Jyoti Nivas College
Autonomous
Post Graduate Centre

Tech on Tap

E-Journal - Department of MCA

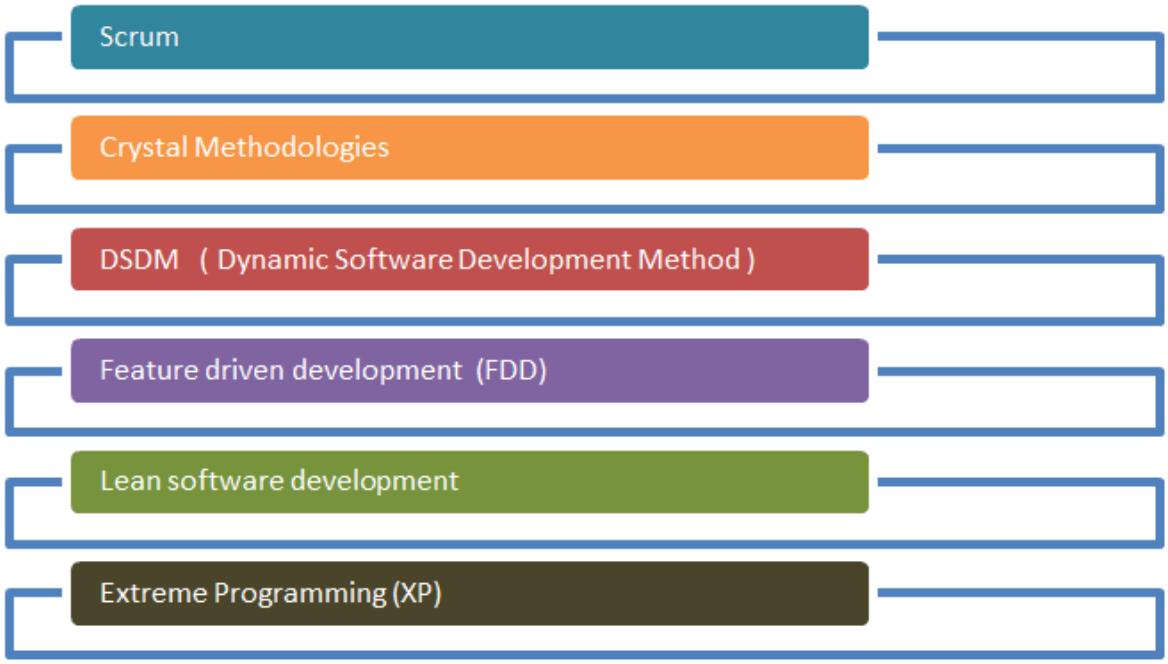
Issue: 2 **August 2018**

This issue tries to illustrate various Agile Software Development Methodologies and some related tools to enhance the performance of various software development tasks. AGILE methodology is a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project.

The agile software development emphasizes on four core values.

1. Individual and team interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Various Agile Methodologies



This document tries to give you a crisp idea about most of these methods and related tools

AGILE MODELING

Nandini D R [17MCA16]
Harshitha M [17MCA10]

INTRODUCTION:

Agile modeling is a supplement to other agile development methodologies such as Scrum, extreme programming (XP), and Rational Unified Process (RUP). It is explicitly included as part of the disciplined agile delivery (DAD) framework. Agile Modelling (AM) is a chaordic, practices-based methodology for effective modelling and documentation.

AM is based upon three components that each consist of sets of ideals and strategies.
These components are:

- Values.
- Principles.
- Practices.

VALUES:

In Agile Modeling, there are five principal values.

- **Communication:** Communication between the designers and the customer, and between the designers and developers.
- **Simplicity:** Simplicity both by clarifying design problems and representing programming concepts in a more understandable or generalized manner.
- **Feedback:** Models help designers to obtain feedback quickly, allowing them to act on the advice they receive.
- **Courage:** Courage to make decisions, and also to admit mistakes and change direction if necessary.
- **Humility:** Humility to recognize the abilities of other team members and value their contributions.

PRINCIPLES:

Agile Modeling contains two kinds of principles: core principles and supplementary principles

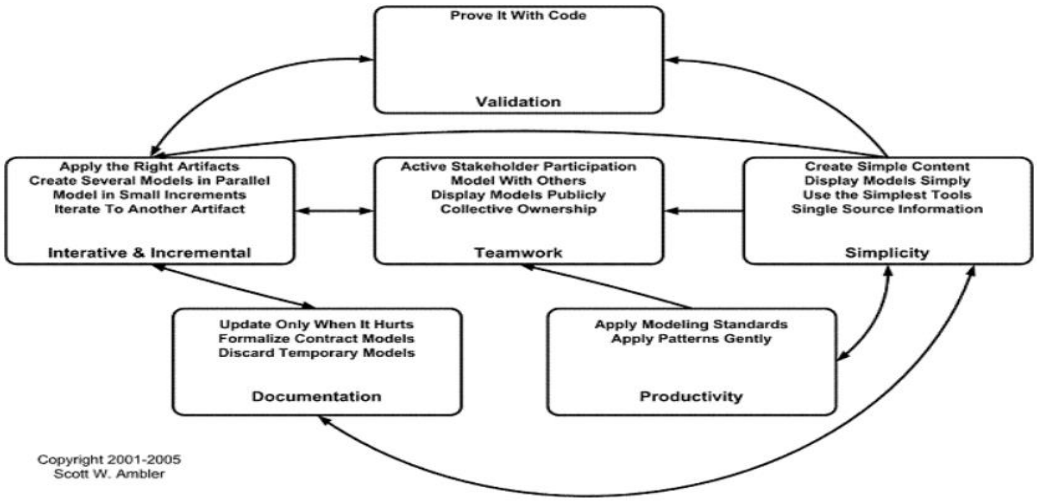
- **Assume Simplicity:** "The simplest solution is the best solution."
- **Embrace Change:** Requirements and available technologies change over time. An approach to development can only be successful if it embraces the reality of change
- **Enabling the Next Effort is Your Secondary Goal:** Ensure robustness of the current system before you even think about the next version.
- **Incremental Change:** Don't insist that a model be perfect the first time. Start with an incomplete or generalized model, and refine it over time.

PRACTICES:

Agile Modelling subdivides practices into two groups: core and supplementary practices.

- **Active Stakeholder Participation:** The idea is to have the customer and end user participate in requirements elicitation and prioritization of requirements.
- **Apply the Right Artifact:** Know the types of models available, and choose one that concisely conveys information while allowing flexibility to change the design

- **Collective Ownership:** Applies to all models and all 'artifacts'.
- **Create Several Models in Parallel:** This technique helps to create a fuller understanding of the system or subsystem, while helping to determine which model represents the system most simply, fully and accurately.



How Am Practices Fit Together

AM USED FOR:

- You are taking an agile approach to development in general.
- You plan to work iteratively and incrementally.
- The requirements are uncertain or volatile.
- The primary goal is to develop software.
- The active stakeholders are supportive and involved.
- The development team is in control of its destiny.
- The developers are responsible and motivated.
- Adequate resources are available for the project.

Agile Modelling does not work well in organizations that have a prescriptive culture, or for teams that are very large and/or distributed

CONCLUSION:

Agile models are based on iterative software development. An independent working module is built after the completion of iteration. Iteration should not consume more than two weeks to complete a code. Agile methodologies invite the developers to get involved in testing, rather than a separate quality assurance team.

Agile methodologies are suitable in changing environments because of new practices and principles that enable a team to develop a product in short duration.

REFERENCE:

http://www.agilemodeling.com
http://www.wikipedia.org
http://www.xprogramming.com

DYNAMIC SYSTEMS DEVELOPMENT METHOD (DSDM)

ROJA RANI K (17MCA23)

AISHWARYA M (17MCA01)

INTRODUCTION

Dynamic systems development method (DSDM) is an agile project delivery framework, initially used as a software development method. First released in 1994, DSDM originally sought to provide some discipline to the rapid application development (RAD) method. The DSDM Agile Project Framework covers a wide range of activities across the whole project lifecycle and includes strong foundations and governance, which set it apart from some other agile methods. The DSDM Agile Project Framework is an iterative and incremental approach that embraces principles of agile development, including continuous user/customer involvement.

DSDM Atern

Atern is a vendor-independent approach that recognizes that more projects fail because of people problems than technology. Atern's focus is on helping people to work effectively together to achieve the business goals. Atern is also independent of tools and techniques enabling it to be used in any business and technical environment without tying the business to a particular vendor.

Principles

There are eight principles underpinning DSDM Atern. These principles direct the team in the attitude they must take and the mindset they must adopt to deliver consistently.

1. Focus on the business need
2. Deliver on time
3. Collaborate
4. Never compromise quality
5. Build incrementally from firm foundations
6. Develop iteratively
7. Communicate continuously and clearly
8. Demonstrate control

Core techniques

- Time boxing: is the approach for completing the project incrementally by breaking it down into splitting the project in portions, each with a fixed budget and a delivery date. For each

portion a number of requirements are prioritized and selected. Because time and budget are fixed, the only remaining variables are the requirements

- Moscow: is a technique for prioritizing work items or requirements. It is an acronym that stands for:
    - MUST have
    - SHOULD have
    - COULD have
    - WON'T have
- Prototyping: refers to the creation of prototypes of the system under development at an early stage of the project. It enables the early discovery of shortcomings in the system and allows future users to 'test-drive' the system. This way good user involvement is realized, one of the key success factors of DSDM, or any System Development project for that matter.

Comparison to other development frameworks

DSDM can be considered as part of a broad range of iterative and incremental development frameworks, especially those supporting agile and object-oriented methods. Like DSDM, these share the following characteristics:

- They all prioritize requirements and work though them iteratively, building a system or product in increments.
- They are tool-independent frameworks. This allows users to fill in the specific steps of the process with their own techniques and software aids of choice.
- The variables in the development are not time/resources, but the requirements. This approach ensures the main goals of DSDM, namely to stay within the deadline and the budget.
- A strong focus on communication between and the involvement of all the stakeholders in the system. Although this is addressed in other methods, DSDM strongly believes in commitment to the project to ensure a successful outcome.

References

Keith Richards, Agile project management: running PRINCE2 projects with DSDM Atern. OGC – Office of Government Commerce.

EXTREME PROGRAMMING (XP)

ANNAPURNA PAILA

TEJASWINI.G

ABSTRACT : Extreme programming is light weighted software development methodology based on the widely recognized and effective paradigms of code inspections, iterative spiral development, integrated product development teams (i.e., full-time customer involvement), frequent builds, programmer teams (i.e., pair programming), design patterns, refactoring, coding standards, risk analysis, and regression testing. XP emerged from the object-oriented programming community.

DEFINITION: Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.

Extreme Programming Advantages:

- The main advantage of Extreme Programming is that this methodology allows software development companies to save costs and time required for project realization. Time savings are available because of the fact that XP focuses on the timely delivery of final products. Extreme Programming teams save lots of money because they don't use too much documentation. They usually solve problems through discussions inside of the team.
- Simplicity is one more advantage of Extreme Programming projects. The developers who prefer to use this methodology create extremely simple code that can be improved at any moment.
- The whole process in XP is visible and accountable. Developers commit what they will accomplish and show progress.
- Constant feedback is also the strong side. It is necessary to listen and make any changes needed in time.
- XP assists to create software faster thanks to the regular testing at the development stage.
- Extreme Programming contributes increasing employee satisfaction and retention.

DISADVANTAGES:

- Some specialists say that Extreme Programming is focused on the code rather than on design. That may be a problem because good design is extremely important for software applications. It helps sell them in the software market. Additionally, in XP projects the defect documentation is not always good. Lack of defect documentation may lead to the occurrence of similar bugs in the future.
- One more disadvantage of XP is that this methodology does not measure code quality assurance. It may cause defects in the initial code
- XP is not the best option if programmers are separated geographically.

Extreme Programming Principles: The fundamental principles of Extreme Programming are –

- **Rapid feedback**: Rapid feedback is to get the feedback, understand it, and put the learning back into the system as quickly as possible.
- **Assume simplicity**: To assume simplicity is to treat every problem as if it can be solved with simplicity.
- **Incremental change:** In any situation, big changes made all at once just do not work. Any problem is solved with a series of the smallest change that makes a difference.
- **Embracing change:** The best strategy is the one that preserves the most options while actually solving your most pressing problem.
- **Quality work:** Everyone likes doing a good job. They try to produce the quality that they are proud of. The team**:**
  - Works well
  - Enjoys the work
  - Feels good in producing a product of value

APPLICATION OF EXTREME PROGRAMMING:

Extreme Programming remains a sensible choice for some projects. Projects suited to Extreme Programming are those that:

- Involve new or prototype technology, where the requirements change rapidly, or some development is required to discover unforeseen implementation problems
- Are research projects, where the resulting work is not the software product itself, but domain knowledge
- Are small and more easily managed through informal methods

CONCLUSION

XP agile software development methods have several features and aspects to support projects for large or small organization, and the projects that's need short or long period of time to be finished. This methods have team and contractors (product owner) effect of project process with responsibility when changing requirements, added the fields of project where to be applied, where this research take the engineering field as the main filed applied, so to optimize the work process and solve drawbacks for each method, the new method is proposed to deal the fields time and size of organization for XP.

Feature Driven Development

Keerthi K Phirangi

Poojashree.N

Introduction:

FDD is an iterative and incremental software development process. It is a lightweight or Agile method for developing software. FDD blends a number of industry-recognized best practices into a cohesive whole. These practices are driven from a client-valued functionality (feature) perspective. Its main purpose is to deliver tangible, working software repeatedly in a timely manner.

Brief History on Feature Driven Development:

FDD was initially devised by Jeff De Luca to meet the specific needs of a 15-month, 50-person software development project at a large Singapore bank in 1997. This resulted in a set of five processes that covered the development of an overall model and the listing, planning, design, and building of features. The first process is heavily influenced by Peter Coad's approach to object modelling. The second process incorporates Coad's ideas of using a feature list to manage functional requirements and development tasks. The other processes are a result of Jeff De Luca's experience. There have been several implementations of FDD since its successful use on the Singapore project..

Founders are:

- Eric Lefebvre
- Mac Felsing

How it Works:

FDD is a model-driven short-iteration process that consists of five basic activities. For accurate state reporting and keeping track of the software development project, milestones that mark the progress made on each feature are defined. This section gives a high level overview of the activities. In the figure on the right, the meta-process model for these activities is displayed. During the first two sequential activities, an overall model shape is established. The final three activities are iterated for each feature.

Feature of Feature Driven Development:

- **Domain Object modeling:** Domain Object Modelling consists of exploring and explaining the domain of the problem to be solved. The resulting domain object model provides an overall framework in which to add features.
- **Configuration Management:** Configuration management helps with identifying the source code for all features that have been completed to date and maintaining a history of changes to classes as feature teams enhance them.

Benefits:

- Feature-Driven Development helps to move larger size projects and obtain repeatable success.
- The simple five processes help to bring work done in a short time and easiest manner.

Conclusion:

Feature-driven development is a process for helping teams produce frequent, tangible working results. It uses very small blocks of client valued functionality, called features. It organizes those little blocks into business-related feature sets. FDD focuses developers on producing working results every two weeks. FDD is better prepared to work with team where developers' experience varies. It offers progress tracking and reporting capabilities. This comforts managers and makes it more attractive for big companies.

RAD (Rapid Application Development)

Khushaboo Kumari(17MCA13)

Kumari Nisha(17MCA14)

RAD model is Rapid Application Development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

Phases in the RAD model:

- **Business modelling:** The information flow is identified between various business functions.
- **Data modelling:** Information gathered from business modelling is used to define data objects that are needed for the business.
- **Process modelling:** Data objects defined in data modelling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.
- **Application generation:** Automated tools are used to convert process models into code and the actual system.
- **Testing and turnover:** Test new components and all the interfaces.

Advantages of the RAD model:

- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of **integration issues**.

Disadvantages of RAD model:

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modelling skills
- Inapplicable to cheaper projects as cost of modelling and automated code generation is very high.

Application of RAD model:

- RAD should be used only when a system can be modularized to be delivered in an incremental manner.
- It should be used if there is a high availability of designers for modelling.

- It should be used only if the budget permits use of automated code generating tools.
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
- Should be used where the requirements change during the project and working prototypes are to be presented to customer in small iterations of 2-3 months.

When to use RAD Methodology?

- When a system needs to be produced in a short span of time (2-3 months)
- When the requirements are known
- When the user will be involved all through the life cycle
- When technical risk is less
- When there is a necessity to create a system that can be modularized in 2-3 months of time
- When a budget is high enough to afford designers for modeling along with the cost of automated tools for code generation

Conclusions:

Requires minimal planning in favour of rapid prototyping. Instead of using codes, Developers use different tools and software development kits and bring them all together to create a software. Developers who are time challenged could use this applications development.

User's feedbacks are important in this development cycle since they will suggest whether the program will fit to their specification and needs. Business will also appreciate this software as it's aimed to answer specific problems.

The Agile Unified Process (AUP)

Mary HarshithaA(17MCA15)

SweetyLenka (17MCA26)

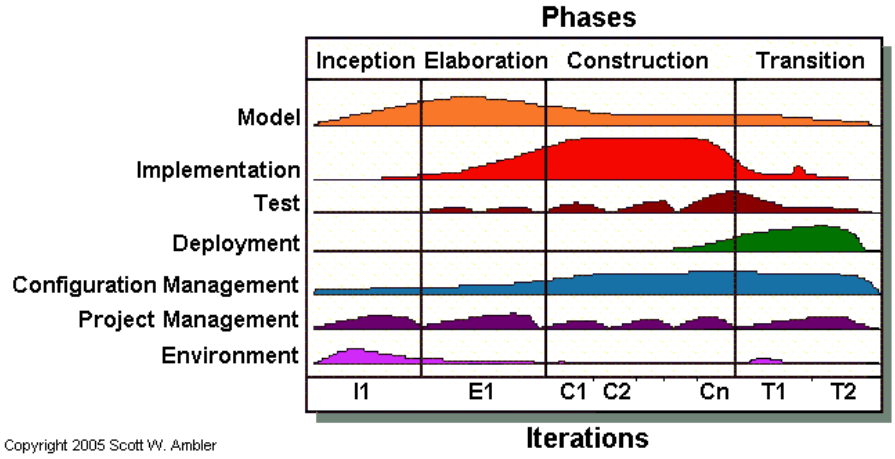
Introduction

Agile Unified Process (**AUP**) is a simplified version of the Rational Unified Process (RUP) developed by Scott Ambler. AUP uses a simple, easy to understand approach to develop business application software using agile techniques and concepts.The Rational Unified Process (RUP) is a software engineering approach whose goal is to produce high-quality software that meets the expectation of its users. The AUP applies agile techniques including test-driven development (TDD), agile modeling (AM), agile change management, and database refactoring to improve productivity.By combining RUP to agile methods, Ambler created a solid process framework that can be applied to all sorts of software projects, large or small. Agile methods provided values, principles, and practices to AUP. These values include individuals and their actions, delivering working software, customer collaboration, and responding to change.

When Ambler created the AUP, he centered the design around the following principles:

- Most people won't read detailed documentation. However, they will need guidance and training now and then. The project should be described simply in a few pages.
- The AUP conforms to the values and principles described by the Agile Alliance.
- The project must focus on delivering essential value rather than unnecessary features.
- Developers must be free to use tools best suited to the task at hand, rather than to comply with an edict.
- AUP is easily tailored via common HTML editing tools.

The Agile Unified Process (AUP) lifecycle.



The AUP is an iterative-incremental process consisting of workflows and phases. A direct comparison between RUP and AUP shows that AUP combines business modeling,

requirements, and analysis and design into a single workflow called Model (The agile unified process).

Model Workflow

The initial step in the AUP is the model workflow which begins with the Inception phase. Here project scope, risks, costs and schedule, and project feasibility are defined, including the preparation for the project environment . A major milestone in the Inception phase is the defined requirements without regards to its final technical implementation.

Implementation Workflow

A major part of the Implementation workflow occurs during the Construction phase where the model is transformed into executable code and tested thoroughly by unit tests.

Test Workflow

During this phase, an objective evaluation of the developed application is performed to ensure that it conform to the quality requirements. These phases include unit-testing, usability testing, and user-acceptance testing

Deployment Workflow

The purpose of this Workflow is to ensure that the newly developed application is successfully deployed.

Configuration Workflow

The Configuration Workflow ensures that the artifacts of the system are tracked and properly versioned.

Project Management Workflow

The Project Management Workflow directs the activities that occur during the project. These activities include risk management, project management, and coordinating external resources to ensure that the project is delivered within scope, cost, and time.

Environment Workflow

The Environment Workflow makes certain that the project occur in an environment suitable enough to ensure the success of the project.

Conclusion

The Agile Unified process (AUP), which is based on agile methods and the Rational Unified Process, provides an iterative-incremental approach to software development. AUP consists of seven workflows, each of which has four phases. The AUP workflow consists of Mode, Implementation, Test, Deployment, Configuration Management, Project Management, and Environment. The AUP phases consist of Inception, Elaboration, Construction, and Transition. The initial project step is the Inception phase that begins with the Model workflow where the project scope, risks, costs and schedule, and feasibilities are defined. The Construction phase is typically the largest phase and is mostly concerned with producing the executable code. The Test workflow ensures that the developed application conform to quality requirements. The Deployment workflow ensures that the developed system is properly deployed, while the Configuration workflow makes sure that the system's artifacts are properly tracked and versioned. The Project Management Workflow directs the activities that occur during the software-development life cycle, while Environment Workflow ensures the project team has all that it needs to be successful. A commercial project that used AUP for its development methodology concluded that AUP provides a flexible and reasonably agile methodology. However, they also found that the organization's culture and management needs to be receptive to Agile methods in order to successful use AUP.

SCRUMBAN

Sanjana K R (17MCA24)

Maithry A

Scrumban is an agile management methodology describing hybrids of Scrum and Kanban and was originally designed as a way to transition from Scrum to Kanban. Today, Scrumban is a management framework that emerges when teams employ Scrum as their chosen way of working and use the Kanban Method as a lens through which to view, understand and continuously improve how they work.

History

As the Kanban method was becoming more popular Scrumban was created as an attempt to make it easier for existing Scrum teams to begin exploring Lean and Kanban concepts

Fundamentally, Scrumban is a management framework that emerges when teams employ Scrum as their chosen way of working and use the Kanban Method as a lens through which to view, understand and continuously improve how they work.

Scrumban is distinct from Scrum in the way it emphasizes certain principles and practices that are substantially different from Scrum's traditional foundation. Among these are:

- recognizing the important role of organizational management (self-organization remains an objective, but within the context of specific boundaries)
- allowing for specialized teams and functions
- applying explicit policies around ways of working
- applying the laws of flow and queuing theory
- deliberate economic prioritization[2]

Scrumban is distinct from the Kanban Method in that it:

- prescribes an underlying software development process framework (Scrum) as its core
- is organized around teams
- recognizes the value of time-boxed iterations when appropriate
- formalizes continuous improvement techniques within specific ceremonie

A Framework for evolution

When Corey Ladas introduced the world to Scrumban in his seminal book of that name he defined it as a transition method for moving software development teams from Scrum to a "more evolved" software development framework. In actual practice, however, Scrumban has itself evolved to become a family of principles and practices that create complementary capabilities unique from both Scrum and the Kanban Method.

Method

In Scrumban, the teamwork is organized in small iterations and monitored with the help of a visual board, similar to Scrum and kanban boards. To illustrate each stage of work, teams working in the same space often use post-it notes or a large whiteboard. In the case of decentralized teams, visual management software such as Assembla, Target process, Eylean Board, JIRA, Mingle or Agilo for Trac are often used. Planning meetings are held to determine what user stories to complete in the next iteration. The user stories are then added to the board and the team completes them, the team working on as few user stories at a time as practical

(the work-in-progress, or WIP, limit). To keep iterations short, WIP limits are thus used, and a planning trigger is set in place for the team to know when to plan next.

Following are methods of scrumban:

1) Iterations

Work iterations in Scrumban are kept short. This ensures that a team can easily adapt and change their course of action to a quickly changing environment. The length of the iteration is measured in weeks. The ideal length of an iteration depends on the work process of each team, and it is recommended not to have iterations exceeding two weeks.

2) On-demand planning

The planning in Scrumban is based on demand and occurs only when the planning trigger goes off. The planning trigger is associated with the number of tasks left in the "To Do" section of the board - when it goes down to a certain number, the planning event is held. The number of tasks that should trigger a planning event is not predefined. The tasks planned for the next iteration are added to the "To Do" section of the board.

3) Prioritization

It is recommended to prioritize tasks during the planning event. This means the tasks are added to the board with marked priorities. It helps the team members to know which tasks should be completed first and which can be completed later.

4) The board

The basic Scrumban board is composed out of three columns: To Do, Doing and Done. After the planning meeting the tasks are added to the To Do column, when a team member is ready to work on a task, he/she moves it to the Doing column and when he/she completes it, he/she moves it to the done column.

5) The team

Scrumban does not require any specific number of team members or team roles. The roles a team has prior to adopting Scrumban are kept when implementing Scrumban. They are reinforced by team members having to choose the tasks to complete themselves. The team roles in Scrumban are more specialized and less cross-functional than what is expected in scrum teams.

6) Pull principle

In Scrumban tasks are not assigned to the team members by the team leader or project manager. Each team member chooses which task from the To Do section they are going to complete next. This guarantees a smooth process flow, where all the team members are equally busy at all times.

7) Triage

Triage usually happens right after feature freeze. With an approaching project deadline, the project manager decides which of the in-development features will be completed and which will stay unfinished.

Tooling

Like other methods, Scrumban can be implemented with a help of various tools. The most basic Scrumban implementation is a physical whiteboard with sticky notes. Electronic solutions, similar to scrum and kanban electronic boards are available as well. They offer a full automation of the board, where it only has to be updated by the team members.

Conclusion

Scrumban releases control to dev. Dev must "grok" the end goal for scrumban to work. Relives some of the forced ritual of scrum. May relieve some of the pressure on overburdened product managers who are serving as product owners.

LEAN SOFTWARE DEVELOPMENT

AMALA SEELAN G V (17MCA02)

R BHAVYASHREE (17MCA21)

Introduction

(**LSD**) is a translation of lean manufacturing principles and practices to the software development domain. Adapted from the Toyota Production System, it is emerging with the support of a pro-lean subculture within the Agile community. Lean offers a solid conceptual framework, values and principles, as well as good practices, derived from experience, that support agile.

Principles of Lean Software Development

There are seven principles of Lean Software Development, drawn from the seven principles of Lean Thinking. These principles are not cook-book recipes for software development, but guideposts for devising appropriate practices for your environment. 2

1. Eliminate Waste

All lean thinking starts with a re-examination of what waste is and an aggressive campaign to eliminate it. Quite simply, anything you do that does not add value from the customer perspective is waste.

2. Amplify Learning

The game of development is a learning game: hypothesize what might work, experiment to see if it works, learn from the results, do it again. People who design experiments know that greatest learning occurs when half of the experiments fail, because this exposes the boundary conditions.

3. Delay Commitment

Delaying commitment means keeping your options open as long as possible. The fundamental lean concept is to delay irreversible decisions until they can be made based on known events, rather than forecasts. Economic markets develop options as a way to deal with uncertainty. Farmers, for example, can buy an option on the future price of grain. If the price drops, they have protected their profits. If the price raises, they can ignore the option and sell at the higher price. Options let people delay decisions until they have more information.

4. Deliver Fast

To those who equate rapid software development with hacking, there seems to be no reason to deliver results fast, and every reason to be slow and careful. Similarly, when Just-in-Time concepts surfaced in Japan in the early 1980's, most western manufacturers could not fathom why they made sense. Everybody knew that the way to make customers happy was to have plenty of product on the shelf, and the way to maximize profits was to build massive machines and keep them busy around the clock. It took a long time for people to realize that this conventional wisdom was wrong. The goal is to let your customer take an options-based approach to making decisions, letting them delay their decisions as long as possible so they can make decisions based on the best possible information. Once your customers decide what they

want, your goal should be to create that value just as fast as possible. This means no delay in deciding what requests to approve, no delay in staffing, immediate clarification of requirements, no time-consuming handoffs, no delay in testing, no delay for integration, no delay in deployment. In a mature software development organization, all of this happens in one smooth, rapid flow in response to a customer need.

5. Empower the Team

In a lean organization, things moves fast, so decisions about what to do have to be made by the people doing the work. Flow in a lean organization is based on local signaling and commitment amongst team members, not on management directives. The work team designs its own processes, makes its own commitments, gathers the information needed to reach its goals, and polices itself to meet its milestones.

6. Build Integrity In

The customer needs to have an overall experience of the System. This is the so-called perceived integrity: how it is being advertised, delivered, deployed, accessed, how intuitive its use is, its price and how well it solves problems. Conceptual integrity means the system's separate components work well together as a whole with balance between flexibility, maintainability, efficiency, and responsiveness. This could be achieved by understanding the problem domain and solving it at the same time. The needed information is received in small batch pieces not in vast chunk preferably by face-to-face communication and not any written documentation. The information flow should be constant in both directions from customer to developers and back, thus avoiding the large stressful amount of information after long development.

7. See the Whole

When you look at them closely, most theories of how to manage software projects are based on a theory of disaggregation: break the whole into individual parts and optimize each one. Lean thinking suggests that optimizing individual parts almost always leads to sub-optimized overall system. Optimizing the use of testing resources, for example, decreases the ability of the overall system to rapidly produce tested, working code. Measuring an individual's ability to produce code without defects ignores the well-known fact that about 80% of defects are caused by the way the system works, and hence are management problems.

Conclusion

To keep customers from changing their minds, raise the maturity of your organization to the level where it can reliably deliver what customers want so fast that they have no time to change their minds. Focus on value, flow, and people, the rest will take care of itself.

Reference

http://synchronit.com/downloads/Lean_Software_Development

Disciplined Agile Delivery (DAD)

Divya P

Kushala M

Disciplined Agile Delivery (DAD) is a people-first, learning-oriented hybrid agile approach to IT solution delivery. It has a risk-value delivery lifecycle, is goal-driven, is enterprise aware, and is scalable.

Why Disciplined Agile Delivery (DAD)?

Because:

- Supporting a robust set of roles.
- Being a hybrid.
- Being open.
- Supporting several delivery lifecycles
- Addressing all aspects of solution delivery.
- Providing choices, not prescriptions.

Roles in Disciplined Agile Delivery

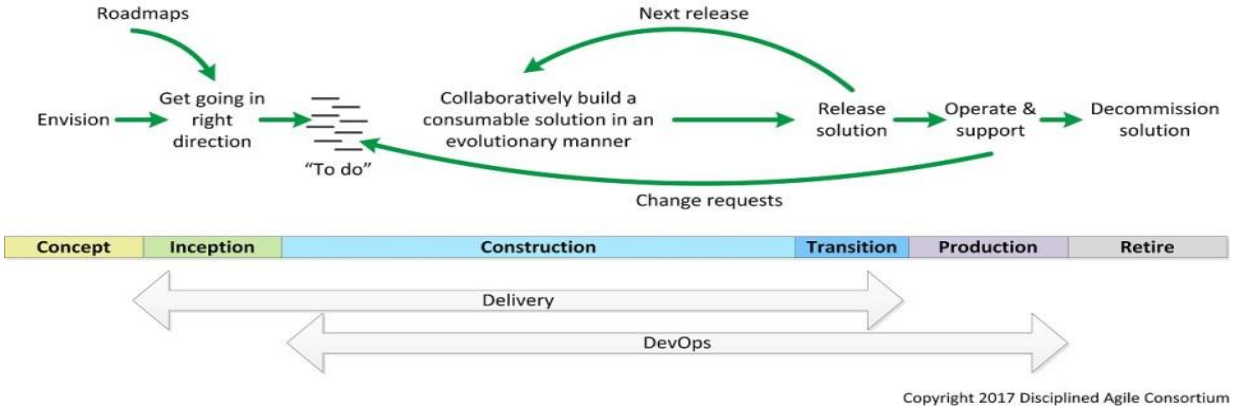
Primary roles : Primary roles will occur on all DAD projects regardless of scale.

Secondary roles : typically occur only at scale and sometimes only for a temporary period of time

A Hybrid Framework

- Disciplined Agile (DA) is a hybrid framework that builds upon the solid foundation of other methods and software process frameworks.
- DAD adopts practices and strategies from existing sources and provides advice for when and how to apply them together.
- Methods such as Scrum, Extreme Programming, Kanban, and Agile Modeling, provide the process bricks and DAD the mortar to fit the bricks together effectively.

A high-level view of the delivery lifecycle



Five versions of the lifecycle are supported

- The Agile/basic lifecycle
- The Lean/advanced lifecycle.
- The Continuous Delivery: Agile lifecycle.
- The Continuous Delivery: Lean lifecycle.
- The Exploratory/Lean Startup lifecycle.

Choice is Good: Goal-Driven

- Supports process tailoring by making process decisions explicit.
- Enables effective scaling
- Makes your process options very clear
- Takes the guesswork out of extending agile methods.
- Makes it clear what risks you're taking on and thus enables you to increase the likelihood of success.
- Hints at an agile maturity model

DAD Teams are Enterprise Aware: Enterprise awareness is one of the principles of the Disciplined Agile (DA) framework. The observation is that DAD teams work within your organization's enterprise ecosystem, as do all other teams.

The DAD framework provides a better foundation for tactically scaling agile in several ways:

- DAD promotes a risk-value lifecycle.
- DAD promotes self-organization enhanced with effective governance.
- DAD promotes enterprise awareness over team awareness.
- DAD is context-sensitive and goal driven, not prescriptive.

KANBAN

Rakshitha G L (17MCA22)

Sushma M N (17MCA25)

Introduction:

Kanban is a scheduling system for lean manufacturing and just-in-time manufacturing. Taiichi Ohno , an industrial engineer at Toyota, developed kanban to improve manufacturing efficiency. The system takes its name from the cards that track production within a factory. For many in the automotive sector Kanban is known as "Toyota name plate system" and as such the term is not used by some other auto makers.

Kanban became an effective tool to support running a production system as a whole, and an excellent way to promote improvement. Problem areas are highlighted by measuring lead time and cycle time of the full process and process steps. One of the main benefits of kanban is to establish an upper limit to work in process inventory to avoid overcapacity.

Brief History:

The system originates from the simplest visual stock replenishment signaling system- an empty box. This was first developed in the UK Spitfire factories during the war and was known as the "two bin system." In the late 1940s, Toyota started studying supermarkets with the idea of applying shelf-stocking techniques to the factory floor. In a supermarket, customers generally retrieve what they need at the required time—no more, no less. Furthermore, the supermarket stocks only what it expects to sell in a given time, and customers take only what they need, because future supply is assured. This observation led Toyota to view a process as being a customer of one or more preceding processes, and to view the preceding processes as a kind of store.

How it Works:

1. Visualize Work

By creating a visual model of your work and workflow, you can observe the flow of work moving through your Kanban system. Making the work visible along with blockers, bottlenecks and queues instantly leads to increased communication and collaboration.

2. Limit Work in Process

By limiting how much unfinished work is in process, you can reduce the time it takes an item to travel through the Kanban system. You can also avoid problems caused by task switching and reduce the need to constantly re-prioritize items.

3. Focus on Flow

By using work-in-process (WIP) limits and developing team-driven policies, you can optimize your Kanban system to improve the flow of work, collect metrics to analyze flow, and even get leading indicators of future problems by analyzing the flow of work

4. Continuous Improvement Once your Kanban system is in place, it becomes the cornerstone for a culture of continuous improvement. Teams measure their effectiveness by tracking flow, quality, throughput, lead times and more. Experiments and analysis can change the system to improve the team's effectiveness

Types of kanban system:

In a kanban system, adjacent upstream and downstream workstations communicate with each other through their cards, where each container has a kanban associated with it. Economic Order Quantity is important. The two most important types of kanbans are:

- Production Kanban: A Production kanban, when received, authorizes the workstation to produce a fixed amount of products. The Production kanban is carried on the containers that are associated with it.
- Transportation Kanban: A Transportation kanban authorizes the transportation of the full container to the downstream workstation. The Transportation kanban is also carried on the containers that are associated with the transportation to move through the loop again.

Benefits:

- Flexibility
- Focus on continuous delivery
- Reduction of wasted work / wasted time
- Increased productivity
- Increased efficiency
- Team members ability to focus

Conclusion:

Successful Kanban implementations can have a profound impact on all the Lean metrics. Carrying excessive levels of inventory ties up money that could be used for purposes that are more beneficial to the company. Not every company is the same, and you must look at each part and supply individually to determine what the best replenishment system will be to implement better control.

Refrences:

https://en.wikipedia.org/wiki/Kanban

https://leankit.com/learn/kanban/what-is-kanban/

LEAN KIT

ARSHIYA ANJUM(17MCA07)

AMALI SUNITHA(17MCA03)

Introduction:

Lean Kit is a visual project delivery tool that enables teams of all types and across all levels of the organization to apply Lean management principles to their work. Whatever workflow methodology you use — including Kanban, Scrum, Waterfall and anything. **Kanban** is one of the Lean tools designed to reduce the idle time in a production **process**.

Lean Kit is a highly flexible platform designed for the practical implementation of Kanban. It provides a real-time, shared understanding of activity and status, making it the ideal tool to manage both project and process work.

Brief History on Lean Kit:

Lean ideals began in the 1950s with Toyota automobile manufacturing in Japan. Lean came out of what was developed by Taiichi Ohno as the Toyota Production System (TPS). The system used visual signals to indicate inventory needs precisely when items were needed, to reduce overall waste and optimize the entire system of production.

Founders are:

1. Tim Mulron
2. Stephen Franklin
3. Tom Jerry
4. Daniel Norton

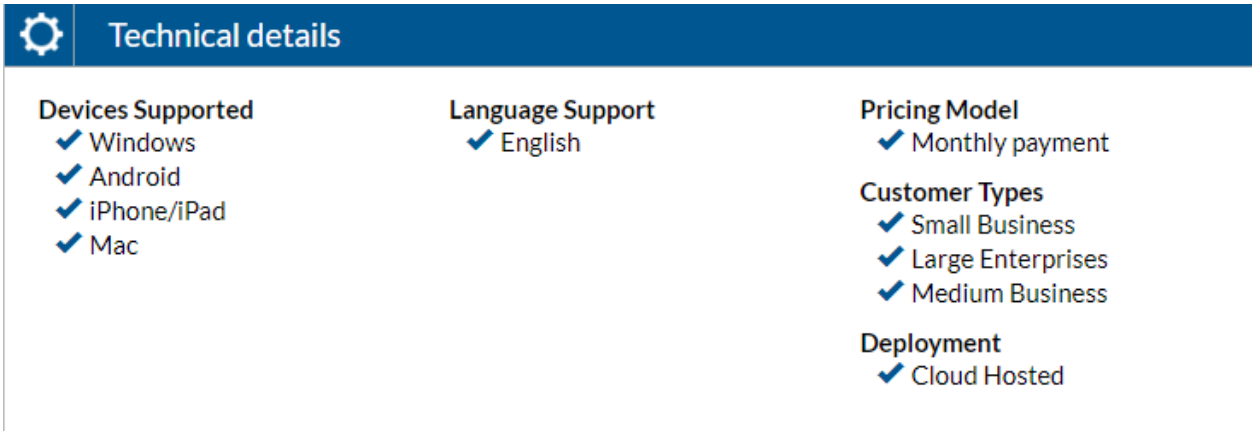
How it Works:

It creates a visual model of your team's work and workflow that's unlike those shown in list-based tools. The intuitive software provides a single, consolidated view of the work that teams can easily access and interact with across multiple locations, devices, and systems. The built-in collaboration features make it easy to ask questions and share status updates. Teams can display their Lean Kit boards on a touchscreen monitor to collaborate on work planning and execution. Today's CEOs need to innovate and evolve faster than ever before. Lean Kit is pioneering Lean across all business functions to help organizations deliver more customer value — faster.

Logo of Lean Kit:

This LOGO marks a significant milestone in our journey, providing an opportunity to reflect on everything that the original Lean Kit brand stood for and everything that we aspire to become. -
Chris Hefley

Technical Details of Lean Kit:



Benefits:

- Lean Kit can easily be accessed anytime, anywhere from a browser or mobile device. Mobile apps for iOS and Android let you enables updating, checking of work status, and adding comments while on the go.
- Making the work visible — along with blockers, bottlenecks, and expedites — gives the team a shared understanding of project activity and status.
- With Lean Kit's tiered board approach, team leaders are able to visualize team initiatives at a high level and it keeps track of the subtasks for each work item using task boards.
- Lean Kit easily integrates with other systems. It can synchronize projects between JIRA, MS Visual Studio/TFS and MS Visual Studio Online, MS Project Server, GitHub and Oracle Primavera.

Conclusion:

Using a humanistic, rather than mechanistic, approach to productivity, Lean Kit wants to help pave the way for the future of work by creating a visual tool that reinvents the way teams get stuff done. It manages backlogs, accommodate file attachments up to unlimited sizes and enable comment threads, emails and RSS updates with unlimited board sharing.

References:

https://books.google.co.in/books?id=IJ1gAgAAQBAJ&printsec=frontcover&dq=lean+kit+pdf&hl=en&sa=X&ved=0ahUKEwiR5oPP7uvcAhVPWisKHfQdCP8Q6AEINjAC#v=onepage&q&f=false
https://books.google.co.in/books?id=IqXcCwAAQBAJ&printsec=frontcover&dq=lean+kit+pdf&hl=en&sa=X&ved=0ahUKEwiR5oPP7uvcAhVPWisKHfQdCP8Q6AEIPTAD#v=onepage&q&f=false

JIRA AGILE

APURVA (17MCA05)

VARSHA SAHAY (17MCA28)

INTRODUCTION

JIRA Software is an AGILE project management tool that supports any agile methodology, be it scrum, kanban, or your own unique flavor. From agile boards to reports, you can plan, track, and manage all your agile software development projects from a single tool. JIRA lets you prioritize, assign, track, report and audit your 'issues', from software bugs and helpdesk tickets to project tasks and change requests .More than just an issue tracker, JIRA is an extensible platform that you can customize to match your business processes . JIRA improves productivity by cutting down on time wasted on tracking issues and coordination. JIRA improves quality by ensuring all tasks are recorded down with all the details and followed up till completion.

FEATURES OF JIRA AGILE

- **Scrum boards:-**Agile teams can stay focused on delivering iterative and incremental value, as fast as possible, with customizable scrum boards.
- **Kanban boards:-**Flexible kanban boards give your team full visibility into what's next so you can continuously deliver maximum output in minimal cycle time.
- **Agile reporting:-**Teams have access to more than a dozen out-of-the-box reports with real-time, actionable insights into how their team is performing sprint over sprint.
- **Agile roadmap planning:-**With the Portfolio for Jira app you can create a roadmap that's connected to your work in Jira Software, track progress across teams and projects and easily share your plan with stakeholders.
- **Secure:-**JIRA provides fine-grained enterprise level security.
- **Track:-** Keeps track of all activities,changes and work logged against issues.
- **Extensible:-**With over 100 plugins contributed by the community.
- **Open:-**An open API, full source code access allow for further integration and customization of JIRA functionality Akeles Consulting.

ADVANTAGES OF JIRA AGILE

- **Flexible:-** JIRA started out as being a bug tracking system. It has become a general purpose issue/task tracking solution. You can use it to track any "tasks".
- **Customizable:-** Every aspect of the system can be customized through configuration (not custom development). It does add a bit complexity to the JIRA admin, but it is still a better trade off than writing code.
- **Workflow:-**JIRA has a robust and customizable workflow. Workflow can be designed graphically.
- **Extensible :-** JIRA can be extended further through add on/plugin system. There are a wide range of add on/plugin available, including free and paid ones. It is very likely that you will find the plugin you need for your project.
- **Release Management :–** JIRA has added release management features in rencent releases. You can use it as a release management tool as well.

- **Usability :–** There is no need to attend any training to start using JIRA. It has becoming slightly complex over the years, but the usability is still very good compared with other tools.
- **Community -** JIRA has a great community of users and third-party vendors because it is so widely used.

DISADVANTAGES OF JIRA AGILE

- Mass-editing and scripting to, say, mass-rename a set of labels to a new one or many
- Command line interface would be nice, for power users
- Version number is difficult to find leading to difficulty judging what version we run thus which features we are truly missing
- Workflow edits aren't protected from errors/inconsistencies like mapping a label to null
- Workflows can be loop-to-self, be massively inefficient, should be a workflow "linter".

CONCLUSION

JIRA has fair, upfront licensing policy and is battle tested by the world corporations. The information delivered in real-time are provided in a convenient format. The powerful and flexible search enables precise search in seconds.

REFERENCES

1. https://confluence.atlassian.com

ANALYSIS OF CLOUD IDES FOR SOFTWARE DEVELOPMENT

Y.HARI CHANDANA (17MCA29)

AFNAAN K

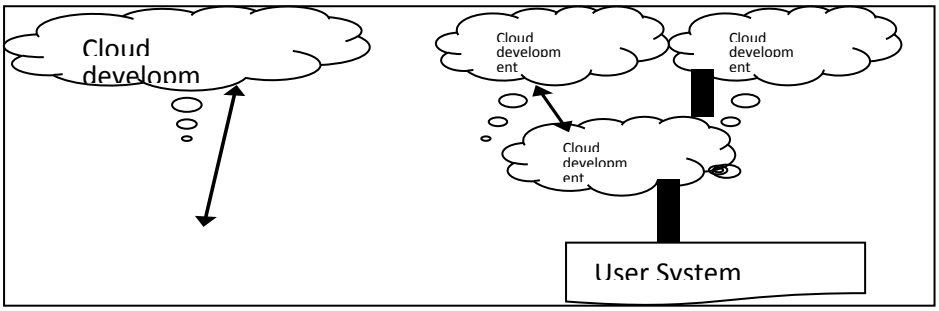
ABSTRACT:

In today world using cloud services has become a convenience for software developers to develop application. An integrated development environment (IDE) facilitates the development of apps in shorter software delivery timelines. A cloud based IDE is an „next gen" solution and still in the incubation stage because the cloud based private workspace is yet to evolve fully.as of now, there still is not any good java-supporting cloud IDE available. Evaluation of different cloud IDEs is discussed in this paper

INTRODUCTION:

The development process plays a vital role in developing a software product. The software development process consists of requirement gathering designing, coding, testing, deployment and maintenance. The basic requirements for software development are hardware and software tools, a server to deploy and test the application and a coding development environment. Cloud based development brings in a lot of cost savings and efficiency into the process of software development. The cloud provides a deployment environment for developers and the developer do not require building, developing or maintaining this environment but can share it easily from the public environment.

Software development and the cloud IDE



THE CLOUD IDE : The basic motive behind the cloud IDE based platform is to create a virtual development environment. It can provide the following features for the software development process :

a. programmer can develop an application in one or more software technologies such as Java, C++, HTML, JavaScript and so on.
b. It has an application container to deploy the application on-the-fly.
c. It offers repository management to store, retrieve and share code between groups of developers.
d. In addition to these basic features, we need to add security to limit the access to only the selected group of developers. This can be achieved by an authentication system that protects the development environment from public us

Advantages of working in cloud-based IDEs

- Programming workspace is a single and centralized environment in which multiple people can co-build, co-edit and co-debug .Easy and instant access to codes and libraries through the web
- Enables developers to work on the go without the need to occupy systems with loads of heavy codes and data
- Accessible through all devices and browsers

ANALYSIS OF OPEN SOURCE CLOUD IDES :

The popular open source cloud IDEs include Cloud9, code anywhere, Eclipse Orion, Coder In the following section the various features and advantages and disadvantages of these tools based on experience of users are discussed.

CLOUD9

Cloud9 IDE is capable of handling projects from JS, HTML, PHP AND Ruby. it support development in JavaScript, GIT, Selenium, HTML/CSS, XML, XQuery, PHP, Ruby and Node.js. it can be used for web application development. Cloud9 supports repositories like GIT, the distributed revision control tool Mercurial, and Apache subversion. It also facilitates deployment, which can be done directly from Cloud9 to Joyent and Heroku. Its limitation is that it is not useful for Java application development or J2EE application development/handling. Advantages: Supports a variety of languages and various repository tools.

Disadvantages: can be used for simple application development but not for large or complex applications.

CODE ANYWHERE

This is lightweight browser based IDE tool. It supports web application development. It support HTML, CSS, Java Script, PHP, MySQL, and more.

This tool is used to develop and test code from anywhere i.e from mobile device also as it is also available as a mobile application. Codeanywhere is supported in IOS, Android and BlackBerry based mobile devices and tablets. Advantages : Lightweight and easy to use. Support for mobile devices.

disadvantages: it is more useful for web page development and it doesnot support a variety of programming languages.

CONCLUSION

For a developer or a small company that lacks the capability to install an IDE locally forsoftware development or wants to develop a program on-the-fly, the tools described above would be helpful to experiment on,prior to choosing a development environment. With the adoption of several simple security precautions and an open mindset, any software developer can reap the rewards of Cloud software suites without any real danger.

Continuous Integration Using Jenkins

DIVYA G M (17MCA08)

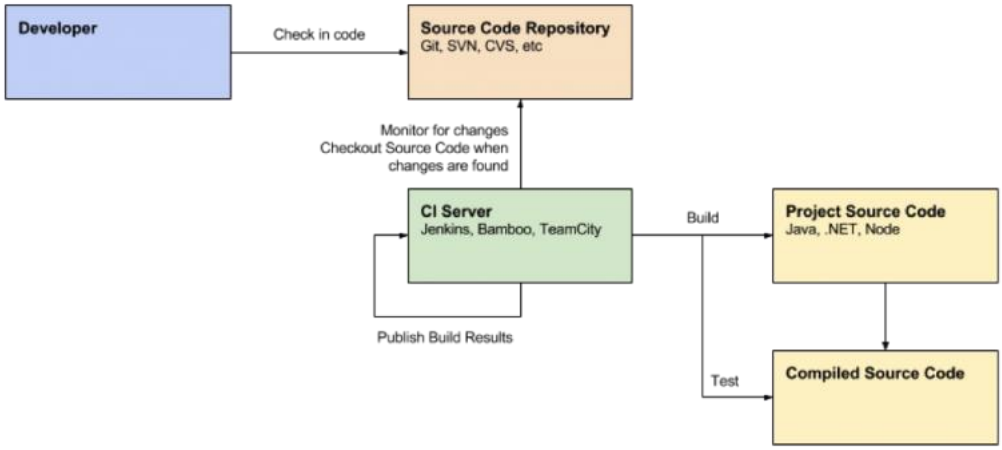
INTRODUCTION

Jenkins is one of the most popular open-source continuous integration and continuous delivery servers available today. It began as a product called Hudson, developed at Sun Microsystems in 2004-2005, before it was forked from Hudson and renamed Jenkins in 2011, as the result of a dispute between the Hudson community and Oracle .Kohsuke Kawaguchi, the creator of Hudson/Jenkins became the Chief Technical Officer for Cloud bees in 2014 and Cloud bees now commercially offers Jenkins as a cloud solution.

Jenkins provides support for all popular source code management (SCM) systems, including Git, Subversion, Mercurial, and CVS, popular build tools like Maven, Ant, Gulp, and Grunt, as well as testing frameworks and report generators. Jenkins plugins provide strong support for technologies like Docker and ECS, which enable the creation and deployment of cloud-based microservice environments, both for testing as well as production deployments. And, in 2016, Jenkins released powerful delivery pipeline support using a Groovy Domain Specific Language (DSL) in what they refer to as "*Pipeline as Code*".

ABOUT CONTINUOUS INTIGRATION

Continuous Integration was popularized by an article that Martin Fowler wrote in September 2000 and subsequently updated in May 2006. Continuous integration introduces a new server application called a Continuous Integration Server (CI Server) that monitors a source code repository and, when a developer checks in code, the CI Server checks out that code, builds it, and executes its unit tests. This is summarized in figure 1.



IMPLIMENTATION

Jenkins can run in several different ways:

- It can run as a WAR file deployed to a Servlet container, such as Apache Tomcat
- It can run in a Docker container, either locally or on a public or private cloud
- It can run as a Software as a Service, hosted by a company like Cloud Bees

CONCLUSION

Continuous integration, through automation of both builds and tests, dramatically improved application release cycles by rendering integration as a non-issue. As continuous integration evolved, the industry created the notion of continuous delivery and continuous deployment to render production deployments a non-issue. These automated tools allow us to go from checking in code to deploying a new version of our application to production.

This series introduced continuous integration, continuous delivery, and continuous deployment and presented Jenkins as an open source tool that enables all of this. In the next article we will setup an application for continuous integration, including executing unit tests and static code analysis, and subsequent articles will review how to implement a deployment pipeline and realize continuous delivery and deployment.

REFERENCES

- J. Humble and D. Farley, Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, 2010.
- K. Kawaguchi, "Jenkins CI," http://jenkins-ci.org/, [Online; accessed 03-Mar-2013].
- M. Fowler and M. Foemmel, Continuous Integration. ThoughtWorks, 2006.
- K. Beck and C. Andres, Extreme Programming Explained: Embrace Change. Addison-Wesley Professional,2004