ESS INTEGRATED CONTROL SYSTEM INTEGRATION SUPPORT AND THE AGILE METHODOLOGY PROPOSAL

Miha Reščič, Cosylab, Ljubljana, Slovenia
Leandro Fernandez, Annika Nordt, ESS, Lund, Sweden

Abstract

The stakeholders of the European Spallation Source (ESS) Integrated Control System (ICS) reside in four main parts of the ESS facility: accelerator, target, neutron instruments and conventional facilities. In order to maintain and support the standardised hardware and software platforms for controls all of the stakeholders' integration requirements and efforts must be strictly harmonised.

This called for a decision by the ICS to perform the majority of the work in a package titled 'Integration Support', ranging from FPGA code development to EPICS integration. This exposes a high number of interfacing systems and devices. Planning of such activities for each system makes the standard waterfall planning model highly inefficient and risky.

In order to properly address the planning risks the agile methodology is proposed - from product owners and teams to scrums and sprints, everything to offer a better and more efficient integration support to controls stakeholders.

ICS AND AGILE DEVELOPMENT

ICS has committed itself to deliver a complete, working integrated control system for ESS by 2025. This has major implications on costing, planning and cross-project coordination, but does not change the primary driving force behind ICS: it is fundamentally a service providing organisation and the control system users satisfaction is one of the key performance indicators.

Integration Support represents a major part of the ICS work. It is heavily user oriented and is at the same time heavily dependent upon by stakeholder systems as well as at risk of delays and issues because of new / unknown equipment. In this context a traditional working model will not be able to provide the required flexibility. It needs a new methodology that promotes:

- Individuals and interactions over processes and tools
- Working software *and hardware* over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Interestingly, the above points describe the main concepts of the Agile Manifesto [1].

CONTROL BOX AND INTEGRATION SUPPORT

It is important to emphasize there are three fundamental concepts that allow ICS to standardize the hardware and software domains.

Control Box

The term has been used previously in relation to ESS [2, 3] but it has been only recently that it has been officially defined as:

> Control Box is the interface between any entity requiring control and/or monitoring functions and ICS.

This does not define the Control Box as an object in technical terms but it does establish it as the only interface to ICS. The above definition is complemented by a technical definition of a generic Control Box configuration:

> The generic Control Box consists of a CPU board (including the operating system and EPICS) and a Timing System receiver.

Integration Support

With the (generic) technical interface defined above we can define Integration Support:

> Integration Support is the effort and/or Control Boxes and/or other custom equipment provided by ICS that is required for deploying the control system components which provide monitoring and control functionality to the stakeholder systems.

The Shopping List

The above definitions cover the interface to ICS and the work performed by ICS, but not the actual control of the HW standardisation. The latter is handled by a set of lists named *the Shopping List*:

> The Shopping List is the centralised set of lists of components approved for use in Control Boxes for Integration Support to provide stakeholder system monitoring and control functionality.

The goal of ICS is to have a finite list of available HW components, with well defined responsibilities and scope of support for ICS integrators and stakeholders to choose from.

ICS INTERFACES

ICS stakeholders range from either a complete, standalone system (e.g. air conditioning) to a complex but standardised facility wide system providing a service to other parts of the facility (e.g. the vacuum system) or a complete accelerator component (e.g. the Ion Source).

By introducing Control Boxes and Integration Support we reduce the number of interfaces of any stakeholder to two:

Technical interface: the bidirectional interface between the Stakeholder System and Control Box.

User interface: the bidirectional interface between the Control System and Stakeholder.

PROJECT PLANNING

There must be a certain level of planning (tasks, milestones, effort allocation) presented to ESS management for all activities within ICS. Therefore, all activities that will be executed in agile way will be encapsulated within a high-level time and effort constraint with major milestones (e.g. *user-story writing, testing, pre-release*) of process exposed to management.

Such planning approach will be (after enough data is acquired) used later to iteratively improve the planning process, since there will be a set of known metrics acquired, e.g. $StoryPoints/UnitOfTime$ or $StoryPoints/FTE$. This approach is inspired by the book *Agile estimating and planning* from Mike Cohn [4].

REQUIREMENTS AND USER STORIES

There are two defined interfaces between the ICS product and the stakeholder: the *technical* and the *user* interface and both should be covered by user stories. However, the technical interface should be very well technically specified, because it involves hardware and logical connections.

Requirements

This introduces the need for separate requirements and specification documentation covering the technical interface that is very detailed and must be approved by the stakeholders before the Integration Support efforts begin. Such documentation reduces the probability of interface issues in the later testing and commissioning stages.

It is then ICS responsibility to translate the requirements and specifications into user stories and use them in the planning and development process.

User Stories

On the other hand, the user interface should be covered with user stories, solicited from the stakeholder in a series of meetings. The stories should be kept simple and should avoid becoming too technical and restrictive.

It is the responsibility of ICS to groom the stories and break them down into smaller, better manageable entities. The stakeholder should be involved only if a major change is required, but these changes should be reduced to a minimum.

ROLES

In order to implement the agile process the roles of the people involved must be well defined. This determines the responsibilities of people, as well as the development team.

Product owner The Product Owner should be the lead developer (integrator) for the stakeholder system. The Owner's responsibilities include: writing user stories with the stakeholder, writing, iterating and approving the technical requirements with the stakeholder, organising the backlog, planning the sprints, etc.

Development Team The product Development Team is selected from the ICS team members based on the stakeholder system specific requirements. The whole ICS team covers a wide range of core competencies: from HW and SW to ICS specific knowledge of CS Services, Timing system and other systems.

The product specific Development Team is selected and defined by the Product Owner taking into account the effort / time restrictions imposed by the project planning.

Scrum Master The Scrum Master is be selected and assigned based on a meaningful set of stakeholder systems and the area of work (e.g. Beam Diagnostics). This will allow the Scrum Master to have an overview over related product development and prevent possible divergences between related or dependent projects.

Management ICS Management should ensure that the required resources are available to build Development Teams and put Product Owners in place. It should also ensure that the high-level planning reflects the development activities. This information is retrieved from the Scrum master and the Product owners.

MEETINGS

Because there are meetings that ICS team members must attend external to the process described in this document the overall number of meetings should be reduced to a sustainable number.

Daily Scrum

The Daily Scrum is a must and should be held every day and attended by Scrum Master, Product Owner and all members of the product Development Team, including off-site developers.

Weekly Reflection meeting

In addition to the daily meeting there are regular weekly meeting in the format of *Scrum of Scrums* (Scrum across teams, [5]) should guarantee the synchronisation between different product Development Teams.

These meetings should also be attended by Product Owners of (internal) ICS products, e.g. the Timing System because many of the stakeholder specific ICS products will have strong dependencies on such ICS internal products.

iWeeks

iWeek (integration week) is an ICS term for a dedicated monthly time frame when many of stakeholder-oriented activities are clustered together in order to achieve higher efficiency. iWeek should also be considered as the end of a Sprint.

Below is a list of activities that should be performed during an iWeek:

Product Acceptance: at the end of each Sprint the stakeholder must perform official product acceptance in order to continue with the next stage of development.

Sprint Retrospective: the finished sprint has to be evaluated to gather information that will improve the future sprints but also the calculate metrics (see [4]) that help with planning and better estimation of future work.

Backlog Grooming: the backlog should be revisited after the end of each sprint to reflect the contemporary development state (and possible user story change requests from the stakeholder).

Sprint Planning: after the product is accepted and finished sprint is evaluated the next iteration of the development is defined.

SPRINTS

The ICS Sprint length should be set to about 30 days, primarily to match the iWeek schedules but also to keep the development cycles short (the exact length will vary because of calendar restrictions).

THE TOOLS

The above described process should be backed up by a set of tools that allow good user interaction and working software and hardware. Such tools are a part of the ICS Development Environment [6] and are listed below.

Code versioning and repositories: ICS is using Mercurial [7] as its code repository because of good distributed development properties, advanced branching / merging capabilities and wide user base.

Continuous integration: Hudson [8] is the selected continuous integration tool because of many good features, e.g. Mercurial integration, testing framework integration, etc. Jenkins [9] will be evaluated as a possible alternative.

Testing frameworks: various frameworks (e.g. googletest [10], Cucumber [11], JUnit [12]) are being evaluated to determine the most suitable framework.

Deployment infrastructure: RPM [13] build tools (Apache Maven [14] as a part of CODAC [15] and a combination of Apache ANT + Ivy [16]) and RPM repositories are setup to allow quick deployment of nightly, stable and other builds of software.

Task tracking: Request Tracker (RT, [17]) is under evaluation as a suitable task tracking tool, but more research on the topic is required to find a suitable candidate (possible candidates range from simple Trello [18] to enterprise solutions like Atlassian Jira [19])

Issue tracking: Bugzilla [20] is the selected issue tracking tool and will be integrated with Mercurial and Hudson.

CONCLUSIONS

This article presents an optimistic plan how to structure Integration Support efforts to meet a wide range of stakeholders. The implementation of the proposal starts in June 2013 and will continue throughout summer till October 2013, when the results and lesions learned will be presented at ICALEPCS conference [21].

REFERENCES

[1] Manifesto for Agile Software Development, `http://agilemanifesto.org/`

[2] T. Satogata, I. Verstovsek, K. Zagar, J. Bobnar, S. Peggs and C.G. Trahern, ESS Controls Strategy and Control Box Concept, Proceedings of PCaPAC 2010

[3] E. Laface, M. Rescic, The ESS Control Box, Proceedings of IPAC2012

[4] Mike Cohn, Agile Estimating and Planning, Pearson Education, 2005

[5] `http://en.wikipedia.org/wiki/Scrum_(development)`

[6] G. Trahern, Status of the ESS Control System, Proceedings of ICALEPCS2011

[7] `http://mercurial.selenic.com/`

[8] `http://hudson-ci.org/`

[9] `http://jenkins-ci.org/`

[10] `http://code.google.com/p/googletest/`

[11] `http://cukes.info/`

[12] `http://junit.org/`

[13] `http://en.wikipedia.org/wiki/RPM_Package_Manager`

[14] `http://maven.apache.org/`

[15] `http://www.iter.org/org/team/chd/cid/codac`

[16] `http://ant.apache.org/`

[17] `http://bestpractical.com/rt/`

[18] `https://trello.com/`

[19] `http://www.atlassian.com/software/jira/overview`

[20] `http://www.bugzilla.org/`

[21] `http://www.icalepcs2013.org/`