

Agile Unified Process

Siddharth Prabhudas
United Health Group
Unitech Cyber Park
Sec-39, Gurgaon
Haryana
Siddharth_prabhudas@uhc.com

Abstract

Most software development is a chaotic activity, often characterized by the phrase "code and fix". The software is written without much of an underlying plan, and the design of the system is cobbled together from many short term decisions. This actually works pretty well as the system is small, but as the system grows it becomes increasingly difficult to add new features to the system. The original movement to try to change this introduced the notion of methodology. These methodologies impose a disciplined process upon software development with the aim of making software development more predictable and more efficient. They do this by developing a detailed process with a strong emphasis on planning inspired by other engineering disciplines. Engineering methodologies have been around for a long time, they've not been noticeable for being terribly successful.

Agile methodologies developed as a reaction to these methodologies. These new methods attempt a useful compromise between no process and too much process, providing just enough process to gain a reasonable payoff. The result of all of this is that agile methods have some significant changes in emphasis from engineering methods. The most immediate difference is that they are less document-oriented, usually emphasizing a smaller amount of documentation for a given task.

Is tailoring required for implementation of Agile Methodology? The answer is YES. This article will emphasize on the best practices of Agile Methodology and Rational Unified Process in the name AGILE UNIFIED PROCESS. It speaks about SCRUM – A disciplined Management Methodology. It will also define a framework where AUP can be integrated with CMM and ISO.

Keywords

RUP: Rational Unified Process, AUP: Agile Unified Process, Usage model, Domain Model, Interface Model, ROI: Return on Investment, TDD: Test Driven Design, Re-factoring, Acceptance Test, FLOOT :Full Lifecycle Object-Oriented Testing

1. Introduction

Agile Unified Process is a simplified version of Rational Unified Process which takes the best practices of RUP and Agile Methodology. AUP is nothing but looking at RUP from an agile practitioner's point of view. RUP is considered to be a rigid artifact driven replacement for waterfall. But its software development processes and practices can be made flexible to fit into the right size of methodology of your project. Similarly Agile

Methodology is characterized by an emphasis on people, communication, working software, and responding to change, what it does not speak about is documentation. So I am advocating for a middle line between RUP and Agile which can fit into any software engineering practices starting from product development to service oriented companies.

2. Business Model for AUP

I am redefining the phases of RUP to fit into Agile Unified process

2.1 Inception:

1. Project Scope is established for a new project.
2. If it is an existing project then along with the new requirement, all associated and existing defects are also studied as a part of the scope.
3. Requirements are prioritized and Requirement Stack is maintained.
4. Architectural vision is identified.
5. A high level estimation is made which includes modeling.

2.2 Elaboration:

1. Requirements are further analyzed in detail level. This phase is very critical as we are concentrating on model. At this level we should do three kinds of modeling: Usage model to explore how users will work with your system, Domain Model which identifies fundamental business entity types and the relationships between them and Interface Model which speaks about the interaction of one system with other systems.
2. Initial Architecture Modeling is given importance here – This is nothing but the effort to try to identify an architecture that has a good chance of working.
3. Use Case Model, Domain Model, Interface Model and initial architecture models takes better shape in further iterations – but then the goal is not to freeze the documents but to give the team something so that they can start working.
4. Based on the detail requirement, estimation is also revisited.
5. As the picture shows, this is again a multi iteration activity.

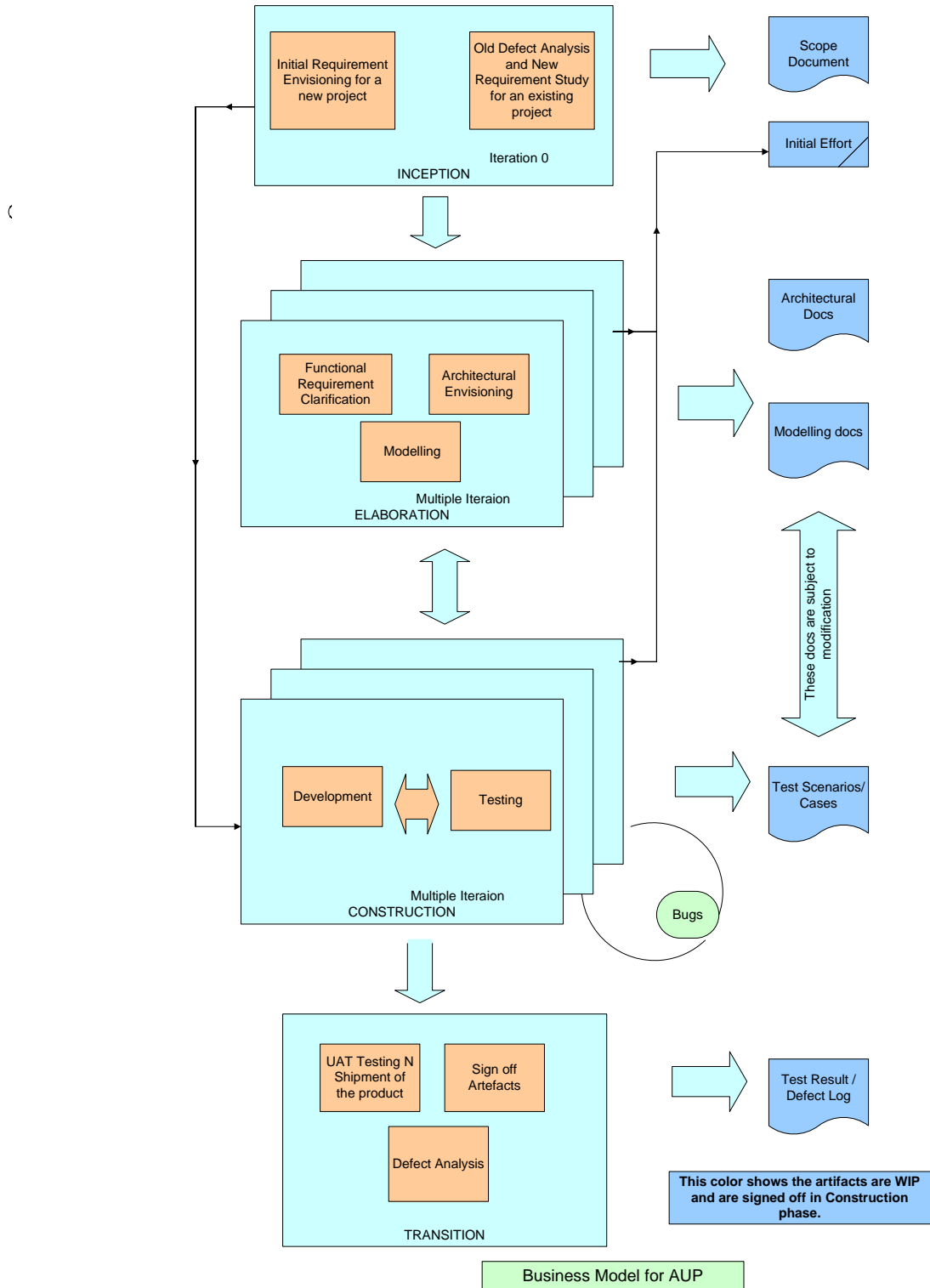
2.3 Construction:

1. Model storming should be encouraged. It's always advisable to model storm for several minutes and then code. Model storming requires very few audience but then the goal is to work towards solving the issues and come up with solutions. This is typically a Q N A session.
2. Model storming should be done using agile practices like Test Driven Design (TDD) and Re-factoring. This works because model storming helps you to think in cross functional and cross application perspective, test driven design helps you focus on a particular entity and re-factoring means you evolve your design through small steps to ensure that you deliver high quality work.

3. Agile encourage detail modeling through executable specification and acceptance tests. Acceptance tests can be considered as detail requirement and developer tests can be considered as detail design. Both when combined give the flexibility to the developer to document less.
4. Agile Unified Process speaks about model but then it does not speak how. It all depends upon the user to choose whether feature driven or use case driven model or both is/are beneficial to them.
5. Testing and Development which forms the major part of this phase are not two separate activities, test and develop approach should be followed. Bugs from one iteration becomes the input to the next iteration and in terms a robust architecture is formed. Testers and developers are part of the same team and should work towards a common goal.
6. Stake holders should also be part of this phase and should actively participate as a team.
7. There is a very close relation between Inception and Construction phase and artifacts produced during these phases are dependent on the outputs of both the phases.
8. Depending on the need and based on the product back log we can go for number of iterations in the same phase.

2.4 Transition:

This is nothing but a transition phase. Defects which could not be catered to in the current release are passed on to future releases. One of the major activities of this phase is to sign off the artifacts so that they can be base lined for the future releases.



3.0 Best practices of AUP

1. Sign off Artifacts: Artifacts are not signed off till the Transition Phase. They get updated as and when required.
2. Need based Artifact: Produce Artifacts which are very much required for the success of the project.
3. Discuss Visualize and Write: If you are thinking of producing an artifact, first discuss then visualize and the last thing that you should do is to document.
4. Single Repository: Try to store all the artifacts in a single repository.
5. Requirement prioritization: The most important thing in agile world is to prioritize your requirements based on the ROI.
6. Stakeholder participation: Stakeholders should participate in each and every phase and in a timely manner.
7. Requirement Envisioning: At the beginning of the project, quality time needs to be invested in defining the scope of the project and to create the initial prioritized stack of requirements. If it is a new project, new and executable requirements will be studied and if it is an existing project, a good amount of time should be invested in studying the existing defects/issues related to the scope requirement which becomes the part of the final scope.
8. Executable specifications: Specifications should be written in executable form so that design and coding can be carried out effectively.
9. Model Storming: In iterations, it's advisable to do little bit of model storming for few minutes to explore the details behind a requirement or to think through a design issue.
10. Model in advance: Plan your modeling. If a complex requirement is popping up in the priority queue, then model for that requirement in the previous iteration.
11. Multiple Models: Nobody is perfect; each and every model has its own advantages and disadvantages. So developers should be provided with a wide range of models to choose from, so that they can effectively utilize those models at right time and at right place.
12. Effective Communication: There is no hierarchy maintained for communication. Stakeholders, developers, analysts, testers and management working on a project are part of the team. So anybody can reach out to anyone to get clarification.
13. Accept Change: Change is always inevitable and should be acceptable in the software world. Multiple iterations, multiple modeling and above all little bit of planning in terms of effort also justify for accepting change in requirement.

4.0 AUP Software Development (How it works)

Agility is the ability to both create and respond to change in order to profit in a turbulent business environment. Agile software development is an approach to software development that is people oriented, that enables people to respond effectively to change

and that result in the creation of working systems that meets the needs of its stakeholders.

I would like to iterate – AUP *Software Development is not:*

- “Code and fix”
- An excuse not to document
- An excuse not to model
- An excuse to short-change quality
- An excuse to ignore enterprise concerns

Agile Software Development Values

- Individuals and interactions
- Working software
- Customer Collaboration
- Respond to Change

Over

- Processes and tools
- Comprehensive documentation
- Contract negotiation
- Following a plan

When I am talking so much about AUP – I should define its *Principles:*

- 1 Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- 2 Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- 3 Deliver working software frequently from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- 4 Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- 5 The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- 6 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- 7 Simplicity--the art of maximizing the amount of work not done--is essential.
- 8 The best architectures, requirements, and designs emerge from self-organizing teams.

5.0 AUP Software Development Methods

There are two basic agile software development methodologies. One is XP and the other is Scrum. For AUP, I will say Scrum will be the best methodology. Scrum is one of the AUP methods for project management. It has been called a "hyper-productivity tool", and has been documented to dramatically improve productivity in teams previously paralyzed by heavier methodologies. Its intended use is for the management of software



development projects, and it has been successfully used to "wrap" Extreme Programming and other development methodologies. However, it can theoretically be applied to any context where a group of people need to work together to achieve a common goal - such as setting up a small school, scientific research projects or planning a wedding.

5.1 Best practices of Scrum

1. Customer is always a part of the development team.
2. Frequent intermediate deliveries with a working functionality are **MUST**.
3. Frequent mitigation and risk plans are being developed by the **DEVELOPMENT TEAM**.
4. Daily status discussion with the team is **DONE** on a daily basis.
5. Things included in the daily discussion are:
 - a. What did you do from yesterday?
 - b. What are you planning to do tomorrow?
 - c. Do you have any problems?
6. Transparency is **ALWAYS** there in planning and different modules.
7. If a feature has a problem, no delivery shall take place.
8. A frequent stakeholder meeting, to see the progress, is a **MUST**.
9. No problems are buried under the carpet. No one is penalized for a problem.
10. Energized workplace and working hours is a **MUST**. "More number of hours" does not mean more "output."

I will not be taking much time in introducing the audience about how sprint works because it depends upon the flexibility of the audience as to how they want to make it work. However, I would like to make the user aware of **Project Execution using AUP**:

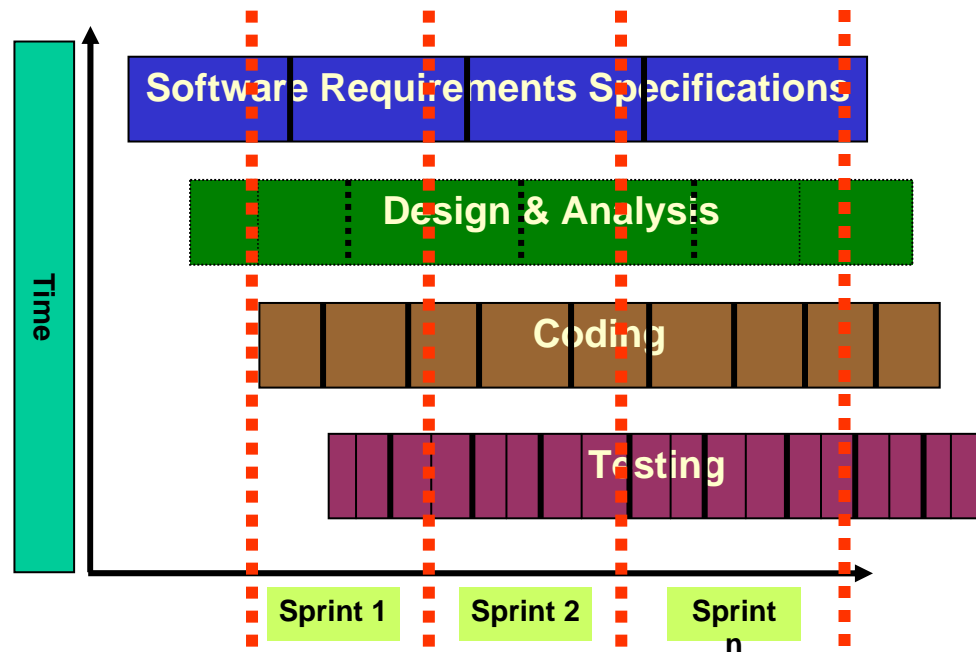
1. Create the Product Backlog – Product owner creates the product Backlog by prioritizing the organizational values.
2. Create a Release Backlog – It is selected by team at outset of Sprint which keeps on changing during Sprint as information is discovered.
3. Plan the 30-day sprint with a Sprint Planning Meeting.
4. Create the Sprint goal and the Sprint Backlog – This is generally done during the Sprint planning meeting.
5. Manage the daily sprints through the Daily Scrum and Sprint Burn down Charts.
6. Remove Impediments.
7. Deliver Product increment.
8. Review product increment as part of Sprint Review meeting.
9. Update the Product Backlog.

Since you are now introduced to the scrum process, you can better understand how the requirement can be managed through Scrum – Whenever a new requirement comes or the priority of the existing requirement change then the Product Owner updates the Product Backlog and Sprint Backlog. All requirements in a sprint backlog will be developed to correspond to a high-level design document (A combination of the Scope document and Design Document). This will satisfy two objectives: First the details of all sprint features

and functionality will be available. And second, a design specification will be available for the team to refer to. The Sprint backlog is reviewed at the Sprint Review Meeting to ensure that all requirements have been met. The Product & Release Backlogs are dynamic documents, and will be kept under configuration control.

5.2 Complete Project Lifecycle

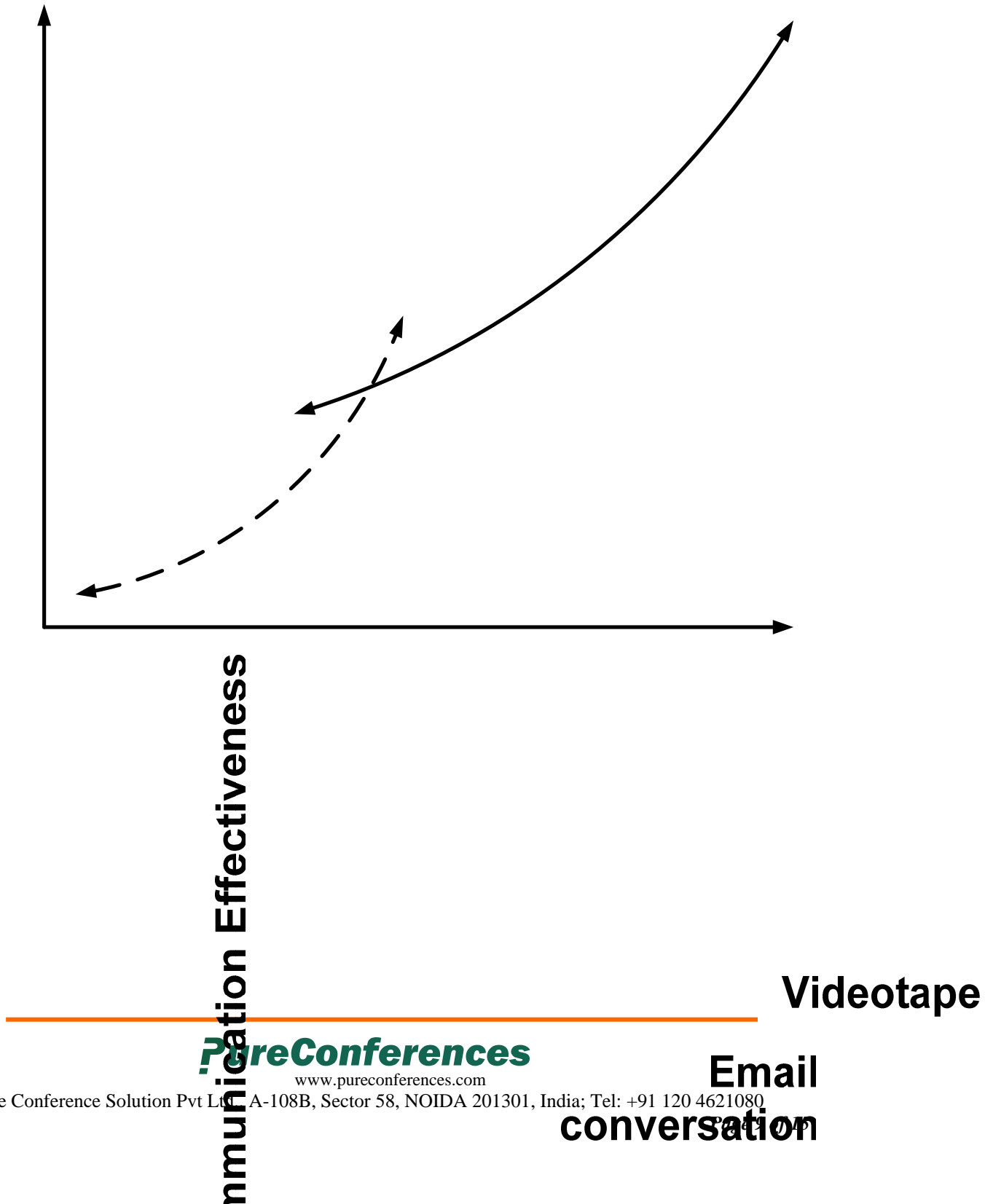
The Complete project Lifecycle is just divided in various sprints and in each and every sprint various phases of the life cycle like Requirement Specification, Design, Coding and Testing are becoming the parts of sprints.



Complete Project Lifecycle

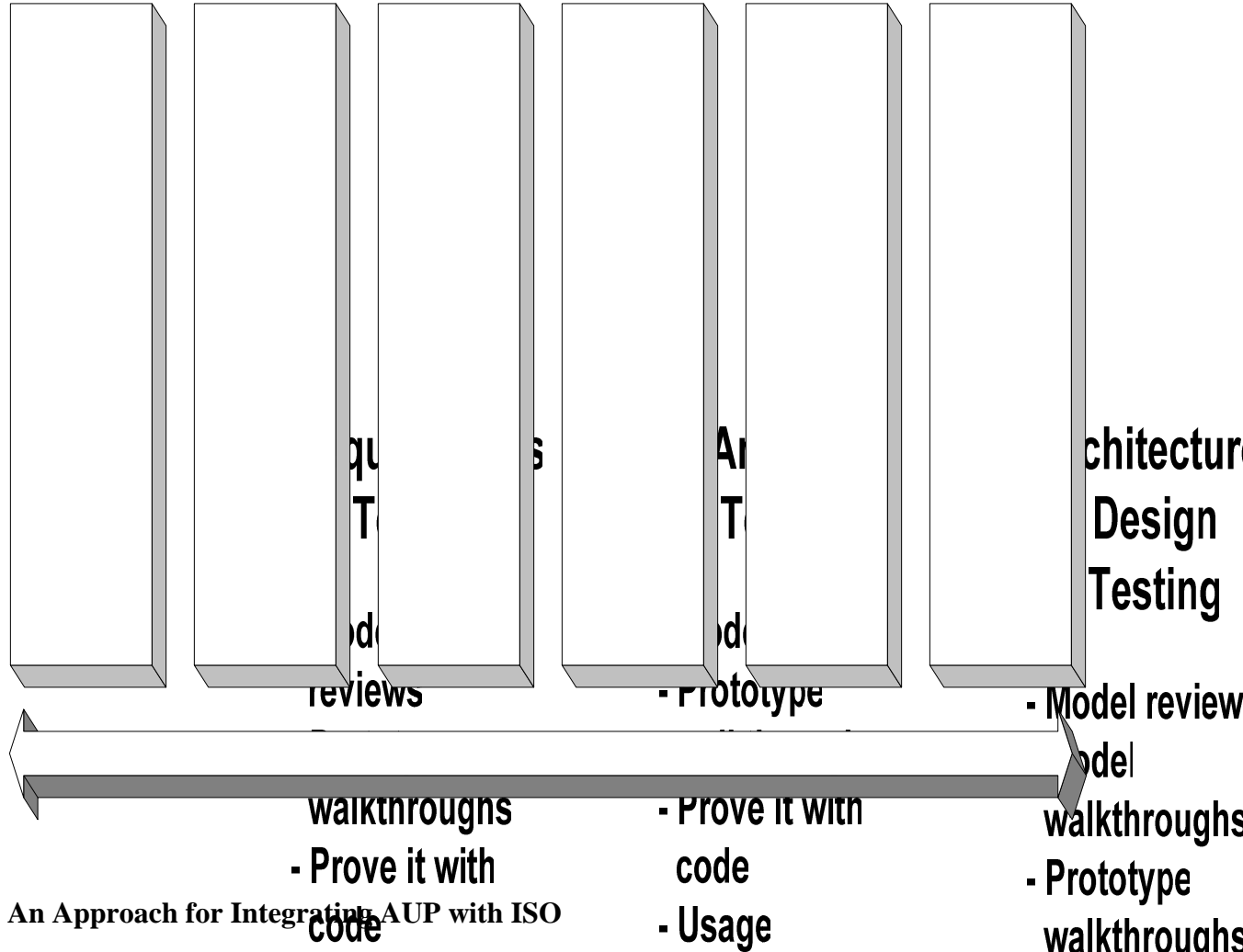
6.0 Some of the lucrative features of Agile Unified Process

6.1 Richness in Communication -



6. 2 Full Lifecycle Object-Oriented Testing (FLOOT)

Testing is done horizontally and vertically and in each and every phase so end of the day we are assuring a quality product. In AUP we are doing all kinds of testing:



7.0 An Approach for Integrating AUP with ISO

Conventional thinking concludes that agile and ISO cannot be compatible and are two opposite poles. ISO is often characterized as heavy on process/ heavy on documentation and is burdensome. And Agile is all about no documentation but this convention is wrong. ISO 9001:2000 is not only compatible with agile, but can provide just enough structure to help ensure your agile processes are followed. ISO 9001:2000 is not only compatible with agile, but can provide just enough structure to help ensure agile processes are followed. In this paper we discuss how to reconcile agile development's focus on speed and lean development with ISO 9001's need for documentation, traceability and control.

Agile Unified Process Clauses and Principles	Integration of Agile principles with ISO - 9001:2000
Value individuals and interactions more than process and tools.	ISO 9001:2000 series should stress the need for the involvement of people.
Value working software more than comprehensive documentation. AUP supports to some extent documentation which is required from a knowledge transition perspective.	ISO 9001:2000 series value working software and also relax the requirements for documented procedures compared to other ISO series.
Value customer collaboration more than contract negotiation	ISO 9001:2000 series also states that organizations should understand customer needs and strive to exceed customer expectations.
AUP stress upon the need of modeling during design and developmental phase. This in turns results in a faster development and minimize heavy design documentation.	ISO does not have its own design and developmental model. If agile practices are strictly followed then ISO can never go against it.
Value responding to change more than following a plan	ISO certainly require a plan to exists and be followed and any change in that needs to undergo change management process. From an agile perspective limiting change management process and approval process will not be a problem.
Agile's highest priority is to satisfy the customer through early and continuous delivery of valuable software.	ISO never conflicts with this. Just that the name highest priority should be taken into consideration.
Agile welcomes changing requirement even late in the development process.	ISO also strives for exceeding customer expectations so accepting changing requirement should not be a problem. The artifacts involved in the change management can be minimized by using tools like scrum works tool.
Deliver working software from a couple of weeks to a couple of months with a preference to shorter time scale.	There is nothing ISO can conflict with this. However, agile methods aimed at larger projects.
Business people and developers should work together daily throughout the project.	There is nothing ISO can conflict with this.

The most efficient and effective method of conveying information to and within a development team is face to face integration or through telephonic or video conferencing.	ISO should value people over documentation. Verbal communication should be encouraged more than email exchange.
Working software or daily meetings may be through scrum are the only means of tracking progress.	ISO believes in maintaining different tracking sheets to measure progress. But all these can be minimized if it is measured through scrum work tools.
Continuous attention to technical excellence and good design enhances agility.	ISO would never argue with a desire for technical excellence and good design.
In AUP, daily scrum is practiced which is an excellent way of bringing all the team together to discuss on progress and issues and in turn increase the visibility with the stakeholders.	ISO also tries to track the issues and challenges in various ways. Tracking progress is also a burdensome task if complete ISO philosophy is followed. So I do not think scrum practices can any way be challenged from an ISO perspective so far as tracking is concerned.

8.0 Using CMMI in Agile Software Development Assessments

Capability Maturity Model (CMM) and Capability Maturity Model Integration (CMMI) are broadly used for organizational maturity and process capability. CMMI is appreciated because of its extensive descriptions of how the best practices are fit together. Integrating Agile with CMMI is nothing but assessing in such a manner which takes agile context into account while still producing useful results.

CMMI Goals	AUP practices
Manage Requirement	Product backlog is created and maintained during sprint planning meeting. These product backlogs are made into sprint backlogs through daily sprint meetings, prioritized and worked upon. Stakeholders also make active participation in this activity.
Establish Estimates	Through sprint meetings, estimates are made and these estimates are revisited every day which makes robust and realistic estimates at the end of the day.

Develop Project plan - Project plan mentions the details of the activities to be performed during a project.	Most of the activities that need to be done as part of the project are captured as product backlog. Still then Agile Unified process do not rule out the possibility of maintaining a Project Plan.
Obtain commitment to the plan	AUP builds a self organizing team comprising developers, stakeholders and testers. I will say it's a self organizing team as we are meeting everyday to discuss where we stand in terms of planning.

9.0 Case Study: Electronic Shelf Label solution for retailers

The Client

A hi-tech product company, into a hardware intensive product with a team of hardware engineers, wanted to roll out the product for electronic shelf labeling market.

Business Scenario

ESL management system helps store chains in strategic and dynamic price management of the items being sold. The system includes hardware like base stations, transceivers and electronic shelf labels (ESL's). System supported both serial and Ethernet based communication channel. Faster time to market such a product required parallel development of hardware and software. With market pressures of providing better features during different stages of progressive development, it was difficult for hardware team to keep changing the hardware from time to time. Cost of making hardware changes for minor feature addition was very high as compared to making software more intelligent for better hardware management.

The Solution

Company X's highly reliable, multithreaded, scalable software solution helped the client in effectively penetrating the market with new features being supported progressively throughout the development. Since the development process needed parallel development of hardware and software, company X's software team made extensive use of mocks replacing the real hardware.

The Business Process

Company X initially followed RUP, the Product was developed through an iterative process, and the customer was made the integrated part of the software development which helped in getting clear requirements, and defining clear direction to the development process. The engagement spanned for more than 3 years. During initial phase more stress was given on getting detailed requirements from the client. These

requirements helped in creating a stable, robust, component based architecture. As Company X progressed, they found RUP to be very big and complex, very broad and needs much of customization to fit into the company, demands big initial efforts and investments to start using it. It is also too much document heavy. Then the company made a drastic change in the approach and tried to be more agile which helped in focusing on development as per dynamic customer requirements. Development process required ceaseless interaction of both hardware and software team. Teams made extensive use of video conferencing/ messenger communication to update each other on regular basis. Daily standup meetings about the progress were carried out both onsite and offsite, and notes were exchanged amongst both the teams. Extensive unit test cases helped in thorough unit and integration testing of the system.

When the company was at its pick with a bunch of talented IITians and doing its best, the flip side of too much agile was very much felt.

1. Active business involvement and close collaboration were required throughout the development cycle. This was very engaging, rewarding and ensured delivery of the right product. However, these principles were very demanding on the business representative's time and required a big commitment for the duration of the project.
2. Requirements emerged and evolved throughout the development. This creates the very meaning of agile - flexibility. There are two big flip sides to this principle though. One is the potential for scope creep, which created risk of ever-lasting projects. The other is that there was much less predictability, at the start of the project and during, about what the project is actually going to deliver. This made it harder to define a business case for the project, and harder to negotiate fixed price projects.
3. Agile development became rather intense for developers. The need to really *complete* each feature 100% within each iteration, and the relentlessness of iterations, became mentally quite tiring for the developers.
4. Agile does not talk much about documentation which good - the folks started concentrating much on the practical and productive side of the project. But change is always inevitable in an organization. People started leaving the organization because of their personal reasons and carried the knowledge with them. As there was no proper documentation available, new resources joining the organization could not catch up with the project.

At this point of time the Company felt the necessity of coming up with a model is in between Agile and RUP. So it picked up the best practices of RUP and Agile and came up with AUP.

Today Company X is a very successful company with client satisfaction at one side and employee satisfaction at the other.



BIOGRAPHY

Siddharth Prabhudas
Project Leader
UnitedHealth Group Information Services

Siddharth Prabhudas, MCA, a project Leader with UnitedHealth Group Information Services, has around 7 years of experience in software Quality Control and Quality Assurance. He is well versed in various processes like Rational Unified Process and Agile Methodology. He carries years of experience in Health Insurance and Retail Domain. Siddharth has expertise in various Black box and White box test techniques in Web and Client Server Application. He is well acquainted in Automation Test tools like QTP and Test Director. Lately he was involved in ISO - 9001:2000 implementations at UnitedHealth Group and has demonstrated sound skills in various processes like Risk Management, Resource Management and Configuration Management. Siddharth loves writing technical articles and has produced various White papers in the past.