Agile Record

The Magazine for Agile Developers and Agile Testers

www.agilerecord.com free digital version made in Germany ISSN 2191-1320

October 2011

issue 8

Agile ALM and collaborative development

by Michael Hüttermann

Application Lifecycle Management (ALM) synthesizes technical and functional elements to provide a comprehensive approach to common project activities and phases, addressing build, configuration, deployment, release, test, quality, integration, and requirements management. With its interdisciplinary approach, Agile ALM integrates project roles, project phases and artifact types. Agile ALM enriches an ALM with agile values and strategies. An agile approach to ALM improves product quality, reduces time to market, and makes for happier developers. Agile ALM is one of the ways in which ALM helps to provide structure for agile. In a nutshell, Agile ALM

- Helps overcome process, technology, and functional barriers (such as roles and organizational units).
- Spans all artifact types as well as development phases and project roles.
- Uses and integrates lightweight tools, enabling the team to collaborate efficiently without any silos.
- Makes the relationship of given or generated artifacts visible, providing traceability and reproducibility.
- Defines task-based activities that are aligned with requirements. This means that the activities are linked to requirements, and that all changes are traceable to their requirements.

One essential aspect of agile ALM is collaborative development, which is what we'll discuss next.

Collaborative development

Writing software collaboratively means that all stakeholders work on the solution, constructively, and they have shared goals. The agile testing matrix (see figure 1, a skeletal based on Lisa Crispin's version of Brian Marick's diagram) defines test quadrants, which are integrated and aligned with the outside-in approach. According to C. Kessler and J. Sweitzer, the main drivers of outside-in are:

- Understanding your stakeholders and the business context
- Mapping project expectations to outcomes more effectively
- Building more consumable software, making systems easier to deploy and use
- Enhancing alignment with stakeholder goals continuously

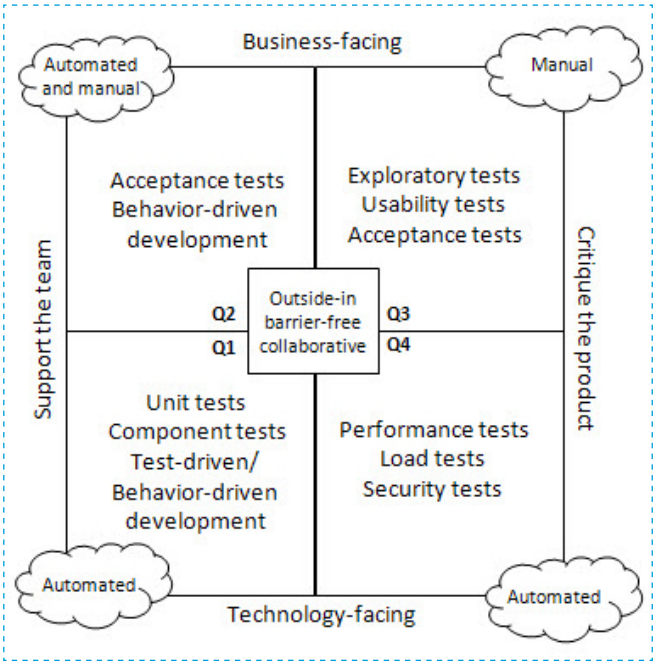

Figure 1: Agile testing matrix with test quadrants that are aligned with the outside-in and barrier-free approach to software development.

Acceptance tests determine whether a system satisfies its specified acceptance criteria. This helps the customer to decide whether to accept the software. This means that acceptance tests also tell the programmers what the customer doesn't want them to do. Executable specifications allow you to use tools that read the specifications automatically (either after manually starting the process, or continuously as part of a continuous integration pro-

cess), process them against the system under test, and output the results in an objectively measurable, efficient, and readable way. The domain expert specifies the tests in simple formats (for example based on HTML or Excel), and the program writes the results after running the tests against the system under test. Merging different mediums for documentation (single-sourcing product information) reduces the amount of traditional documentation, because the specification is the system's functional documentation and therefore can be efficiently validated against the current software state. This leads to a living documentation that is always up-to-date. There are many different ways of how to implement acceptance tests, and many different integrated tool chains can be used, for example chains based on Fit/FitNesse. Beside tools, you can also utilize platforms and languages to implement acceptance tests, which is what we'll discuss next.

Barrier-free, with BDD flavored acceptance tests

Behavior-driven development (BDD) promotes a special approach to writing and applying acceptance tests that's different from the traditional test-driven development (TDD), although BDD also promotes writing tests first. With BDD, tests are like (functional) specification stories, normally in a given/when/then format. This specification-oriented technique also uses a natural language to ensure cross-functional communication and to understand business concepts. BDD provides a ubiquitous language for analysis and emphasizes application behavior over testing. Scala and Groovy, both 1st class citizens on the Java virtual machine, offer interesting features for setting up a polyglot ecosystem, leveraging existing platforms by providing solutions that involve special purpose languages. With Scala and Groovy, you can write tests, which helps to overcome various barriers:

- Barriers between project phases and project activities (because coding and testing move together more closely)
- Barriers between artifact types (because code and executable specifications are written on the same unified infrastructure)
- Barriers between project roles (because tests are written collaboratively, with mechanisms to use terms close to the problem domain)
- Barriers between tools (because the same tools are used for programming and testing)

Overcoming common project barriers leads to what I call a barrier-free approach. In the Java ecosystem, you can write BDD flavored acceptance tests with Scala and the Scala specs2 library, or with Groovy and the Groovy easyb library.

Summary

Agile ALM spans many disciplines in software engineering. Agile ALM helps to provide structure for agile and helps to approach ALM in a determined, pragmatic way. One essential aspect of agile ALM is collaborative development that can help to improve the probability of success of your project.

Reference

Michael Hüttermann, *Agile ALM*, Manning, 2011

Carl Kessler and John Sweitzer, *Outside-in Software Development*, IBM Press, 2008

Lisa Crispin and Janet Gregory, *Agile Testing*, Addison Wesley, 2009

Brian Marick's test matrix: http://www.exampler.com/old-blog/2003/08/21/

> About the author

Michael Hüttermann
(Java Champion, SCJA, SCJP, SCJD, SCWCD) is freelance developer, architect, coach, author and tutor for Java/JEE, ALM/SCM and agile software development. Further information: http://huettermann.net.