

White Paper



Practicing Agile software development on the Windows® Azure™ platform

Amit Dumbre, Sathya Priya Senthil, Sidharth Subhash Ghag

Abstract

For several of us in the software industry, Agile software development methodology is not new. If we observe the software industry, different methodologies have evolved over years to help IT cope-up with rapidly evolving business requirement changes. Today's business is becoming more dynamic and demanding than ever before, and to address this, a new methodology aptly called as Agile evolved. Agile is an iterative approach to developing software, primarily used by companies for software development, to match up with fast and changing client demands.

On the technology front too, technologies have evolved to help enterprises be more dynamic and on-demand. As per Gartner, cloud computing is one of the most disruptive technologies of 2009. Cloud promises to deliver software more economically, on-demand and with lower on-boarding overheads.

When we move towards cloud computing, several questions flash in our minds; can we apply agile to cloud projects? Would cloud development degrade agile software development? How to implement agile software development practices with cloud application development? What could be the advantages, if any? So on and so forth. In this paper we will try to find the answers to these questions. We will attempt to discuss the benefits of applying the Agile methodology to Azure™ application development and the potential challenges encountered and the solutions to addressing the challenges.

This paper will provide insights to architects and managers who are looking for ways to improve their existing software development practices using agile software development practices (Scrum model).

Need to be agile

One of the leading US based health & care Support Company which provides proactive, custom health and care plan for individuals at each phase of life.

To win a market, businesses are transforming their IT investments into a competitive advantage which drives them towards revenue growth and opens new markets and opportunities for businesses. Along with this, they are continuously evolving to adopt new services or change their existing services to meet customer demands.

This forces the businesses to be more demanding and dynamic in nature because of which the application requirements are also changing frequently, even during large application development we may expect some requirement changes from the client side, due to changing business demands. In such a scenario, following the traditional waterfall model, where benefits start realizing only after a long application development cycle, the projects could face the risk of delivering the software a little too late. A potential risk of expectation mismatch, with respect to client requirements and the delivered software, would exist.

Following are some of the main advantages of using agile methodology over traditional models:

- Faster time to market
- Quick ROI
- Shorter release cycles
- Better quality
- Better adaptability and responsiveness to business changing requirements
- Early detection of failure/failing project

A crash course on agile methodology

For those new to Agile, this section provides a quick peek into the agile life-cycle model. There are several Agile programming methodologies such as Scrum, Extreme Programming, Lean software development and TDD (Test Driven Deployment). Here we discuss Scrum, which is one of the most popular ones.

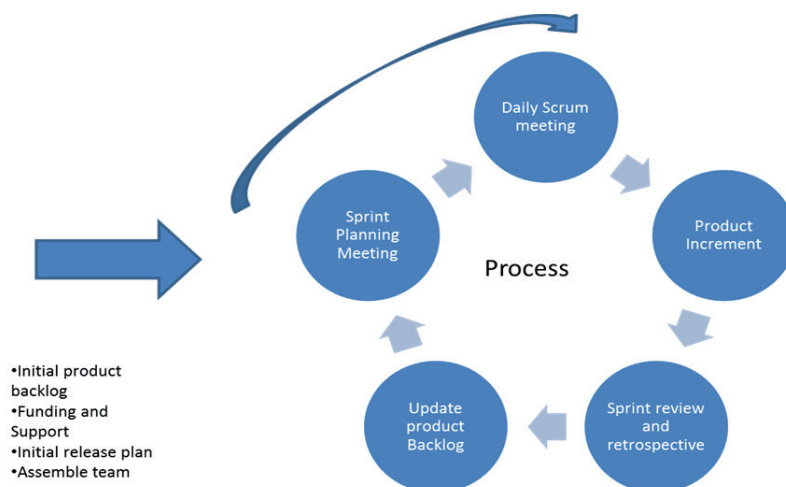


Figure 1: Scrum Methodology

With Scrum, a team comprises of a product owner (very important stakeholder), scrum owner, a scrum master and scrum team members. The product delivery is divided into a number of small tasks and each task is planned to be completed within a short cycle called Sprint. At the end of each working cycle, potentially shippable software is given to the product owner according to the requirements understanding. If the software has any defects then they will be logged in a product backlog and depending on the priority, the defects are fixed and delivered.

Important terminologies:

- **Sprint:** Each software delivery lifecycle of scrum is referred to as sprint and lasts for 2- 4 weeks.
- **Product Backlog:** List of prioritized business requirements desired for the product.
- **Sprint Backlog:** List of System requirements/user stories that the scrum team is committed to for the current sprint.

Important activities of Scrum:

- **Sprint Planning Meeting:** Depending on the highest priority, work will be taken up for the current sprint.
- **Daily Scrum Meeting:** Attended by all, each developer will explain about their previous day's work and the work to be done. This will serve to synchronize the work.
- **Frequent demos to product owner and business stakeholders:** At the end of week there will be demos to product owner/business stakeholders to show that this sprint is in the right track.
- **Sprint Review:** Will be conducted at the end of each sprint and product owners will give feedback about the fresh product delivered. This feedback will result in adding items to product backlog.
- **Sprint Retrospective:** Another meeting at the end of each sprint with all the team members, to identify opportunities to improve on the new sprint.

Challenges faced with agile development today

With Agile models, the business expects that products are delivered incrementally earlier rather than later. And project managers have to plan their delivery cycles in short spans, utmost within 2-4 weeks, to meet those expectations. Some key challenges which the projects face and which possibly hamper product delivery include:

- **High effort and cost involved in infrastructure** - Setting up the infrastructure required for development and testing, leave alone production, is often a repetitive but also a very time consuming activity. Projects spend considerable time and effort in the environment construction phase, often leading to delays.
- **Availability of skilled resources** to setup, manage and certify the software development and deployment infrastructure. For smaller projects (OR small-sized projects), it is often seen that team members are forced to play multiple roles. This may lead to setting up environments which are deficient in meeting the high QoS standards of performance, security, reliability, etc., which are the demands of any production ready environment.
- **Ability to build applications from multiple locations** is another challenge in projects especially with teams that are geographically distributed. Source code and infrastructure has to be always available and accessible from anywhere. Administrators have to authorize network access and restrict access to local resources.

Though both on-premise and Azure™ application validation process will follow similar phases in the testing life cycle, as depicted in the diagram below, the execution of these phases will have few significant differences which we shall highlight in the subsequent sections.

Why Azure™?

Azure™ Services Platform is an application platform on the cloud that provides a wide range of core infrastructure services such as compute and storage, along with building blocks which can be consumed from both on-premises environments and the internet to develop cloud based applications. It offers Platform as a Service (PaaS) capabilities which allows applications to be built, hosted and run from within Microsoft® managed data-centers using programmable API's or interoperable service exposed by the platform.

Here we shall highlight those aspects of Azure™ which can help bridge the challenges faced in building application following Agile today:

- **Setting up infrastructure:**
Infrastructure is readily available on Azure™ cloud. System maintenance and system updates of the cloud server will be taken care of by Microsoft®. Projects can save considerable time and effort spent on setting up the environment.
- **Availability of skilled resources:**
Even a novice can work with deployment. Microsoft® has set up an auto managed cloud infrastructure on high QoS standards of performance, security, reliability, etc. This helps with the project on-boarding processes to be completed quickly and without the need for any skilled or specialized resources.
- **Ability to build applications from multiple locations:**
Cloud addresses this by giving a common production URL. So that anytime, anywhere, any team member can access production/staging applications without any issues. Infrastructure will be available, for example: Developer can go and deploy his/her solution without setting up the deployment environment. Some more benefits of Azure™:

Business users can enjoy:

- Faster 'time to market' their products and services
- Pay per use

Developers benefit from:

- Easy setup and deployment of applications to Go Live
- No maintenance of the infrastructure which runs the development apps
- Access to a global environment
- Leveraging a familiar development environment and being able to use their existing skills on .NET™.

Now looking at the above benefits we will see how to set up a development environment on the Azure™ platform which can help enhance your agile practices.

Setting up an agile environment with Azure™

We have already seen the lifecycle of agile methodology and its important steps. In this section we will focus on how we can carry out agile development for an application which needs to be deployed on Azure™. We will also see in what ways the Azure™ Agile development compares with the traditional on-premise agile application development approach.

The following diagram depicts a typical enterprise software development environment that is set up during the application sprint development. With Agile development one or more teams can work on different sprints (project backlog) simultaneously.

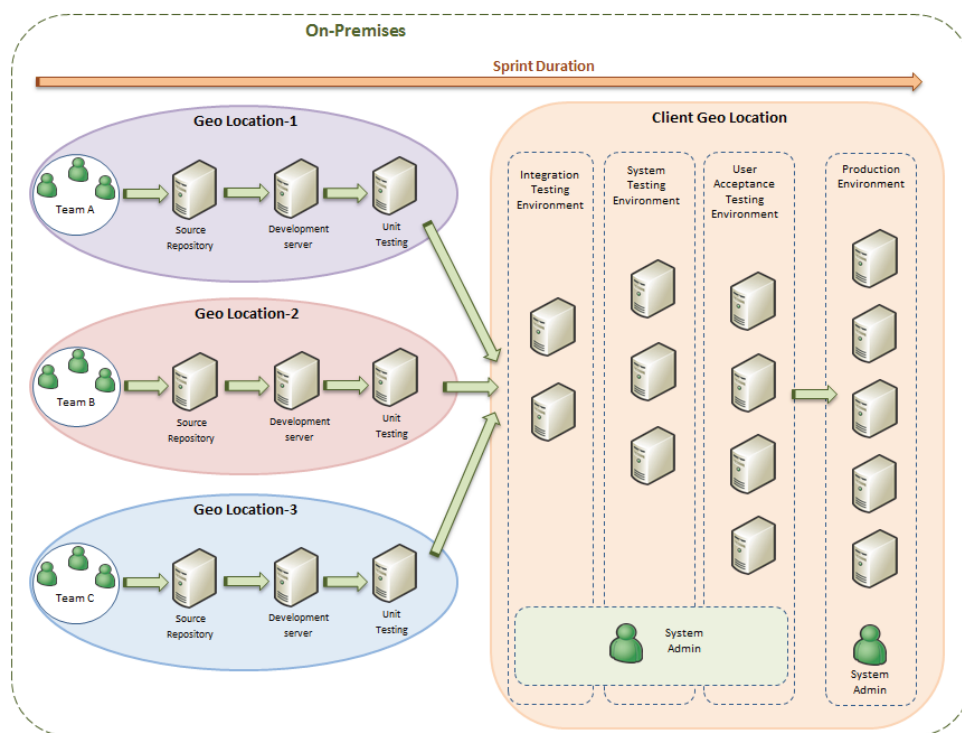


Figure 2: A typical software development environment setup for an on-premise agile development model

As shown in the figure, teams can be spread across multiple geographical locations and each team will have their own separate development environments along with source code repository deployed either at multiple locations, or being available from common source code repository available across various development locations. Sprint deliverables from each team, after unit testing are moved to a different testing environment at client location. If these deliverables pass through all the testing phases such as integration testing, system testing and user acceptance testing (UAT), then it is finally deployed to production environment.

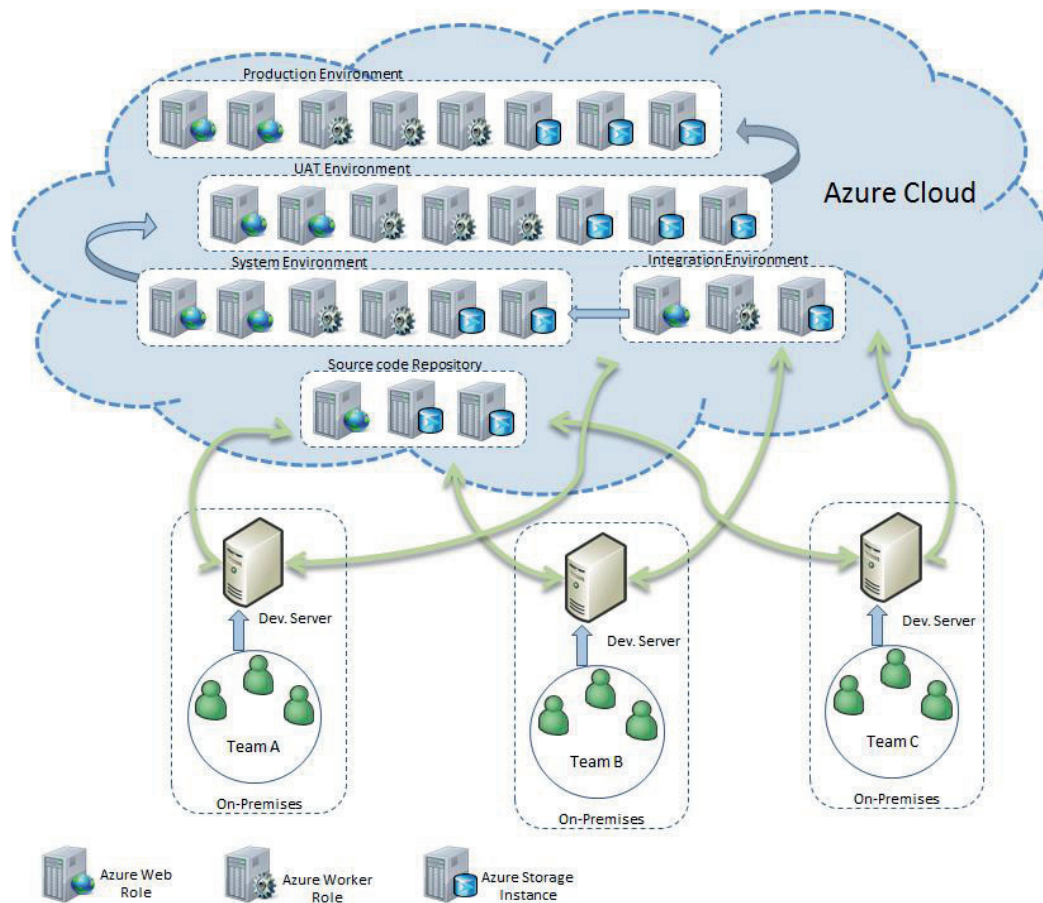


Figure 3: A typical software development environment setup for an Azure based agile development model

The above figure depicts the offloading of various testing and production environments along with the source code repository such as SVN to a more cost effective and on-demand Azure™ cloud platform. While on-premise, one would require to have only the development environment and the necessarily toolsets required to build the unit test applications.

However one need to note that with Azure™, while provisioning a single compute instance to host an application, the user is provisioned with a single production environment and only a single staging environment for users to test their apps before release to production. This would mean that for enterprises to be able to realize the complex development environment, such as depicted in figure 2, on Azure™ would be different and a similar one-to-one mapping from the traditional to cloud deployments on Azure™ cannot be realized as-is.

Also this limitation is imposed only for compute instance and not for storage (including both Azure™ storage such as Table, Blobs & Queues and SQL Azure™). For storage, Azure™ provisions you only a single storage instance which is Table, Blob, Queues or SQL Azure™. One would have to provision a separate storage instances for every environment which would be required as a part of the software development environment.

Here we show you how such a scenario can be addressed in light of meeting enterprise class software development requirements such as that of figure 2.

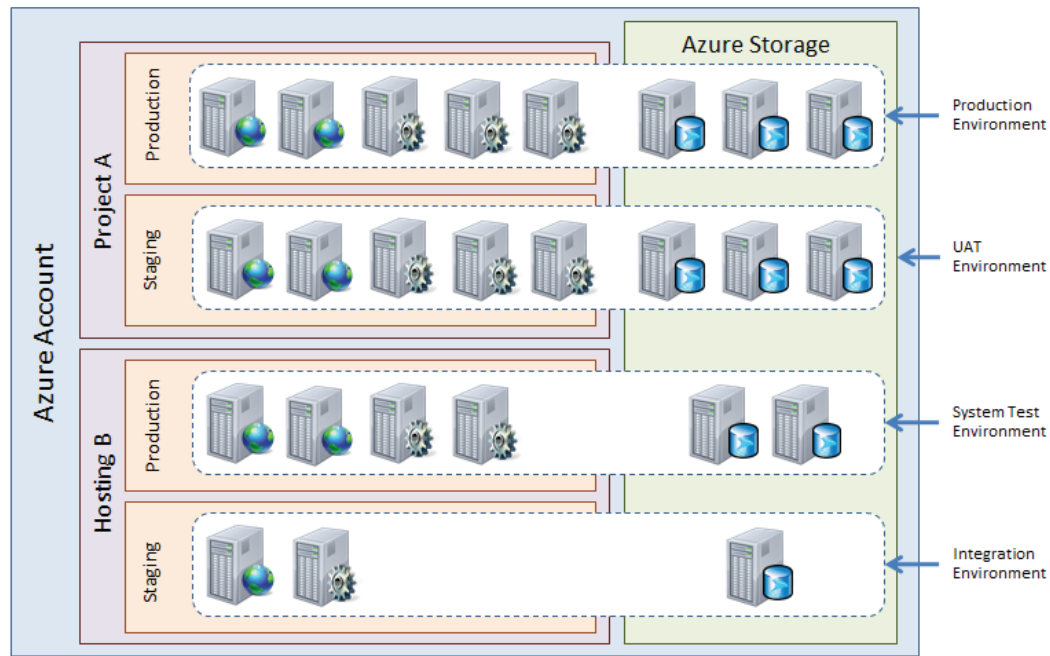


Figure 4: Creating different logical environments using Azure

To meet the requirements of the traditional scenario depicted in figure 2, a project would have to provision two Azure™ compute instances (hosting instance) along with the respective number of storage accounts depending on the application architecture and test validation criteria defined.

In a typical agile software development process, the software code release to production is preceded by a series of interim validations where the application is passed through different testing environments so as to ensure that necessary checks are carried out on the different aspects of an application such as performance, functionality, security, etc., and meet the requirements stated.

We can utilize Azure™ 's staging and production environments, associated with a single Azure™ hosting service, for creating different logical testing environments as shown in above figure.

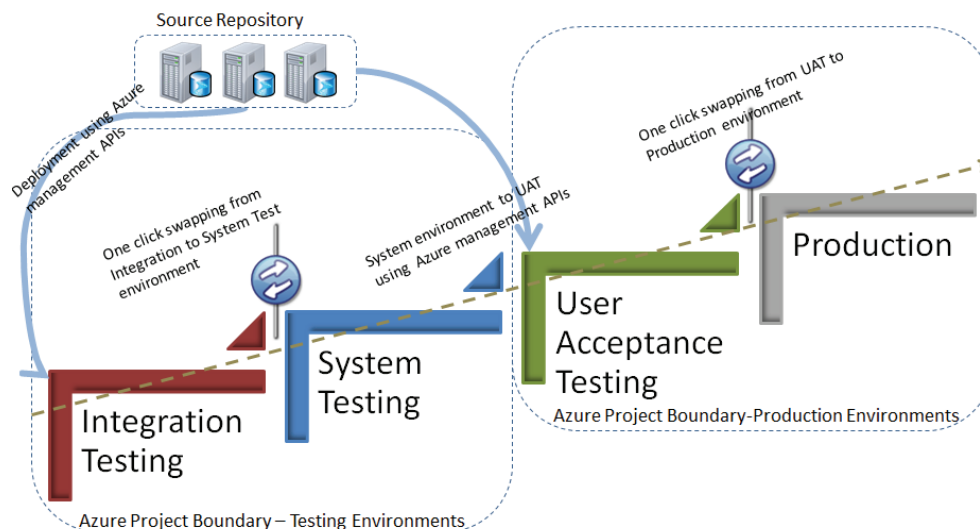


Figure 5: Azure cloud deployment across various environments

With this in perspective and the limitation around provisioning of hosting accounts depicted in figure 5, the above figure 6 depicts the process of moving a deployment package on Azure™ from one environment to another.

If we need to progress the application from one environment to another within the same compute instance, i.e., from Integration testing to System testing (or vice-versa) or UAT to Production (or vice-versa), it is a simple one click process called "Swap" managed from the Azure™ development portal and the promotions would be implicitly handled by the Azure™ management layer. However if we need to move an application across different Azure™ compute hosted instances such as from System Testing to UAT, we need to explicitly upload the deployment package into UAT from either the blob storage or the Azure™ development portal.

Automating Deployment on Azure™

We can make use of Azure™ management APIs for automating the application deployment process across the Azure™ project boundaries. These API's give programmatic control over the Azure™ deployment process, and help to move the deployed application. This is not only from one environment to another if they are in the same hosted instance but also across hosted instances.

Using this type of cloud setup we can observe the following differences as compared to on-premises setup:

- Azure™ provides staging and production environments on the cloud which have elastic scalability provided on-demand and suited to meet the requirements of the agile philosophy to a larger extent.
- Only the development and unit testing is carried out on premises.
- Cloud staging environment can be used to create different test environments on cloud such as Integration, System and UAT.
- If required we can also maintain application source code repository on Azure™ cloud using Windows® Azure™ compute instances and blob storage (not shown in picture).
- In the traditional development setup, developers test their application with a production-like environment and not really the same infrastructure as that in production, primarily owing to cost or the effort in setting such an infrastructure up. The developers end up having to make calculated assumptions on the ability of their app to deliver on its promises and hoping that the calculations done are highly accurate. However Azure™ elastic and on-demand platform will help address this challenge, since the compute staging environment of Azure™ provides infrastructure along with the ability to support service SLAs similar to that of the compute production instance and thus developers being supported with the ability to deliver better software that can be tested with higher accuracy rather than by guesstimating.
- Code can be promoted from one environment to another rather seamlessly without developers having to write verbose deployment scripts or instruction manuals to setup the application in the respective environments. The Azure™ platform will manage the promotions across different environments rather implicitly with minimalistic manual intervention.

How does Azure™ help software delivery to be more agile?

Now going forward in this section, we will discuss some of the important steps in the agile life cycle wherein, we believe; Azure™ can play a vital role in alleviating the process of building gen-next applications.

Infrastructure

Agile Scrum methodology requires infrastructure in place before the development starts (Iteration 0) so that your development teams are ready to deploy production quality incremental products on each iteration end and also at any time when requested by the customer. Windows® Azure™ gives you the option to have this infrastructure available from day one.

Traditionally, at the beginning of any project development lifecycle, considerable time and effort is spent on procuring hardware and software licenses required for setting up the development environments, leaving alone the maintenance during the course of the development. This initial setup activity would also incur a high capital expenditure from your project budget. Money is spent and capital locked-in without earning any revenue for business. These high startup costs are generally incurred in the traditional procure-and-install approach. However, with the Azure™ subscribe-and-use service consumption model, businesses can start reaping benefits and revenue much quicker. Combined with Agile, products will start getting delivered faster.

In the traditional approach, one needs to prepare the infrastructure, which would include taking care of OS and other required software installations. This would include additional overhead of handling software and hardware patching/upgrade. Azure™ does not provide access to the infrastructure but it provides ownership of a pre-fabricated hardware and software infrastructure where one can consistently deploy and run an application directly without any overheads of maintaining the hardware and software required for application development.

To summarize the above discussion:

- Software development demands flexibility of infrastructure.
- Agile development requires all infrastructures in place even before the start of first iteration.
- Azure™ helps in providing infrastructure from day one.
- No need to invest upfront in infrastructure and load your project budget.
- Cloud infrastructure is more reliable and more risk-free as compared to maintaining the infrastructure on-premise.
- Additional overhead of software and hardware upgrade and patching is taken care of automatically, by the Azure™ platform.

Realizing application full development cycle

Azure™ facilitates agile practices in order to meet customer and user requirements faster. Microsoft® provides an integrated development environment for developers to build cloud applications faster. Microsoft® facilitates Rapid Application Development (RAD) of cloud applications through Microsoft® Visual Studio™, which is an Integrated Development Environment (IDE) to build, test and deploy Azure™ applications. Project templates, Tools and SDK's are some of the assets provided out of the box integrated with the IDE for building cloud apps faster and with the right resources from a single development environment.

The development eco-system provided for Azure™ by leveraging Visual Studio™ and the .Net framework enables developers to leverage their existing skills of building .NET™ applications and thus seamlessly latch onto building applications for Azure™ with relatively low effort to adapt to the this newer technology.

Another important requirement of agile projects is of being able to provide end users with access to the functionality being developed at shorter intervals frequently during the course of the development for review and eliciting immediate feedback. With the Azure™ build, test and deploy strategy integrated with Visual Studio™ IDE projects are enabled to provide

developers the capability to do so rather seamlessly and thus shortening the overall time for them to reach out to the end users faster.

Summarizing the high points as follows:

- Cloud enables faster development/deployment as Visual Studio™ IDE is integrated with cloud environment. There is little or significant difference for a developer in terms of development experience, comparing regular development process and cloud application development.
- Cloud provides libraries and APIs which will help the developer for rapid application development of cloud applications.
- Enables developers to have end users access the application functionality for eliciting feedback from anywhere with near zero effort.

Deployment considerations

With the traditional approach, application deployment to the various environments could be very cumbersome and a highly methodical task, similar to a Project Manager planning for deployment and write release notes, the developers have to prepare installation scripts and package the application binaries to be ready for deployment. Many a times there could be separate packaging and deployment teams that would handle deployment of the applications across the different development regions.

Now with the Azure™ cloud platform, the automated deployment and upgrade process which the platform provides will help minimize or eliminate some of the lengthy steps while planning the deployment process. The steps such as preparing separate release notes or even writing verbose installation scripts can be considerably eliminated. For the development team, unlike the traditional approach, deployment to Azure™ is a simple upload process. We simply need to follow the following two steps to deploy the application in production environment:

1. Change configuration of staging application:

We can select the well tested application which is at staging and ready for deployment. From the Azure™ deployment portal change the service configuration file in order to point application to production databases and we can change the number of instance of the application required in production environment.

2. Swap the Staging Environment to Production:

Second step is to swap the staging environment to production environment by just clicking the swap button on the Azure™ deployment portal. This process makes staging environment change to production and production environment change to staging one.

Summarizing deployment high points:

- Uploading the code on the cloud on regular interval is possible.
- Deployment process is simplified and cut down to a single step upload of application on cloud.
- No need of any separate packaging efforts as it is can be packed and published easily using Visual Studio™ IDE.
- User can easily swap application between Production environment and Staging environment without any additional steps.

Environment creation and accessibility

With the traditional development, different environments such as unit testing, integration, system testing, production, etc., have to be setup by project teams to have the products being released to undergo rigorous test and validation cycles before releasing for production use. This is a task which requires considerable time and effort to make the various environments ready. Also since there is no standardization and automation involved this can often be a process which is error prone.

Getting the production environment ready mainly takes more than 2 weeks it requires experts like architects, network and infrastructure specialist. With Agile, the main focus should be to minimize overall time of the development process but because of these environment setup challenges the process lifecycle gets stretched initially.

With the teams distributed across different geographical locations, in scenarios such as outsourcing, global delivery models, etc, different environmental challenges are imposed. In such cases project teams have to plan for access of resources from different locations within their network as well as on the inside the partner domain. Some of the tasks involved and which often requires careful deliberation includes setting up private VPN links, providing access to an isolated environment for developers outside the corporate domain to test their code, setting up firewalls with the necessary security privileges, provide vendor users access to the data source etc.

The Azure™ platform principally is available always and on-demand. Also Azure™ API's provide accessibility to the platform over ubiquitous http-based standards such as REST. These capabilities make compute and storage resources for geographically dispersed developers available always and from anywhere.

Scrum Testing and Integration

Scrum many times follows Test Driven Design approach. This results in frequent testing and integration of the application code. Agile requires testing during many phases of the software development lifecycle like testing internal deployed software, Pilot test release and Final acceptance testing. This slows down the process.

Using the Azure™ development environment once one testing has been done, we can do the changes in present code for further test and do the incremental deployment in cloud environment; we just have to share a cloud environment URL to tester/customer through which they can access the application for testing.

Prototyping and Demos

Agile software development practices require that developers deliver software by means of rapid prototyping and eliciting immediate feedback from the end user. This could be potentially a challenge if the development team works out of different locations not in the same domain as that of the end user. In such cases, as discussed in the previous section, accessibility would be a challenge.





























With the Azure™ platform, being available always and anywhere will help address these challenges. Working prototypes build by developers locally and deployed on Azure™ can now be made accessible immediately to prospective customers and for eliciting feedback with a shorter turn around. Additionally the risk associated with failure of these prototypes would also be minimalistic compared to the traditional on-premise deployment model wherein considerable investments had to be made on setting up the infrastructure and make the prototypes accessible.

Till now we have seen the advantages of using Azure™ in agile kind of development, we can refer to a typical scenario to see how Azure™ helps. Suppose an enterprise is working on some client product that need to be hosted in DMZ and need to provide access to client for getting the services. Now to host the product in DMZ enterprise network service group need to be involve who would be responsible for hosting the product, but they would require to cross some internal hurdles of following certain policies and procedures to host product which is again a very time consuming job. Above which if client requirements for that particular product is changing frequently then it's very difficult to match the client expectations. In such situation

agile development using Azure™ as a platform would bring all the advantages which we have covered so far. This enables enterprise capable enough to cope up client frequent requirement changes for the product.

Normal agile and Azure™ -agile development some comparison

Following are some high level comparison we have made between normal agile development and agile development using Azure™ as a technology. The table shown below gives comparison based on cost involved and time require for different factors which affects agile development.

#	Factors affecting Agile Development	Cost involved		Time Required	
		without Azure	with Azure	without Azure	with Azure
1	Creating and Managing Different Test Environments.				
2	Creating and Managing Production Environments.				
3	System administrator requirement for handling systems and deployments.				
4	Development effort				
5	Deployment effort				
6	On-demand provisioning of infrastructure to handle instant team ramp-up.				
7	Prototyping and Demos				





 High
  Moderate
  Low
  Not Applicable.

Figure 6: Cost-Time comparison w.r.t. some agile development factors

Also below figure shows the number of stages involved in agile iteration and normal time taken for each of the activity in the iteration. These are the steps commonly followed by many agile service providers (reference: http://www.rallydev.com/agile_products/lifecycle_management/). We have also shown the time based comparison between normal agile iteration lifecycle and agile iteration life cycle with Azure™ .

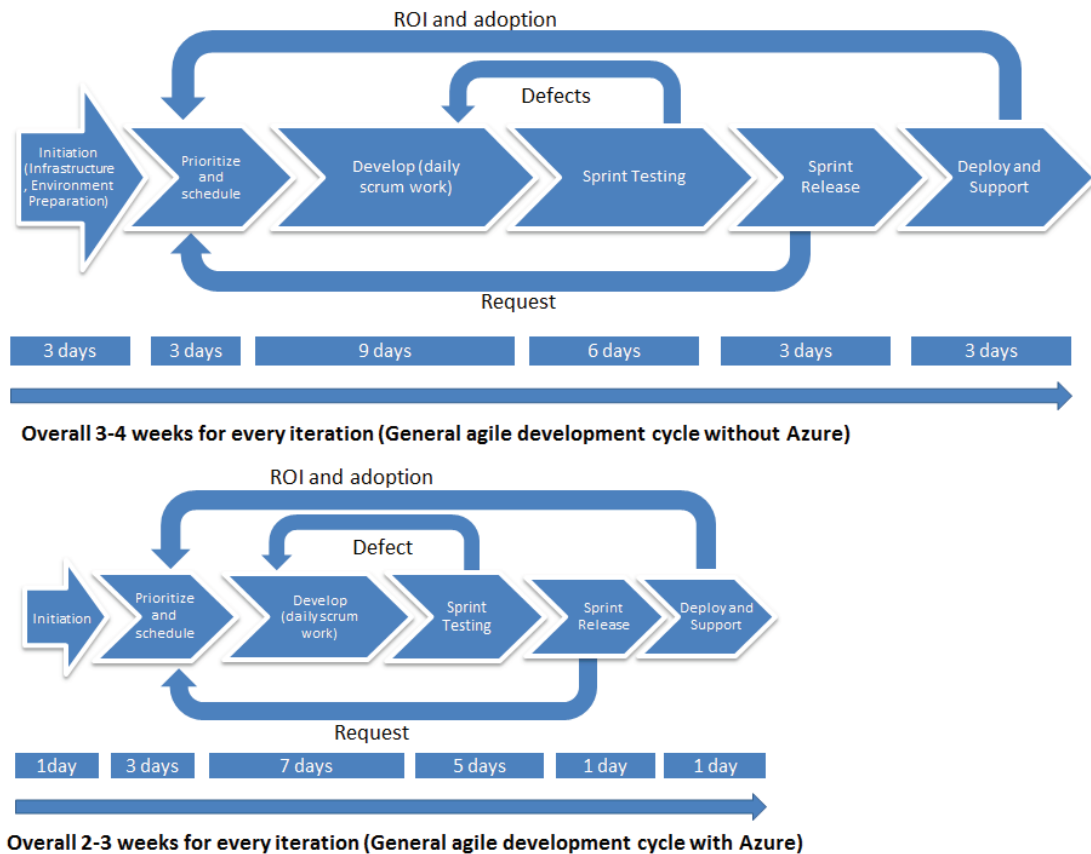


Figure 7: General agile development cycle with Azure™

Challenges using Azure™

Assuming you are choosing the Agile Scrum methodology for Azure™ application development, we may come across following challenges while adopting Azure™ technology for agile development.

Developing desktop or client server application

If you want to develop a smart client windows application and want to choose agile methodology with Azure™ technology, then wait! There are limitations from Azure™ technology side as Azure™ cloud is mainly useful to develop web application or applications which are service enabled but it is not suitable for development of desktop apps. In such scenarios one may think of moving towards other cloud vendors such Amazon (AWS), Rack space, etc., that play in the infrastructure as a service space and which provides the infrastructure for deployment of such application.

Client policies on data security

Enterprises may have IT policies which govern data and data access security. Like policies which do not permit to enterprise critical data to be moved outside the enterprise boundaries. In such scenario we may need to take a middle hybrid approach of keeping the database on-premise and moving the application logic to cloud.

This also may cause following overhead:

- **Performance in terms of data access**

In such case we have to expose the database to application as a service, it may be either REST or SOAP based. This additional service access layer will add up an overhead and performance will be slightly low compared having database in the cloud.

- **Database scalability**

Being on-premise, one cannot take advantage of the auto scalability features for database provide on cloud. We need to manually take care of scaling up/out of database as per the load requirement. This will additionally increase operational cost for running and maintaining the database.

Managing different testing environments

With Azure™ hosting, as discussed previously in this paper, a single hosting instance provisioned will have only two environments (Staging and Production) provided to users per instance. In case of complex project environments which are usually setup by IT in enterprises, this provisioning may fall short of meeting the requirements to operate multiple test and production environments simultaneously and which may require some reasonable degree of isolation so as to be able to run multiple versions of code simultaneously. For this, projects need to provision additional hosting instances to be able to configure and operate more environments and that which has been hypothetically explained in *Figure 6*.

Debugging on Cloud

Unlike the traditional on-premise approach where application debugging is very exhaustively supported by a strong Visual Studio™ suite of testing tools such as VS Debugger, memory profilers, etc., debugging applications on Azure™ is a challenge. This is primarily because Azure™ applications are hosted in a separate location with limited administrative access to the hosting instance and thus the inability to deploy and run debugging tools on a provisioned hosting instance. So in order to debug an application in cloud we have some alternatives and which are listed below:

- Use the development fabric, a simulator of the Azure™ cloud to test applications locally. This will help developers diagnose and address most of the functional issues as the VS debugger is well integrated with the development fabric. Hence developers can expect a similar development experience as that with building any .NET™ application on-premise
- Instrument using traces and logs in your application even after deploying to cloud
- Use Windows® Azure™ diagnostics APIs to collect and store a variety of system level diagnostic data which would be useful to analyze and diagnose application behavior on cloud. This is useful for:
 - Detecting and troubleshooting problems
 - Performance measurement
 - Capacity planning, etc.

Conclusion

This paper has attempted to provide architects and project managers an insight into how cloud technologies and platforms such as Azure™ help augment enhances existing agile practices to deliver software faster. Thus make IT in a capable position to deliver Gen Next solutions which are able to meet rapidly changing business requirements.

Azure™ as a platform not only brings in significant savings to projects in terms of both cost as well as effort but also provide essential capabilities, as we have discussed in this paper, to help deliver software faster. The agile software development practices are significantly enhanced with Cloud technologies and Microsoft® Azure™ Platform as a Service (PaaS) providing developer centric capabilities helps in furthering the cause to deliver software in that fashion. Traditional Agile practices and areas of improvement by which Azure™ helps bridge these gaps by providing an on-demand and scalable environment have been discussed.

Happy developing software with the clouds!

Authors

Sidharth Subhash Ghag (Sidharth_Ghag@infosys.com) is a Senior Technology Architect with Microsoft® Technology Center (MTC), Infosys. With over twelve years of industry experience, he currently leads solutions in Microsoft® Technologies in the area of Cloud Computing. In the past he has also worked in the areas of SOA and service-enabling mainframe systems. He has been instrumental in helping Infosys clients with the service orientation of their legacy mainframe systems. Currently he helps customers adopt Cloud computing within their Enterprise. He has authored papers on Cloud computing and service-enabling mainframe systems. Sidharth blogs at <http://www.infosysblogs.com/cloudcomputing>

Amit Dumbre (Amit_Dumbre@infosys.com), Technology Architect with Microsoft® Technology Centre, Infosys and has more than 10 years of industry experience of developing and designing solutions in Microsoft® Technologies such as windows applications, ASP.net, web services, WCF services etc. He is currently working and exploring Windows® Azure™ platform and implementing applications using technology stack of Azure™ and deploying those on Azure™ platform.

Sathya Priya Senthil (SathyaPriya_Senthil@infosys.com), Senior Systems Engineer with Microsoft® Technology Center (MTC) in Infosys. She has nearly 4 years of industry experience on Microsoft® .NET™. She works on implementing and deploying projects on the Windows® Azure™ platform.

Acknowledgement

Authors would like to acknowledge **Sameer Vajre**, Senior Project Manager, CMEX, Infosys Technologies Ltd. for his help with the review of this paper.



For more information, contact askus@infosys.com

About Infosys

Many of the world's most successful organizations rely on Infosys to deliver measurable business value. Infosys provides business consulting, technology, engineering and outsourcing services to help clients in over 30 countries build tomorrow's enterprise.

For more information about Infosys (NASDAQ:INFY), visit www.infosys.com.