PhUSE 2016

Paper AD11

Clinical SAS Application Development in a GRID Environment

Giri Balasubramanian, PRA Health Sciences, Chennai, India
Edwin Ponraj Thangarajan, PRA Health Sciences, Chennai, India

ABSTRACT

Application Development Methodologies and Process has been evolving over a period of time with the changing need & demand from business community. Transformation of SDLC Methodologies has started from Waterfall model in 1980's to Agile in 1990's and then to Lean model and DevOps Implementation with Agile Principles. Each of the models has its own merits & demerits and is judged based on agility that is needed for a kind of business. The session would be dwelling these models that are prevalent and bring about a case study to demonstrate the agility of recent models for development of SAS® based application development using different programming languages such as C# which needs to be deployed in a SAS® GRID Environment. Case study would focus on how a most recent methodology say DevOps is adopted to ever changing needs of users in building a SAS® based application and what kinds of toolsets, processes to be in place while still adhering to Agile principles for continuous integration, delivery & deployment of solutions.

INTRODUCTION

SDLC, Software Development Life Cycle is a process used by software industry to design, develop and test high quality software's. SDLC aims to produce high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. SDLC consists of the models and methodologies that development teams use to develop the software systems, which the methodologies form the framework for planning and controlling the entire development process. A Systematic development process is essential to govern the entire process in order to ensure that the end-product comprises of high degree of integrity and robustness, as well as user acceptance.

It is necessary to bear in mind that a model is different from a methodology in the sense that the former describes what to do whereas the latter, in addition, describes how to do it. So a model is descriptive whilst a methodology is prescriptive.

Broadly, the SDLC models have been split between traditional and agile development based. Traditional models starts with Waterfall, V-Model, Iterative, Prototyping whereas Agile based varies from Scrum, Kanban and Extreme Programming. In the recent past this has extended to include Lean Model and, DevOps. This paper would be highlighting the SDLC models and dwell more into the application of agile techniques for different types of development work.

OVERVIEW OF MODELS

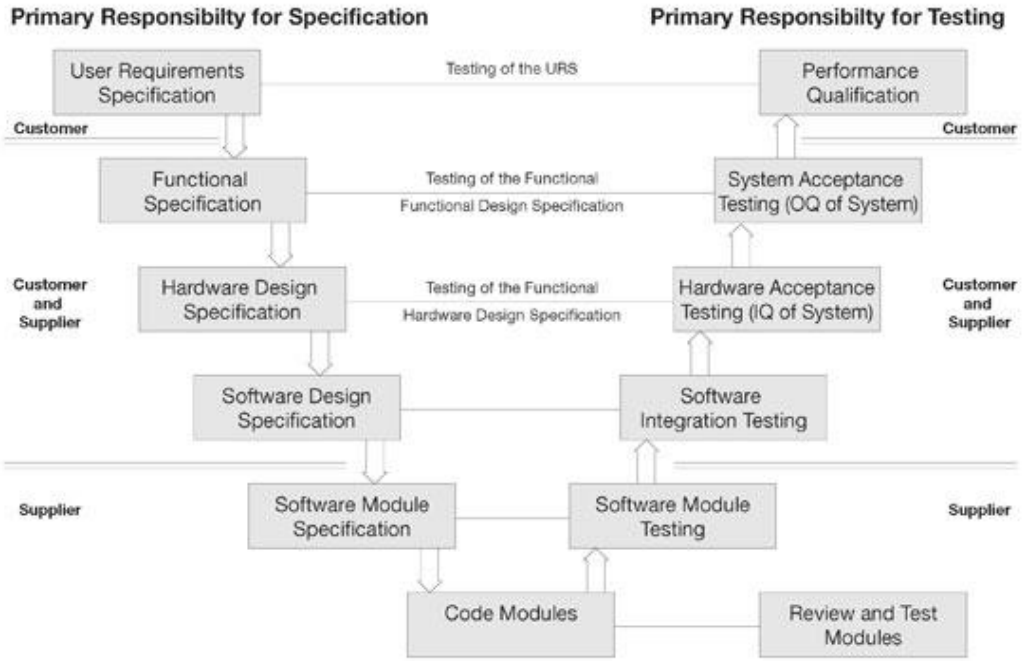
WATERFALL MODEL

Waterfall is a linear - sequential development approach, in which development is seen as flowing sequentially through phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance. The sequential phases in waterfall model are Requirement Gathering & Analysis, System Design, Implementation, Integration & Testing, Deployment of System, and Maintenance. Each of these phases would start with an input from the earlier phase.

Waterfall model was tweaked by the users into different forms to include iterative model into the sequential process.

V-MODEL

V-Model is an extension of Waterfall model and is also known as Verification and Validation Model. Execution path is sequential as in Waterfall model. Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development in V-model. Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence, higher chance of success over the waterfall model.



PROTOTYPE MODEL

Prototype Model is an extension of Waterfall in downstream except in the early phase a prototype is created and reviewed with the sponsor prior to venturing into development work. Prototype model is adopted where the new concept or solution is developed to determine the actual need of the sponsor and finalized software requirement specification is evolved. Based on that, actual system is developed using water fall model. At times, it is also referred as Rapid Application Development.

ITERATIVE MODEL

Iterative Model enables the users to develop the system with partial specification and then proceed with further development once the next level of specifications is refined and available. This process is then repeated, producing a new version of software for each of the cycle. The working version of the system is available early in the process and makes it less expensive to implement changes.

SPIRAL MODEL

One of the most flexible SDLC methodologies, the Spiral model takes a cue from the Iterative model and its repetition; the project passes through four phases over and over in a “spiral” until completed, allowing for multiple rounds of refinement. This model allows for the building of a highly customized product, and user feedback can be incorporated from early on in the project. But the risk you run is creating a never-ending spiral for a project that goes on and on. Spiral Model takes care of risk analysis as part of it primary phase and tries to assist in avoiding risk items especially mission-critical projects.

AGILE MODEL

Agile is an evolution addressing the issues behind various models prevalent in 80’s and 90’s and is a blend of iterative & incremental model.

Following twelve principles are the characteristics that differentiate a set of agile practices from a heavyweight process. Development frameworks that follow these values and principles are known as agile techniques.

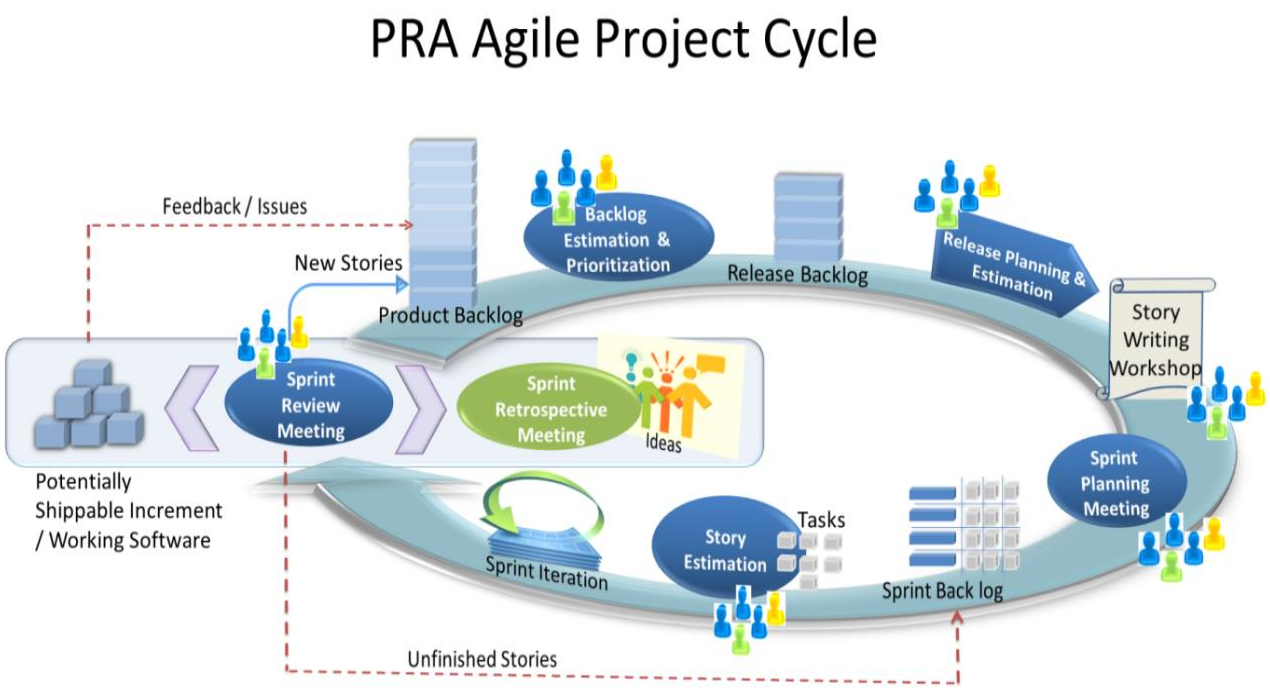
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.

- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Many developers have lived through the nightmare of a project with no practices to guide it. The lack of effective practices leads to unpredictability, repeated error, and wasted effort. Customers are disappointed by slipping schedules, growing budgets, and poor quality. Developers are disheartened by working ever-longer hours to produce ever-poorer software. The professional goal of every software developer and every development team is to deliver the highest possible value to employers and customers. Yet our projects fail, or fail to deliver value, at a dismaying rate. The principles and values of agile software development were formed as a way to help teams break the cycle of process inflation and to focus on simple techniques for reaching their goals.

Agile promotes continuous iteration of development and testing throughout the software development life cycle of the project. There are different methodologies of Agile that can be adopted based on the scope of project. To name few are, Scrum, Extreme Programming, and Kanban. More recently, Lean Development came into being, which is an application of Agile and DevOps principles at Scale.

The figure below presents a more detailed view of what we call agile lifecycle which extends Scrum's construction lifecycle followed at PRA Health Sciences. In addition to this being a more detailed view of the lifecycle, there are several interesting aspects to this lifecycle. This life cycle is extended to all development and support activities within the Information Technology group giving flexibility to resource management and keeping the business involved as part of the development process by way of transparency.



CASE STUDY - I

Clinical SAS® Application Development in SAS® GRID environment is considered here for the case study to demonstrate the agility of adopting DevOps Implementation using agile principles.

Clinical SAS® Application uses more than one language for the development work namely C# which is supported at the backend by an Oracle database and uses the SAS® GRID & Metadata server for the load distribution and single sign on capability.

Clinical SAS® based Application Development is designed to handle the following business capabilities:

- Extraction of Clinical Trial Data from different sources
- Data Standardization and generation of Submission Data Standards as per CDISC Guidelines
- Creation of Analysis datasets as per Analysis Data Model
- Analysis & Reporting (Creation of Tables, Listing, Figures)

The development methodology was revisited by PRA based on the learning from the user community on:

- Usability,
- Business expectation on the availability,
- Continuous process improvement for our internal business.

It was decided to retain the agile principles since there were benefits that were seen post adoption of the model since 2013. While doing so, PRA was able to bring in UXDD – User Experience Driven Development model with DevOps Implementation methodology. This shift is towards the Lean Development model.

DevOps is a set of measures undertaken by Developers & Operations team to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.

UXDD – USER EXPERIENCE DRIVEN DEVELOPMENT

UX Designers are brought in to develop a UI since the designers see their validity as coming from the user research that informs their design. The UX Designers are supported by a Design team comprising members from the business users who would play a role to validate the UI sketch designed. UX Driven Development works on the following manifesto:

- Collaborative design over designing on an island
- Solving user problems over designing the next “cool” feature
- Measuring KPIs over undefined success metrics
- Applying appropriate tools over following a rigid plan
- Nimble design over heavy wireframes, comps or specs

DEVELOPMENT MODEL, METHODOLOGY, PROCESS AND DEPLOYMENT

Based on the assessment & need, it was decided to split the activities under different phases:

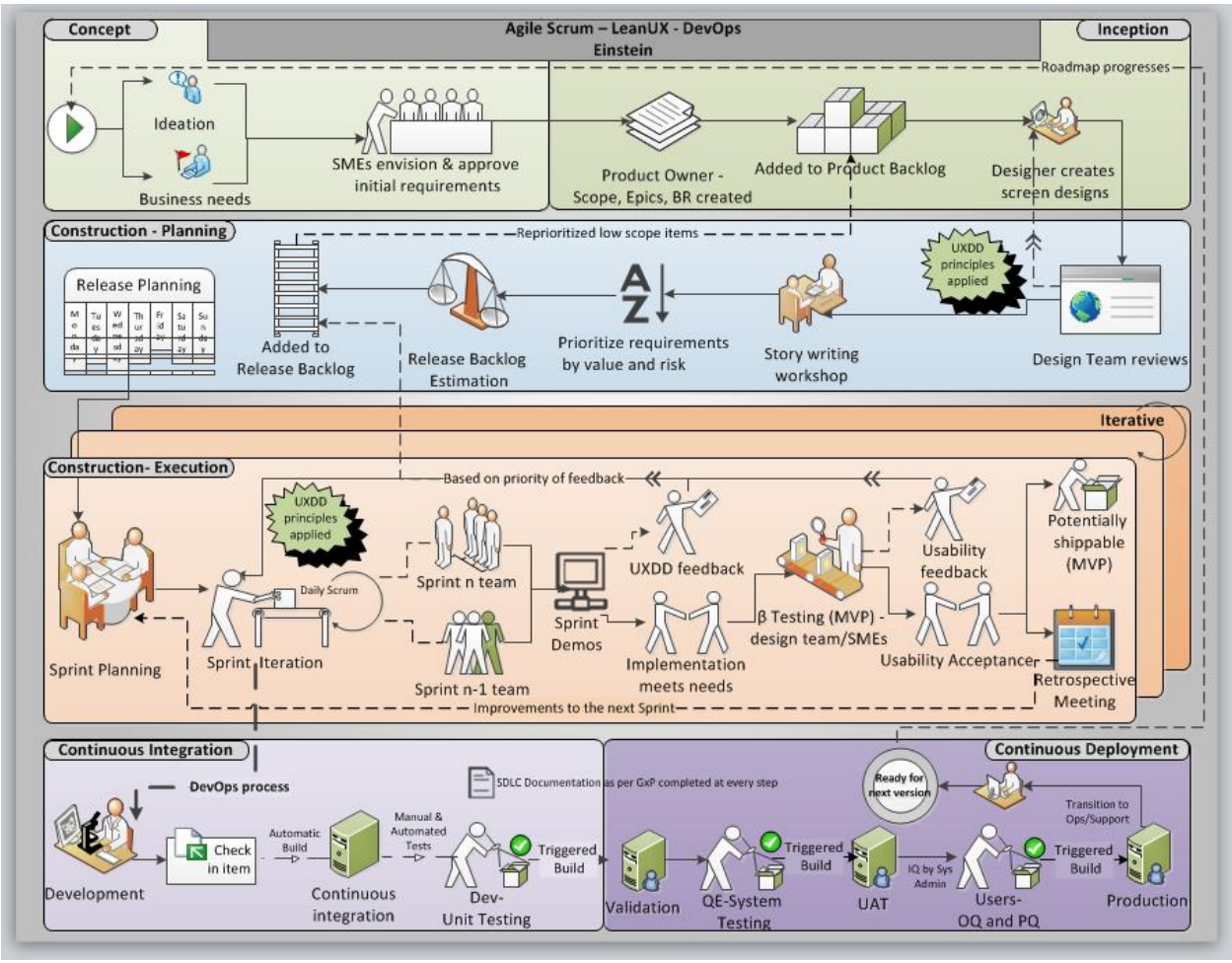
- **Concept / Inception**
  It is an ideation phase where the SMEs from business would be involved in proposing new requirements as well changes to the existing ones aligning with the sponsor & regulatory needs. Design Team is formed from the business who engages with UX Designer closely to come up UI templates to meet the expectations and vetted with SMEs. Outcome of this phase is the identification of business requirements, user stories, UI Design, UI User flow sequence with mock live data and acceptance criteria.

- **Construction – Planning**
  Agile –Scrum gets kicked off as part of the planning phase whereby the user stories are analyzed, sequenced, prioritized as part of release back log. Items that are not part of release backlog are pushed to product backlogs. Acceptance criteria are vetted along with the information on business value and benefits. The business requirements are grouped under Minimum Viable Products based on the business value & benefits and is considered as part of milestone planning.

- **Construction – Execution / Development**
  As part of agile-scrum methodology, Sprint analysis & planning is initiated by the scrum master. Sprint iterations are planned as part of the development effort and sprint demos are planned with design team. The feedbacks from design team are looped back to the business requirements & user stories and decided for inclusion as part of sprint planning & iteration. This sequence of sprint work would be repeated to complete the identified MVPs and released to beta environment for verification & usage by design team. Feedback from design team and SMEs are captured as part of learning and might lead to correcting the interface or process which have impact on the Business Requirement and the respective user story. Each MVP / Milestone once verified by the design team leads to a potentially shippable version as part of the development is concerned.

- **Continuous Integration & Deployment**
  DevOps implementation is added to the Agile-Scrum based development to bring in CI. Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build & test execution, allowing teams to detect problems early. Each integration cycle is verified by an automated build (including test) to detect integration errors as quickly as possible. Scrum teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly. CI clubbed with Continuous Deployment (CD) facilitates the release of product to production quickly. The cycle of deployment can be decided based on the nature of the application and its impact in the business cycle. Continuous Delivery is the natural extension of Continuous Integration: an approach in which teams ensure that every change to the system is releasable, and that we can release any version at the push of a button.

Each of the phases with its activities has been depicted in the diagram below for better understanding. The Model, Methodology, Process and Deployment taken up at PRA are blend of Agile driven with Lean UX and DevOps implementation.



It was assessed in detail while going about the model, methodology, process & deployment for continuous product development initiatives that should meet the user experience, assess the minimum viable product acceptance criteria, agile in development to accommodate the feedback with reprioritization with continuous integration, development & deployment. Hence, it was narrowed towards tailor made blend of items taking each of the benefits into account.

DevOps practices impact all three phases.

1. **Inception Phase**: During the inception phase, release planning and initial requirements specifications are done. Considerations of operations (Ops) will add some requirements for the developers, but maintaining backward compatibility between releases and having features be software switchable are two of these requirements. The form and content of operational log message impacts the ability of Ops to troubleshoot a problem. Release planning

includes feature prioritization but it also includes coordination with operations personnel about the scheduling of the release and determining what training the operations personnel require to support new release. Release planning also includes ensuring compatibility with other packages in the environment and a recovery plan if the release fails. DevOps practices make incorporation on many of the coordination-related topics in release planning unnecessary, whereas other aspects become highly automated.

2. **Construction Phase**: During the construction phase, key elements of the DevOps practices are the management of code branches, the use of continuous integration and deployment and incorporation of test cases for automated testing. There are also agile practices but form an important portion of the ability to automate the deployment pipeline. A new element is the integration and automated connection between construction and transition activities.

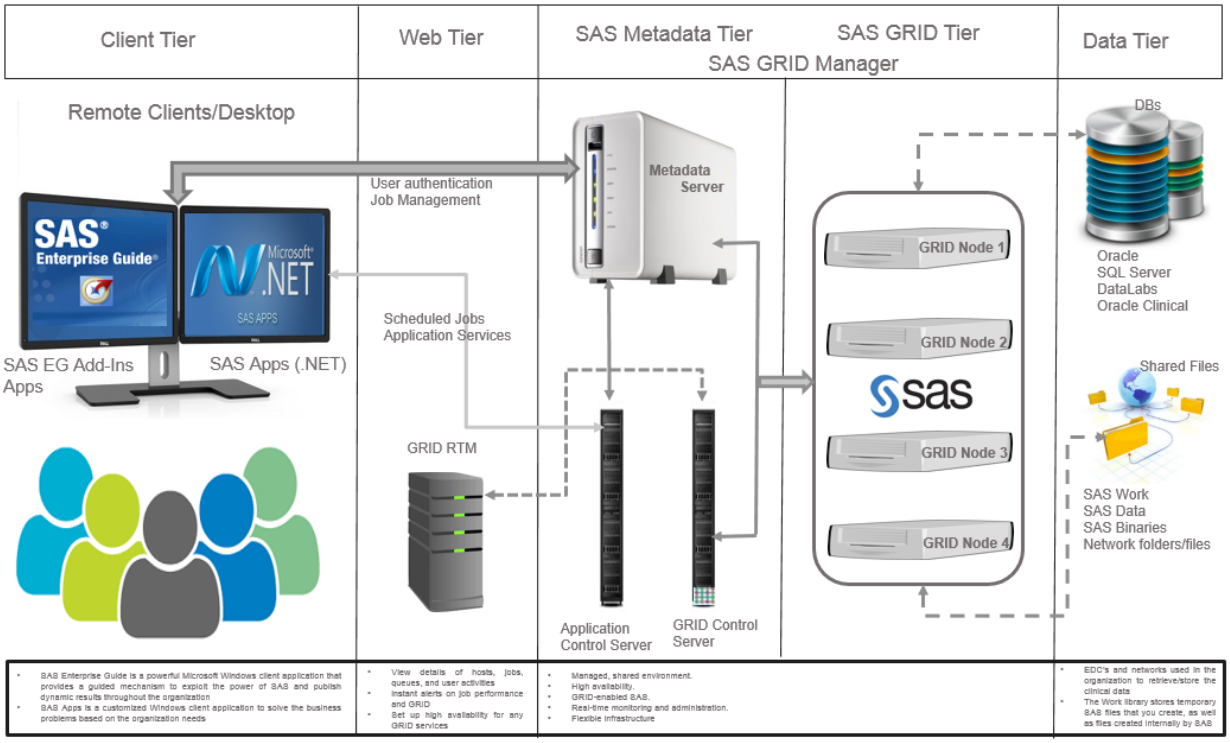
3. **Deployment Phase**: In the transition phase, the solution is deployed and the development team is responsible for the deployment, monitoring the process of deployment, deciding where to roll back and when, and monitoring the execution after deployment. The development team has a role of 'reliability engineer', who is responsible for monitoring and troubleshooting problems during deployment and subsequent execution

SAS® BASED APPLICATION DEVELOPMENT IN A GRID ENVIRONMENT

Case study presented involves a multi-language enterprise class application development using the SAS Engine as backend in a Windows Platform server. The landscape of the Development, Validation and Deployment (Production) Servers are split to house different tools like SAS EG, .NET framework, SAS Metadata server and its supporting GRID servers, database repository for storing the configuration information, and a file server to house the clinical trial data.

GRID servers are kept minimal for Development and Validation environment and are scaled to more than 4 GRID servers in a production deployment based on the work load. Citrix servers are used for running the SAS EG and .NET based application and the number of Citrix's servers in production is again based on the number of users trying to access at peak load during the day. The application is accessible in different time zones and availability of the system is considered critical for the Operations team involved as part of DevOps and importance is given on the down time for any upgrades or deployment that needs to be done as part of Continuous Deployment cycle. DevOps based implementation is considered to be a best fit for this kind of complex enterprise class development & deployment environment where the down time has to be very minimal keeping the business continuity as priority.

The system landscape of the SAS based application development in a GRID environment is depicted in the below sketch. Each of the SAS software components requires operations support to make it working and application developed is deployed in different tiers because of its enterprise class capability which can scale to handle large number of concurrent users.



TOOLS & INFRASTRUCTURE

Development toolsets & infrastructure landscape and availability play an important role for the Clinical SAS based Application Development. This is again coordinated with Ops as part of DevOps Implementation.

Requirements	Development (CI)	Validation (CI & CD)	Production (CD)
Language			
SAS®	SAS 9.4	SAS 9.4	SAS 9.4
	SAS Enterprise Guide 6.1 or later	SAS Enterprise Guide 6.1 or later	SAS Enterprise Guide 6.1 or later
	SAS Integration Technologies	SAS Integration Technologies	SAS Integration Technologies
C#	.NET Framework 4.6.1	.NET Framework 4.6.1	.NET Framework 4.6.1
	MS Visual Studio 2015		
	C# 6.0		
Tools	MS Team Foundation Server 2015	MS Test Manager 2015	Jenkins V1.642.1
	MS Test Manager 2015	Jenkins V1.642.1	Microsoft Office Tools 2013
	Oracle 11g	Oracle 11g	Oracle 11g
	Jenkins V1.642.1	SAP Crystal Report for VS2015	SAP Crystal Report for VS2015
	SAP Crystal Report for VS2015	SAS ASSIST 9.4	Internet Information Services (IIS) Manager
	Beyond Compare 3.3.12	Microsoft Office Tools 2013	
	Microsoft Office Tools 2013	Internet Information Services (IIS) Manager	
	Internet Information Services (IIS) Manager		
Automation Testing	Unified Functional Testing 12.0.1	Unified Functional Testing 12.0.1	
	Jenkins V1.642.1	Jenkins V1.642.1	

CASE STUDY - II

Another case study to be looked into and frequently used in the recent past is effective use of Custom task feature in SAS EG and SAS Add-In for Microsoft Office. Custom task can be considered a replacement of SAS/AF where it provides a simplified, focused UI to enable users to perform a task that is specific to their trial analysis or sponsor. Users need to have or develop skills in .NET Windows UI Design, develop windows executables and package them as .NET assemblies and deploying by way of Drop-In or Add-In mode in SAS EG.

Custom Task Building features enable the SAS programmers to start developing small applications of interest revolving around analytics with a good presentation layer of their own design. It enables them to share such applications within their group while working on the data to decipher or present meaningful information to their sponsors. Such applications which are globally used by a group of SAS programmers can be deployed in the common folder so that it can be used by all in a team and such work involves the support of SAS & System Administrators.

Before initiating such move, the concerned SAS programmer has to do minimum set of validation on their custom task(s) based on the nature & scope of the Add-In and type of usage across team. Such activities lead to multiple roles to be played by the SAS programmers and require an undocumented process to be in place with the support of SAS & SYS Admin. Under these circumstances, Agile Model comes into play where the methodology can be adopted by the team since it is simple. By

this way, Agile increases the team productivity and employee satisfaction. It minimizes the waste inherent in redundant meetings, repetitive planning, excessive documentation, quality defects, and low-value product features. By improving visibility and continually adopting to programmers changing priorities, agile improves user engagement & satisfaction, brings the most valuable products and features to market faster and more predictably, and reduces risk in their day to day work.

BENEFIT(S)

Case study development model being presented as part of the Clinical SAS Based Application Development provides innumerable benefits to the Business, Development and Operations team.

Key benefits are highlighted below:

Collaborative and interdisciplinary – It intends to ‘‘foster more open, collaborative, and iterative processes, and to break through the organizational red tape that can stifle creativity’’. In this case study , Lean UX team consist of a handful number of SMEs, design experts depending on the nature of project and the size of organization from different fields / disciplines – such as Clinical Programmers, Analysis Programmers, Director level experienced person, Business Analyst, designer and developer, allowing team-wide collaborations and feedback throughout the development process. Collaborations in Lean UX is not only limited among the experts. Collaborations with sponsors is also placed at the heart of practice, allowing us to have a shared understanding of problems and solutions with sponsors.

Faster time to market: It’s not uncommon for the integration and test/fix phase of the traditional phased software delivery lifecycle to consume weeks or even months. When teams work together to automate the build and deployment, environment provisioning, and regression testing processes, developers can incorporate integration and regression testing into their daily work and completely remove these phases. It facilitates avoidance of large amounts of re-work that plague the phased approach.

Higher quality: When developers have automated tools that discover regressions within minutes, teams are freed to focus their effort on user research and higher level testing activities such as exploratory testing, usability testing, and performance and security testing. By building a deployment pipeline, these activities can be performed continuously throughout the delivery process, ensuring quality is built in to products and services from the beginning.

Better products: Continuous delivery makes it economic to work in small batches. This means developers can get feedback from users throughout the delivery lifecycle based on working software. Techniques such as A/B testing enable us to take a hypothesis-driven approach to product development whereby it facilitates to test ideas with users before building out whole features. This means, DevOps can avoid the 2/3 of features built that deliver zero or negative value to our businesses.

Lower costs: Any successful software product or service will evolve significantly over the course of its lifetime. By investing in build, test, deployment and environment automation, we substantially reduce the cost of making and delivering incremental changes to software by eliminating many of the fixed costs associated with the release process.

Happier teams: Peer-reviewed research has shown continuous delivery makes releases less painful and reduces team burnout. Furthermore, when we release more frequently, software delivery teams can engage more actively with users, learning which ideas work and which don’t, and seeing first-hand the outcomes of the work they have done. By removing the low-value painful activities associated with software delivery, we can focus on what we care about most—continuously delighting our users.

CONCLUSION

This paper has outlined the different Development Models, Methodologies and Process which are in practice and deals about sharing a case study of a complex SAS based application development in GRID Environment. Case study demonstrated the application of different Development Models, Methodologies and Process that need to be considered while deciding the nature of development and the kind of benefits that is brings to the business, sponsor and the team on the whole. This paper would enable the Development team to further explore the application of different models in their organization and re-evaluate based on the retrospective results accounted by their managers in terms of business benefit, time & cost.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Giri Balasubramanian
Company: PRA Health Sciences
Address: 40, II Main Road, R.A. Puram
City / Postcode: Chennai, TamilNadu 600 028, India
Email: BalasubramanianGiri@prahs.com
Web: www.prahs.com