Unit II Agile Development Process

- Agile Development: Agile manifesto, agility and cost of change,
- Agile principles, myth of planned development
- Tools for Agile Project Management
- Scrum- process flow, scrum roles, events and artifacts
- Scrum cycle description, product backlog, sprint planning meeting
- sprint backlog, sprint execution, daily scrum meeting
- maintaining sprint backlog and burn-down chart, sprint review and retrospective.
- Agile Practices: test driven development, refactoring, pair programming, continuous integration, exploratory testing versus scripted testing

What is Agile?

- Agile --readiness for motion, move quickly and easily, dexterity in motion

- Agility

  The ability to both create and respond to change in order to profit in a turbulent business environment

  - Companies need to determine the amount of agility they need to be competitive.

Agile Software Development

- **Agile software development** is a conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project.

- Software developed during one unit of time is referred to as an iteration, which may last from one to four weeks.

- Agile methods also emphasize working software as the primary measure of progress.

Agile Software Development: Intro

- Characteristics of Agile Software Development
  -- Light Weighted methodology
  -- Small to medium sized teams
  -- vague and/or changing requirements
  -- vague and/or changing techniques
  -- Simple design

1. Agile Software Development

- **Agile software development** is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.
  - Methods
  - Iterative
  - Incremental

- It promotes **adaptive** planning, **evolutionary** development and delivery, a time-boxed **iterative** approach, and encourages rapid and flexible response to **change**.

- It is a conceptual framework that promotes **foreseen interactions** throughout the development cycle.

- The *Agile Manifesto*[introduced the term in 2001. (Wiki, 21 Aug 12)

- Let's take this definition apart.

a. Software Development Method

- A **software development methodology** or **system development methodology** in *software engineering* is a framework that is used to structure, plan, and control the process of developing an *information system*.

b. Software Engineering / Software Development

- **Software Engineering** (WIKI) (SE) is the application of a systematic, disciplined, quantifiable approach to the design, development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

- **Software development** is the process of conceiving, specifying, designing, programming, documenting, testing, and bug fixing involved in creating and maintaining applications, frameworks, or other software components.(WIKI)

c. Information System

- An **information system** (IS) - is any **combination** of information technology (**IT**) and people's **activities** that support operations, management and decision making.

- In a very broad sense, the term information system is frequently used to refer to the **interaction** between people, processes, data and technology.

- In this sense, the term is used to refer
  - **not only** to the information and communication technology (ICT) that an organization uses,
  - **but also** to the way in which **people interact** with this technology in support of business processes. (Wiki)

Iterative and Incremental Development

Iterative and Incremental development is at the heart of a cyclic software development process developed in response to the weaknesses of the waterfall model.

It starts with an initial planning and ends with deployment with the cyclic interactions in between.

Iterative and incremental development are essential parts of the Rational Unified Process, Extreme Programming and generally the various agile software development frameworks.

It follows a similar process to the “plan-do-check-act” cycle of **business process improvement**.

e. Time-Boxed Approach

- In time management, a **time box** allots a **fixed** period of time for an activity.

- **Timeboxing** plans activity by allocating time boxes.

2. Introductory Thoughts

- Fears regarding software development led to a number of pioneers / industry experts to develop the **Agile Manifesto** based up some firm values and principles.

- Practitioners had become afraid that repeated software failures could not be stopped without some kind of **guiding process** to guide development activities.

Common Fears

- Practitioners were afraid that
  - The project will produce the wrong product
  - The project will produce a product of poor quality
  - The project will be late
  - We'll have to work 80 hour weeks
  - We'll have to break commitments
  - We won't be having fun.

Agile Alliance/Agreement

- Several individuals, **The Agile Alliance**,
  - motivated to make activities
  - such that certain outputs and artifacts are **predictably** produced.
  - Around 2000, these notables got together to address common development problems.

- Goal: outline values and principles to allow software teams to
  - **develop quickly** and
  - **respond to change.**

- These activities stand up in large part to **runaway processes**.
  - Failure to achieve certain goals was met with ‘more process.’ Schedules slipped; budgets bloated, and processes became even larger.

- The Alliance (17) created a statement of **values**: termed the **manifesto** of the Agile Alliance.

- They then developed the **12 Principles of Agility.**

The Manifesto for Agile Software Development

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

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

Kent Beck et al

Value 1: Individuals and Interactions over Processes and Tools

- **Strong players**: a must, but can fail if don't work together.
- **Strong player:** not necessarily an 'ace;' work well with others!
  - Communication and interacting is **more important** than raw talent.

- '**Right' tools** are vital to smooth functioning of a team.

- **Start small**. Find a free tool and use until you can demo you've expand it. Don't assume bigger is better. Start with white board; flat files before going to a huge database.

- **Building a team** more important than **building environment.**
  - Some managers build the environment and expect the team to fall together.
  - Doesn't work.
  - Let the team build the environment on the **basis of need**.

Value 2: Working Software over Comprehensive Documentation

- **Code** – not ideal medium for communicating rationale and system structure.
  - Team needs to produce human readable documents describing system and design decision rationale.

- **Too much documentation is worse than too little**.
  - Take time; more to keep in sync with code; Not kept in sync? it is a lie and misleading.

- **Short rationale and structure document**.
  - Keep this in sync; Only highest level structure in the system kept.

Value 3: Customer Collaboration over Contract Negotiation

- Not possible to describe software requirements up front and leave someone else to develop it within cost and on time.
- Customers cannot just cite needs and go away
- Successful projects require **customer feedback on a regular and frequent basis** – and not dependent upon a contract.

Value 4: Responding to Change over Following a Plan

- **Course of a project cannot be predicted far into the future.**
- **Better planning strategy – make detailed plans for the next few weeks, very rough plans for the next few months, and extremely rough plans beyond that.**
- Need to know what we will be working on the next few weeks; roughly for the next few months; a vague idea what system will do after a year.
- **Only invest in a detailed plan for immediate tasks**; once plan is made, difficult to change due to commitment.
  - But rest of plan remains flexible. The lower resolution parts of the plan can be changed with relative ease.

What is Agility?

- Effective (rapid and adaptive) response to change (team members, new technology, requirements)
- Effective communication in structure and attitudes among all team members, technological and business people, software engineers and managers。
- Drawing the customer into the team.
  Planning in an uncertain world has its limits and plan must be flexible.
- Organizing a team so that it is in control of the work performed

- **Rapid, incremental delivery of software**

An Agile Process

- Is driven by customer descriptions of what is required (scenarios). Some assumptions:
  - Recognizes that plans are short-lived (some requirements will persist, some will change. Customer priorities will change)
  - Develops software iteratively with a heavy emphasis on construction activities (design and construction are interleaved, hard to say how much design is necessary before construction. Design models are proven as they are created.)
  - Analysis, design, construction and testing are not predictable.
- Thus has to Adapt as changes occur due to unpredictability
- Delivers multiple 'software increments', deliver an operational prototype or portion of an OS to collect customer feedback for adaption.

Extreme programming

- Perhaps the best-known and most widely used agile method is XP.
- Extreme Programming (XP) takes an ‘extreme’ approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.

Extreme Programming (XP)

- **The most widely used agile process, originally proposed by Kent Beck in 2004. It uses an object-oriented approach.**
- XP Planning
  - Begins with the listening, leads to creation of "user stories" that describes required output, features, and functionality. Customer assigns a value(i.e., a priority) to each story.
  - Agile team assesses each story and assigns a cost (development weeks. If more than 3 weeks, customer asked to split into smaller stories)
  - Working together, stories are grouped for a deliverable increment next release.
  - A commitment (stories to be included, delivery date and other project matters) is made. Three ways: 1. Either all stories will be implemented in a few weeks, 2. high priority stories first, or 3. the riskiest stories will be implemented first.
  - After the first increment "project velocity", namely number of stories implemented during the first release is used to help define subsequent delivery dates for other increments. Customers can add stories, delete existing stories, change values of an existing story, split stories as development work proceeds.

Extreme Programming (XP)

- XP Design (occurs both before and after coding as refactoring is encouraged)
  - Follows the KIS principle (keep it simple) Nothing more nothing less than the story.
  - Encourage the use of CRC (class-responsibility-collaborator) cards in an object-oriented context. The only design work product of XP. They identify and organize the classes that are relevant to the current software increment.
  - For difficult design problems, suggests the creation of "spike solutions"—a design prototype for that portion is implemented and evaluated.
  - Encourages "refactoring"—an iterative refinement of the internal program design. Does not alter the external behavior yet improve the internal structure. Minimize chances of bugs. More efficient, easy to read.
- XP Coding
  - Recommends the construction of a unit test for a story *before* coding starts. So implementer can focus on what must be implemented to pass the test.
  - Encourages "pair programming". Two people work together at one workstation. Real time problem solving, real time review for quality assurance. Take slightly different roles.
- XP Testing
  - All unit tests are executed daily and ideally should be automated. Regression tests are conducted to test current and previous components.
  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionality.

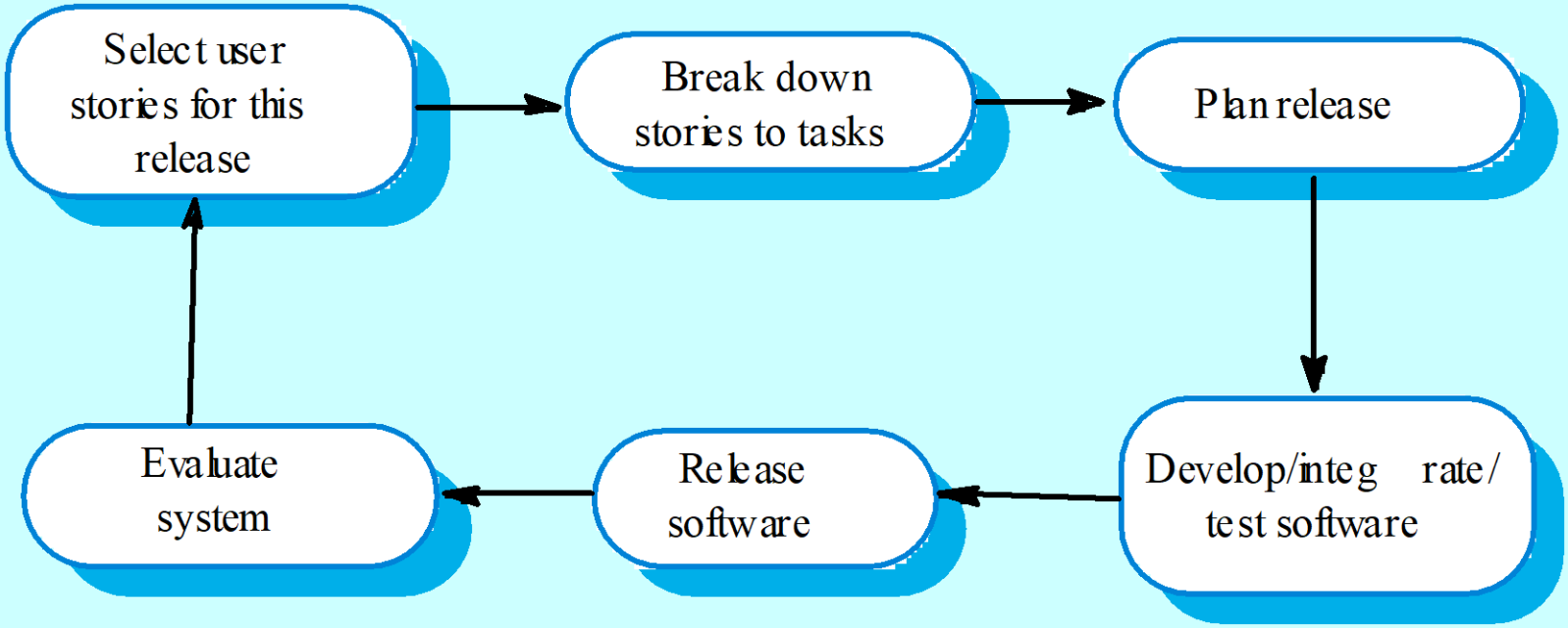
Extreme programming practices 1

Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development ŌTasksÕ
Small Releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple Design	Enough design is carried out to meet the current requirements and no more.
Test first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

Extreme programming practices 2

Pair Programming	Developers work in pairs, checking ea ch otherÕs work and providing the support to always do a good job.
Collective Ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers own all the code. Anyone can change anything.
Continuous Integration	As soon as work on a task is complete it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of over-time are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site Customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

The XP release cycle



Agility Principles - I

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. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. 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.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity – the art of maximizing the amount of work not done – is essential.

11. The best architectures, requirements, and designs emerge from self–organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Principles (12)

- The following principles are those that differentiate agile processes from others.

Principle 1: Our Highest Priority is to **Satisfy the Customer** through **Early** and **Continuous Delivery** of Valuable Software

- Number of practices have significant impact upon quality of final system:
- 1. Strong **correlation** between **quality** and **early delivery of a partially functioning system.**
  - The less functional the initial delivery, the higher the quality of the final delivery.
- 2. Another strong **correlation** exists between **final quality** and **frequently deliveries of increasing functionality**.
  - The more frequent the deliveries, the higher the final quality.

- **Agile processes deliver early and often**.
  - Simple system **first** followed by systems of **increasing functionality** every few weeks.
  - Customers may use these systems in production, or
  - May choose to review existing functionality and report on changes to be made.
  - Regardless, they must provide meaningful **feedback**.

Principle 2: Welcome Changing Requirements, even late in Development. Agile Processes harness change for the Customer's Competitive Advantage.

- This is a statement of **attitude**.
- Participants in an agile process are **not afraid** of change.
  - Requirement changes are good;
  - Means team has learned more about what it will take to satisfy the market.

- Agile teams work to keep the **software structure flexible**, so requirement change impact is minimal.
-
- Moreso, the **principles of object oriented** design help us to maintain this kind of flexibility.

Principle 3: Deliver Working Software Frequently

(From a couple of weeks to a couple of months with a preference to the shorter time scale.

- We deliver working software.
  - **Deliver early and often**.
- The **goal** of delivering software that satisfies the customer's needs.

Principle 4: Business People and Developers Must **Work Together Daily** throughout the Project.

- For agile projects, there must be **significant** and **frequent interaction** between the
  - customers,
  - developers, and
  - stakeholders.

An agile project must be **continuously guided**.

Principle 5: Build Projects around Motivated Individuals. (Give them the environment and support they need, and trust them to get the job done.)

- **An agile project has people the most important factor of success.**
  - All other factors, process, environment, management, etc., are considered to be second order effects, and are subject to change if they are having an adverse effect upon the people.

- **Example:** if the office environment is an obstacle to the team, **change the office environment**.
- If certain process steps are obstacles to the team, **change the process steps.**

Principle 6: The Most Efficient and Effective Method of Conveying Information to and within a Development Team is **face-to-face Communications**.

- In agile projects, developers ***talk*** to each other.
  - The **primary mode of communication is conversation**.
  - Documents may be created, but there is no attempt to capture all project information in writing.
- An agile project team **does not demand written** specs, written plans, or written designs.
  - They may create them if they perceive **an immediate and significant need**, but they are not the default.
  - The **default is conversation**.

Principle 7: Working Software is the Primary Measure of Progress

- Agile projects measure their progress by measuring the amount of **working software**.
  - Progress **not measusred** by phase we are in, **or**
  - by the volume of produced documentation **or**
  - by the amount of code they have created.
- **Agile teams are 30% done when 30% of the necessary functionality is working.**

Principle 8: Agile Processes promote sustainable development

The sponsors, developers, and users should be able to **maintain a constant pace** indefinitely.

- An agile project is not run like a 50 yard dash; it is run like a marathon.
  - The team does not take off at full speed and try to maintain that speed for the duration.
  - Rather they run at a fast, but **sustainable**, pace.

- Running too fast leads to stress, shortcuts.
- Agile teams pace themselves.
  - They don't allow themselves to get too tired.
  - They don't borrow tomorrow's energy to get a bit more done today.
  - They work at a **rate** that allows them to maintain the highest quality standards for the duration of the project.

Principle 9: Continuous Attention to Technical Excellence and Good Design enhances Agility.

- **High quality is the key to high speed**.
  - The way to go fast is to **keep the software as clean and robust as possible.**
  - Thus, all agile team-members are **committed** to producing only the **highest quality code** they can.
  - They do not make messes and then tell themselves they'll clean it up when they have more time.
  - **Do it right the <u>first</u> time!**

Principle 10: **Simplicity – the art of maximizing the amount of work not done – is essential.**

- Agile teams take the simplest path that is consistent with their goals.
  - They don't anticipate tomorrow's problems and try to defend against them today.
  - **Rather they do the simplest and highest quality work today, confident that it will be easy to change if and when tomorrows problems arise.**

Principle 11: The Best Architectures, Requirements, and Designs emerge from **Self-Organizing Teams**

- An agile team is a self organizing team.
  - Responsibilities are **not handed to individual team members** from the outside.
  - Responsibilities are **communicated** to the team as a whole, and the **team determines** the best way to fulfill them.

- Agile team members work together on all project aspects.
  - Each is allowed input into the whole.
  - No single team member is responsible for the architecture, or the requirements, or the tests, etc.
  - The team shares those responsibilities and each team member has influence over them.

Principle 12: At regular Intervals, the **Team reflects** on how to become **more** effective, **then tunes and adjusts** its **behavior** accordingly.

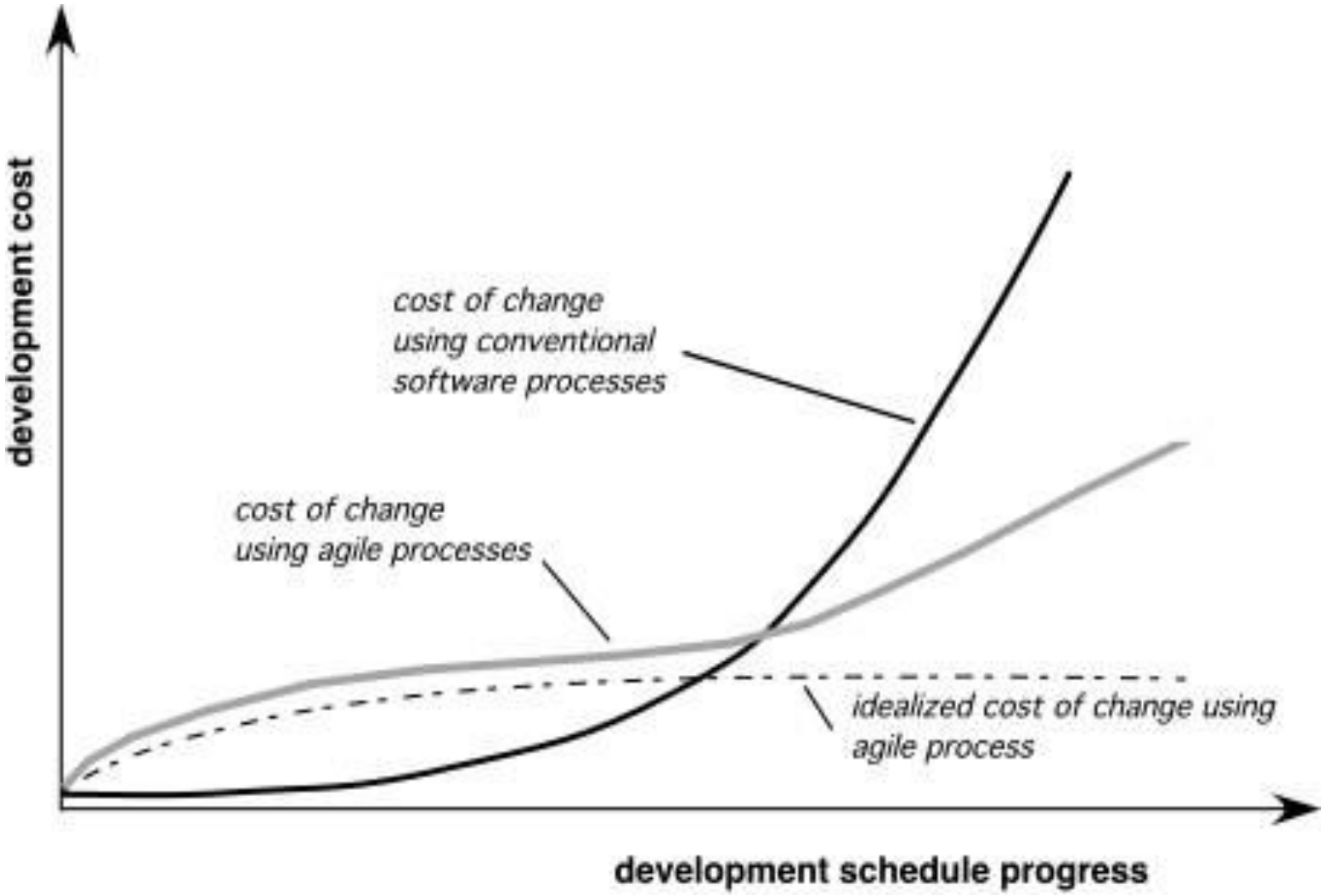
- An agile team continually adjusts its organization, rules, conventions, relationships, etc.
- An agile team knows that its environment is continuously changing, and knows that they must change with that environment to remain agile.

Agility and the Cost of Change

- Conventional wisdom is that the cost of change increases nonlinearly as a project progresses. It is relatively easy to accommodate a change when a team is gathering requirements early in a project. If there are any changes, the costs of doing this work are minimal. But if the middle of validation testing, a stakeholder is requesting a major functional change. Then the change requires a modification to the architectural design, construction of new components, changes to other existing components, new testing and so on. Costs escalate quickly.

- A well-designed agile process may "flatten" the cost of change curve by coupling incremental delivery with agile practices such as continuous unit testing and pair programming. Thus team can accommodate changes late in the software project without dramatic cost and time impact.

Agility and the Cost of Change



Tools for Agile Project Management

In agile development, the emphasis on building the right product as per customer needs. Therefore, the agile testers needs to monitor their project constantly. There are many tools available for this purpose.

1. Agilean

First one on the list of agile project management tools is Agilean. It is a SaaS enterprise workflow automation and project management software solution that is basically created to be used by small-medium IT enterprises. The main features of Agilean include project planning, execution, monitor, impediments and response plan, stand up meeting automation, release management, retrospective analysis, and visualized reports.

Wrike

Another great one from the list of the best agile project management tools is Wrike that is one of the best in terms of integrating email with project management, having main features inside. It is built to scale and drive results by giving you the flexibility you might need to manage multiple projects and teams at one place. Along the Agile process, you going to get the accurate, up-to-date information and you can insert & add any important information inside. Planning should be easier, there will be always accurate information and real-time reports and analytics that going to save your time and help in analyzing the situation. This is for sure one of the resources that your team might d like to use daily: customization supportive & collaboration tools and many other things that will keep your team focused.

3. JIRA

JIRA is a tool developed for bug and issue tracking and project management to software and mobile development processes. The JIRA dashboard has many useful functions & features which are able to handle different issues easily. Some of its key features and issues are: issue types, workflows, screens, fields and issue attributes. Some of these features you won't find elsewhere. The dashboard on JIRA can be customized to match your business processes.

JIRA is a defect tracking tool which is used for Agile testing as well as project management. This tool is not only used for recording, reporting but also integrated with code development environment.

Features:

JIRA Query Language helps to create quick filters with a single click
Estimations help your team become more accurate and efficient
Reporting functionality gives team critical insight into their agile process
Extensive reporting functionality gives your team critical insight into their agile process.
Allows creating custom workflows of any size which is helpful to build, test, and release software

4. Trello

I guess many of you already know about Trello, one of the most used and well-known project management applications. It has both free and premium accounts that give you a great chance to use most of the common functions. The structure of Trello is based on the kanban methodology. All the projects are represented by boards, that contain lists.

Every list has progressive cards that you make as drag-and-drop. Users that are related to the board can be assigned to said cards. Also, it has many nice, small but not less useful features I would like to indicate: writing comments, inserting attachments, notes, due dates, checklists, colored labels, integration with other apps, etc. Additionally, Trello is supported by all mobile platforms. What I also like about Trello is that this tool can be used both for work, like we do it in Apiumhub, and personal processes.

7. Backlog

Backlog is an all-in-one online project management tool for task management, version control, and bug tracking. With features like subtasking, custom issue fields, and Gantt charts, it's easy for teams to define, organize, and track their work.

Burndown charts, Git & SVN repositories, and Wikis help developers review, track, release, and document their code. And targeted notifications keep everyone in the loop along the way. Bringing together the organizational benefits of project management with the power and convenience of code management, Backlog enhances team collaboration across organizations large and small.

6. Jmeter:

The Apache JMeter application is an open source agile performance testing tool. It is used to load functional test behavior and measure performance of the website.

Features:

- Ability to load and performance test different applications/server and protocols
- Full featured Test IDE for fast Test Plan recording
- It offers complete portability and 100% Java purity
- Data analysis and visualization plugins offers great extensibility
- Functions can be used to offer dynamic input to test or provide data manipulation
- Easy Continuous Integration using third party libraries for tools like Maven, Gradle,and Jenkins

. Pivotal Tracker is a tool that helps developers for planning project for software development. It's mainly based on agile development methods. However, it works effectively with all kinds of projects.

Features:

Support for ActiveResource

Ability to get a list of all the projects

Project transparency at a glance

Move and edit multiple stories at once

List of other tools for Agile project management

1. Kanbanize
2. Assembla
3. Appium:
4. Selenium
5. Bug Shooting
6. Usersnap
7. SoapUI
8. qTest Scenario
9. Qmetry
10. QAcomplete
11. Enterprise Tester:

ETC…

Agile Methods

- Agile methods:
  - Scrum
  - Extreme Programming
  - Adaptive Software Development (ASD)
  - Dynamic System Development Method (DSDM)

Thank You