Editor: **Sarah Gregory**
Intel Corporation
sarah.c.gregory@ieee.org

Requirements Engineering and Continuous Deployment

Nan Niu, Sjaak Brinkkemper, Xavier Franch, Jari Partanen, and Juha Savolainen

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 to manage configurations, versioning, and rollback, organizations build, test, integrate, and deploy continuously. For example, Facebook adopts continuous deployment, releasing software to production in short cycles as soon as it's ready.[1] So, the company significantly increases the frequency of its mobile releases. Although hundreds of Android hardware variants of the app exist, Facebook's Android release has gone, over four years, from a release every eight weeks to a release every week.[1]

This high-speed software development, manifested in practices such as continuous deployment, changes how 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 are continuous deployment and its related practices influencing requirements practitioners and researchers? What do they see as the most promising synergies between requirements engineering (RE) and continuous deployment? What are the most pressing challenges for the community?

These were among the questions we set out to explore in the RE in the Age of Continuous Deployment panel at the 25th IEEE International Requirements Engineering Conference (RE 17; re2017.org). On a gorgeous early-September day in Lisbon, the panel offered strong opinions and engaged in diverse conversations, accompanied by heated debates and controversies. Here, we summarize the panelists' and audience's contributions. In particular, we highlight the two strongest synergies and present a prominent pain point for requirements practices in fast-paced, agile-like projects.

No Documented Requirements? We've Got User Stories

The first synergy was the wide adoption of user stories in practice. The Agile Manifesto's statement of "working software over comprehensive documentation"[3] has led to viewing requirements writing as taboo in agile development. This is especially true regarding writing a central, up-front requirements specification that's correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable, and traceable.[4] Not only did practitioners feel that following such recommendations as "software before documentation" was difficult,[5] they also externalized bits of information (such as user stories) 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).[6] Even though developing realistic user requirements for Glow was impossible, stories helped Ian and his colleagues make progress. The stories popular among agile practitioners are user stories.[7]

Figure 1 shows a user story that captures a requirement's elements in a structured way: who it's for, what it expects from the system, why it's important, and how its implementation looks.

Is this a good user story? Probably not if you assess it on the basis of the criteria of "problem-oriented"

("Done looks like" hints at the solution) and "atomic" (the conjunction "and" in "zoom and pan" indicates more than one feature). Other desiderata for user stories? Unambiguous, complete, conflict-free ....[7] Sound familiar? (Hint: see IEEE 830-1998.[4])

For the Scholar@UC project, this user story is good because "# 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 (for example, see the pull request comments at github.com/uclibs/scholar_uc/pull/1109). In short, so long as user stories provoke a more detailed understanding of the requirements throughout development, they can serve as anchors for further discussions with customers.[8]

Linguistically Linking Continuous Practices

The second synergy the panel discussed is linguistic tooling integrated into agile development. Delivering value to customers at a much accelerated pace drives continuous deployment and related practices. One panelist pointed out that, despite continuous practices such as integration, delivery, and deployment, agility in general is a requirements risk management method.

Because self-knowledge is power, it's powerful to acknowledge that we lack perfect foresight to predict in detail what our customers need. We then become more powerful through mitigating our lack of knowledge by putting what we perceive as the working software in our customers' hands, explicitly testing high-risk assumptions in short feedback cycles.

Different organizations and projects manage the cycles differently. Figure 2 shows two examples. As Figure 2a shows, a story is only one of the anchors to further the understanding of requirements. A *feature* in a presentation from a group at Intel refers to a relatively large portfolio item, whereas a *task* is intended to be performed in days to fulfill the requirements. The *execution* must align with the strategic value stream.[9]

Cognizant's practices (see Figure 2b) illustrate the interdependency between requirements and testing. Even test-driven development (TDD) advocates who want developers to create tests before writing new functional code need to realize that TDD requires a thorough understanding and documentation of the requirements.[10] The low adoption of TDD practices[8] reflects the intrinsic requirements challenge: if writing good (agile) requirements is hard, so is writing good (agile) tests.

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" in Figure 1 can be traced in various artifacts in Figure 3 through the term "zoom" and its variants. The challenge is to build linguistic models for stakeholder tasks and integrate the tooling into the native development environments.

Unfortunately, we found no testing artifacts to linguistically link to Scholar@UC's user story in Figure 1. However, recent research on automated acceptance tests that are created as part of behavior-driven development[11] has helped instrument more ubiquitous traceability between agile requirements and production code.[8]

It's a challenge to realize that not all the known or documented user stories should be traced. In fact, many are discarded (for example, some "# Early Adopter"-tagged Scholar@UC stories used for elicitation), and many more get added during development and are reflected only in development artifacts. If agility is about better managing requirements risk and if agile requirements processes should be spread evenly throughout development,[8] 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 verify the results, remove bias when providing feedback on incremental delivery of the product, grow test assets matching customer requirements at the system boundary, ... This list of topics went on in our panel, and the topic 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 looks 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 from the Scholar@UC project, whose goals are digital preservation and discovery. The project maintains approximately 150 user stories. See github.com/uclibs/scholar_use_cases. IIIF = International Image Interoperability Framework.

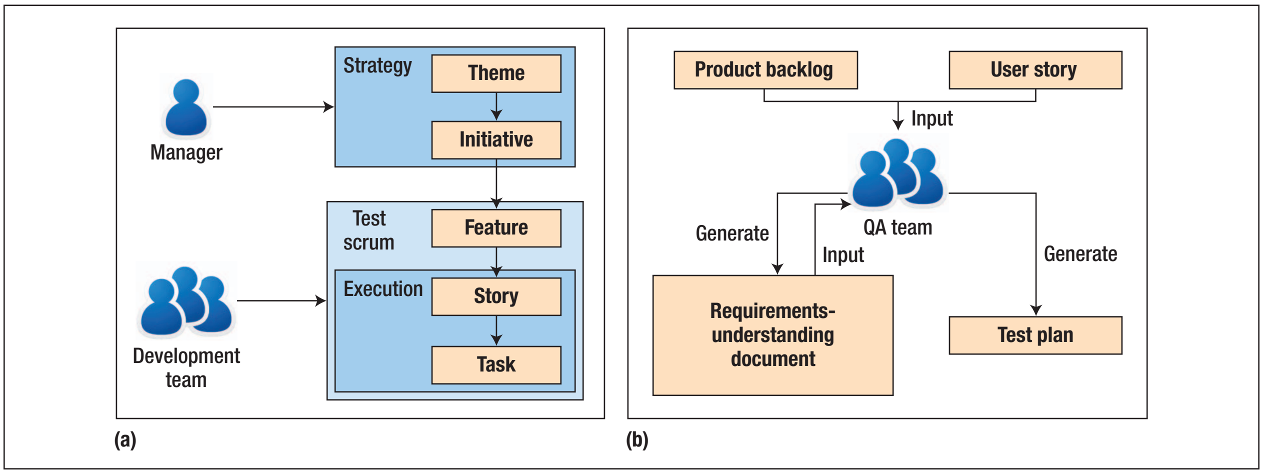
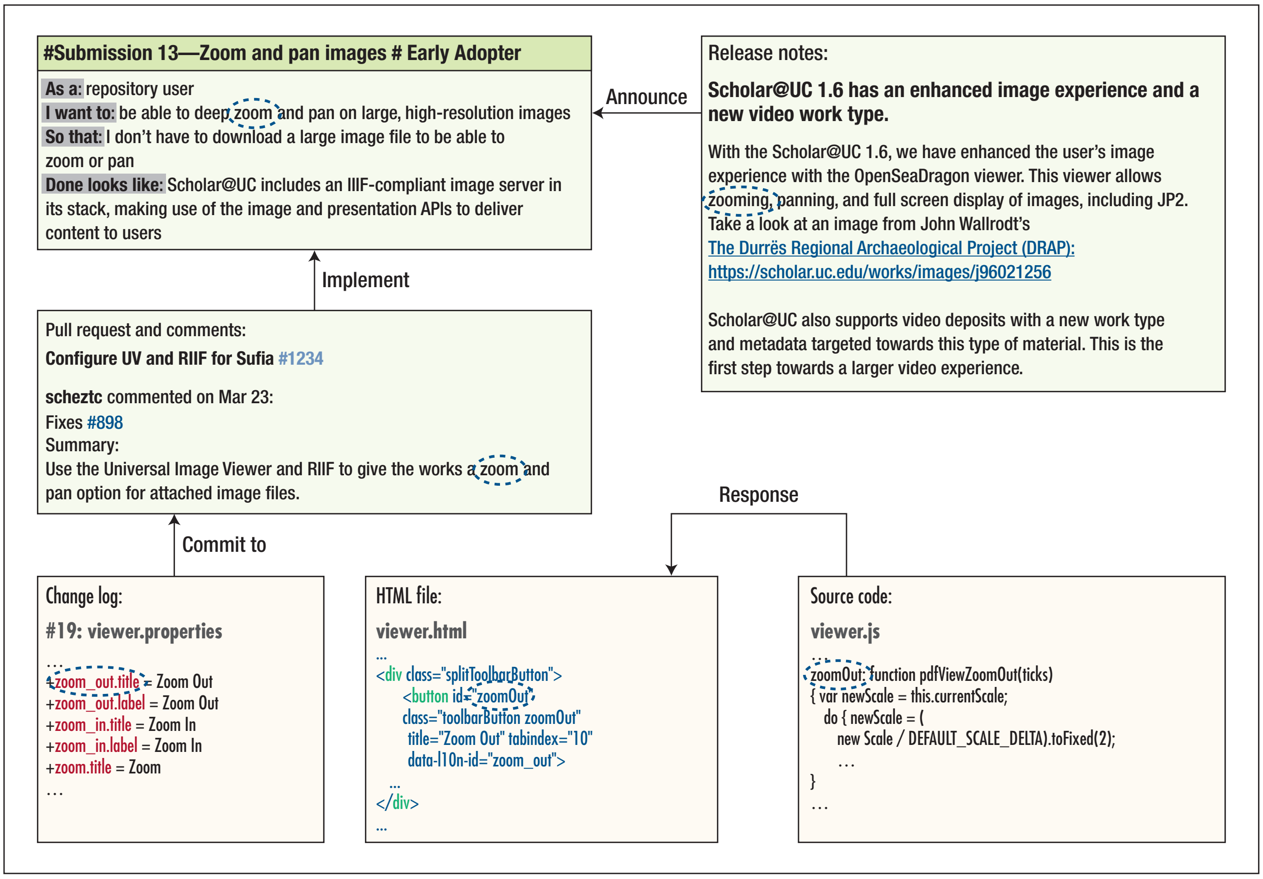


FIGURE 2. Two ways to manage short feedback cycles. (a) One Intel team's experience with the Scaled Agile Framework.[9] (b) Cognizant's application of agile development.[10] These examples show user stories' different roles in different projects. QA = quality assurance.



#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 looks 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 Universal 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:

viewer.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. The challenge is to build linguistic models for stakeholder tasks and integrate the tooling into the native development environments.

The Pain Point: Nonfunctional Requirements

"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 a paper that won the RE 17 most-influential-paper award.[12] This observation held true on multiple occasions during our panel.

Rapid development and continuous deployment quickly deliver requirements to the customers and continually allow the consequences of development decisions to emerge. As the software accumulates a critical mass of features and becomes more mature, tradeoffs must be considered between speedy deployment and other nonfunctional 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.[13]

For Figure 1's user story, compatibility issues such as whether to support the JP2 image format arose in the pull request (see github.com/uclibs/scholar_uc/pull/1109). Improving quality attributes such as compatibility clarifies the requirements' meaning in terms of which phenomena belong to the machine, the environment, and their intersections.[2] It also shapes testing, build, integration, deployment, and other continuous practices.

FIGURE 4. Crowning the inaugural MVP (most valuable panelist) at RE 17. Congratulations, Juha—with a tolerable error rate, and, yes, fault tolerance is yet another nonfunctional requirement!

Our panel represents a step toward a continuous dialogue between requirements practitioners and researchers. We hope the participants had as much fun attending the panel as we did running it (see Figure 4). To carry on this dialogue, members of the RE community need to propose new topics of interest, organize workshops and special issues, offer tutorials on writing good requirements, and challenge myths (for example, that user stories are agile ways of specifying requirements). We can't wait for the panels and interactive events at RE 18 in Banff, Canada (www.re18.org).

Acknowledgments

We thank Wentao Wang from the University of Cincinnati, who helped run the panel and prepared examples and additional materials.

References

1. T. Savor, "Continuous Mobile Deployment," presentation at 2016 ACM SIGSOFT Int'l Symp. Foundations of Software Eng. (FSE 16), 2016.
2. M. Jackson, "The Meaning of Requirements," *Annals of Software Eng.*, vol. 3, no. 1, 1997, pp. 5–21.
3. K. Beck et al., "Manifesto for Agile Software Development," 2001; agilemanifesto.org.
4. *830-1998—IEEE Recommended Practice for Software Requirements Specifications*, IEEE, 1998; doi .org/10.1109/IEEESTD.1998 .88286.
5. C. Ebert and M. Paasivaara, "Scaling Agile," *IEEE Software*, vol. 34, no. 6, 2017, pp. 98–103.
6. I. Sommerville, "Simplicity Considered Harmful," keynote address, 25th Int'l Requirements Eng. Conf. (RE 17), 2017; re2017.org/slides /RE2017_Keynote_Ian.pdf.
7. G. Lucassen et al., "Improving Agile Requirements: The Quality User Story Framework and Tool," *Requirements Eng.*, vol. 21, no. 3, 2016, pp. 383–403.

ABOUT THE AUTHORS

NAN NIU is an assistant professor in the University of Cincinnati's Department of Electrical Engineering and Computer Science. Contact him at nan.niu@uc.edu.

JARI PARTANEN is the head of quality and environment at Bittium. Contact him at jari.partanen@bittium.com.

SJAAK BRINKKEMPER is a professor in Utrecht University's Department of Information and Computing Sciences. Contact him at s.brinkkemper@uu.nl.

JUHA SAVOLAINEN is the senior director of software at Danfoss. Contact him at juhaerik.savolainen@danfoss.com.

XAVIER FRANCH is a professor in the Polytechnic University of Catalonia's Software and Service Engineering Group. Contact him at franch@essi.upc.edu.

8. L. Cao and B. Ramesh, "Agile Requirements Engineering Practices: An Empirical Study," *IEEE Software*, vol. 25, no. 1, 2008, pp. 60–67.
9. Y. Weltsch-Cohen, "Implementing SAFe MDO (Intel) Test Case," 2014; www.scaledagileframework.com/wp-content/uploads/2014/09/Implementing-SAFe-MDO-test-case.pdf.
10. N.I. Sayed, "The Case for Agile Testing," Cognizant, 2014; www.cognizant.com/InsightsWhitepapers/The-Case-for-Agile-Testing-codex891.pdf.
11. G. Lucassen et al., "Behavior-Driven Requirements Traceability via Automated Acceptance Tests," *Proc. 25th Int'l Requirements Eng. Conf. Workshops* (RE 17), 2017, pp. 431–434.
12. M. Glinz, "On Non-functional Requirements," *Proc. 15th IEEE Int'l Requirements Eng. Conf.* (RE 07), 2007, pp. 21–26.
13. J. Cleland-Huang, "Safety Stories in Agile Development," *IEEE Software*, vol. 34, no. 4, 2017, pp. 16–19.