**Agile Product & Project Management Advisory Service**
**Executive Update Vol. 12, No. 16**

# Generalists, Specialists, and Generalizing Specialists

by Steve Berczuk

Software development is complicated, requiring highly specialized skills. Agile methods such as Scrum advocate self-organizing, cross-functional teams and promise predictable, efficient delivery of useful functionality. To have teams composed of people who are specialists in each of the relevant skills for your project, you need to know what the skills are before you plan the iteration. Unless the work for an iteration matches the available people, you will have idle hands and incomplete work. You are also unlikely to have an expert in every area you need. Having teams of generalists address some of the problems, but it's a reasonable concern that generalists aren't going to deliver the best solution for a task.

An organization adopting agile software development needs to create teams that can be responsive to changing requirements and also commit to delivering high-quality functionality at the end of each development iteration. This *Executive Update* discusses the concept of generalizing specialists and how to form teams that have the right balance of specialized skills to deliver good software and the flexibility to allow for efficient delivery.

## SPECIALIZATION AND TEAMWORK

At first glance, you might think that the best way to deliver good software quickly is to have teams of people who have deep skills in required areas. There are a number of reasons why this is not the most effective approach to developing software when you have changing requirements and need to be agile. Some of these reasons are as follows:

- When a project starts, you don't know the balance of work across skills, so it is almost impossible to staff a team with the "right" skills.
- Even if you have the correct skills on the team at the start, work allocation might not match staffing allocation.
- As discussed in a recent *Harvard Business Review* blog post, it's not just the individuals on a team, but their interactions that lead to productivity and creativity.[1]
- You want the team to be small enough that the entire team can participate in planning and design sessions when appropriate. And ideally, you want your agile team to be composed of people fully committed to the project at hand.

Thus, you want a team that can comprise a broad set of competencies in a relatively small number of people. Specialists don't provide this for the reasons just discussed. Generalists, who can do all things — and not excel at any — don't address the need for deep knowledge on the team. According to Cutter Senior Consultant Scott Ambler, generalizing specialists combine most of the flexibility of generalists, with the ability to apply deep technical skills to a problem in a specific domain.[2] This allows you to deliver functionality efficiently and develop creative solutions while also keeping team members learning.

A generalizing specialist model is counter to the way people are used to managing and working. This *Update* provides some advice about how to balance generalists, specialists, and generalizing specialists.

## SPECIALIZATION AND THE DEVELOPER

It takes several different skills to deliver working software. A typical team developing e-commerce applications might have user interface (UI) developers, middleware developers, a release engineer, and database developers. Among each of these groups, there may be subspecialities (within middleware developers, for example, there might be an expert in object-relational

mapping, a framework expert, and a messaging expert). These divisions illustrate that there are various dimensions in which people on a team have special skills. Some of the ways that developers specialize include:

- **Layer.** The technologies for UI development are different than those for server development or database administration.
- **Phase.** Coding, build, and testing are often considered distinct skills, even though they often occur in close proximity, especially on agile teams.
- **Domain.** Independent of technology, some people on a team might have experience or knowledge about financial services, or someone might have a particular aptitude for algorithm work or integration work.
- **Technology.** One person on the team might have knowledge of a framework you use or a third-party technology that you use. Someone might be especially proficient in a programming language (e.g., a language lawyer in the Chief Programmer Team model[3]).

While many good software developers have skills that span these dimensions, people have aptitudes, interests, and experiences that can affect how long it takes to solve a problem. If you were to assign work to developers, it might seem reasonable to assign and select work based on the skill set of the person. However, by assigning work rather than having teams select work at the start of an iteration, you can limit the productivity of the team and lose some of the benefits of agility that come from agile, self-organizing teams.

## AGILE AND SELF-ORGANIZING TEAMS

Agile projects are self-organizing. According to codeveloper of Scrum, Ken Schwaber, this means that team members decide how the work gets done and commit to meeting deliverables.[4] This combination means that teams can be flexible in addressing requirements and be more productive. Self-organizing is not just agile dogma; it enables the benefits of agility. When teams self-organize, they can take the time to find a good solution quickly and leverage everyone's skills.

An important aspect of this process is that the team decides how best to deliver the functionality. Quite often, a feature that seems as if it might be best done one way (e.g., as a stored procedure in the database) might actually be better implemented in another way (e.g., in the application tier). You might not know the skills required to implement a feature until the team does some design work. The collaboration between the various generalizing specialists on a team makes this clear.

There may also be occasions when much of the work in an iteration seems to involve the expertise of one person on the team who may not be available. In these situations, you need to decide whether to defer the task or allow the team to have someone who is less familiar (but competent) with the technology work on the tasks. This flexibility allows the team to focus on business needs first, rather than let technical capacity be the sole determinant of priority.

In some cases, work naturally involves more than one discipline, so dividing work will add overhead. For example, consider a simple tool with a UI that answers a question based on information in a database table. You could have a UI developer do the front end, a database developer define the schema, a build engineer write the build script, and add the project to the integration server. But if one developer did all the work across the lifecycle, you could see results more quickly and then iterate.

While having a team composed of multi-talented people who are equally adept at all areas of technology and domain understanding sounds like it could be beneficial, it's also an unreasonable goal. There is so much to know about any one domain that it isn't reasonable for someone to have the time to digest all the information there is about current practices in more than a couple of domains.

The most flexible agile teams are composed of generalizing specialists. There may be someone on the team who is an excellent UI developer but is able to work on data architecture, for example. If your team can adopt a generalizing specialist mindset, you can get more done more quickly and improve morale.

The *Executive Update* is a publication of the Agile Product & Project Management Advisory Service.  Print ISSN: 1946-7338 (*Executive Report*, *Executive Summary*, and *Executive Update*); online/electronic ISSN: 1554-706X.

## Advantages of Generalizing Specialists

Ambler defines a generalizing specialist as someone who:[5]

- Has one or more technical specialties (e.g., Java programming, project management, database administration)
- Has at least a general knowledge of software development
- Has at least a general knowledge of the business domain in which her or she works
- Actively seeks to gain new skills in both existing specialties as well as in other areas, including both technical and domain areas

Having generalizing specialists onboard can help your team be more efficient and come up with better solutions. Some of the advantages of having teams of generalizing specialists are:

- **Better flow** — both in the lean sense of maintaining the flow of work through the system and the optimal experience sense[6] of creating an environment where people can focus
- **A more reliable path to the "best" solution** — as team members have some understanding of the various approaches and are less likely to focus solely on what they know best
- **Professional development** — as developers want to learn, and with every new challenge there is an opportunity to extend their expertise

Fostering a culture of generalizing specialists can make for a better team and more productive team members.

## Core Skills and Specializations

There are some skills all members of an agile team should have to avoid bottlenecks. They include:

- **Understanding the basics.** Build and configuration management team members should understand the basics of their build and software configuration management systems so that they can execute, debug, and make simple modifications to the build process. The build serves the developer.
- **Testing.** A good understanding of testability makes for more reliable and modular code, and testing closer to when the time code is constructed is more efficient.

### THE CHALLENGE OF HYPERSPECIALIZATION

In a recent *Harvard Business Review* article, Malone et al. discuss the benefits of a hyperspecialization approach, using an example from software development where UI design work is farmed out to a UI design expert.[1] Most of the examples in the article involve well-defined tasks. Agile methods give you an advantage when there is vagueness in the requirements or the solution. If you have a situation that works, such as the one in the HBR case, then specialists might be answer.

[1]Malone, Thomas W., Robert J. Laubacher, and Tammy Johns. "The Big Idea: The Age of Hyperspecialization." *Harvard Business Review*, July 2011.

- **Experience in scripting languages.** Almost every project needs tools to automate processes and testing. Being able to write scripts to automate routine processes helps teams make configurable, deployable code.
- **UI development.** While user experience can be a complex subject, many tools have a UI component by means of which a user interacts with the application. Familiarity with the UI framework of choice can help those with expertise in the middle tier develop robust APIs and can allow someone to own the first draft of an end-to-end implementation.
- **Database skills.** This applies if you have a data-driven application.

Being a "master of all" isn't practical and having people on your team who can do everything isn't realistic. Here are some areas where you want to have specialization:

- The primary development language(s) of the project
- Deeper database skills
- User experience

## Onward to Effective Cross-Functional Teams

In author Atul Gawande's book about improvement in medical professions, he relates a story of a clinic in rural India that is poorly staffed and poorly funded, and where doctors manage to successfully perform procedures that are outside the realm of their training.[7] Of necessity, this group of people performed work outside the range of their expertise and saved lives. They

were able to do this because they frequently met to discuss their cases with a desire to learn from each other. While software development is different than medical care, the value of being able to fill gaps in a team is the same.[8] There are lessons software teams can take away from these stories about how to use team resources and leverage people's skills, including:

- For generalizing specialists to be effective, the team needs to have an ethic of continual learning, both from outside resources and each other.
- When deciding who should work on a task, consider the overall flow of the project. If the specialist is available, then you may as well have the "expert" work on the task (after the team agrees on an approach). But if waiting for the specialist would delay the project and there are other people on the team who don't have a full backlog of work, consider having someone else work on it. The specialist can provide advice and perhaps improve on the work later. But you will have functional software sooner.

Deciding who the "right" person for a job is involves balancing project schedule, skills, and interests. While the temptation to categorize work into a layer or domain and assign it to the domain expert is great, you might find that projects are done more quickly when the team collaborates to find a solution. Some steps you can take to build and leverage a team of generalizing specialists are:

- **Don't presume how a feature will be implemented.** Try to keep feature definitions in terms of user (or client) interactions, rather than assuming a database or UI implementation.
- **Have the team collaborate to decide how to address development tasks.** There are several possible solutions to a problem, and what seems right at the start might not be the best after further exploring the problem.
- **Ensure that the team has a good understanding of schedule and feature priorities.** This will help the team to decide whether the task should wait for the expert or whether someone else should give it a start.
- **Encourage team members to sign up for tasks rather than have them assigned.** This is especially important when a person is working in a new area so that they he or she feels more committed.

## SPECIALIZATION AND GENERALISTS

Software development is a complex, rich discipline and it is difficult, if not impossible, for a developer to learn about all the tools, technologies, and frameworks that exist and still have time to write code. So a certain degree of specialization is inevitable. But when you form a team, you want people who have deep expertise on one or more of the things you are working on and who are familiar with some of the other aspects of the application. A team with this mix of skills and a desire to communicate and share knowledge can deliver useful software more quickly.

## ENDNOTES

[1]Taylor, Bill. "Great People Are Overrated." *HBR Blog Network*, 20 June 2011 (http://blogs.hbr.org/taylor/2011/06/great_people_are_overrated.html).

[2]Ambler, Scott W. "Generalizing Specialists: Improving Your IT Career Skills." Ambysoft (www.agilemodeling.com/essays/generalizingSpecialists.htm).

[3]Brooks, Frederick P. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley Professional, 1995.

[4]Schwaber, Ken. *Agile Project Management with Scrum*. Microsoft Press, 2004.

[5]Ambler. See 2.

[6]Csikszentmihalyi, Mihaly. *Flow: The Psychology of Optimal Experience*. Harper Perennial, 1991

[7]Gawande, Atul. *Better: A Surgeon's Notes on Performance*. Metropolitan Books, 2007.

[8]Berczuk, Steve. "Specialization, Generalization, and Effectiveness in Software Teams: Clinical Metaphors." *Accidental Simplicity*, 4 July 2011 (http://steveberczuk.blogspot.com/2011/07/specialization-generalization-and.html).

## ABOUT THE AUTHOR

Steve Berczuk is an engineer and ScrumMaster at Humedica, where he's helping to build next-generation clinical informatics applications based on SaaS. The author of *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*, he is a recognized expert in software configuration management and agile software development. Mr. Berczuk is passionate about helping teams work effectively to produce quality software. He has a master's degree in operations research from Stanford University, a bachelor's degree in electrical engineering from MIT, and is a Certified Practicing ScrumMaster. He can be reached at steve@berczuk.com.

 