Requirements Engineering and Continuous Deployment

Nan Niu, University of Cincinnati, USA
Sjaak Brinkkemper, Utrecht University of Cincinnati, The Netherlands
Xavier Franch, Polytechnic University of Catalonia, Spain
Jari Partanen, Bittium, Finland
Juha Savolainen, Danfoss, Denmark

Abstract:
This column summarizes the panel on requirements engineering and continuous deployment held in the 25th IEEE International Requirements Engineering Conference. We highlight two synergistic points (user stories and linguistic tooling) and one challenge (non-functional requirements) in fast-paced, agile-like projects, and outline a couple of ideas to further the dialog.

Introduction

In a rapidly evolving IT environment, many companies seek to test and launch digital products and services faster and at lower cost. With tools and processes helping manage configurations, versioning, and roll-back, organizations build, test, integrate, and deploy continuously. For example, Facebook adopts the practice of *continuous deployment*, trying to release software to production as soon as it is ready in short cycles [1]. As a result, the company makes significant progress in increasing the frequency of its mobile releases. Although there exist hundreds of Android hardware variants, Facebook's Android release has gone from a release every 8 weeks to every 1 week over a period of 4 years [1].

The high-speed software development, manifested in practices like continuous deployment, changes the way that requirements are engineered. A big change lies in the ability to quickly observe the effects of the software (or the "machine" according to Michael Jackson [2]) in the environment, and to evaluate the observations against stakeholders' needs and desires. How is continuous deployment, together with its related practices, influencing requirements practitioners and researchers? What do they see as the most promising synergies between requirements engineering and continuous deployment? The most pressing challenges for the community to tackle?

These were the types of questions that we set out to explore in a panel at the 25th edition of the premier requirements engineering conference (re2017.org/). In a gorgeous early-September day in Lisbon where unconditional hospitality was experienced by every RE'17 participant, the panel offered sharp opinions and engaged in diverse conversations, accompanied by heated debates and controversies. In this column, we summarize the contributions made by the

DOI: 10.1109/MS.2018.1661332

panelists and the audience, with examples and additional materials prepared by Wentao Wang from University of Cincinnati who also helped to run the panel in Lisbon as a proud student volunteer. In what follows, we highlight two strongest synergies and then present a prominent pain point for requirements practices in fast-paced, agile-like projects.

No documented requirements? We've got "User Stories"

"Working software over comprehensive documentation" (agilemanifesto.org) has led writing requirements to be seen as a taboo in agile development, especially writing a central and upfront software requirements specification (SRS) that should be correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable, and traceable (doi.org/10.1109/IEEESTD.1998.88286). Not only were practitioners puzzled by "software before documentation" [3], but they externalized bits of information to better understand stakeholders' needs.

"People love stories. People relate to stories." Ian Sommerville shared in his RE'17 keynote about his requirements approach to Scotland's digital learning environment—Glow (connect.glowscotland.org.uk/). Even though developing realistic user requirements for Glow was impossible, stories helped Ian and his colleagues to make progress. The stories popular among agile practitioners are *user stories* [4]. Figure 1 shows an example where the essential elements of a requirement are captured in a structured way: *who* it is for, *what* it expects from the system, *why* it is important, and *how* its implementation looks like.

#Submission 13 - Zoom and pan images # Early Adopter
As a: repository user I want to: be able to deep zoom and pan on large, high-resolution images So that: I don't have to download a large image file to be able to zoom or pan Done look like: Scholar@UC includes an IIIF-compliant image server in its stack, making use of the image and presentation APIs to deliver content to users

Figure 1 A sample user story of Scholar@UC whose goals are digital preservation and discovery. The project maintains about 150 user stories. Please see: github.com/uclibs/scholar_use_cases .

Is this a good user story? Probably not if assessed based on the criteria of "problem-oriented" ("Done look like" hints at the solution) and "atomic" (the conjunction "zoom *and* pan" indicates more than one feature). Other desiderata for user stories? Unambiguous, complete, conflict-free… [4]. Sounds familiar (hint: doi.org/10.1109/IEEESTD.1998.88286)? For the Scholar@UC project, this is a good user story because the hashtag "# Early Adopter" signals the elicitation focus of engaging enthusiastic users in agile development. Moreover, stakeholder needs, desires, and preferences become clearer as the implementations go on (e.g., see the pull request comments github.com/uclibs/scholar_uc/pull/1109). In short, as long as the user stories provoke more detailed understandings of the requirements throughout development, they fit the purpose of serving as anchors for further discussions with customers [5].

Linguistically linking continuous practices

Delivering values to customers at a much accelerated pace drives continuous deployment and related practices. One of the panelists pointed out that, despite the continuous practices like integration, delivery, and deployment, agility in general is a requirements risk management method. Since self-knowledge is power, it's powerful to acknowledge that we lack perfect foresight to predict what our customers need in detail. We then become more powerful in mitigating our lack of knowledge by putting what we perceive as the working software to the hands of our customers, by explicitly testing high risk assumptions, and by doing so in short feedback cycles.

The cycles are managed very differently in different organizations and projects. Figure 2 shows two examples. As the left of Figure 2 shows, a story is only one of the anchors to further requirements understandings. A feature in Intel's application of Scaled Agile Framework refers to a relatively large portfolio item, whereas a task is intended to be operated in days to fulfill the requirements. The executions need to align with the strategic value stream [6]. Cognizant's practices, as shown in the right of Figure 2, illustrates the interdependency between requirements and testing. Even for test-driven development (TDD) advocates who want developers to create tests before writing new functional code, it is important to realize that TDD requires a thorough understanding and documentation of the requirements [7]. The low adoption of TDD practices [5] reflects the intrinsic requirements challenge: if writing good (agile) requirements is hard, so is writing good (agile) tests.

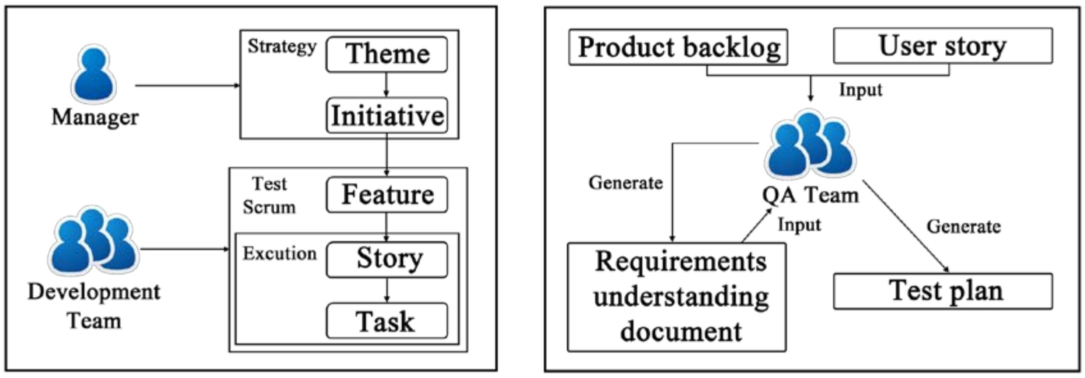
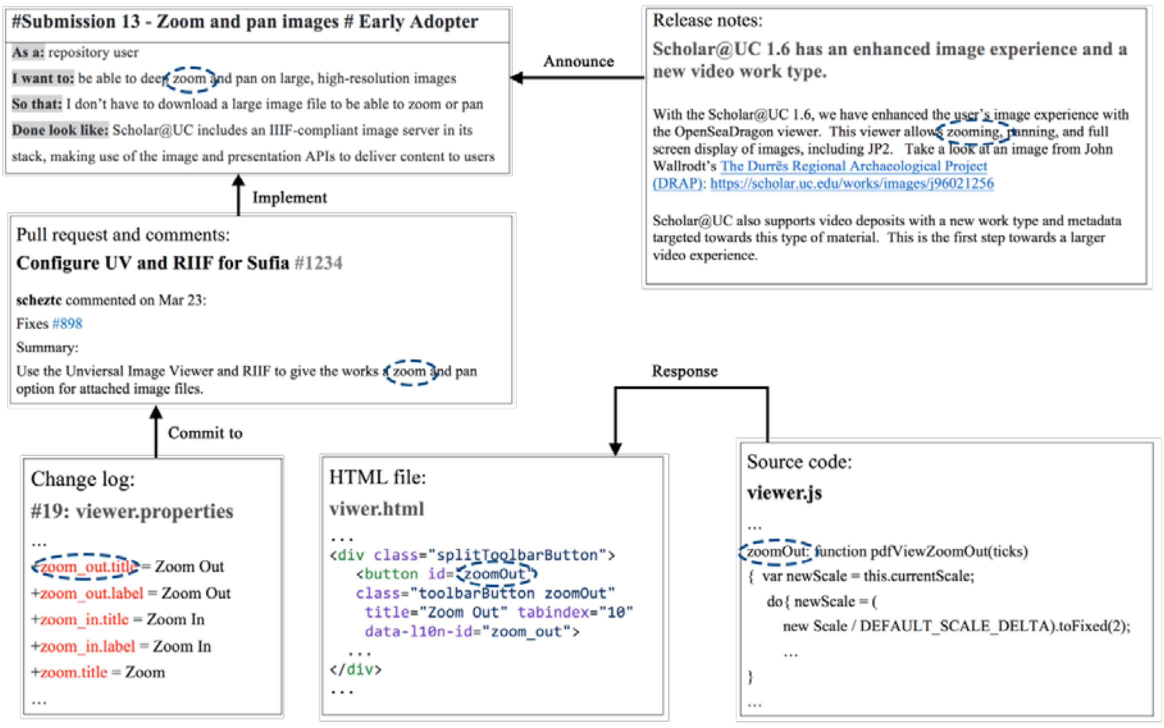


Figure 2 Intel's experience of Scaled Agile Framework (left; adapted from [6]); Cognizant's application of agile software development (right; adapted from [7]).

Figure 3 illustrates one panelist's vision of using linguistic tooling to support agile development. The feature of being able to "deep zoom and pan on large, high-resolution images" presented in Figure 1 can be traced in various artifacts shown in Figure 3 via the specific term "zoom" and its variants. The challenge is to build linguistic models for stakeholder tasks and to integrate the tooling into the native development environments. Unfortunately, we didn't find any testing artifacts to linguistically link to Scholar@UC's user story shown in Figure 3, but recent work on using automated acceptance tests that are created as part of the behavior-driven development (BDD) [8] helps instrument more ubiquitous traceability between agile requirements and production code [5].

A traceability challenge is to realize that not all the known/documented user stories should be traced. In fact, many are discarded (e.g., some "# Early Adopter" tagged Scholar@UC stories used for elicitation purposes) and many more get added during development and reflected only in development artifacts. If agility is about better managing requirements risks and agile requirements processes should be evenly spread throughout development [5], then practitioners need the support—linguistic tooling and other kinds—to identify a delta (what's new and where's the departure), find unarticulated hidden needs, clarify how to test the high-risk assumptions and how to verify the results, remove bias when providing feedback on the product increment delivery, grow test assets matching customer requirements at the system boundary… The list went on in our panel and the compilation is especially valuable for researchers and tool vendors.



#Submission 13 - Zoom and pan images # Early Adopter

As a: repository user
I want to: be able to deep zoom and pan on large, high-resolution images
So that: I don't have to download a large image file to be able to zoom or pan
Done look like: Scholar@UC includes an IIIF-compliant image server in its stack, making use of the image and presentation APIs to deliver content to users

Release notes:

Scholar@UC 1.6 has an enhanced image experience and a new video work type.

With the Scholar@UC 1.6, we have enhanced the user's image experience with the OpenSeaDragon viewer. This viewer allows zooming, panning, and full screen display of images, including JP2. Take a look at an image from John Wallrodt's The Durrës Regional Archaeological Project (DRAP): https://scholar.uc.edu/works/images/j96021256

Scholar@UC also supports video deposits with a new work type and metadata targeted towards this type of material. This is the first step towards a larger video experience.

Pull request and comments:

Configure UV and RIIF for Sufia #1234

scheztc commented on Mar 23:
Fixes #898
Summary:
Use the Unviersal Image Viewer and RIIF to give the works a zoom and pan option for attached image files.

Change log:

#19: viewer.properties

```
...
+zoom_out.title = Zoom Out
+zoom_out.label = Zoom Out
+zoom_in.title = Zoom In
+zoom_in.label = Zoom In
+zoom.title = Zoom
...
```

HTML file:

viwer.html

```
...
<div class="splitToolbarButton">
   <button id="zoomOut"
   class="toolbarButton zoomOut"
    title="Zoom Out" tabindex="10"
    data-l10n-id="zoom_out">
  ...
</div>
...
```

Source code:

viewer.js

```
...
zoomOut: function pdfViewZoomOut(ticks)
{  var newScale = this.currentScale;
    do{ newScale = (
        new Scale / DEFAULT_SCALE_DELTA).toFixed(2);
        ...
}
...
```

Figure 3 Linguistically linking software artifacts in agile development

Continuous deployment is about speed. What about safe deployment, larger-scale deployment, etc.?

"If you want to trigger a hot debate among a group of requirements engineering people, just let them talk about non-functional requirements." Martin Glinz wrote in most influential paper [9] awarded at RE'17. This held true in multiple occasions during our panel. Rapid development and continuous deployment quickly deliver the requirements to the customers and also continually allow the consequences of development decisions to emerge. As the software cumulates a critical mass of features and becomes more mature, tradeoffs must be considered between speedy deployment and other non-functional requirements such as safe and scalable deployment. For example, safety stories can be added to the sprint backlog for software systems that have safety implications at lower levels of the criticality spectrum [10]. For Figure 1's user story, compatibility issues such as whether the image format .jp2 should be supported arose in the pull request (github.com/uclibs/scholar_uc/pull/1109). Improving the quality attributes like compatibility not only clarifies the meaning of requirements in terms of which phenomena belong to the machine, the environment, and their intersections [2], but also shapes testing, build, integration, deployment, and other continuous practices.

Continuous dialog

Our panel represents the effort of continuous dialog of requirements practitioners and researchers on contemporary issues. We wish the participants had fun as we did by running the panel (see Figure 4). To further the dialog, topics of interest to conferences shall be modernized, workshops and special issues organized, tutorials on writing good and better requirements (continuously) offered, and myths (e.g., user stories are agile ways of specifying requirements) challenged. We can't wait for the panels and interactive events at RE'18 in Banff, Canada (www.re18.org/).

Figure 4 Crowning the inaugural MVP (most valuable panelist) at RE'17. Congratulations, Juha—with tolerable error rate, and yes, "fault tolerance" is yet a non-functional requirement!

References

1. Tony Savor, Facebook, "Continuous Mobile Deployment", http://www.cs.ucdavis.edu/fse2016/program/showcase/ Last accessed: November 2017.

2. Michael Jackson. "The meaning of requirements". *Annals of Software Engineering*, Volume 3, 5-21, 1997.

3. Christof Ebert and Maria Paasivaara. "Scaling agile". *IEEE Software*, Volume 34, Issue 6, 98-103, Nov/Dec 2017.

4. Garm Lucassen, *et al.* "Forging high-quality user stories: towards a discipline for agile requirements". *Proc. 23rd IEEE International Requirements Engineering Conference (RE'15)*, 2015, pp. 126-135.

5. Lan Cao and Balasubramaniam Ramesh. "Agile requirements engineering practices: an empirical study". *IEEE Software*, Volume 25, Issue 1, 60-67, Jan/Feb 2008.

6. Yariv Weltsch-Cohen, Intel, "Implementing SAFe MDO (Intel) test case", http://www.scaledagileframework.com/wp-content/uploads/2014/09/Implementing-SAFe-MDO-test-case.pdf Last accessed: November 2017.

7. Naziya Iqbal Sayed, Cognizant, "The Case for Agile Testing", https://www.cognizant.com/InsightsWhitepapers/The-Case-for-Agile-Testing-codex891.pdf Last accessed: November 2017.

8. Garm Lucassen, *et al.* "Behavior-Driven Requirements Traceability via Automated Acceptance Tests". *Proc. 2nd Just-in-Time Requirements Engineering Workshop (JIT RE'17)*, 2017, pp. 431-434.

9. Martin Glinz. "On Non-Functional Requirements". *Proc. 15th IEEE International Requirements Engineering Conference (RE'07)*, 2007, pp. 21-26.

10. Jane Cleland-Huang. "Safety Stories in Agile Development". *IEEE Software*, Volume 34, Issue 4, 16-19, July/Aug 2017.