Agile Project Management with Azure DevOps

Concepts, Templates, and Metrics

Joachim Rossberg

Apress®

Agile Project Management with Azure DevOps

Concepts, Templates, and Metrics

Joachim Rossberg

Apress®

Agile Project Management with Azure DevOps: Concepts, Templates, and Metrics

Joachim Rossberg
Kungsbacka, Sweden

ISBN-13 (pbk): 978-1-4842-4482-1 ISBN-13 (electronic): 978-1-4842-4483-8
https://doi.org/10.1007/978-1-4842-4483-8

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Joan Murray
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover image designed by Freepik (www.freepik.com)

To my kids, Amelie and Eddie.
Love you forever.

Table of Contents

About the Author

Joachim Rossberg is an expert in Agile project management, SAFe, DevOps processes, and Azure DevOps/TFS, working for Solidify in Sweden. An IT consultant for more than two decades, primarily in the role of product owner, Agile coach and trainer, and project manager, he has extensive experience as a system developer and designer and holds certifications in PSM I & II, SAFe SPC, PSPO I, PSD I, CSPO, and PAL I. He is the author of seven books.

About the Technical Reviewer

Gregor Suttie lives near Glasgow, Scotland, and has been in the IT industry for 20-plus years. He started with Visual Basic 6 and Com +, along with Visual Interdev and the original Active Server Pages. After that period in his career, he moved on to .Net and has been developing with it since its original release. More recently, he has been concentrating on learning Azure, which he had never used until the end of 2018.

During the last quarter of 2018, Gregor sat for eight Azure beta exams and received most of his results by January 2019. He is a certified Azure developer and is one transition exam away from being a certified Azure architect. He is still waiting for the results of his Azure DevOps beta exam.

Gregor has a passion for learning, and you can access his blog at http://gregorsuttie.com and on twitter at @gregor_suttie. He is enthusiastic about Azure and DevOps, and he has been using them daily at work and has learning more about Azure DevOps since its release. Every week he discovers something new. When the opportunity arose to be a technical reviewer for a book about Azure DevOps, he jumped at the chance!

The book covers a few different areas, but its guts cover Azure Boards and how to go about using this feature successfully. Gregor hopes you enjoy reading the book as much as he did reviewing it.

Introduction

Throughout the years, many things have happened with Team Foundation Server (TFS) and the Visual Studio Team System. The product has gone through several names and is now called Azure DevOps and Azure DevOps Server. It's kind of hard to keep track of them all.

This book is written with Agile leaders in mind, such as product owners, Scrum masters, Agile project/product managers, and the like. Members of Agile teams can benefit a lot as well.

In this book, I write about how we can use the features and functionality of Azure DevOps, and customize our work process. I cover Kanban board customizations in both Azure DevOps and Azure DevOps Server/TFS.

In Chapter 1, we start with a brief introduction to a concept called *application life cycle management*, which, to this day, I still think describes the areas of DevOps well. After that, we look at DevOps (Chapter 2) and then at Agile concepts (Chapter 3) such as Scrum, Kanban, SAFe, and more. In Chapters 4 and 5, I talk about the underlying logic of Azure DevOps. You will see what work items are and how we can customize them to fit our way of working. Chapter 6 covers Agile practices in Azure DevOps. We examine things like test-driven development, the Scaled Agile Framework, (SAFe) framework, and how we can use Azure DevOps to implement SAFe and more. Before we look at a fictional project implementation in Chapter 8, Chapter 7 takes you through some Agile metrics that are good to monitor.

CHAPTER 1

Introduction to Application Life Cycle Management

What do you think about when you hear the term *application life cycle management* (ALM)? During a seminar tour in 2005 in Sweden, presenting on Microsoft's Visual Studio Team System, we asked people what ALM was and whether they cared about it. To our surprise, many people equated ALM with Operations and Maintenance. This is still often the case when we visit companies, although today more people are aware of the term.

Was that your answer as well? Does ALM include more than just Operations? Yes, it does. ALM is the thread that ties together the development life cycle. It involves all the steps necessary to coordinate development life cycle activities. Operations are just one part of the ALM process. Other elements range from requirements gathering to more technical things such as the build-and-deploy process.

These days we do not talk as much about ALM as a concept as we used to. These days we talk more about DevOps. But, let's start by talking some ALM in this chapter to lay a foundation for DevOps and Azure DevOps, as Microsoft calls it.

J. Rossberg, *Agile Project Management with Azure DevOps*,
https://doi.org/10.1007/978-1-4842-4483-8_1

Microsoft renamed Visual Studio Team Services (VSTS) to Azure DevOps at the end of 2018. As of this writing, Microsoft's Team Foundation Server (TFS) is in version 2018. In the near future, the TFS name will be changed to Azure DevOps Server.

Aspects of the ALM Process

All software development includes various steps performed by people playing specific roles. There are many different roles, or disciplines, in the ALM process, and some of them are defined in this section. (The process could include more roles, but we focus on the primary ones.)

Look at Figure 1-1, which illustrates ALM and some of its aspects. You can see the flow from the birth of an application, when the business need first arises, to when the business need no longer exists and the application dies. Once the thought of a new application (or system) is born, many organizations do some portfolio rationalization. This means you discuss whether an existing system can handle the need or whether a new system has to be developed. If a new system must be built, you enter the *software development life cycle* (SDLC) and develop the system, test it, and deploy it into operation. This is the point at which you do a handover so that Operations personnel can maintain and refine the system. Once in production, the system (hopefully) delivers the intended business value until retirement. While in operation, the system usually is updated or undergoes bug fixes; at such times, *change requests* (CRs) are written. For each CR, you go through the same process again.

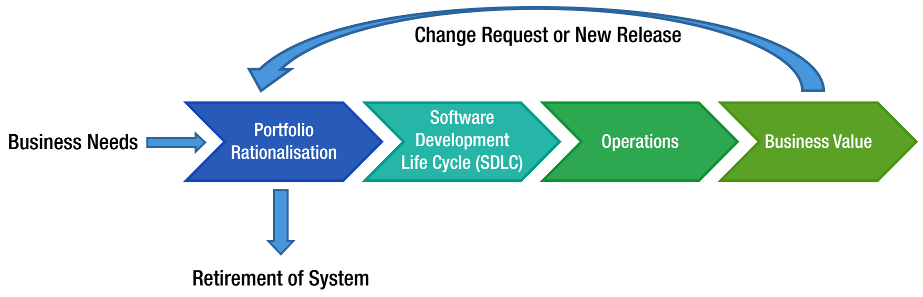
Change Request or New Release
Business Needs
Portfolio Rationalisation
Software Development Life Cycle (SDLC)
Operations
Business Value
Retirement of System

Figure 1-1. *The application life cycle management process*

It's essential to understand that all business software development is a team effort. The company personnel who play specific roles collaborate on projects to deliver business value to the organization. If you don't have this collaboration, the value of the system will most likely be considerably less than it could be. One step up from the project level, at the program level, it's also important to have collaboration between all roles involved in the ALM process so that you perform this process in the most optimal way.

The roles in the ALM process include, but aren't limited to, the following:

- *Stakeholders*: Stakeholders are usually the people who either pay for the project or have decision-making rights about what to build. We like to also include end users in this group, so not only management has a stake in a project.

- *Business manager*: Somebody has to decide that a development activity is going to start. After initial analysis of the business needs, a business manager decides to initiate a project to develop an application or system that delivers the expected business value. A business manager, for instance, must be involved in the approval process for a new suggested project, including portfolio rationalization, before a decision to go ahead is made. Information technology (IT) managers are also part of this process because IT staff will probably be involved in the project's development and deployment into the infrastructure.

- *Project manager, product owner, or Scrum master*: Suitable individuals are selected to fill these roles, and they are prepared to work on the project after the decision to go ahead is made. Ideally, these people continue leading the project all the way through, so that you have continuity in project management.
- *Project management office (PMO) decision makers*: These individuals are also involved in planning because a new project may change or expand the company's portfolio.
- *Business analyst*: After requirements collection starts, the business analyst has much to do. Usually, initial requirements are gathered when the business need arises, but the real work often begins after portfolio rationalization. A business analyst is responsible for analyzing the business needs and requirements of the stakeholders to help identify business problems and propose solutions. Within the system's development life cycle, the business analyst typically performs a collaborative function between the business side of an enterprise and the providers of services to the enterprise.
- *Architect*: The architect draws an initial picture of the solution. In brief the architect draws the blueprint of the system, and the system designers or engineers

use this blueprint. The blueprint includes the level of freedom necessary in the system: scalability, hardware replacement, new user interfaces, and so on. The architect must consider all these issues.

- *User experience (UX) design team*: UX design should be a core deliverable, not something you leave to the developers to handle. Unfortunately, this design often overlooked; it should be given more consideration. It's important to have close collaboration between the UX team (which could be just one person) and the development team. The best solution is to have a UX expert on the development team throughout the project, but sometimes that isn't possible. The UX design is important in making sure users can perceive the value of the system. You can write the best business logic in the world, but if the UX is designed poorly, users probably won't think the system is any good.
- *Database administrators (DBAs)*: Almost every business system or application uses a database in some way. DBAs can make your databases run like lightning, with good uptime, so it's essential to use their expertise in any project involving a database. Be nice to them; they can give you lots of tips about how to make a smarter system. Alas, for DBAs, developers handle this work more and more frequently. This means developers are expected to have vertical knowledge and not just focus on coding.

- *Developers*: "Developers, developers, developers!" as Microsoft Chief Executive Officer (CEO) Steve Ballmer shouted in a famous video. And who can blame him? These are the people who work their magic to realize the system by using the architecture blueprint drawn from the requirements. Moreover, developers modify or extend the code when CRs come in.
- *Testers*: I'd rather not see testing as a separate activity. Don't get me wrong: It's a role, but testing is something you should consider from the first time you write down a requirement and should continue doing during the whole process. Testers and test managers help to secure quality, but modern development practices include testing by developers as well. For instance, in test-driven development (TDD), developers write tests that can be automated and run at build time or as part of checking in to version control.
- *Operations and maintenance staff*: When an application or system is finished, it's handed over to operations. The Operations staff takes care of it until it retires, often with the help of the original developers, who come in to do bug fixes and new upgrades. Don't forget to involve these people early in the process, at the point when the initial architecture is considered, and keep them involved with the project until everything is done. They can provide great input about what can and can't be done within the company infrastructure. So, Operations is just one part—although an important one—of ALM. In Chapter 3, this book talks about DevOps, which is a practice that ties developers and Operations personnel more closely.

All project efforts are done collaboratively. No role can act separately from the others if you're to succeed with any project. It's essential for everybody involved to have a collaborative mind-set and to have the business value as their primary focus at every phase of the project.

If you're part of an Agile development process, such as a Scrum project, you might have only three roles: product owner, Scrum master, and team members. This doesn't mean the roles just described don't apply though! They're all essential in most projects; it's just that, in an Agile project, you may not be labeled a developer or an architect. Rather, you're a team member, and you and your comembers share responsibility for the work you're doing. We go deeper into the Agile world later in Chapter 4.

Four Ways of Looking at ALM

ALM is the glue that ties together the roles just discussed and the activities they perform. Let's consider four ways of looking at ALM (Figure 1-2). We've chosen these four because we've seen this separation in many of the organizations with which we've worked or individuals to whom we've spoken:

1. *SDLC view*: The SDLC view is perhaps the most common way of looking at ALM, because development has "owned" management of the application life cycle for a long time. This could be an effect of the gap between the business side and the IT side in most organizations, and IT has taken the lead.

2. *Service management or operations view*: Operations has also been (in our experience) unfortunately separated from IT development. This has resulted in Operations having its own view of ALM and the problems that can occur in this area.

3. *Application portfolio management (APM) view*: Again, perhaps because of the gap between business and IT, we've seen many organizations with a portfolio ALM strategy in which IT development is only one small part. From a business viewpoint, the focus has been on how to handle the portfolio, not on the entire ALM process.

4. *Unified view*: Fortunately, some organizations focus on the entire ALM process by including all three of the preceding views. This is the only way to take control of, and optimize, ALM. For a chief information officer (CIO), it's essential to have this view all the time; otherwise, things can get out of hand easily.

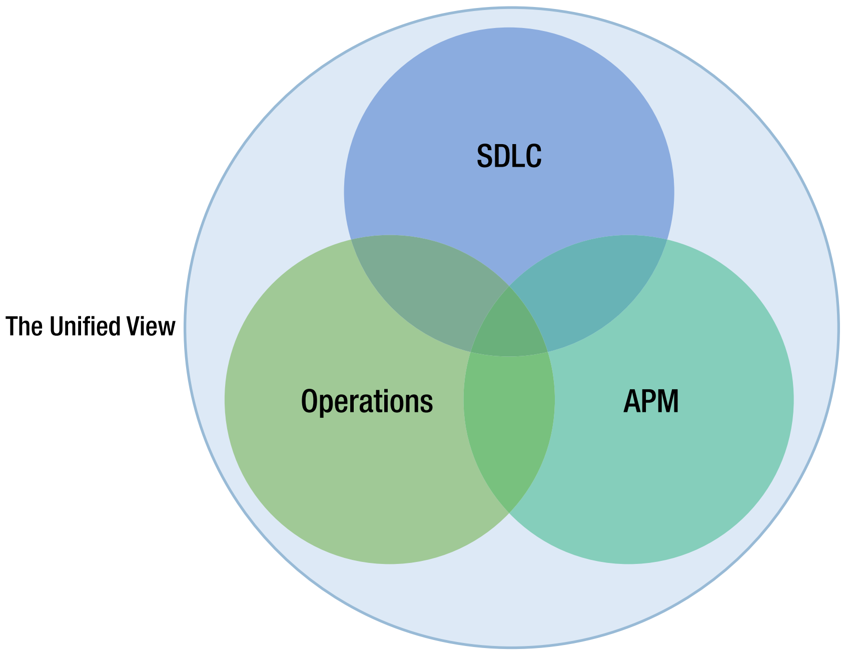

Figure 1-2. *The four ways of looking at ALM*

Let's look at these four views in more detail, starting with the SDLC view.

The SDLC View

Let's consider ALM from an SDLC perspective first. In Figure 1-3, you can see the different phases of a typical development project and the roles most frequently involved. Keep in mind that this is a simplified view for the sake of this discussion. We've also tried to fit in the different roles from the ALM process presented earlier.

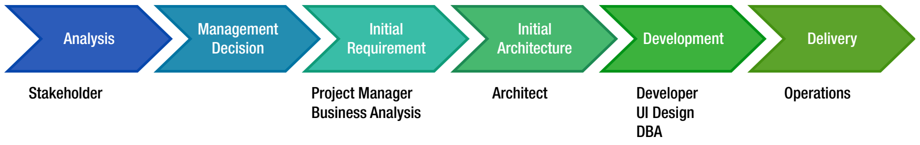

Figure 1-3. *A simplified view of a typical development project*

First, somebody comes up with an idea based on an analysis of business needs: "Hey, wouldn't it be great if we had a system that could help us do [whatever the idea is]?" It can also be the other way around: The idea comes first, and the business value is evaluated based on the idea.

An analysis or feasibility study is performed, costs are estimated, and (hopefully) a decision is made by IT and business management to start an IT project. A project manager (PM) is selected to be responsible for the project and begins gathering requirements with the help of business analysts, PMO decision makers, and users or others affected. The PM also starts planning the project in as much detail as possible at this moment.

When that is done, the architect begins looking at how to realize the new system, and the initial design is chosen. The initial design is evaluated and updated based on what happens during the project and how requirements change throughout the project. Development beings, including work performed by developers, user interface (UI) designers, and DBAs (and anyone else not mentioned here but important for the project).

Testing is, at least for us, something done all along the way—from requirements specification to delivered code—so it doesn't get a separate box in Figure 1-3. We include acceptance testing by end users or stakeholders in the Development box. After the system has gone through acceptance testing, it's delivered to Operations for use in the organization. Of course, the process doesn't end there. This cycle is generally repeated over and over as new versions are rolled out and bug fixes are implemented.

What ALM does in this development process is support the coordination of all development life cycle activities by doing the following:

- Makes sure you have processes that span these activities.
- Manages the relationships between development project artifacts used or produced by these activities (in other words, provides traceability). These artifacts include UI mockups done during requirements gathering, source code, executable code, build scripts, test plans, and so on.
- Reports on progress of the development effort as a whole so you have transparency for everyone regarding project advancement.

As you can see, ALM doesn't support a specific activity. Its purpose is to keep all activities in sync. It does this so you can focus on delivering systems that meet the needs and requirements of the business. By having an ALM process that helps you synchronize your development activities, you can determine more easily whether an activity is underperforming and thus take corrective actions.

The Service Management or Operations View

From a Service Management or Operations view, you can look at ALM as a process that focuses on the activities that are involved with the development, operation, support, and optimization of an application so that it meets the service level that has been defined for it.

When you see ALM from this perspective, it focuses on the life of an application or system in a production environment. If, in the SDLC view, the development life cycle starts with the decision to go ahead with a project, here it starts with deployment into the production environment. Once deployed, the application is controlled by the Operations crew. Bug fixes and CRs are handled by them, and they also pat it on its back to make it feel good and run smoothly.

This is a healthy way of looking at ALM in our opinion: Development and Operations are two pieces of ALM, cooperating to manage the entire ALM process. You should consider both pieces from the beginning when planning a development project; you can't have one without the other.

The APM View

In the APM view of ALM, you see the application as a product managed as part of a portfolio of products. APM is a subset of project portfolio management (PPM).[1] Figure 1-4 illustrates this process.

This view comes from the Project Management Institute (PMI). Managing resources and the projects on which they work is very important for any organization. In Figure 1-4, you can see that the product life cycle starts with a business plan—the product is an application or system that

[1]The Project Management Institute (PMI) is the world's leading not-for-profit professional membership association for the project, program, and portfolio management profession. Read more at `www.pmi.org`.

is one part of the business plan. An idea for an application is turned into a project and is carried out through the project phases until it's turned over to Operations as a finished product.

When business requirements change or a new release (an upgrade, in Figure 1-4) is required for some other reason, the project life cycle starts again, and a new release is handed over to Operations. After a while (maybe years), the system or application is discarded (this is called *divestment*, which is the opposite of investment). This view doesn't speak specifically about the operations part or the development part of the process but should instead be seen in the light of APM.

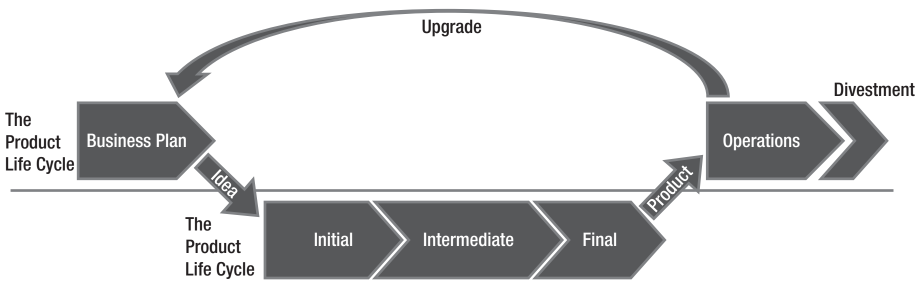

Figure 1-4. *The APM view of ALM*

The Unified View

Finally, there is a unified view of ALM. In this case, an effort is made to align the previous views with the business. Here you do as the CIO would do: You focus on business needs, not on separate views. You do this to improve the capacity and agility of a project from beginning to end. Figure 1-5 shows an overview of the unified ALM view of a business.

You probably recognize this figure from Figure 1-1. We want to stress that with the unified view, you need to consider all aspects—from the birth to the death of an application or a system—hence the curved arrow that indicates continuous examination of an application or system and how it benefits the business

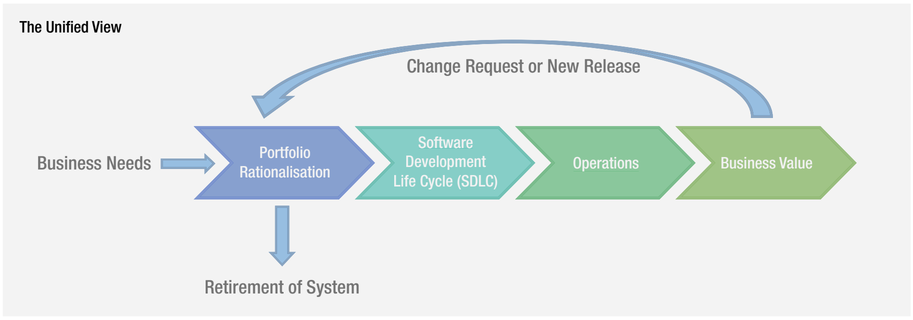
The Unified View
Change Request or New Release
Business Needs
Portfolio Rationalisation
Software Development Life Cycle (SDLC)
Operations
Business Value
Retirement of System

Figure 1-5. *The unified view takes into consideration all three previously mentioned views*

Three Pillars of Traditional ALM

Let's now look at some important pillars of ALM that are independent of the view you might have (Figure 1-6). These pillars were first introduced by Forrester Research.[2]

[2]Dave West, "The Time Is Right For ALM 2.0+," Forrester Research, www.forrester.com/The+Time+Is+Right+For+ALM+20/fulltext/-/E-RES56832?objectid=RES56832, October 19, 2010.

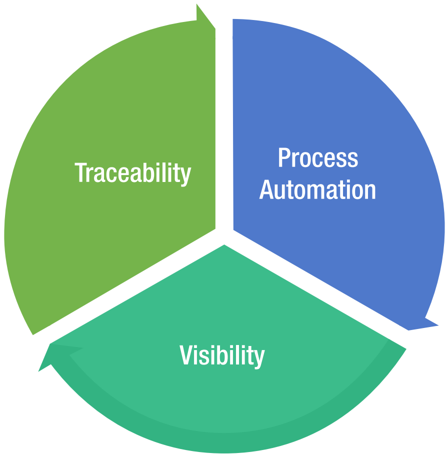

Figure 1-6. *The three pillars of ALM*

In the following sections, we examine these pillars in greater detail, starting with traceability.

Traceability

Some customers we've seen have stopped doing upgrades on systems running in production because their companies had poor or no traceability in their systems. For these customers, it was far too expensive to do upgrades because of the unexpected effects even a small change could have. The companies had no way of knowing which original requirements were implemented where in the applications. The effect was that a small change in one part of the code might affect another part, which would come as a surprise because poor traceability meant they had no way of seeing the code connection in advance. One customer claimed (as we've heard in discussions with many other customers) that traceability can be a major cost driver in any enterprise if not done correctly.

There must be a way to trace requirements all the way to delivered code—through architect models, design models, build scripts, unit tests, test cases, and so on—not only to make it easier to go back into the system when implementing bug fixes, but also to demonstrate that the system has delivered the things the business wants.

You also need traceability to achieve internal as well as external compliance with rules and regulations. If you develop applications for the medical industry, for example, you must comply with Food and Drug Administration (FDA) regulations. You also need traceability when CRs come in so you know where you updated the system and in which version you performed the update.

Automation of High-Level Processes

The next pillar of ALM is automation of high-level processes. All organizations have processes. For example, approval processes control handoffs between the analysis and design or build steps, or between deployment and testing. Much of this is done manually in many projects, and ALM stresses the importance of automating these tasks for a more effective and less time-consuming process. Having an automated process also decreases the error rate compared to handling the process manually.

Visibility into the Progress of Development Efforts

The third and last pillar of ALM is providing visibility into the progress of development efforts. Many managers and stakeholders have limited visibility into the progress of development projects. The visibility they have often comes from steering group meetings, during which the PM reviews the current situation. Some would argue that this limitation is good; but, if you want an effective process, you must ensure visibility.

Other interest groups, such as project members, also have limited visibility of the entire project despite being part of the project. This is often a result of the fact that reporting is difficult and can involve a lot of manual work. Daily status reports take too much time and effort to produce, especially when you have information in many repositories.

A Brief History of ALM Tools and Concepts

You can resolve the three pillars of ALM manually if you want to, without using tools or automation. (ALM isn't a new process description, even though Microsoft, IBM, Hewlett-Packard (HP), Atlassian, and the other big software houses are pushing ALM to drive sales of their respective ALM solutions.) You can, for instance, continue to use Excel spreadsheets or, like one of our most dedicated Agile colleagues, use sticky notes and a pad of paper to track requirements through use cases/scenarios, test cases, code, build, and so on, to delivered code. It works, but this process takes a lot of time and requires much manual effort. With constant pressure to keep costs down, you need to make the tracking of requirements more effective.

Of course, project members can simplify the process by keeping reporting to the bare minimum. With a good tool or set of tools, you can cut time (and thus costs) and effort, and still get the required traceability you want in your projects. The same goes for reporting and other activities. Tools can, in our opinion, help you be more effective and also help you automate much of the ALM process into the tools.

Having the process built directly into your tools helps prevent the people involved from missing important steps by simplifying things. For instance, the Agile friend we mentioned could definitely gain much from this, and he is looking into Microsoft's TFS to determine how that set of tools can help him and his teams be more productive. Process automation and the use of tools to support and simplify daily jobs are great, because they can keep you from making unnecessary mistakes.

Serena Software Inc. is one supplier of ALM tools, and the company has interesting insight into ALM and related concepts. According to Serena Software, there are eight ALM concepts[3]:

[3]Kelly A. Shaw, "Application Lifecycle Management for the Enterprise," Serena Software Inc., www.serena.com/docs/repository/company/serena_alm_2.0_for_t.pdf, April 2007.