

AN AGILE PROCESS FRAMEWORK FOR CLOUD APPLICATION DEVELOPMENT

The logo for CSC (Computer Sciences Corporation) is a red hexagon with the letters "CSC" in white.

Debrata Das

Kirti Vaidya

CSC

kvaidya2@csc.com

CSC Papers

2011

Keywords: Cloud computing, PaaS, Platform as a Service, Agile development, Architecture, Framework

ABSTRACT

By its very nature, cloud-based development offers an organization a high degree of agility; correspondingly, the developmental processes themselves should be agile in nature. This paper provides an overview of the challenges faced in developing cloud applications, and discusses an agile framework that offers a repeatable process for building cloud-based applications. With the growing maturity of cloud computing technologies, enterprises are now exploring their benefits beyond the known efficiencies gained in data center infrastructure. Cloud technologies are being applied and leveraged in different types of applications, a trend fueling the recent growth in the number of Platform as a Service (PaaS) vendors and technologies. PaaS provides the bedrock for cloud-based application development, deployment and management. Given the new architectural paradigms introduced by cloud, we expect to see more and more applications built merely by assembling and orchestrating services, rather than developed based on indigenously built software components. This necessitates a fresh look at existing software development lifecycle processes.

INTRODUCTION

The growth of Cloud Computing adoption in the industry has been more focused on leveraging the infrastructure optimization and automation benefits of Cloud computing. While this is certainly a good first step, it is only one of the many benefits of Cloud Computing – traditional applications can also take advantage of Cloud Computing in terms of almost infinite scalability and unprecedented reliability. This is enabled by the Platform-as-a-Service (PaaS) technologies on the Cloud. The use of PaaS in traditional applications and the corresponding development process to aid Cloud applications is the key focus of this paper.

The paper provides an overview of application architecture and development considerations for the Cloud environment and its challenges. It explains Agile development and proposes an Agile process framework for Cloud based application development. It examines the business benefits CSC would reap from building a consulting service offering that includes enablers for global execution and delivery of Cloud development projects. The paper presents a roadmap to realize the service offering - deliver an Agile process website for Cloud based development, pilot it on a live project, and finally publish the capability as CSC's intellectual property and service offering, in 2010.

OVERVIEW – DEVELOPING FOR THE CLOUD

Cloud Computing is a lot more than how organizations can quickly and efficiently provision and manage computing capability. It also represents a fundamental shift in the mindset on how applications on the Cloud need to be built and run. Before we delve into the details of such applications, it is important to understand the basic concepts of the Cloud and how they relate to applications. Once that is established, we will explore an Agile process framework for Cloud application development.

NIST'S DEFINITION OF CLOUD COMPUTING

The National Institute of Standards and Technology (NIST) has taken the lead on defining the Cloud computing in terms of its service models and deployment models. The illustration below is a great example of describing a Cloud ecosystem.

CLOUD SERVICE MODELS [NIST]

Service models are used to categorize different types of services that can be available from a Cloud Computing environment. Here are some definitions from NIST about each of the service models.

Cloud Software as a Service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a Cloud infrastructure.

The applications are accessible from

various client devices through a thin client interface such as a web browser (e.g., web-based email). The consumer does not manage or control the underlying Cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

Cloud Platform as a Service (PaaS)

The capability provided to the consumer is to deploy onto the Cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider.

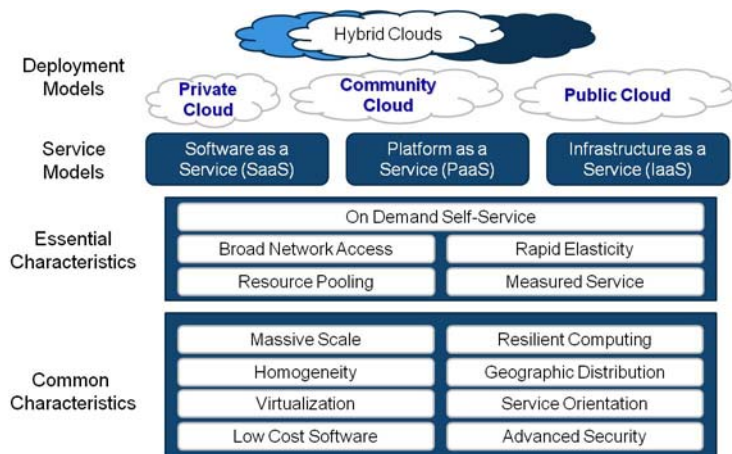


Figure 1 – Depiction of NIST Cloud Computing Definition [1].

The consumer does not manage or control the underlying Cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.

Cloud Infrastructure as a Service (IaaS)

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications.

The consumer does not manage or control the underlying Cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

CLOUD DEPLOYMENT MODELS [NIST]

Private Cloud

The Cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.

Community Cloud

The Cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.

Public Cloud

The Cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling Cloud services.

Hybrid Cloud

The Cloud infrastructure is a composition of two or more Clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., Cloud bursting for load-balancing between Clouds).

PLATFORM AS A SERVICE

The PaaS model is perhaps the least understood of the three service models, because it covers such a wide range of capabilities. To understand the PaaS service model, one must first focus on the fact that it is about application development and delivery, not lower level infrastructure or higher level software services. PaaS vendors provide environments for customers to build and run applications, they do not make the underlying operating systems available to the customer.

PaaS has two distinct, but related, purposes:

- **Application delivery:** all the pieces needed for an application to execute within a Cloud computing environment
- **Application development:** all the functions that are needed to build, test, and deploy applications for Cloud computing environments

Not every service provider will have a solution for every part of *Figure 1 – Depiction of NIST Cloud Computing Definition [1]*. For example, many service providers assume that any application deployed into their Cloud will be interactive, so they don't have the ability to support scheduling services. Other service providers, such as Microsoft's Windows Azure service, provide that ability.

Figure 2 – PaaS Component Sample Services below shows sample services that would fall into the various components of the PaaS layer. These capabilities are representative of the components necessary and unique to the PaaS layer. There are capabilities that a typical PaaS application would leverage from other parts of the overall model, such as depending on Incident Management services.

This still encompasses a wide variety of environments, but PaaS providers and vendors separate into three categories:

- **Application framework as a service:** full solution that includes support for customization of the application using an interface that abstracts the technologies used (i.e., no low level coding needed). An example of a service provider is Salesforce.com; their environment allows for new applications to be created in a very rapid manner without traditional coding



Figure 2 – PaaS Component Sample Services.

- **Application delivery as a service:** focuses on the core components and APIs to execute an application and fully leverage the service provider’s Cloud computing environment. These environments tend to be technology-centric (e.g., .NET, Java, Python). Providers in this category include Google App Engine, Spring Source, and Microsoft Windows Azure
- **Functional service:** specializes in a particular function. As an example, both SOASTA’s CloudTest and SkyTap’s Cloud Development and Test Solution map very well to the Testing Tools component

CLOUD APPLICATIONS & MULTI-TENANCY

Just as PaaS providers can be categorized as described above, Cloud applications too can vary in their architectural models, depending on your choice of the Cloud environment for the application. At the two extremes of these architecture models spectrum are the “Cloud Hosted” and “Cloud Optimized” applications. A discussion on these models is important to understand the development challenges of a Cloud application. A crucial factor that plays into the selection of the architecture model is the degree of “multi-tenancy” required for your application.

Simply put, “multi-tenancy” refers to the capability of an application to support multiple clients. Wikipedia defines “multi-tenancy” as

“Multitenancy refers to a principle in software architecture where a single instance of the software runs on a server, serving multiple client organizations (tenants). With a multitenant architecture, a software application is designed to virtually partition its data and configuration so that each client organization works with a customized virtual application instance.”

A “Cloud Hosted” application leverages multi-tenancy at the Infrastructure layer i.e. a Cloud IaaS provider will be able to share its infrastructure to support multiple client applications. E.g. Amazon EC2, Rackspace etc.

A “Cloud Optimized” application supports multi-tenancy at all the different layers (infrastructure, application server, database etc.) by leveraging a PaaS platform. E.g. Salesforce.com’s Force.com.

The visual below illustrates some of the key factors to consider about your need for multi-tenancy and then choose the appropriate Cloud service model. Cloud apps should consider business and technical implications to identify the appropriate level of multi-tenancy.

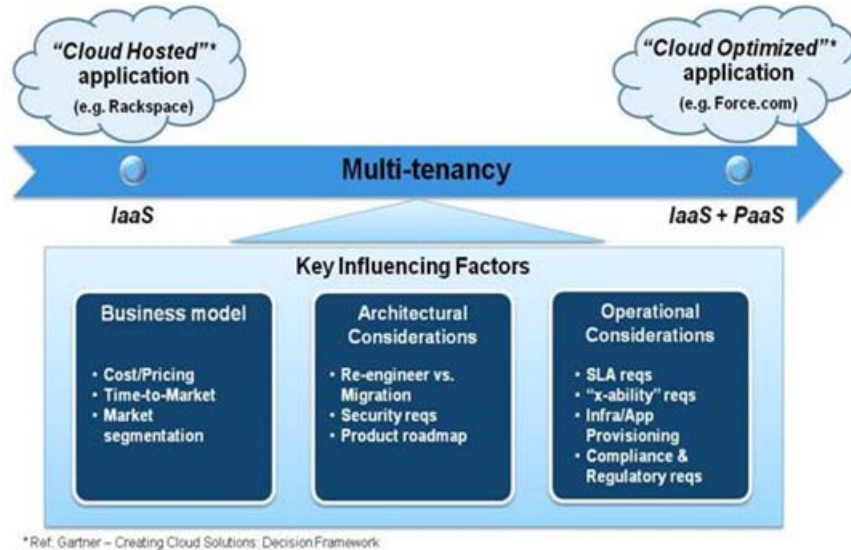


Figure 3 – Multi-tenancy.

CLOUD HOSTED” APPLICATION ARCHITECTURE

When an application is deployed to a IaaS Cloud environment, purely to leverage its virtualized environment, we call it a “Cloud Hosted” application i.e. the application is only leveraging the virtualization, rapid provisioning, pay-as-you-go model and such other benefits of the infrastructure service model (Refer NIST definition for IaaS above).

It does not change the behavior or architecture of the application itself. As a result, you are relatively limited in your ability to scale an application – for example, if an application has not been inherently developed to scale and it is deployed as a “Cloud Hosted” application, you can only add additional instances of the application on the Cloud to address increased demand. Of course, this assumes that shared memory, maintaining session state etc. are not required or are addressed by special means. While this is true for pure IaaS providers, newer technologies such as those offered by VMware vMotion are minimizing the impact in a Private Cloud setting.

For a SaaS application on a Cloud, each client is provided with a virtual instance of a SaaS application instance, hosted on an IaaS platform. Only the infrastructure requirements of the SaaS application are addressed.

The illustration below shows the architecture of a “Cloud Hosted” application.

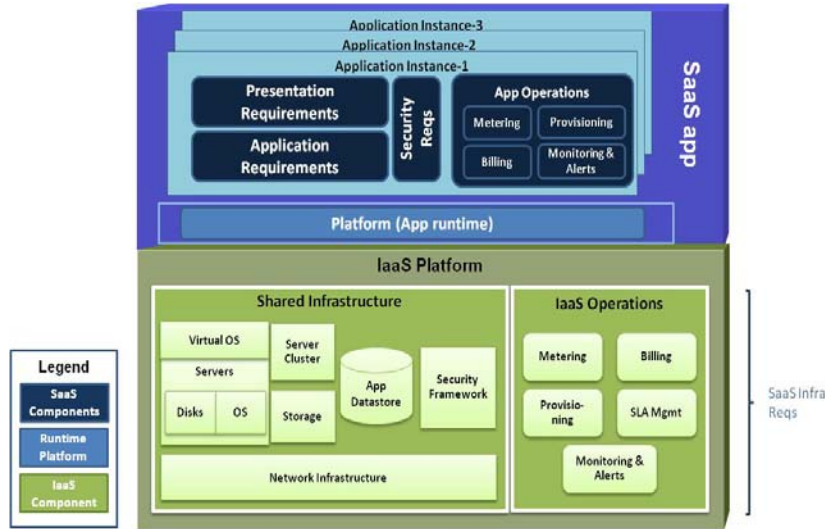


Figure 4 – Cloud-Hosted Application Architecture.

“CLOUD OPTIMIZED” APPLICATION ARCHITECTURE”

On the other hand, “Cloud Optimized” SaaS applications assume a far greater degree of support from a PaaS platform to provide cross-application layer multi-tenancy.

As highlighted in the picture below, a full-featured PaaS platform provides an elaborate framework for SaaS applications. A PaaS provider is expected to not only provide application specific capabilities such as user Interface framework, scalable business rules/process support, security framework etc. but also management tools and portals for subscribers (i.e. customers who are using the PaaS provider Cloud environment for their applications) for configuring, managing and monitoring applications. A good example of a “true” PaaS platform is Salesforce.com’s Force.com, where you have most of the attributes highlighted below.

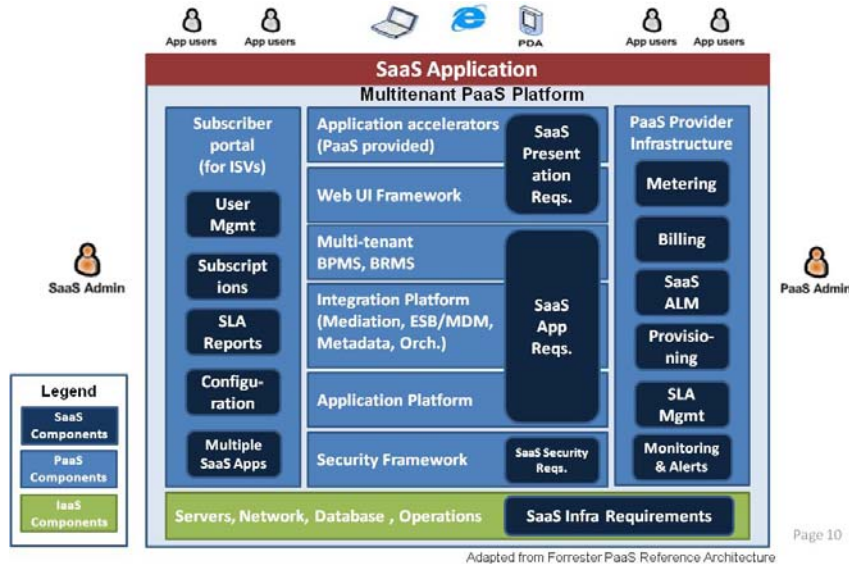


Figure 5 – Cloud-Optimized Application Architecture.

A “Cloud Optimized” application is often compared to a traditional on-premises SOA application. While a full discussion on the subject of SOA and Cloud is beyond the scope of this whitepaper, it is sufficient to say that Cloud applications are architecturally required to be service oriented with a focus on statelessness, low coupling, modularity, and semantic interoperability.

As illustrated above, a true PaaS platform provides a extensible features for business process magamanet, business rules,service orchestration etc – which are synonymous with SOA realization requirements.

“CLOUD HOSTED” VS “CLOUD OPTIMIZED”

Now that we have reviewed the architectures for a “Cloud Hosted” and “Cloud Optimized” applications, here is a short summary of the key differences between the two architectural models.

Application Characteristic	“Cloud Hosted”	“Cloud Optimized”
Size of workload	Typically small; may not need to use full server resources	Capable of very large, compute-intensive tasks that may need several servers
Architecture	Typical single instance apps where vertical scaling is sufficient	Loosely coupled multi-instance apps based on dedicated frameworks that enable scaling across huge server farms
Fault Tolerance	Hardware-provided	Built into the application architecture; hardware independent
State Management	Traditional, app or database oriented	State is maintained in distributed data stores for redundancy and resiliency
Server Virtualization	Must-have. Server optimization is often the driver for this model	Can be optional, though may not be recommended. If needed, the app stack can be moved to bare-metal servers.
Portability	Limited; Virtual Machines may be tied to Cloud provider’s infrastructure	Higher. Application architecture components and frameworks should be capable of running on any environment

Table 1 – Cloud Hosted and Optimized Applications

WHY IS CLOUD DEVELOPMENT DIFFERENT?

The Cloud Computing environment is very different compared to a traditional data center in terms of how applications are deployed, configured, run, and managed. This requires certain fundamental attributes to be present for an application built for a Cloud environment. The following table highlights some of the key differences between tradional applications and Cloud based applications.

Traditional Apps	Cloud-based Apps
Application components co-located in same environment (deployed as one bundle)	Components are mostly scattered around one or many Clouds (deployed as multiple modules)
Run-time infrastructure is structured and controlled (infra. features can be assumed during design/development)	Run-time infrastructure is un-structured and managed by Cloud fabric (infra. features change constantly and at run-time)
Security is enforced by application architecture (LDAP look-up based authentication/authorization)	Security is built into the service contracts (WS-Security, SAML, etc)
Support for Multi-tenancy is typically not required	Multi-tenancy support is assumed
Typical user base is known at Design-Time and is consistent (need to scale to the known user volume)	User base may not be known and could be dynamic (potential need to scale to hundreds of thousands of users)
Deployment requires traditional packaging and deployment tools (Application server admin console, ANT, etc)	Along with traditional tools, deployment requires knowledge and utilization of vendor specific Cloud API and tools
Enhancements and Upgrades require down-time and follow a “all-in-one” approach (deploy common version in all runtimes at same time)	No down-time required for enhancements and upgrades (versions can be deployed to all or parts of the Cloud)
Components interact with non-SOA contracts (method calls, RMI, CORBA, HTTP, etc)	Standard SOA Service based interaction between components is assumed (SOAP and REST interfaces)
Business functionality is realized by using “controller” components that calls methods (functions) of business components	Service Orchestration (BPEL, etc) is used to realize business functionality – invoke one or many business services
Application is tested in controlled environment (Unit/Integration/System)	Application (integration) is tested on the Cloud to ensure seamless orchestration between services on one or many Clouds (focus on across the network communication, security mechanisms, etc)

Table 2 – Traditional versus Cloud Applications

CHALLENGES FOR THE DEVELOPMENT TEAM

Just as Cloud based applications are “required” to be different, they present unique challenges to application architects and developers. Here are some key challenges that one may encounter while building apps for the Cloud.

- Remember Cloud has layers – Structure Applications at design time to map to Cloud layers
 - **Infrastructure** (server and installed software)
 - **Storage** (RDBMS, Google's Bigtable, Amazon's SimpleDB, etc)
 - **Platform** (solution stacks such as Ruby on Rails, LAMP, Python Django, etc)
 - **Application** (custom app, Google Docs, Salesforce, Facebook, Flickr, etc)
 - **Services** (custom web services, PayPal, Google Maps, etc)
 - **Client** (thin/thick desktop clients, mobile clients like Symbian, Android, iPhone, etc)
- Organize business solutions into :
 - Services – Independent business tasks that can be started/stopped separately across the Cloud
 - Service Applications - Applications that get deployed "into" a service (Java web app into a Tomcat service)
- Design with SOA principles in mind
 - Standardized Service Contracts, Loose Coupling, Abstraction, Reusability, Service Autonomy, Statelessness, Composability
- Orchestrate Cloud-enabled services using custom logic or Cloud APIs to build complex business applications
- Security (Authentication/Authorization) - Use WS-security, SAML (or other methods) tokens that can be passed seamlessly between application and services
- Use standard service-access techniques e.g.: SOAP, REST, .NET and Spring Remoting
- Use common storage device (e.g.: SAN) accessible to all workers in the Cloud for storing resources (configuration files, databases, etc)
- Remember interoperability before using Cloud Fabric specific APIs
- Regard resources as services (storage, database, JMS queues, MOM, etc) and build service-enabled access to them
- Design framework components (logging, auditing, security, etc) as services
- Consider mitigations for the following at design time:
 - **Reliability** – Choose a service provider with mirrored sites
 - **Data security** – Check vendor policies before hosting sensitive data
 - **Vendor lock-in** – Build methods to migrate data out of a provider, Avoid design using vendor specific APIs

When possible, Cloud applications should be packaged as set of individual modules (one service – one module). This has several advantages as under:

- Enables staggered provisioning and deployment into the Cloud
- Enables deploying modules into different workers on the Cloud or different Clouds (e.g: Credit card processing service can be on public Cloud while authentication service stays in the private trusted Cloud)
- Enables scaling-up of individual modules as required
- Allows for Cloud Fabric to manage provisioning, load balancing and distribution, workload management and transaction reliability
- Enables versioning of services and individual deployment without affecting other modules

CLOUD DEVELOPMENT PROCESS SELECTION

Since Cloud development is different and poses its own challenges to the development team, it is important to consider a Cloud development process which leverages the strengths and benefits of Cloud development while helping the team manage the challenges.

A software application development process provides information on *who* (roles) does *what* (workflows) *with* what (work products) and *how* (guidance). Projects typically follow a specific process during their lifecycle.

So far most application development processes have been published in static media like books or websites. Process authoring tools (e.g. Eclipse Process Framework Composer, i.e. EPF Composer [4]) typically publish static, passive websites. Very few tools help publish “active processes” which can be orchestrated into projects. It can be argued that such an active process for Cloud application development should be included in PaaS Components Sample Services (ref. *Figure 2 – PaaS Component Sample Services* above) within the application development partition.

GOALS FOR A CLOUD DEVELOPMENT PROCESS

The software development community is turning to the Cloud primarily because of the benefits it provides. In the PaaS area, these include flexibility of platforms, the ability to offload on-premise workload, and consequent adaptability as the project’s resource demands change. This results in agility and speed. Any developmental process we choose should match these benefits. It should certainly not impede the value of the Cloud at the least. E.g. traditional *Waterfall Processes* (see below) would be counter-productive unless some specific conditions like repetition of work are met.

We propose the following ten goals for a Cloud development process, *each focussing on Cloud-specific nuances*. The process should be:

1. Quick
2. Resourceful
3. Adaptable
4. Cloud focussed
5. Minimal
6. Sufficient
7. Lean
8. Configurable
9. Practice based
10. Active

QUICK

The Cloud development process should facilitate quick discovery and assembly of resources and services available within the Cloud in order to build a software application at a high speed to market.

RESOURCEFUL

Resourceful [9]: “able to meet situations : capable of devising ways and means”. For a process this implies having sufficient ways and means within itself that help practitioners assemble Cloud applications with the most current resources.

ADAPTABLE

The process should be structured so that practicing communities could adapt it to how Cloud technology and vendor offerings change.

CLOUD FOCUSED

Based on Why is Cloud Development Different? above, the process should guide practitioners on leveraging the Cloud for application assembly and deployment. It should also help mitigate challenges listed above in Challenges for the Development Team.

MINIMAL

There are two ways of authoring a process. It could either include everything from which a practitioner would subtract what is not needed. Or it could include just the minimum guidance to which a practitioner would add if and as needed. We propose the latter.

SUFFICIENT

The process should be minimal but should still have sufficient guidance to assemble most Cloud based application types within its scope at a given time.

LEAN

A process may be minimal, but it could still lead to unnecessary work and work products. A goal for the Cloud development process should be to imbibe lean software development principles [9].

CONFIGURABLE

Method content for the process should be easily configurable, and also, should lend itself to easy integration into other process configurations. The EPF Composer tool [4] allows authoring Method Content which can be selectively extended and associated with process configurations.

PRACTICE BASED

Contemporary software development processes are being redefined using practices as building blocks (this is also a goal for CSC's AgileCSC process which is being integrated into Catalyst). A practice could be seen as complete method content including roles, workflows, work products, and guidance pertaining to one objective; e.g. architecture for Cloud application development.

ACTIVE

We propose that the Cloud development process is made to be an active process in that, a new team charged with developing a Cloud application would not only just refer to its static website but would be able to instantiate it as a project that includes configurable tasks with all necessary guidance and templates, along with collaboration facilities. This will ensure constant ground up streamlining and currency of the process.

PROCESS CHOICES

Now to choose a specific process type for authoring the Cloud development process framework, we consider three broad types of processes which exist at present:

WATERFALL PROCESSES

In the Waterfall process, various project disciplines like Requirements, Analysis & Design etc. become sequential phases. Phase gates depend on signed document exchange. The process is good for repetitive work, but is otherwise perceived as rigid. The 'realized' risk typically rises with time, since risk mitigation opportunity diminishes with time.

ITERATIVE PROCESSES

In an Iterative process, various project disciplines like Requirements, Analysis & Design etc. are interspersed within iterations which typically occur sequentially, and could be combined into phases. Iteration gates depend on specific objectives. The process is good for exploratory work. The 'realized' risk typically diminishes with time, since risk management happens early and within every iteration.

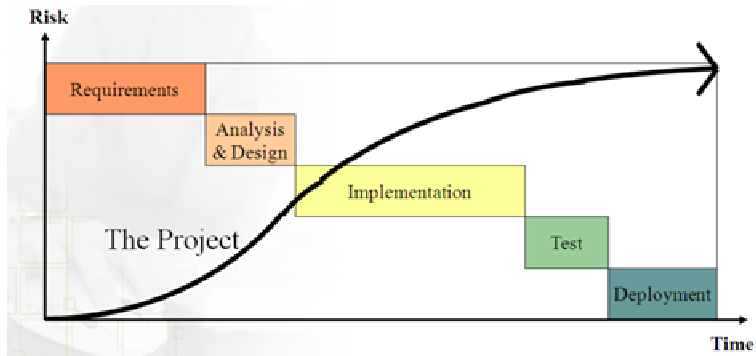


Figure 1 – Waterfall Process.

Some popular iterative processes include variations on the Unified Process:

- IBM's Rational Unified Process (RUP) [1]
- Eclipse's Open Unified Process (OpenUP) [4]

AGILE PROCESS

Agile processes are also typically iterative as above. However spiritually they draw from the Agile Manifesto [5]:

Some popular Agile processes include:

- Scrum
- Extreme Programming (XP)

A pragmatic definition of Agility is [8]:

Agility: "the quality or state of being Agile"

Agile: "having a quick resourceful and adaptable character"

This definition provides three terms that are key to agility within an information systems project.

(What) Quick: Businesses want a quick "Time-to-Market" – faster than the speed of competition – is the mantra.

(How) Resourceful: To achieve such agility, resourcefulness is important.

(How) Adaptable: The project must possess adaptability to be resourceful.

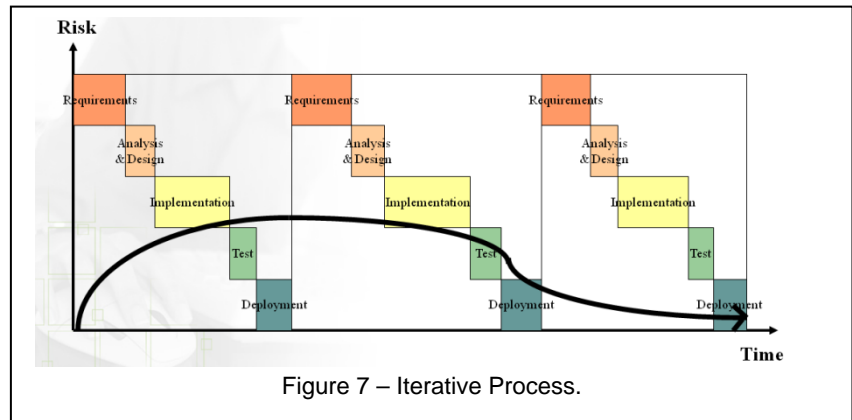


Figure 7 – Iterative Process.

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- Individuals and interactions** over processes and tools
- Working software** over comprehensive documentation
- Customer collaboration** over contract negotiation
- Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Figure 8 – Manifesto for Agile software Development.

THE CLOUD DEVELOPMENT PROCESS

Based on *the Goals for a Cloud Development Process* above, an Agile process is chosen as the baseline upon which the Cloud Development process would be built. Today's Agile processes like Scrum and XP deliver agility on the terms mentioned above. If configured appropriately, they could facilitate quick discovery and assembly of resources and services available within the Cloud in order to build a software application.

CLOUD AGILE FRAMEWORK (CAF) FOR CLOUD DEVELOPMENT

Cloud Agile Framework (CAF) is the proposed Agile process framework for Cloud application development.

GOAL BASED PROCESS AUTHORIZING

The following table lists recommendations considered in proposing Cloud Agile Framework (CAF) in order to satisfy the goals outlined before.

Goal	Recommendations
1. Quick	Base the process on AgileCSC
2. Resourceful	
3. Adaptable	
4. Cloud focused	Base it on Error! Reference source not found. and Error! Reference source not found. Error! Reference source not found..
5. Minimal	Incorporate Lean Principles [9].
6. Sufficient	
7. Lean	
8. Configurable	Author the process with the EPF Composer tool [4].
9. Practice based	Build the process by leveraging practices as in Error! Reference source not found. Error! Reference source not found..
10. Active	Attach the Cloud Agile Framework (CAF) website published from EPF Composer, to a PaaS based process orchestration tool.

Table 3 – Cloud Development Process Goals and Recommendations

Cloud Agile Framework (CAF) will be based on CSC's *AgileCSC*, which is an empirical Agile project management framework used to deliver increments of high value to the customer iteratively. A major difference will be that Cloud Agile Framework (CAF) will additionally include relevant technical practices including one for Cloud development, as discussed below.

Cloud Agile Framework (CAF) will leverage the following process definitions:

1. Scrum [6]: a very widely used Agile management process
2. OpenUP [4]: divides a project into four logical phases and baselines the architecture in the second phase
3. Work Components (from CSC – Covansys, now a part of MSS): an objective based, team-centric work definition [3]

PRACTICES

Practices are defined as "chunks' of process for adoption, enablement, and configuration" [4]. Refer to the following table which shows practices [4] configured into processes. It proposes practices that Cloud Agile Framework (CAF) should leverage. They include a new technical practice, "Cloud Application Development" which we propose should be authored.

Practices	Cloud Agile Framework (CAF)	AgileCSC	Agile (Scrum &XP)
Management			
Risk-Value Lifecycle (4 Phases)	X	X	
Agile Practices (from Scrum)	X	X	X
Iterative Development	X	X	X
Release Planning	X	X	X
Whole Team	X	X	X
Team Change Management	X	X	X
Technical			
Concurrent Testing			X
Continuous Integration	X		X
Evolutionary Architecture	X		X
Evolutionary Design	X		X
Shared Vision	X	X	X
Test Driven Development			X
Cloud Application Development	X		

Table 4 – A Practice-Process Framework

CSC'S "CLOUD APPLICATION DEVELOPMENT" PRACTICE

The following architectural activities will be included in CSC's "Cloud Application Development" practice.

Evaluate Service Model and determine providers

Evaluate the Services model for the Cloud application and determine service provider(s) – PaaS provider, external entities, cross-Cloud etc. and the required orchestration between the services.

Determine integration feasibility

Determine the integration requirements and feasibility of the Cloud application with in-premises systems and data stores.

Consider security and confidentiality

Consider the security and confidentiality requirements of the application from a Cloud suitability perspective.

Consider performance

Review performance requirements and latency tolerance. Develop architecture to minimize performance impact due to use of Cloud environment.

Evaluate Cloud Deployment Models

Evaluate various Cloud deployment models for suitability with the application requirements.

ROLES

There will be three types of roles in a Cloud Agile Framework (CAF) project:

- Product Owner (or customer): Will own the requirements and prioritize them
- Scrum Master: Will own the process and ensure that teams remain unhindered and productive
- Team: Will be composed of cross-functional roles like Architect, Designer, Developer, and Tester. They will 'self organize' in that, they will work on any responsibility that they could effectively fulfil and benefit the project, at any given time

WORKFLOWS

A Cloud Agile Framework (CAF) project's workflow will be described in terms of units of work (Work Components) enacting across four phases.

INCEPTION PHASE

The objectives of Inception phase are to set up the project, define initial scope, and understand architectural choices.

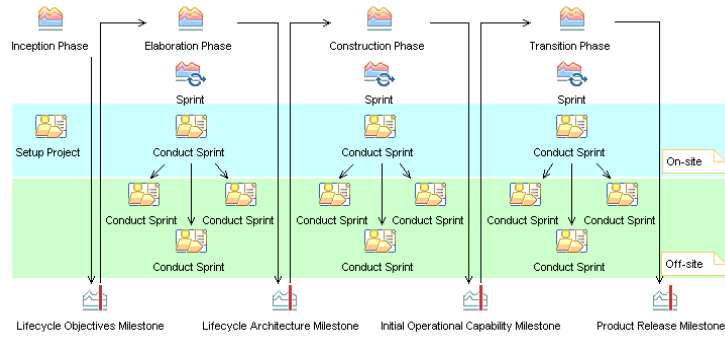


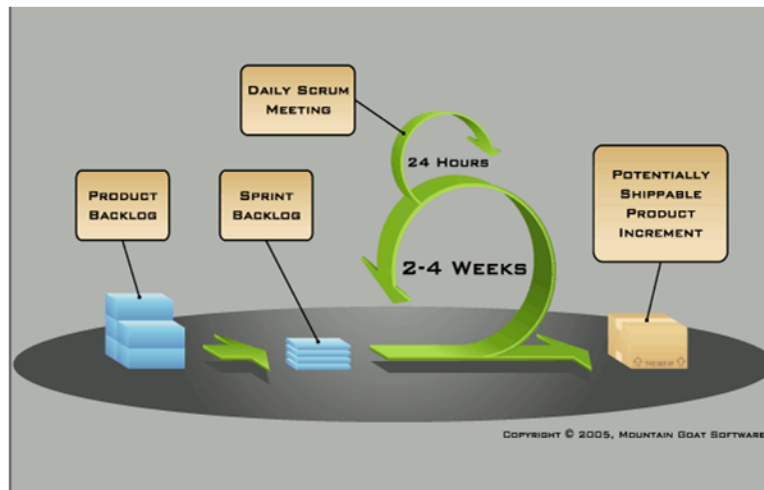
Figure 9 – Cloud Agile Framework (CAF) Workflow.

Work Component: Setup Project will be enacted in Inception as follows.

- The project will be set up. This will include finalization of the team distribution in terms of units of up to 9 member each, deployed on-site and off-site / offshore as necessary
- An initial list of features will be collected in a User Story [6] format as a prioritized list called the Product Backlog. It will include business stories as well as supplemental stories reflecting Cloud related requirements
- The architecture will also be discussed / discovered. This will include leveraging PaaS resources, and architecting the application resources and services to be assembled from the Cloud

ELABORATION PHASE

The objectives of Elaboration phase are to ensure that the teams are able to work in harmony (along with their tooling), to produce an initial, incremental working application to demonstrate that right kind of features are being captured, and architectural risks (incl. those pertaining to Cloud development) have been understood.



Scrum

An established agile management process.

Benefits

Business Driven, developer run, agile management.

Figure 10 – CAF Sprints (based on Scrum).

Work Component: Conduct Sprint will be enacted by each Team of up to 9 members within fixed duration iterations called Sprints.

The following describes a typical 10 day Sprint (ref. figure above):

- Day 1: Sprint Planning Meeting
 - User Stories from the Product backlog will be prioritized primarily by architectural risk and framework building considerations and then by business need to see an early result. The stories will be listed in a Sprint Backlog
 - Team members will list tasks for each story to be tackled in the Sprint, take task ownerships, and estimate durations
- Day 2 – 9: Assembly, Development, and Testing, Daily Scrum Meetings
 - The Team will assemble the application using resources from the Cloud, develop connections and code as necessary, and conduct testing
 - Every morning the Scrum Master will conduct a 15 min standing Daily Scrum Meeting. Each team member will convey what they did the previous day, what they planned to do on that day, and their impediments which could include any Cloud connectivity / capacity/ capability related issues. The Scrum Master will spend the day to resolve these issues
- Day 10: Sprint Review and Retrospective Meetings
 - The Team will conduct a Sprint Review Meeting in which they will show the Product Owner a “potentially shippable” increment of application functionality
 - The Team will conduct a Sprint Retrospective Meeting in which they will collect feedback and adapt the process

CONSTRUCTION PHASE

The objective of Construction phase is to bring the application to a ‘Done’ state as agreed by business, pending User Acceptance and handover.

Work Component: Conduct Sprint will be enacted by each Team of up to 9 members within fixed duration Sprints, in Construction, as described for the Elaboration phase.

One difference will be that the stories will be prioritized primarily by the Product Owner. The product will be ready for transition to the user community.

TRANSITION PHASE

The objective of Transition phase is to bring the application to a ‘Done’ state as agreed by business, subsequent to User Acceptance, transition the application, and conclude the project.

Work Component: Conduct Sprint will be enacted by each Team of up to 9 members within fixed duration Sprints, in Transition, as described for the Elaboration phase.

The stories will be prioritized primarily by the Product Owner. The product will be transitioned to the user community.

WORK PRODUCTS

Three key management work products within Cloud Agile Framework (CAF) will be:

- The Product Backlog: a prioritized list of desired features
- The Sprint Backlog: a prioritized list of tasks that the Team has identified for the current Sprint
- The Increment of Product Functionality: delivered at the end of each Sprint

Additional work products pertaining to the technical practices will also be developed as needed.

ACCESS

Cloud Agile Framework (CAF) process will be published as a website. It will be authored using Method Authoring Method on the EPF Composer tool [4].

Currently method content of the AgileCSC process is being leveraged to define Agile practices including iRise visualization within Catalyst. It is proposed that the Cloud Agile Framework (CAF) website be similarly offered across CSC's portal as a part of our Catalyst process framework. Additionally it is proposed that the Cloud Agile Framework (CAF) website published from EPF Composer be attached to a PaaS based process orchestration tool so as to make it active.

DEVELOPMENT TOOLS

As evidenced by the preceding sections, development for the Cloud brings its own unique set of requirements. The scenario is further complicated by different types of Clouds offered by various vendors – each with its own set of development tools, IDEs, SDKs and deployment considerations.

Cloud environments such as Amazon EC2 are great for the “Cloud Hosted” applications as described earlier. You do not require special tools to develop for EC2 but it does provide extensive mechanisms to deploy, manage and monitor your applications. In addition, it also provides APIs to interact with EC2 infrastructure. These APIs are very useful for any customizations or extensions that may be useful or necessary for your applications. There are third-party tools such as RightScale that run with EC2 and provide significant Cloud environment and workload management capabilities.

As we move towards the other end of the “multi-tenancy” spectrum i.e. towards the “Cloud Optimized” applications, development tools tend to vary in capabilities and complexity. Let's look at a few such tools from key Cloud providers such as IBM, Microsoft, and Google. Needless to say, there are a host of other Cloud providers with a wide range of capabilities and features but a full discussion on this topic is beyond the scope of this document.

IBM

IBM is now offering **IBM Rational Software Delivery Services** [11] for Cloud Computing. This capability set provides comprehensive application lifecycle management (ALM), which you can provision as a service on the Cloud. These services provide you with tooling specifically designed to handle different workloads and development and delivery activities and are based on the IBM Jazz™ initiative. Although these tools/services are optimized to run in private Cloud environments, they can also be extended, in some cases, to the Public Clouds as well. Examples of services on the Cloud include collaborative application management, quality management, test management, test lab management, requirements definition and management, performance measurement and management, and delivery automation.

The services are offered in three flexible delivery models:

- **IBM CloudBurst for Development & Test** – a pre-integrated set of hardware, storage, virtualization and networking, with a sophisticated built-in service management system to allow clients to rapidly deploy an internal/private Cloud environment

- **IBM Smart Business Development & Test Cloud** – a private Cloud service behind the client's firewall, built and managed by IBM. The service now includes enhanced capabilities for collaborative Cloud development using Rational Software Delivery Services for Cloud Computing
- **IBM Smart Business Development & Test on the IBM Cloud** – application development and test featuring Rational Software Delivery Services for Cloud Computing over IBM's secure, scalable Cloud

GOOGLE APP ENGINE

Google App Engine is the development, test and deployment platform on Google's Cloud environment.

Figure 11 – Google App Engine.

Google App Engine offers a platform to run web applications on Cloud and secure-connect with enterprise data.

- An Eclipse-based development IDE that helps develop and deploy Java/Python based apps on Google App Engine
- Google App Engine
 - Platform as a Service - Java/Python
 - Memcache - distributed in-memory data cache
 - Built in URL Fetch and image manipulation support
- App Engine datastore: A Schema-less object data store
- Easy and tight integration with Google Apps (Email, Calendar, Collaboration & Docs)
- Postini – Email Security and archiving
- Secure Data Connector – Secure connect with corporate infrastructure for easy integration with on-premises systems and data

MICROSOFT AZURE

Similar to Google App Engine, Microsoft Azure is a Cloud-based development, test and deployment environment, primarily targeted towards the Microsoft .NET applications. Extensions are available for other languages and platforms as Eclipse IDE plug-ins.

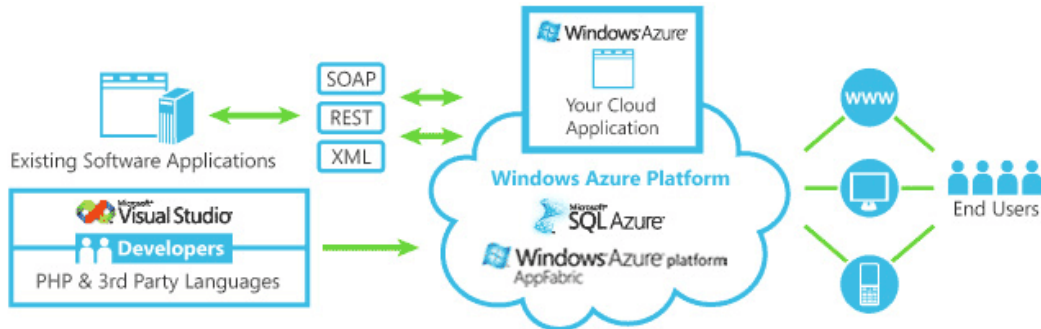


Figure 12 – Microsoft Azure [12].

Windows Azure offers a platform that is relatively easily implemented alongside an Enterprise environment that has significant number of Windows based applications.

- The recently released Visual Studio 2010 provides excellent integrated development capabilities with Azure. Prior versions can use Azure add-ins to interact with Azure.
- Windows Azure: operating system as a service
- Microsoft SQL Azure: fully relational database in the Cloud
- Windows Azure platform AppFabric: makes it simpler to connect Cloud and on-premises applications

ECLIPSE PROCESS FRAMEWORK COMPOSER

The open source Eclipse organization has produced a tool called Eclipse Process Framework Composer (EPF Composer) [4] using which processes can be authored. The latest version of the tool supports practice based process authoring as we have proposed for Cloud Agile Framework (CAF) (AgileCSC was also authored using EPF Composer). A process library is created to include its method content in terms of roles, workflows, work products, and guidance. One can attach this content within various process configurations which can be individually published. The outcome is a static process website which contains hyperlinked information and guidance on the process. It is possible to inherit previously authored content. The content and tool can be used under a free Eclipse Public License.

Covansys, which is now a part of CSC MSS, was a committer and active participant in authoring the EPF Composer tool and its first library, the OpenUP process.

EMERGING INNOVATIONS

We anticipate that the tools and process landscape around Cloud-based development will continue to evolve as Cloud computing/PaaS technologies mature, standards emerge, vendors/platforms consolidate, and new lessons are learned in integrating with Cloud environments.

However, in our discussions with various enterprises, there appears to be a real need for a framework and associated toolset that can help “broker” or “orchestrate” services between Cloud providers and the enterprise. This “orchestration layer” will sit between the traditional data center and various Cloud service providers and would comprise of algorithms that determine the best Cloud service for a particular workload based on lowest cost, highest performance and SLA agreements.

As enterprises adopt more and more Cloud based applications, it is anticipated that they will need to interact with more than one Cloud provider for reasons of service capability/feature, reliability/availability, data security, compliance and regulatory requirements. Business processes supported by Cloud-based applications will need to “traverse” multiple Clouds to accomplish their business requirements.

Although there are some vendors (e.g. Cohesive FT) offering secure and reliable connectivity between Clouds and a management console to manage and monitor environments and associated workloads, a mature set of development tools that can facilitate cross-Cloud development and testing is yet to emerge and this is definitely an area of growth in the future.

We also anticipate that the different nature of Cloud application development will catalyze changes in software development process frameworks; which will become more agile, practice-based, and active.

System integrators such as CSC can expect to maintain a leadership position in Cloud development by either investing in the development of frameworks that broker or orchestrate services between Cloud providers (with associated application development process frameworks) or seeking out capable vendors for partnerships/alliances.

WHAT IS IN IT FOR CSC?

Today Cloud computing is emerging as a progressively crisply defined collection of IT and software development areas as discussed above in NIST's definition of Cloud Computing.

Of its three service models, the focus so far has been on SaaS and IaaS. However we see a lot of potential in PaaS. It is being enabled by many vendor offerings at present, e.g. Salesforce.com's Force.com, which provides the ability to quickly develop business applications and, if desired, integrate them with Salesforce CRM, and Google App Engine, which provides a secure, scalable, Cloud-based development and runtime environment for Java and Python applications. Microsoft has also joined this space with its Windows Azure platform. Then there are start-ups such as Rollbase. The Rollbase platform allows companies to create enterprise-level SaaS applications using do-it-yourself tools in a standard Web browser. Such offerings bring the power of rapid application development to businesses. They also offer an on-demand application sharing service where users can browse, test and install business applications built and published by experienced users and partners.

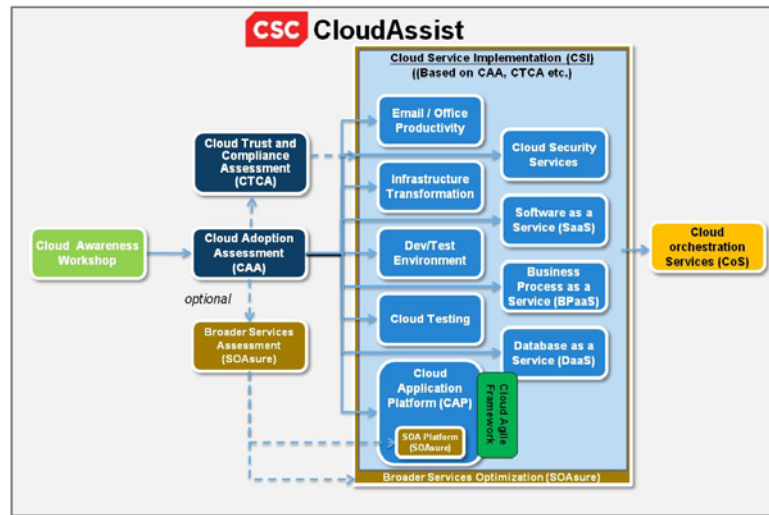


Figure 13 – CSC CloudAssist.

For CSC this implies that though the PaaS service model might not seem very competition rich today, it will become so soon. That will also mean widening of this playing field. With an offering such as Cloud Agile Framework (CAF), we will be better positioned to reap business benefits at that time.

SERVICE OFFERING

The Cloud Agile Framework (CAF) can be easily integrated with CSC CloudAssist offerings by providing an extensible process framework around the Cloud Application Platform services.

Specifically, CAF will provide the following service offerings:

- **Advisory services on Cloud development process framework** – This will provide consultation and guidance to enterprises on executing Cloud application projects with predictable and repeatable results. This will include any customizations necessary for target Cloud platforms (e.g. Google App Engine, Microsoft Azure etc.).

Enterprises struggling to adapt to the new development paradigm around Cloud applications will benefit tremendously with this offering. CAF will represent a balanced mix of our learnings on various Cloud projects and the supporting processes and templates.

- **Cloud application projects delivery enabler** – As CSC takes on more and more Cloud development projects, CAF will be very useful to help plan and deliver such projects. Accelerators such as a customized Cloud application development project plan mapped to client specific processes and templates will give teams a great headstart. Even in pre-sales situations, CAF can represent a significant differentiator.

CAF DELIVERY ENABLERS

- **Cloud Project Plan Template** – A comprehensive project plan template that includes all requirements, design, development, deployment, and management aspects of a Cloud based project along with the required roles and responsibilities.
- **Cloud Reference Architectures** – This could target various Cloud platforms such as Google App Engine, Microsoft Azure etc. This should also focus on specific security and integration requirements. It will be very useful for planning tasks and resources depending on the target Cloud platform.
- **Cloud Application Security Recommendations** – A focussed set of guidelines that can help enterprises understand the security risks of putting their applications on the Cloud and recommendations on mitigating them. Client specific recommendations will translate into actionable tasks and activities for the project team.
- **Application Scalability Recommendations** – Although applications can hypothetically enjoy “infinite” scalability by leveraging a Cloud environment, applications must be specifically built to gain such scale on the targeted Cloud platform. This set of recommendations and associated tasks will focus on application requirements and options for enabling such scalability.

PROPOSED ROADMAP

The roadmap to create CAF and the corresponding service offering within CSC will also spawn another roadmap, that of CAF scaling itself beyond just application development, e.g. for infrastructural, or IaaS projects. The CAF goals of resourfulness, adaptability, and configurability will help make this possible.

CAF ROADMAP

We propose the following roadmap for creating a viable Cloud application development process within CSC, with corresponding service offering that would enable us to monetize the process.

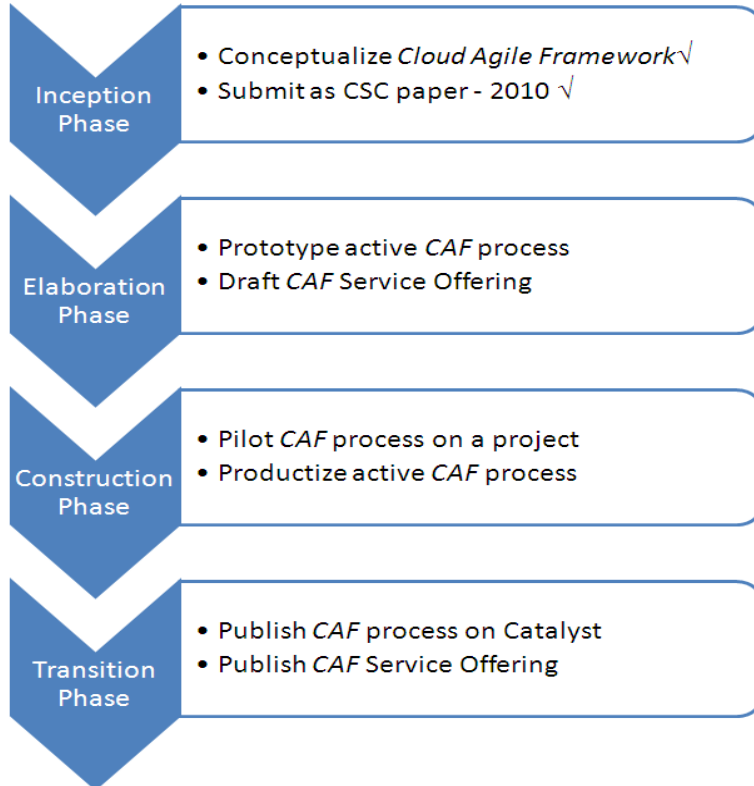


Figure 14 – Cloud Agile Framework Service Offering Roadmap.

The roadmap leverages a four phased approach which is also a part of the proposed Cloud Agile Framework (CAF) process. We propose that CSC plans to accomplish the activities by end of 2010 in order to gain leadership in this market space.

INCEPTION PHASE

Conceptualize Cloud Agile Framework (CAF)

Based on our experience with Cloud computing and software development processes, we conceptualize the Cloud application development process. This has been done, as presented here.

Submit as CSC paper - 2010

We propose the process as a CSC paper 2010 submission. This is work in progress and will be done when this paper is submitted on May 17, 2010.

ELABORATION PHASE

Prototype active CAF process

We author key aspects of the process as method content using the EPF Composer tool, and attach the published output to a suitable tool to make it orchestrable and active.

Draft CAF Service Offering

Draft a formal CSC service offering leveraging the Cloud Agile Framework (CAF) process.

CONSTRUCTION PHASE

Pilot CAF process on a project

Secure a project using the draft service offering as above and pilot the Cloud Agile Framework (CAF) process on it.

Productize active CAF process

Based on experience gained from the pilot project, productize the Cloud Agile Framework (CAF) process, by finalizing its content, and creating training plus other collateral.

TRANSITION PHASE

Publish CAF process on Catalyst

Publish the CAF process on CSC's Catalyst process framework.

Publish CAF Service Offering

Complete the Cloud Agile Framework (CAF) based service offering and publish it within CSC, along with training etc.

SCALABILITY OF CAF

CAF addresses application development under PaaS at present, but will need to soon scale itself, e.g. as a process framework for infrastructural, or IaaS projects. This need will be addressed through our proposal that CAF be authored as a process framework using the EPF Composer tool.

The tool offers a clear demarcation between method content and process. We are proposing to leverage this for scalability, and also, extend it through the use of Cloud based process orchestration tooling to make CAF an active process as per its goal.

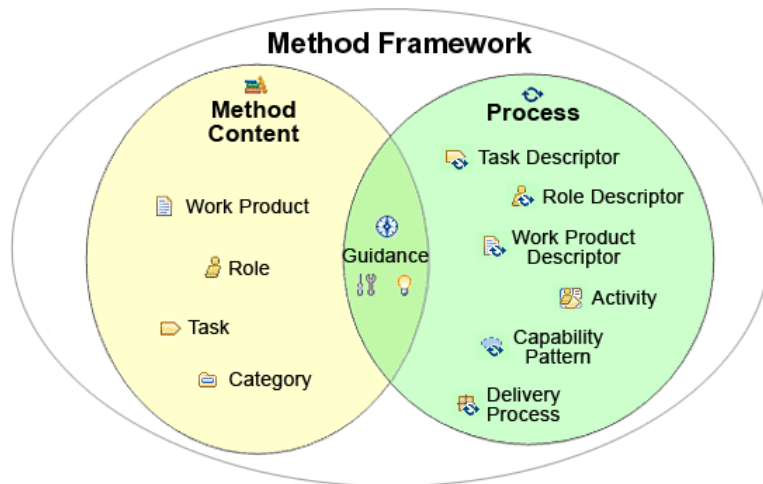


Figure 15 – EPF Composer Topology.

EPF COMPOSER METHOD CONTENT

EPF Composer allows one to author a process' method content as rich text pages capturing information on roles, tasks, work products, and guidance. It allows categorization of these in terms of custom packages. One way to categorize the content is to define horizontal practices, e.g. "Cloud Application Development" (ref. CSC's "Cloud Application Development" Practice above).

The tool offers the capability of extending (aka inheriting and changing or replacing) existing method content in order to create derived method content. This implies that once CSC creates CAF method content on the tool, it would be relatively easy to extend it in future to produce more process elements in the library e.g. for extending it to an IaaS-centric practice, or providing tool mentors from new vendor offerings, etc.

EPF Composer Process

The tool allows configuration of software processes from the method content. E.g. the CAF method content would contain the “Cloud Application Development” practice, descriptors (pointers) to whose elements would be configured into a CAF based Cloud application development process. This process would be published as a static website.

When in future CSC wants to publish another process that is more IaaS focussed, we would use the IaaS based practice created through extension of the PaaS based “Cloud Application Development” practice created now, create the corresponding IaaS-centric process, and publish it as a website.

PROCESS ORCHESTRATION

Processes published by a tool like EPF Composer are static in that, they are available as hyperlinked web pages; and unless teams take interest in keeping them up-to-date, they become unused and obsolete.

In order to avoid this from happening to CAF, we recommended that it should be an ‘active’ process, as a goal (ref. the discussion above on *Access*). This implies that a project manager charged with developing a Cloud application would not only just refer to its static website but would be able to instantiate it as a project that includes configurable tasks with all necessary guidance and templates, along with collaboration facilities. This will ensure constant ground up streamlining and currency of the process. We mention this as a part of CAF roadmap since it would be an important step in keeping it current and relevant so that it is successfully institutionalized within CSC. Such institutionalization is a critical challenge for any process as it involves huge organizational change management.

CONCLUSION

Just as enterprises are adopting Cloud Computing for their infrastructure oriented benefits, the next wave in Cloud Computing is expected to focus on applications and therefore PaaS. It is crucial to understand the requirements and challenges of a Cloud application to be able to fully benefit from a Cloud environment. They are different from traditional development practices and norms because of the fundamental differences in Cloud architecture. Architects and designers must adapt to a new mindset to develop for the Cloud. Correspondingly, development processes and practices must also evolve to address the inherent expectations of agility and flexibility of a Cloud environment.

CSC can expect to gain tremendously by investing into an Agile framework that supports Cloud development. It will help the organization to be ready and prepared as our customers look for guidance and help in developing these new generations of applications. It will enable CSC to maintain a significant edge over its competitors by providing a comprehensive framework and methodology for Cloud oriented applications. Last but not the least, as CSC delivery teams take on more Cloud development projects, the same framework can greatly enhance the chances of predictable and repeatable success.

ACKNOWLEDGMENTS

- Ashok Madhuranath for his invaluable contributions and insight into Cloud applications and development challenges.
- Max M. Ramsay for his whitepaper “Platform as a Service – A working definition”.
- Sreedhar Kajeepeta for his constant guidance and encouragement on Cloud Computing topics.

REFERENCES

- [1] 2009. Effectively and Securely Using the Cloud Computing Paradigm. From **NIST, Information Technology Laboratory**. Peter Mell, Tim Grance. Retrieved May 14, 2010, from <http://csrc.nist.gov/groups/SNS/Cloud-computing/Cloud-computing-v26.ppt> .
- [2] Rational Unified Process (RUP). From **IBM**. Retrieved May 10, 2010, from <http://www-306.ibm.com/software/awdtools/rup/>.
- [3] 2004. RUP Work Components. **IBM**. Retrieved May 10, 2010 from <http://www.ibm.com/developerworks/rational/library/5317.html>.
- [4] Eclipse Process Framework Project (EPF). From **The Eclipse Foundation**. Retrieved May 10, 2010, from <http://www.eclipse.org/epf/>.
- [5] **Manifesto for Agile Development**. Retrieved May 10, 2010, from <http://agilemanifesto.org/> .
- [6] What is Scrum? From **Scrum Alliance**, Retrieved May 10, 2010, from http://www.scrumalliance.org/pages/what_is_scrum.
- [7] Extreme Programming: A gentle introduction. From **Extreme Programming**. Retrieved May 10, 2010, from <http://www.extremeprogramming.org/>.
- [8] 2008. Nurturing Information Systems Development Projects as Agile Ecosystems. **CSC**. Kirti Vaidya. WMSCI 2008.
- [9] In **Merriam-Webster Online Dictionary**. Retrieved May 10, 2010, from <http://www.merriam-webster.com/>.
- [10] **Lean Software Development**. Retrieved May 12, 2010, from <http://www.poppendieck.com/>.
- [11] Leveraging the cloud to transform software delivery. From **IBM**. Nov 2009.
- [12] **Windows Azure Platform**. Retrieved May 12, 2010, from <http://www.microsoft.com/windowsazure> .
- [13] Platform as a Service – A working definition. Max M. Ramsay.



Worldwide CSC Headquarters

The Americas

3170 Fairview Park Drive
Falls Church, Virginia 22042
United States
+1.703.876.1000

Europe, Middle East, Africa

Royal Pavilion
Wellesley Road
Aldershot, Hampshire GU11 1PZ
United Kingdom
+44(0)1252.534000

Australia

26 Talavera Road
Macquarie Park, NSW 2113
Australia
+61(0)29034.3000

Asia

139 Cecil Street
#06-00 Cecil House
Singapore 069539
Republic of Singapore
+65.6221.9095

About CSC

The mission of CSC is to be a global leader in providing technology enabled business solutions and services.

With the broadest range of capabilities, CSC offers clients the solutions they need to manage complexity, focus on core businesses, collaborate with partners and clients, and improve operations.

CSC makes a special point of understanding its clients and provides experts with real-world experience to work with them. CSC is vendor-independent, delivering solutions that best meet each client's unique requirements.

For more than 50 years, clients in industries and governments worldwide have trusted CSC with their business process and information systems outsourcing, systems integration and consulting needs.

The company trades on the New York Stock Exchange under the symbol "CSC."

Disclaimer

The information, views and opinions expressed in this paper constitute solely the authors' views and opinions and do not represent in any way CSC's official corporate views and opinions. The authors have made every attempt to ensure that the information contained in this paper has been obtained from reliable sources. CSC is not responsible for any errors or omissions or for the results obtained from the use of this information. All information in this paper is provided "as is," with no guarantee by CSC of completeness, accuracy, timeliness or the results obtained from the use of this information, and without warranty of any kind, express or implied, including but not limited to warranties of performance, merchantability and fitness for a particular purpose.

In no event will CSC, its related partnerships or corporations, or the partners, agents or employees thereof be liable



to you or anyone else for any decision made or action taken in reliance on the information in this paper or for any consequential, special or similar damages, even if advised of the possibility of such damages.

Copyright © 2011 CSC. All rights reserved