

David Chappell

December 2011

WHAT IS AN APPLICATION PLATFORM?

Sponsored by Microsoft Corporation

Copyright © 2011 Chappell & Associates



DavidChappell
& Associates

Just about every application today relies on other software: operating systems, database management software, even software running in the public cloud. Whatever it does and wherever it runs, all of this software together comprises an *application platform*.

Application platforms play a fundamental role in modern computing environments. Applications and the data they use provide all of the value that information technology brings, and virtually every application depends on an application platform. Since pretty much every organization today relies on applications, there's a clear connection between business value and application platforms¹.

Yet modern application platforms aren't simple. The applications they support run on all kinds of computers, including mobile phones, desktops, on-premises servers, and servers in the public cloud. An effective application platform needs to provide the right set of services on each of these. And different kinds of applications need different things from an application platform. A single-user application running on a phone needs radically different services for execution and storage than does an application that runs in the cloud and supports thousands of simultaneous users. Thinking clearly about all of this diversity requires taking a broad view of application platforms, whatever services they provide and wherever they run.

The goal of this paper is to provide that broad view. We'll start with a general look at the topic, building an abstract model, then end with a specific example: the Microsoft application platform. Before doing any of this, however, we first need to examine the things that every application platform supports: applications themselves.

What is an Application?

By definition, an application platform provides services to applications. But what exactly is an application? This isn't a simple question to answer, and reasonable people can disagree on a precise definition. Still, it's not hard to illustrate typical application styles, and doing this gives us a good sense of the diversity that an application platform must support today. Figure 1 shows the simplest place to start: a standalone application running on a client computer.

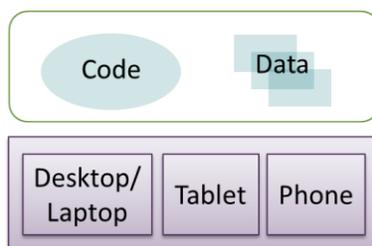


Figure 1: A standalone client application runs on hardware such as desktop computers, tablets, and phones.

Like applications in general, a standalone client application typically includes code and data. This kind of application runs on various kinds of hardware, including desktop computers, laptops, tablets, and mobile phones. Whatever it runs on, the application doesn't need to interact with code or data on any other system; it provides

¹ For a more detailed look at this topic, see *Application Platforms and Business Strategy: Making the Connection*, David Chappell

value all by itself. Some common examples include word processing software, software for reading books and magazines, and many games.

Lots of applications don't run on a single computer, however. Instead, their code (and perhaps their data) is spread across multiple machines. Figure 2 shows a simple example of this kind of distributed application.

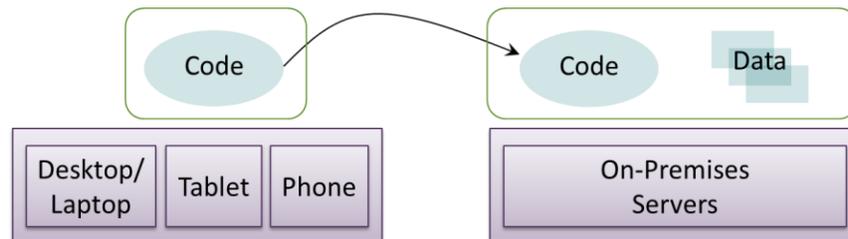


Figure 2: A distributed application includes both client and server components.

In a distributed application, the client once again might run on various kinds of hardware, including PCs, tablets, and phones. This client might be anything from a simple Web browser to complex custom software written specifically for the application. Whatever it is, the client isn't useful unless it's connected to the application's server component. As the figure suggests, this back-end component commonly runs on physical servers in an on-premises data center within an organization. Examples of distributed applications include most of today's business packages, such as ERP systems, as well as many custom applications.

Another important option for creating applications today is to place the code and data for a distributed application on computers in a data center owned by a public cloud provider. Figure 3 shows how this looks.

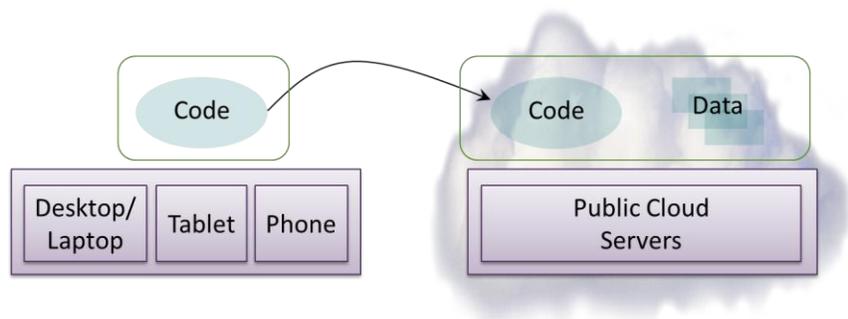


Figure 3: The server component of a distributed application can run in the public cloud.

This picture is nearly identical to the previous diagram—the only difference is the presence of the cloud—so why bother showing it separately? The reason is that the application platform components used with servers in the cloud are often different from those used with servers in on-premises data centers. Viewing public clouds as a distinct entity in their own right is important for understanding modern application platforms.

Yet running code on servers in remote datacenters isn't the only way that applications can use the public cloud. It's also common today for applications running either standalone on clients or as part of a distributed application to use cloud services. Figure 4 illustrates the first of these options.

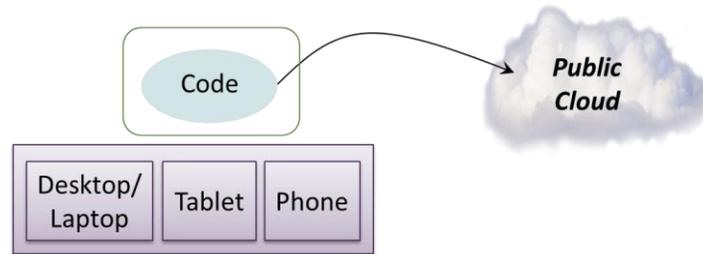


Figure 4: A client application might use one or more services provided from the public cloud.

Here, a standalone client relies on a service provided from the public cloud. This service might store the application’s data, provide maps to show its user, help it connect to other applications, or do something else. Whatever it does, this cloud service is certainly part of the application platform—it provides a service that the application relies on while it executes. From the point of view of the developer, services in the cloud just provide more options to use when designing and building an application.

The line between a standalone client application and a distributed application can be blurry. Many clients have value on their own, but become more useful when using services in the public cloud or acting as the client for a distributed application. For example, a standalone reader application might also allow buying new books through a public cloud service, or a phone app for an ERP system might act as a client for the full distributed ERP application, then let its user work with cached data when it’s not connected. The key point is that because modern applications need to do many different things, an application platform should support all of these options.

Describing Application Platforms: An Abstract View

Applications come in several different styles, as we’ve just seen, and an application platform should support all of them. This is a tall order: Diverse services are required, and they must be provided in various ways. One way to understand this complexity is first to group together the kinds of services an application platform provides, then look at the different contexts in which those groups of services are provided. Figure 5 illustrates the fundamental service groups of a modern application platform.

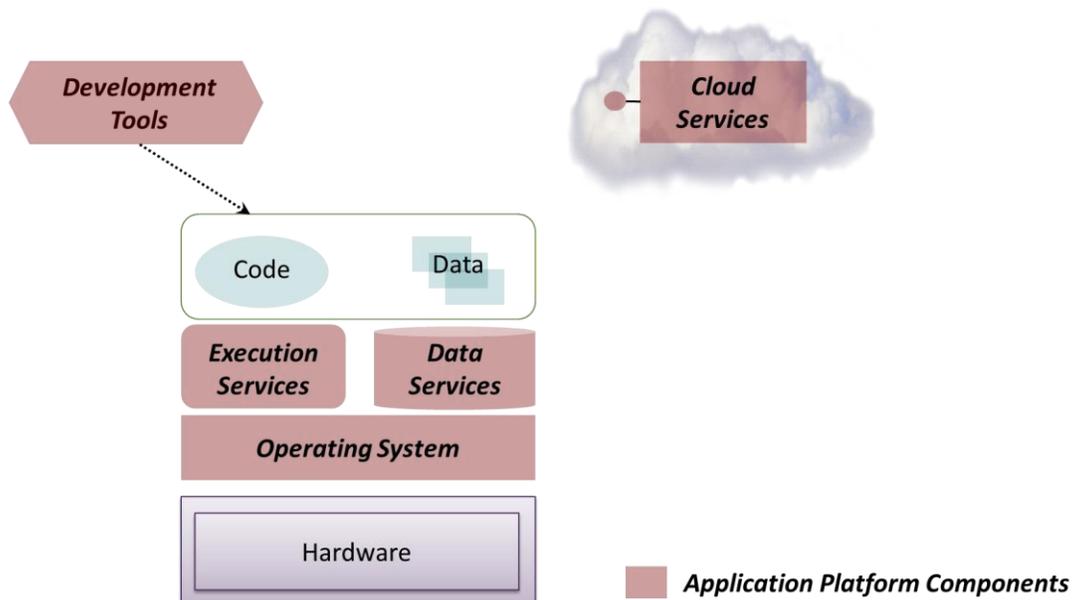


Figure 5: An application platform has five different components, each providing a specific type of services for applications and the people who create them.

The services provided by an application platform can be grouped into five categories:

- An *operating system*, implementing the foundation services on which all applications depend. These include basic storage, such as a file system, and the fundamentals required to run code, such as scheduling.
- *Execution services*, providing libraries and more for running software. This large category includes support for creating user interfaces on clients, communicating with other software, structuring how code executes (e.g., with workflows), and many other things.
- *Data services*, which let applications store and process data. The most important technology in this category today is a database management system (DBMS), but other data services are becoming increasingly important. Streamed data, such as data used for complex event processing or real-time voice, can be useful in many scenarios. The rise of big data, i.e., very large amounts of unstructured information, also requires technologies that go beyond traditional relational database systems.
- *Cloud services*, offering remotely provided functionality that applications can use. Example cloud services today provide information, such as maps, or let applications do things such as search the Internet or connect with other applications.
- *Development tools*, helping development teams create and maintain applications. These tools range from simple code editors to full-featured tool families with support for writing code, testing, deployment, and other aspects of the development process.

Not every application uses all of these components, but a broadly targeted application platform offers all five. More than this, it needs to provide these components across different kinds of hardware. Figure 6 shows how this looks, giving a general picture of a modern application platform.

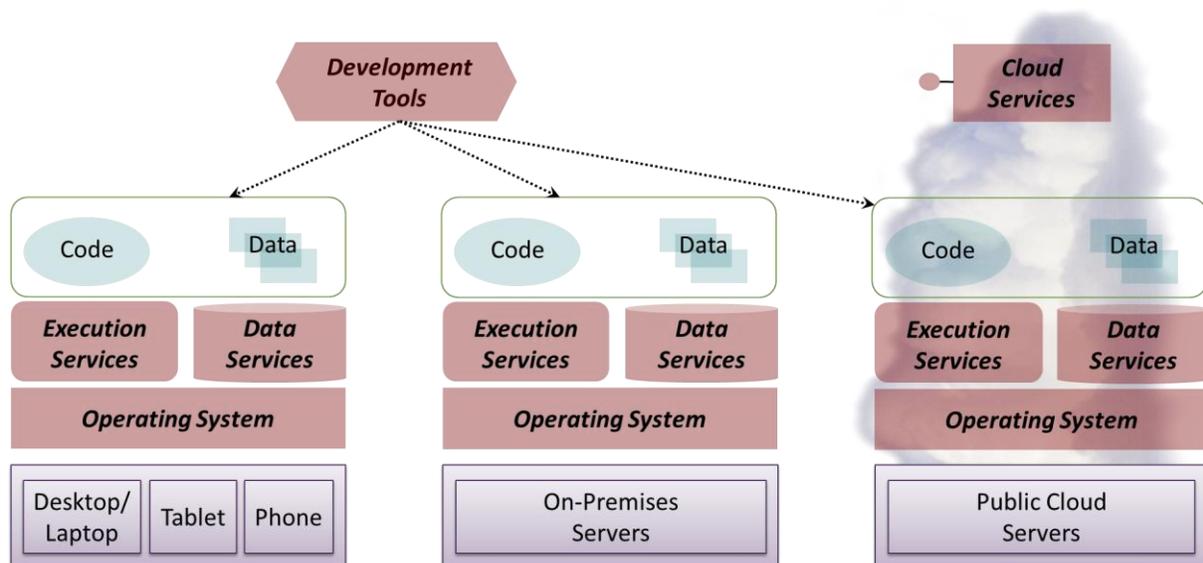


Figure 6: An application platform can offer all five components across multiple hardware options, including clients, on-premises servers, and public cloud servers.

As this diagram suggests, a modern application platform is broad, supporting all kinds of applications. This includes standalone clients, distributed applications with server logic running on-premises or in the public cloud, and applications that use cloud services. It's also consistent, letting developers use the same tools and skills to create a range of applications that run across various kinds of hardware.

Most developers—most organizations, even—don't create all of the application styles that today's application platforms can support. Still, having a general model of the foundation that applications use is useful, if only because it helps us understand what's possible.

An Example: The Microsoft Application Platform

Abstract models are nice, but concrete examples are usually the best way to understand something. The Microsoft application platform provides one illustration of a set of technologies that can be viewed through the lens just described. Figure 7 shows how various parts of the Microsoft platform fit into this model.

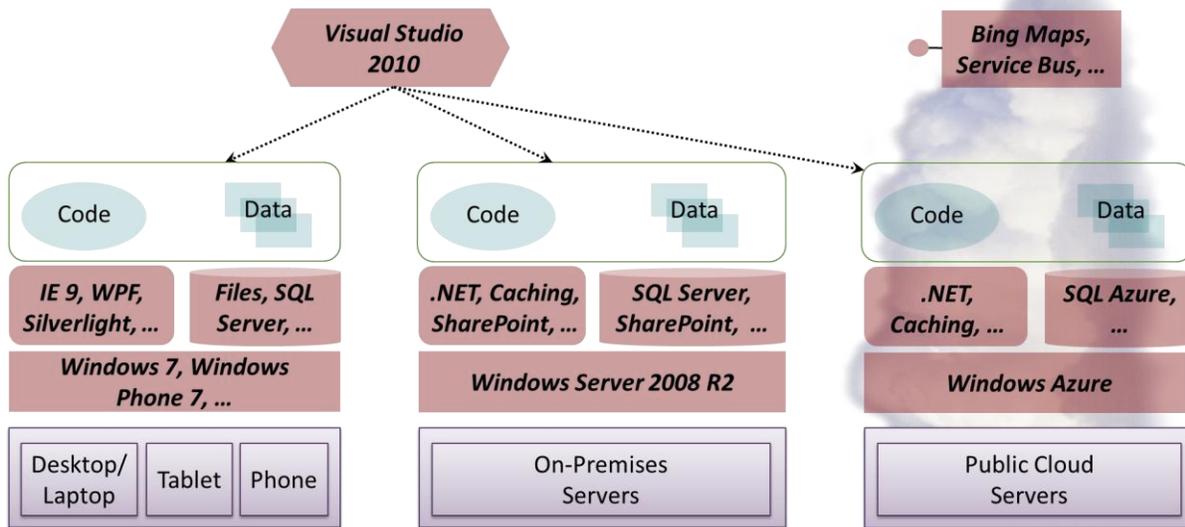


Figure 7: The Microsoft application platform provides a range of services for different kinds of applications, including the examples shown here.

To understand this broad set of technology, it's worth walking through each of the five kinds of application platform components described earlier, looking at examples of what the Microsoft platform provides in each area.

For *operating systems*, the Microsoft application platform includes options for each of the three hardware categories:

- For clients, the choices today include Windows 7 for desktops and laptops and Windows Phone 7 for mobile phones.
- For on-premises servers, the platform provides Windows Server 2008 R2.
- For public cloud servers, the underlying operating system is Windows Azure.

For *execution services*, the options provide some commonality across each of the hardware platforms, but they also differ where required. Those options include:

- For clients, all of the choices support Internet Explorer 9, Microsoft's Web browser, and all also provide at least some parts of the .NET Framework. The technologies for creating native user interfaces vary, however. On Windows 7, an application might use Windows Presentation Foundation (WPF), for instance, while Windows Phone 7 uses Silverlight. Microsoft also supports user interfaces designed for tablets, including touch-based interfaces.
- For on-premises servers, Windows Server 2008 R2 includes the .NET Framework, which provides a large set of supporting services for applications. These include Windows Communication Foundation (WCF) for service-oriented communication, Windows Workflow Foundation (WF) for creating workflow-based applications, and others. Another example is the system's distributed caching service, which helps speed up applications by

storing frequently accessed data in memory. Windows Server 2008 R2 also supports SharePoint, which allows creating applications for collaboration and more, along with other services that application developers can use.

- For public cloud servers, Windows Azure provides the .NET Framework together with a broader set of services. These include a distributed caching service like the one in Windows Server 2008 R2 and built-in monitoring and load balancing for virtual machines.

For *data services*, the Microsoft application platform again offers different technologies for different environments:

- For clients, the options include local storage, such as the Windows file system, along with versions of SQL Server.
- For on-premises servers, the platform provides the full version of SQL Server, including business intelligence offerings such as SQL Server Analysis Services and SQL Server Integration Services. Applications can also use higher-level storage options built on SQL Server, such as SharePoint lists. For working with streaming data, SQL Server provides a technology called StreamInsight, along with programming interfaces to Lync for accessing real-time communications data. And Microsoft has announced plans to support Hadoop for processing large amounts of unstructured data.
- For public cloud servers, the platform includes SQL Azure, a cloud version of SQL Server. Windows Azure also provides scalable, non-relational table storage for structured data and blobs for binary data. When it's available, Hadoop will also run on Windows Azure.

The *cloud services* in the Microsoft application platform address a number of problems that application developers might face. Bing Maps exposes programmatic interfaces that let applications access maps and other geographic information, for example, while Service Bus provides a cloud-hosted intermediary for connecting different parts of an application. Similarly, Microsoft's Access Control service offers cloud-hosted identity services.

The *development tools* in the platform are largely provided as part of Visual Studio 2010. This technology family includes an integrated development environment (IDE), along with tools for manual and automated testing, architectural design, and other aspects of creating applications.

All of these technologies are part of the Microsoft application platform. This isn't an exhaustive list, however—it's possible to take a broader view. For instance, Microsoft Dynamics CRM includes a component called the *xRM framework* that's designed to help developers create a specific kind of business application. Deciding exactly what should be included in any vendor's application platform is an inexact science, but taking a comprehensive view of this area is useful. It helps us remember the wide range of services available for creating and running applications.

Conclusion

Twenty five years ago, application platforms were simple: an IBM mainframe offered CICS while a desktop PC provided a basic operating system. Today, both the hardware and software environments have expanded significantly, and developers have many more options. This requires them to understand a more complex world, but it also lets them create a much broader array of useful applications.

Anybody choosing a foundation for new applications should make that choice based on the entire application platform. As mentioned earlier, a modern platform is *broad*, supporting all kinds of applications, and it's *consistent*, letting the same tools and skills be used to create different kinds of applications running on diverse hardware. While understanding the breadth of a modern application platform takes some effort, today's technologies also offer great rewards. Understanding this fundamental part of the IT world is an essential part of building better applications.

About the Author

David Chappell is Principal of Chappell & Associates (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.