

**David Chappell**

May 2011

# ADOPTING A COMMON ALM FOUNDATION

## WHY IT MAKES SENSE IN A HETEROGENEOUS WORLD



**DavidChappell**  
& Associates

**Sponsored by Microsoft Corporation**

Copyright © 2011 Chappell & Associates

Building software is complex. Most significant projects rely on a diverse team of people, including developers, graphic designers, business analysts, and more, all of whom who must work well together. The different tools these people use add to the complexity. Developers use various integrated development environments (IDEs), for example, designers choose their favorite graphics tools, and business analysts feel most at home with spreadsheets. Different projects also commonly adopt different solutions for version control, test management, and other purposes. Diverse tools are an inescapable fact of life in most development organizations today.

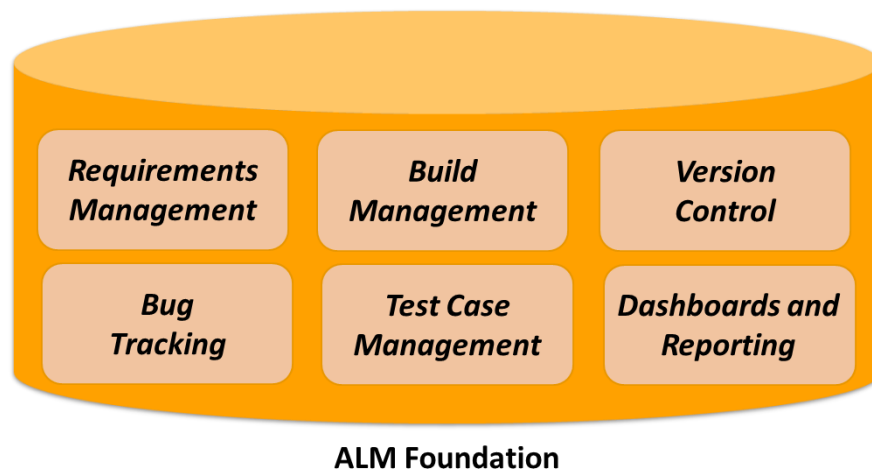
Sometimes, this diversity makes sense. People in different roles certainly need different tools—forcing a business analyst to use an IDE wouldn't make sense—and even people in the same role might make different choices. A developer creating a Java application probably wouldn't be happy using the Visual Studio IDE, for example, just as one building a .NET application wouldn't typically choose Eclipse. Sometimes, though, using different tools adds no value—it just makes life more complex. Especially in an agile environment, where effectively sharing information among everybody on a team is essential, needlessly diverse tools hinder the development process.

Mandating a common set of tools for all of the participants in every development project isn't realistic for most organizations, at least in the short run. Yet it's still possible to provide consistent support for the development process as a whole by adopting a common foundation for application lifecycle management (ALM). This common foundation can connect the tools people are already using while helping an organization move toward less diversity over time. A common ALM foundation also makes sure that everybody has a consistent way to share information about the project. Both of these improvements can make the development process less complex, less risky, and faster.

## What is an ALM Foundation?

---

Understanding this approach requires looking first at the services an ALM foundation provides. Figure 1 shows typical examples.



**Figure 1: A common ALM foundation provides a fundamental set of services for the development process.**

Today's most visible choices for a common ALM foundation include Microsoft Team Foundation Server (TFS), part of Visual Studio 2010, and IBM Rational Team Concert, which is based on Jazz Team Server. Both provide a unified

set of tools for version control, build management, test case management, reporting, and other functions. Even in organizations that use diverse tools, building on this common base can make sense. Here's why:

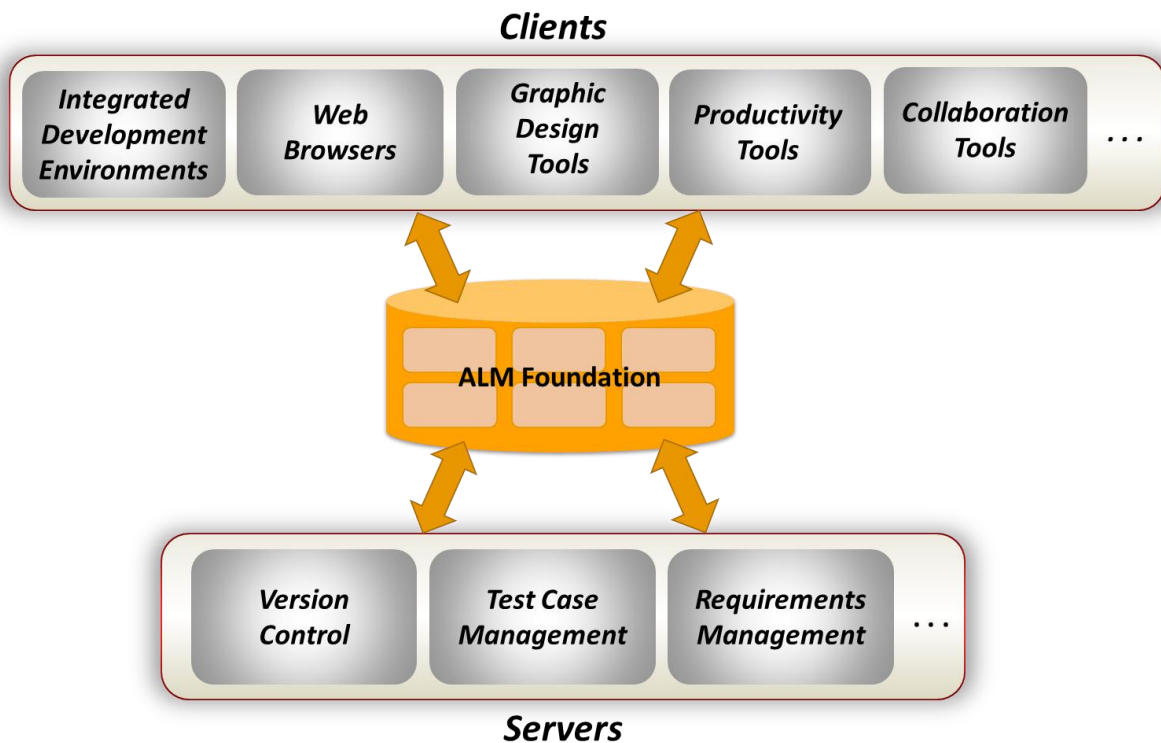
- A common ALM foundation allows storing data about all of an organization's development projects in one place. Because it provides common reporting, developers, ScrumMasters, product owners, and other stakeholders can get a consistent view of this data across all current projects, even for teams using different IDEs and other tools.
- Using the same technologies for version control, test case management, and other aspects of ALM means that everybody on every development team understands these tools. This makes it easier to move people across projects, and it avoids creating camps of competing technology enthusiasts, each wedded to a different solution.
- Deploying a common ALM foundation simplifies the effort and cost of acquiring and managing diverse tools. For example, rather than deploying several different version control systems, it's possible over time to standardize on just one.
- Using a common ALM foundation makes it easier to share code and other development artifacts across different development projects. Even projects targeting different platforms can benefit from storing database schemas, graphics, and other information in a common place, making this information accessible to everybody.
- A shared ALM foundation helps implement the same development process across all teams in an organization. Since everybody relies on the same underlying mechanisms, using a common process gets easier. This simplifies management, and it also allows improvements made by any group to spread quickly to all of the others.

The overarching goal of software development is to produce the right software on time with the lowest possible risk. Using a common ALM foundation can help do this.

## Connecting to a Common ALM Foundation

---

Relying on a common ALM foundation is a worthwhile goal, but the real problem for most organizations today is to begin this journey. Doing this requires meeting people where they are, supporting whatever development tools are already in use. The implication is clear: To be successful, an ALM foundation must fit well into an organization's existing environment. Figure 2 illustrates this idea.



**Figure 2: A common ALM foundation connects all of the software used in the development process.**

As the figure shows, software that needs to connect with an ALM foundation fits into several different categories. For clients, those categories include:

- ❑ Integrated development environments, such as the Visual Studio IDE and Eclipse. Each of these tools creates different outputs for different platforms, and so each one provides platform-specific support for creating those outputs.
- ❑ Web browsers, used by people who need a straightforward way to access information stored in the ALM foundation. Think of a project’s business sponsor, for example, who might wish to check project status or see current code quality metrics. Someone in this role is unlikely to install an IDE, but she still needs a way to interact with the ALM foundation.
- ❑ Graphic design tools, such as Microsoft Expression Studio and others. Any application with a good user interface probably relies on a graphic designer, and graphic designers rely on specialized tools.
- ❑ Productivity tools, such as spreadsheets. Plenty of people feel most at home inside a spreadsheet, and so letting tools like this access the ALM foundation is important. A business analyst might choose to monitor and update requirements using Microsoft Excel, for example.
- ❑ Collaboration tools, such as Microsoft SharePoint. Many organizations use these tools in a variety of ways, which means that connecting them to an ALM foundation is important.

This list isn't exhaustive. Other components, such as tools for defining requirements, might also be important. The development process involves more than just writing code, and so a variety of software needs to connect to the ALM foundation.

Along with diverse clients, different servers also need to be connected. Examples include:

- Version control systems, which provide software configuration management. When an organization adopts a common ALM foundation, it probably still needs to maintain existing source code—at least for a while—in other version control systems. Connecting these existing systems to the ALM foundation, either for synchronization or migration, is a fundamental part of adopting this common approach.
- Test case management, another area where organizations might have different options installed today. As with version control, these must be connected to the ALM foundation in an effective way.
- Requirements management, where existing software must once again be connected with a new ALM foundation.

Making all of these parts work well together is essential to providing good support for the development process. Reports and dashboards, for example, should be able to display information created by any component, while developers using any IDE should be able to rely on tests, requirements, and other information no matter where they're stored. Business analysts and product owners shouldn't need to use different systems to understand the status of different projects.

*How other software connects to an ALM foundation is important.*

Connecting diverse tools to a common ALM foundation is clearly required. But how should these connections be implemented? How other software connects to an ALM foundation is important, and so it's worth devoting some time to thinking about this question.

One approach is to define multi-vendor standards that can be used with different ALM foundations from different vendors. This could potentially let anybody creating a development tool connect to any ALM foundation in a common way. Yet while this might be an appealing idea, there are significant roadblocks standing in the way. They include the following:

- Vendors will support interoperability standards only when they believe that doing so is in their own interest. Yet just a handful of companies provide full-featured ALM foundations. Given this, each of them likely wants to define its own approach for connecting to other tools. The probability of creating true multi-vendor standards in this area is vanishingly small.
- Since they must work across diverse products, multi-vendor standards commonly provide lowest-common-denominator connections. Even though ALM foundations provide similar functions, they do it in different ways. Creating multi-vendor standards for connections to these diverse technologies is likely to result in reduced functionality across all of them.

The alternative to multi-vendor standards is for each ALM foundation to provide its own approach to connecting other tools. This allows full-fidelity interactions, and since the number of tools that must connect to an ALM foundation is relatively small—it's dozens, not hundreds—the total effort required is manageable.

For example, Microsoft provides several options for connecting clients to its ALM foundation. The Visual Studio IDE connects to Team Foundation Server through a plug-in called Team Explorer, while a similar plug-in called Team Explorer Everywhere is available for Eclipse. Other client software can access a group of redistributable .NET and Java APIs provided by TFS. Microsoft itself relies on these APIs to connect components such as Microsoft Excel and

*The probability of creating true multi-vendor standards in this area is vanishingly small.*

the Expression tools for graphical designers, but other organizations can also use them. For example, Tasktop used the Java APIs to implement an Eclipse Mylyn plug-in, providing a task-oriented interface to TFS. Similarly, Ekobit connects Microsoft Outlook to TFS via these APIs.

For servers, Microsoft provides the TFS Integration Platform. Intended as a general-purpose solution, it includes adapters for several existing technologies, including IBM Rational ClearCase, Subversion, and older versions of TFS. Other vendors have provided adapters for other ALM software, such as the Quality Center adapter created by Hewlett Packard. These connections are important: They provide a way for an organization to adopt a common ALM foundation while still getting value from their investments in current development tools.

Don't underestimate the importance of connecting an ALM foundation to other software. For an organization embarking on an entirely new project, it might be possible to use a completely new and consistent set of development tools. For most projects, however, this isn't an option—they must integrate with what already exists in their environment. Connecting these to an ALM foundation is unavoidable, and doing it well is essential.

## Conclusion

---

Every development organization of any size is bound to have some diversity in the tools it uses. Yet minimizing this diversity makes sense. Using different tools is a good thing when it provides real value, such as choosing appropriate tools for different roles or adopting different IDEs that target distinct platforms. When it doesn't, however, choosing common solutions is better. A prime example of this is building on a common ALM foundation. And because most development organizations haven't yet done this, adopting this approach can differentiate an organization from its competitors.

Connecting existing tools to a common ALM foundation is also important. While using multi-vendor standards would be appealing, the truth is that they're not on the horizon. Instead, the provider of an ALM foundation can offer its own integration support for the creators of other components.

When the tools used by a development team work together well, the development process—and the team's performance—improves. Building on a common ALM foundation is a sensible step toward this goal.

## About the Author

---

David Chappell is Principal of Chappell & Associates ([www.davidchappell.com](http://www.davidchappell.com)) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technologies.