map
  - Software development using an Agile approach may appear new to assessors

- Tailor the approach defining the best V&V activities configuration management based on
  - Actual organization
  - Project context
  - Target SIL
  - Tools and testing environment

- Independent Testing
  - Testing of implemented user stories not assigned to the implementers of the same user stories
<table>
<thead>
<tr>
<th>Critical Area</th>
<th>Critical Aspects</th>
<th>Proposed Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poor test automation</td>
<td>• Agile effectiveness depends on the availability of fully automatic test suites</td>
<td>• Run manually a focused test set to verify each new feature</td>
</tr>
<tr>
<td></td>
<td>• Unit tests are easy to automate but system test or overall software tests are often executed manually</td>
<td>• Run manually a sanity test-set to verify major regressions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Execute full regression in release phases and selected integration phases</td>
</tr>
</tbody>
</table>
### Agile-R® Managing of Critical Issues (2/2)

<table>
<thead>
<tr>
<th>Critical Area</th>
<th>Critical Aspects</th>
<th>Proposed Approach</th>
</tr>
</thead>
</table>
| Legacy Software   | Regression       | **Different strategies in increasing order of complexity:**  
|                   |                  | • Test and document only new features  
|                   |                  |   • No confidence about the legacy part, high regression risk  
|                   |                  | • Scrum Zero set-up of a minimum initial documentation and test suite than refine according the new features  
|                   |                  |   • Partial confidence about the legacy part, medium regression risk  
|                   |                  | • Large reverse engineering activity before starting any new development  
|                   |                  |   • Confidence about the legacy part, low regression risk |

Intecs Solutions property. Reproduction prohibited in all forms and means
Agenda

01 Agile Overview

02 A new approach – Agile-R®

03 A case study at Intecs
Agile-R® in Practice – SIRIO LX Use Case

• Experimental application to the development of the Level Crossing Protection System SIRIO LX product (CENELEC SIL4)

• Outside the official development path, not starting from scratch

• Overall case study phases
  • Planning (1 week)
  • Sprint Zero (1 week)
  • 2 Sprints (3 weeks each)
# Case Study – Roles

<table>
<thead>
<tr>
<th>Agile-R® Team Role</th>
<th>Team Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO</td>
<td>1 Sirio Lx System Expert</td>
</tr>
<tr>
<td>PM</td>
<td>1 Internal Scrum Master</td>
</tr>
<tr>
<td>DEV</td>
<td>1 RQM/DES, 2 IMP/INT/TST, 1 VER</td>
</tr>
<tr>
<td>VAL</td>
<td>1 VAL</td>
</tr>
</tbody>
</table>
Case Study – Main Goals

- Understand the impact on the EN 50128 planning phase (with respect to a «legacy» waterfall life cycle)
  - Software Quality Assurance Plan
  - Software Configuration Management Plan
  - Software Verification Plan
  - Software Validation Plan

- Understand the impact on the verification activities
  - It is possible to use the same templates?
  - It is necessary to make some particular adaptation to the «legacy» way to do that?

- Get feedback from the team, composed by domain experts, but not Agile experts

- Learning lessons «from the trenches»
Case Study – Planning Results

- Impact on EN50128 planning phase
  - **Sw Quality Assurance Plan** – minimal impact
    - Deliveries follow traditional Gantt, activities proceed by time-boxed Sprints
    - Agile lifecycle where each «traditional» phase is crossed several times for each functional increment
  - **Sw Configuration Plan** – minimal impact
    - Indication to set a baseline for each phase (Sprint, Integration, Release….)
  - **Sw V&V Plan** – minimal impact
    - Clear indication that «what» and «who» do not change
    - Implementation-V&V as a continuum in order to verify and validate «as soon as possible»
    - Agile concept of «definition of done» shall embed V&V execution
    - V&V activities performed in a iterative, incremental way
Case Study – Feedback

- Positive feedback from the team about the new approach
  - Team members can have a common understanding on the system
  - What is in the Sprint scope and how to demonstrate it is clarified before starting
  - Implementers can run static analysis in the sprint context (no long and boring days to resolve or justify static analysis long after development)
- Complexity of the document verification activities seems lower than in the traditional approach
  - Analysis of requirements traceability related to a feature and documents is easier
Case Study – Lesson Learned

- The visual management approach could be very impressive but the history of previous Sprints is lost…. we tried to freeze with photos but not really practical
  - we decided to use TuLeap or Jira for future projects
- For SIL 4 software the Validator’s work, by its nature, cannot be completely time-boxed
  - The Validator shall have the freedom to perform additional analysis in order to assure the correct behavior
  - This confirms the need for Validation and Release Sprints
- When basic reusable software blocks are needed (e.g. sw timer management, sw queues,…) in our opinion, they should be developed (even partially) in Sprint 0…..like up-front software
Conclusions

- The Agile-R® approach (in our opinion and experience) changes only the «how», not the «what» or «who»

- The «what» and «who» remain strictly compliant to EN 50128 requirements

- Project pitfalls [e.g. a wrong or simplistic design, poor tools, immature test environment] have an impact on the Agile-R® approach in the same way as with a traditional approach but ...
  - You discover it after a short period of time…as we did
  - You can implement counter-measures in an early stage of the project
Thank you for your attention

Q&A