COURSE NAME

SOFTWARE ENGINEERING

CSC 3114

(UNDERGRADUATE)

CHAPTER 3

AGILE SOFTWARE DEVELOPMENT

M. MAHMUDUL HASAN
ASSISTANT PROFESSOR, CS, AIUB
Web: http://cs.aiub.edu/profile/m.hasan
WordPress: https://wordpress.com/view/mhasansuman.wordpress.com

RG Google Scholar LinkedIn

AGILE DEVELOPMENT

"Plan-driven methods work best when developers can determine the requirements in advance . . . and when the requirements remain relatively stable, with change rates on the order of one percent per month."~ *Barry Boehm*

- Agility is the ability to create and respond to change in order to profit in a turbulent business environment
- Companies need to
  - innovate better and faster operations
  - respond quickly to
    - competitive initiatives
    - new technology
    - customer's requirements

AGILE SOFTWARE DEVELOPMENT MODEL

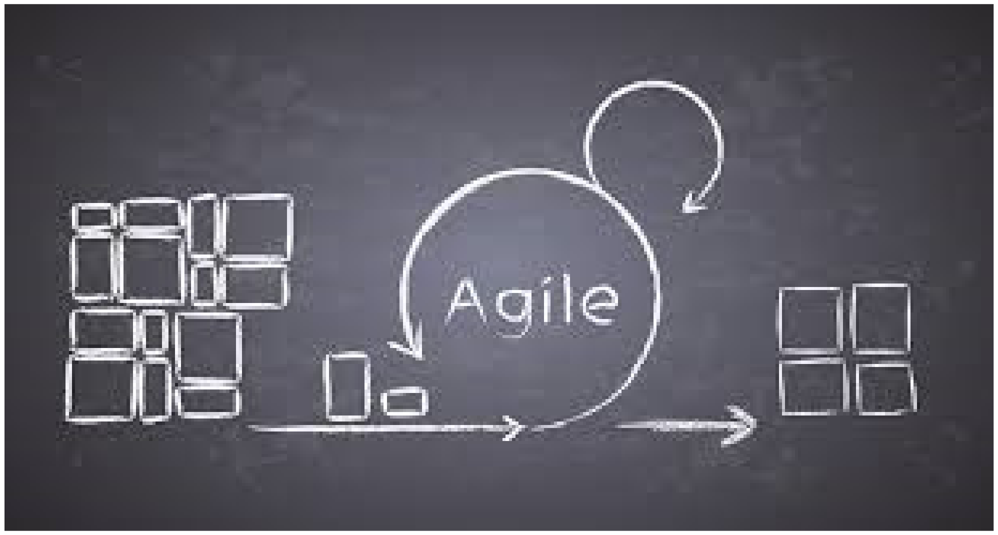
- Subset of iterative and evolutionary methods

Iterative Products

- Each iteration is a self-contained, mini-project with activities that span requirements analysis, design, implementation, and test
- Leads to an iteration release (which may be only an internal release) that integrates all software across the team and is a growing and evolving subset of the final system
- The purpose of having short iterations is so that feedback from iterations N and earlier, and any other new information, can lead to refinement and requirements adaptation for iteration N + 1

TIMEBOX & SCOPE

- The pre-determined iteration length serves as a timebox for the team.
- Scope (set of tasks) is chosen for each iteration to fill the iteration length.
- Rather than increase the iteration length to fit the chosen scope, the scope is reduced to fit the iteration length.



AGILE METHODS VS. PAST ITERATIVE METHODS

- A key difference between agile methods and past iterative methods is the length of each iteration

- In the past, iterations might have been three or six months long
- In agile methods, iteration lengths vary between one to four weeks, and intentionally do not exceed 30 days
- Research has shown that shorter iterations have lower complexity and risk, better feedback, and higher productivity and success rates

AGILE PROPONENTS BELIEVE

- Current software development processes are too heavyweight or cumbersome
  - Too many things are done that are not directly related to software product being produced
- Current software development is too rigid
  - Difficulty with incomplete or changing requirements
- Short development cycle is needed
- More active customer involvement needed

WHAT IS AN AGILE METHOD?

- Agile methods are considered
  - Lightweight (do not concentrate on the whole s/w development at once)
  - People-based rather than Plan-based
- Several agile methods
  - No single agile method
  - Different agile methods can be combined in s/w development (Hybrid)
- No single definition. Agile Manifesto closest to a definition
  - Set of principles
  - Developed by Agile Alliance

AGILE VALUE STATEMENT

- In 2001, Kent Beck and 16 other noted software developers, writers, and consultants (known as Agile Alliance) signed a manifesto as following:

"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

AGILE ASSUMPTION

- It is difficult to predict in advance which software
  - requirements will persist and which will change
  - It is equally difficult to predict how customer priorities will change as the project proceeds
- Design and construction are interleaved in many types of software. That is, both activities should be performed tightly so that design models are proven as they are created. It is difficult to predict how much design is necessary before construction is used to prove the design.
- Analysis, design, construction, and testing are not as predictable (from a planning point of view) as we might like.

AGILE MANIFESTO (POLICY)

1. Our highest priority is to satisfy the costumer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile process harness (control) change for the customer´s competitive advantage
3. Deliver working software frequently with a preference to the shorter timescale
4. Businesspeople and developers must work together daily throughout the project
5. Build projects around motivated individuals. Give them the environment and support their 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

AGILE MANIFESTO (POLICY)

7. 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 persistently
9. Continuous attention to technical excellence and good design enhances agility
10. Simplicity – use simple approaches to make changes easier
11. The best architectures, requirements, and designs emerge from self-organizing teams (iterative development rather defined plans)
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

AGILE VS. PLAN DRIVEN PROCESS

Agile Process	Plan Driven Process
Small products and teams; scalability limited	Large products and teams; hard to scale down
Inappropriate for safety-critical products because of frequent changes	Handles highly critical products
Good for dynamic, but expensive for stable environments.	Good for stable, but expensive for dynamic environments
Require experienced Agile personnel throughout	Require experienced personnel only at start if stable environment
Personnel succeed on freedom and chaos	Personnel succeed on structure and order

HUMAN FACTORS IN AGILE DEVELOPMENT

- Competence/skill/capability
- Common focus
- Collaboration
- Decision-making ability
- Fuzzy (vague) problem-solving ability
- Mutual trust and respect
- Self-organization

SOME OTHER METHODOLOGY

- Extreme Programming (XP)
- Scrum
- Scrumban
- Dynamic Systems Development Method (DSDM)
- Feature-Driven Development (FDD)
- Crystal Methods
- Lean Software Development (LD)
- Adaptive Software Development (ASD)
- Agile UP
- Kanban

AGILE: CASE - I

- You have been called upon to help out a struggling development team. The team is working on developing a mobile ticket app for a professional baseball franchise and has failed to deliver a working product on schedule. They think they are about two weeks behind schedule, but it's hard to know because nothing is being tracked. They need help, and they need it fast! You have heard from other product managers that Agile practices have helped their projects in the past. You only want to implement practices that follow the Agile Manifesto. Which of the following practices would you implement?

  (A) As tasks are started, they are displayed to the entire team in a table

  (B) The client cannot add new features after the initial planning is complete

  (C) The development plan is re-evaluated at regular time intervals

  (D) The software product manager acts as a messenger between the client and the development team

AGILE: CASE - I

- Agile: Case 1 Options:
  - (A) As tasks are started, they are displayed to the entire team in a table
  - (B) The client cannot add new features after the initial planning is complete
  - (C) The development plan is re-evaluated at regular time intervals
  - (D) The software product manager acts as a messenger between the client and the development team
- Note: If you're wanting to implement agile practices, having a development plan that is reevaluated at regular time intervals is a good place to start. This follows the agile value of adapting to change, therefore, answer C is the correct answer. A is not correct, since in agile, you want to track progress by working software or tasks completed, not tasks started. Answer B does not allow client collaboration or adapting to change. And D is incorrect, since you want your development team and client to have constant open communication.

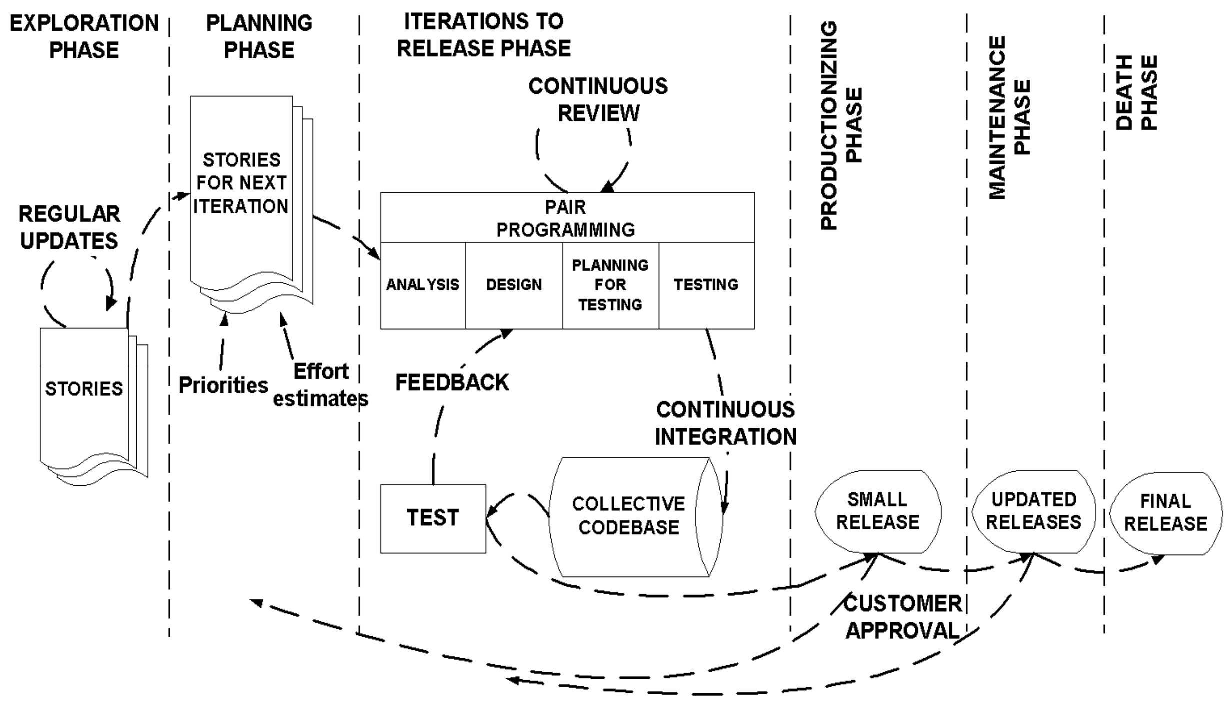
EXTREME PROGRAMMING (XP)

- The software development methodology (practices) are called extreme programming because of being extreme of adapting the agile manifesto in the development process.
- Evolved from the problems caused by the long development cycles of traditional development models (Beck 1999a).
- First started as 'simply an opportunity to get the job done' (Haungs 2001) with practices that had been found effective in software development processes during the preceding decades (Beck 1999b)
- Method is formed around common-sense principles and simple to understand practices
  - No process fits every project, rather, simple practices should be tailored to suit an individual project

XP PROCESS

- The life cycle of XP consists of five phases:

1. Exploration
2. Planning
3. Iterations to Release
4. Productionizing
5. Maintenance and Death



XP PROCESS – EXPLORATION PHASE

- The customers write out the story cards that they wish to be included in the first release
- At the same time the project team familiarize themselves with the tools, technology and practices they will be using in the project
- The exploration phase takes between a few weeks to a few months, depending largely on how familiar the technology is to the programmers

XP PROCESS

❑ Planning phase

- User's stories are written and Estimate the effort of working with the user stories
- Priorities are given to the user stories to be implemented
- Release planning creates the release schedule

❑ Iterations to release phase

- Includes several iterations of the systems before the first release
- Each take one to four weeks to implement
- The first iteration creates a system with the architecture of the whole system. This is achieved by selecting the stories that will enforce building the structure for the whole system
- The customer decides the stories to be selected for each iteration
- At the end of the last iteration the system is ready for production

XP PROCESS

❑ Productionizing phase

- Requires extra testing and checking of the performance of the system before the system can be released to the customer
- New changes may still be found, and the decision has to be made if they are included in the current release
- The iterations may need to be quickened from three weeks to one week
- The postponed ideas and suggestions are documented for later implementation

❑ Maintenance phase

- After the first release is productionized for customer use, the XP project must both keep the system in the production running while also producing new iterations
- Requires an effort also for customer support tasks
- Development velocity may decelerate after the system is in production
- May require incorporating new people into the team and changing the team structure

XP PROCESS – DEATH PHASE

- When the customer does no longer have any stories to be implemented
- System satisfies customer needs also in other respects (e.g., concerning performance and reliability)
- Necessary documentation of the system is finally written as no more changes to the architecture, design or code are made
- Death may also occur if the system is not delivering the desired outcomes, or if it becomes too expensive for further development

XP - ROLES AND RESPONSIBILITY

- **Customer:** writes the stories and functional tests, and decides when each requirement is satisfied. The customer sets the implementation priority for the requirements
- **Programmer:** keeps the program code as simple and definite as possible
- **Tester:** helps the customer write functional tests, also run functional tests regularly, broadcast test results and maintain testing tools

XP - ROLES AND RESPONSIBILITY

- Tracker: gives feedback in XP. He traces the estimates made by the team (e.g., effort estimates) and gives feedback on how accurate they are in order to improve future estimations. He also traces the progress of each iteration and evaluates whether the goal is reachable within the given resource and time constraints or if any changes are needed in the process (i.e., track the progress of the project)
- Coach: Coach is the person responsible for the process as a whole. A sound understanding of XP is important in this role enabling the coach to guide the other team members in following the process (e.g., Trainer in cricket team)
- Consultant: Consultant is an external member possessing the specific technical knowledge needed (i.e., banking domain expert, semantic web, blockchain technology)
- Manager: Manager makes the decisions (time, cost, resource, schedule, risk management)

XP VALUES/PRACTICES/PRINCIPLES

1. Planning:
   - The client and development team work together in planning the product. This entails a larger planning session at project initiation and smaller sessions at each iteration.
   - The larger session determines the product's required features, their priorities, and when they should be released over time. A smaller session focuses on the features to be completed in an iteration and determines the development tasks needed.
2. Simplicity:
   - Design the simplest product that meets the customer's needs. An important aspect of the value is to only design and code what is in the current requirements rather than to anticipate and plan for unstated requirements.
   - So, it would be wasteful in making elaborate designs for something that will change

XP VALUES/PRACTICES/PRINCIPLES

3. Small/short releases (Communication and Feedback):

- Releases must occur as frequent as possible to gain plenty of feedback
- A simple system is "productionized“ rapidly – at least once in every 2 to 3 months. New versions are then released even daily, but at least monthly.
- Smaller releases also allow estimates to be more certain; it is easier to look a week or two ahead, rather than months
- XP has a culture of oral communication and its practices are designed to encourage communication and feedback through short interaction.

“Problems with projects can invariably be traced back to somebody not talking to somebody else about something important.” 2. Small/short releases:

- The development team obtains feedback from the customers at the end of each iteration and external release. This feedback drives the next iteration.

XP VALUES/PRACTICES/PRINCIPLES

4. Metaphor:
   - The system is defined by a metaphors between the customer and the programmers. This "shared story" guides all development by describing easily how the system works.
   - For example, is the shopping cart metaphor in online shopping, which builds upon a concept from real-world shopping.
5. Refactoring:
   - Refactoring is a practice to restructure the internal design of the code without changing its external behavior. It gradually reorganize the code to allow easier extension.
   - Restructuring the system by removing duplication, improving communication, simplifying and adding new feature flexibility.
   - Reworks code that is difficult to understand, because of bug fixes or adaption to requirements changes, into easily understood code with high maintainability.

XP VALUES/PRACTICES/PRINCIPLES

6. Continuous Testing:

- In XP, tests are prepared for a required feature or functionality before its corresponding source code is written known as Test-Driven Development or TDD.
- This means writing tests before you actually need them. Instead of building your product and then testing, where developers write tests around existing code, you want to focus the actual problem that the code is meant to solve. Basically, building your tests first means you're actually testing if your code solves your problem. And not only if your code works (in improve the quality of the software product)
- Automating the tests will allow them to be executed continuously
- There are two main types of tests involved, an *acceptance test* is for the client to verify whether a product feature or requirement works as specified and thus acceptable. A *unit test* is written and run by developers to verify whether low-level functionality (algorithm, data structure) works correctly in some module of the software implementation.

XP VALUES/PRACTICES/PRINCIPLES

7. Pair programming

- Two people write the code at one computer
- One programmer, the driver, has control of the keyboard/mouse and actively implements the program. The other programmer, the observer, continuously observes the work of the driver to identify tactical defects (syntactic, spelling, etc.) and also thinks strategically about the direction of the work.
- Two programmers can brainstorm any challenging problem. Because they periodically switch roles.

XP VALUES/PRACTICES/PRINCIPLES

8. Collective Code Ownership:
   - Although a specific pair of developers might initially implement a particular piece of the product, they do not "own" it. To encourage contributions, other team members can add to it or any other part of the product. Thus, the produced work is the result of the team, not individuals. So, project success and failure resides with the team, not on specific developers.
   - Anyone can add and change any part of the code at any time
9. Continuous Integration:
   - In XP, developers combine their code often to catch integration issues early (daily build)
   - This should be at least once daily, but it can be much more frequent. With tests written first, the tests can also be run frequently to highlight pending work or unforeseen issues.

XP VALUES/PRACTICES/PRINCIPLES

10. 40-Hour Work Week:
    - XP aims to be programmer-friendly. It respects the developer's balance of work and life.
    - At critical situation time, up to one week of overtime is allowed, but multiple weeks of overtime would be a sign of poor management or estimation.
11. On-Site Customer:
    - In XP, the client is situated near the development team throughout the project to clarify and answer questions.
12. Coding Standard:
    - All the developers agree to and follow a coding standard that specifies conventions on code style, formatting, and usage. This makes the code easier to read, and it encourages the practice of *collective ownership* (e.g., meaningful variable names, exception handling)
13. Open workspace & visible Wall graph: A large room without any cubicles is preferred

XP VALUES/PRACTICES/PRINCIPLES

14. Courage and Respect :

- Allow the team to have courage in its actions and decision making. For example, the development team might have the courage to resist pressure to make unrealistic commitments.
- Team members need to care about each other and about the project.
- Quote from Extreme Programming website which will surely improve the moral of the developer and other team member in the project teams.

  " We will tell the truth about progress and estimates. We don't document excuses for failure because we plan to succeed. We don't fear anything because no one ever works alone. We will adapt to changes whenever they happen "

15. Just rules: Team has its own rules that are followed but can also be changed at any time. The changes have to be agreed upon and their impact has to be assessed

XP - ARTEFACTS

- User story cards
  - Paper index cards which contain brief requirement (features, non-functional) descriptions
  - Not a full requirement statement
  - Commitment for further conversation between the developer and the customer
  - During this conversation, the two parties will come to an oral understanding of what is needed for the requirement to be fulfilled
  - Customer priority and developer resource estimate are added to the card
  - The resource estimate for a user story must not exceed the iteration duration

XP - ARTEFACTS

- CRC (Class-Responsibility-Collaboration) cards
  - Paper index card on which one records the responsibilities and collaborators of classes which can serve as a basis for software design
  - The classes, responsibilities, and collaborators are identified during a design brainstorming/role-playing session involving multiple developers
- Task list
  - A listing of the tasks (one-half to three days in duration) for the user stories that are to be completed for an iteration
  - Tasks represent concrete aspects of a user story
  - Programmers volunteer for tasks rather than are assigned to tasks

XP - ARTEFACTS

- Customer Acceptance Tests Case
  - Textual descriptions and automated test cases which are developed by the customer
  - The development team demonstrates the completion of a user story and the validation of customer requirements by passing these test cases.

- Visible Wall Graphs
  - To foster communication and accountability, progress graphs are usually posted in team work area. These progress graphs often involve how many stories are completed and/or how many acceptance test cases are passing.

HOW XP SOLVES SOME SE PROBLEMS

Problem	**Solution**
Slipped schedule	Short development cycles
Cost of changes	Extensive, ongoing testing, system always running
Defect rates	Unit tests, customer tests
Misunderstand the business	Customer is a part of the team
Business changes	Changes are welcome
Staff turnover (replacement)	Intensive teamwork

XP - SCOPE AND USE & LIMITATION

- XP is aimed at small to medium-sized teams because of collective ownership
- Not suitable for any safety critical product development
- Requires the development team to be in one place. Scattering of programmers on two floors or even on one floor is intolerable for XP
- Require experienced team member through the project development time (XP is people based rather than plan based)
- Communication and coordination between project members should always be enabled
- It is not often possible to the full-time client availability (24/7) whenever the developer needs them for feedback

XP: CASE - 1

- Erica is working as a software product manager for a development team that is creating a social media application. Her team spends two months completely planning out every detail of the project, including the design and schedules. The duration of their iteration is hours, not weeks. Her team has depicted their system metaphor on a whiteboard in the office. Which of their practices are considered extreme?

  (A) Two-month planning period

  (B) Iteration duration in hours

  (C) System metaphor on a whiteboard

XP: CASE – I (SOLUTION)

❑ XP: Case I Options:

(A) Two-month planning period

(B) Iteration duration in hours

(C) System metaphor on a whiteboard

❑ Note: In extreme programming, planning should be simple, adaptive and not a two-month, upfront ordeal. Therefore, A is not the correct answer. You want iterations to be as small as possible while still delivering a working product. If the size of the iteration is so small that its duration happens in a matter of hours, then that's pretty extreme. Having a system metaphor is also an extreme programming practice. Therefore, B and C are the correct answers.

XP: CASE - 2

- You're a software product manager working with the development team who's producing an app for a cinema that allows users to check show times and buy tickets. Which of the following, if any, would be acceptance tests?
  - (A) a test that confirms when a user buys a ticket, a valid ticket appears.
  - (B) a test that confirms when a user creates an account, their information is added to the database.
  - (C) a test that confirms when the user logs in, that their log-in information matches an account in the database.
  - (D) a test that confirms when a user selects a show time, movies playing at that time appear.

XP: CASE - 2

- XP: Case 2 Options:
  - (A) a test that confirms when a user buys a ticket, a valid ticket appears.
  - (B) a test that confirms when a user creates an account, their information is added to the database.
  - (C) a test that confirms when the user logs in, that their log-in information matches an account in the database.
  - (D) a test that confirms when a user selects a show time, movies playing at that time appear.
- Note: A & D are referring to acceptance testing while B & C are referring to Unit testing

XP: CASE - 3

- After delivering a prototype, the client sends out a nasty email to the development team, saying that they discovered a bug in the application that calculates the wages incorrectly. The client is very upset. This bug would cost the business owners a great deal of money, and employees would end up getting more money than they have earned. Timmy and Maria were the programmers who co-wrote the unit tests for this feature. Steven and Maria were the programmers who pair programmed the source code. Danielle, the product manager, acceptance tested the product before the release. Who is to blame for this error? Choose all that apply.

  (A) Danielle (B) Timmy (C) Maria (D) Steven

  (E) The other two developers on the team

XP: CASE – 3 (SOLUTION)

- XP: Case 3 Options:

  (A) Danielle (B) Timmy (C) Maria (D) Steven

  (E) The other two developers on the team

- Note: A practice of extreme programming is collective ownership. This means that all members of the team are responsible for errors that occur. Therefore, everyone is at fault.

XP: CASE - 4

- You are a software product manager for a small startup that is working on creating an application that matches parents with local babysitters and nannies. You and your development team decided early on that the team will adopt the Extreme Programming methodology. Your development team is organized into pairs, with each pair working in front of a single workstation. This is a practice of Extreme Programming that we will talk about later. Which of the five aspects of Extreme Programming do you think this improves?

  (A) Communication (B) Simplicity (C) Feedback (D) Respect

XP: CASE – 4 (SOLUTION)

- XP: Case 4 Options:

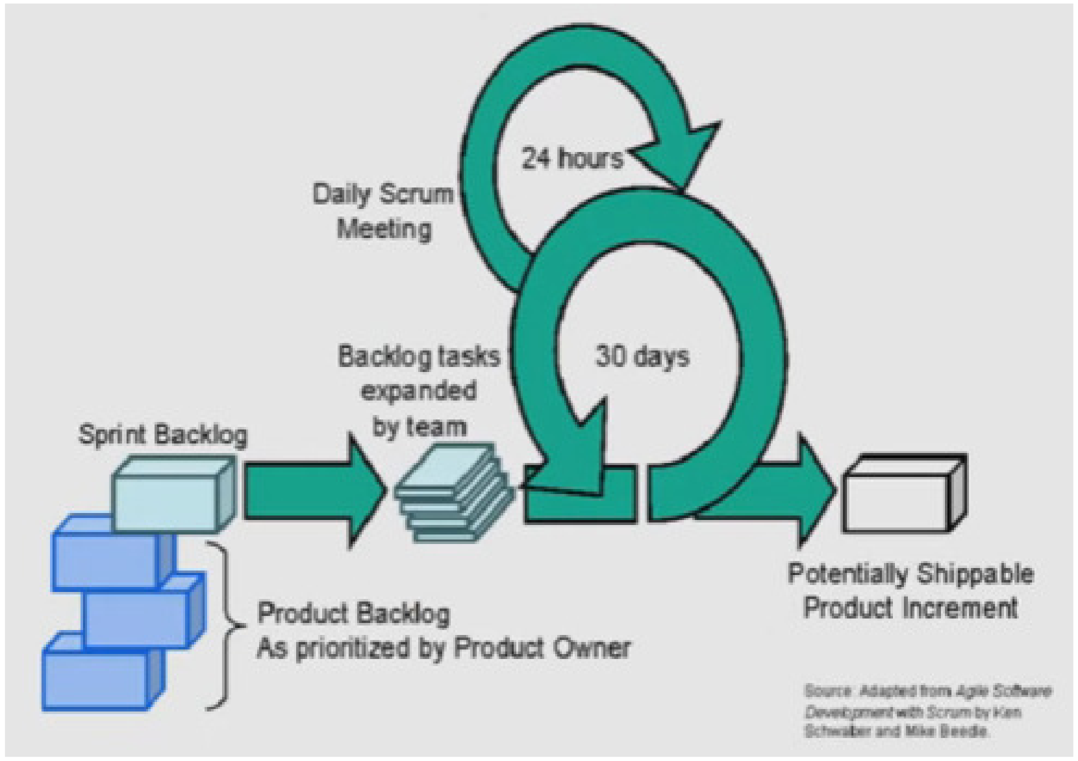
  (A) Communication (B) Simplicity (C) Feedback (D) Respect

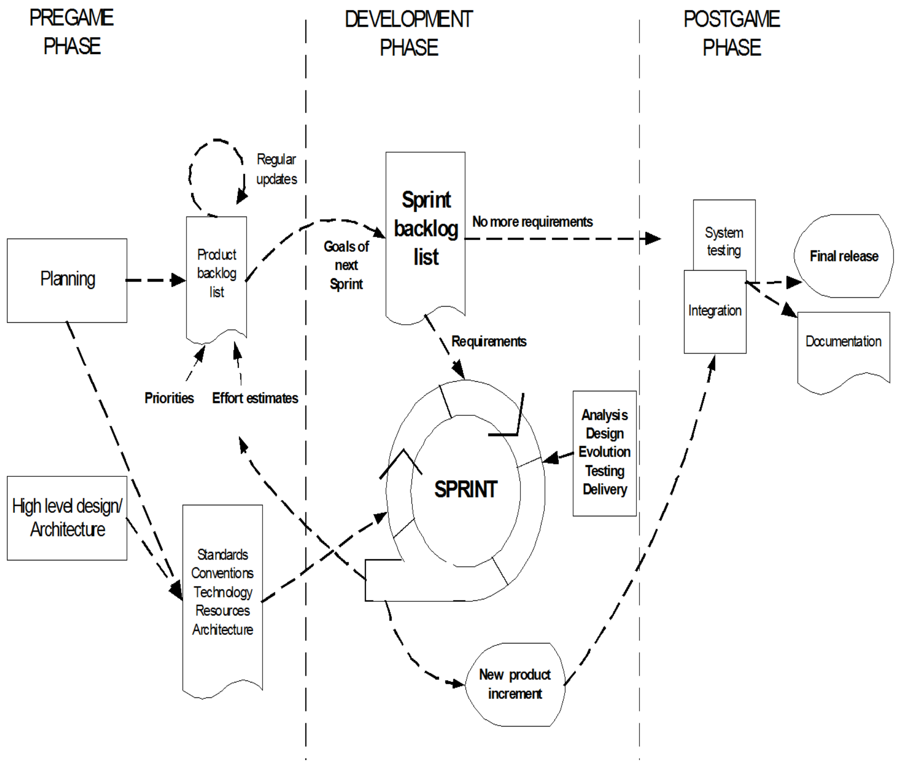
- Note: this practice, called pair programming, involves all aspects of development. Pair programming encourages the two programmers to communicate and work collaboratively through problems. Having two perspectives results in simpler and higher quality code. You also get immediate feedback since you are working directly with someone. It also encourages respect since you can use the complimentary qualities and skills of the programmers to solve the problem. Finally, it improves courage since the programmer does not work alone. Programmers working together are more likely to take a calculated risk than they would alone.

SCRUM

- The first references in the literature to the term 'Scrum‘ point to the article of Takeuchi and Nonaka (1986) in which an adaptive, quick, self-organizing product development process originating from Japan is presented (Schwaber and Beedle 2002).
- The term 'scrum' originally derives from a strategy in the game of rugby where it denotes "getting an out-of play ball back into the game" with teamwork (Schwaber and Beedle 2002).
- Scrum is used by many major tech companies like Adobe, Amazon, Microsoft, and Yahoo.
- SCRUM process includes three phases
  - Pre-game
  - Development (game phase)
  - Post-game

SCRUM PROCESS


24 hours
Daily Scrum Meeting
Backlog tasks expanded by team
30 days
Sprint Backlog
Product Backlog As prioritized by Product Owner
Potentially Shippable Product Increment
Source: Adapted from Agile Software Development with Scrum by Ken Schwaber and Mike Beedle.



PRE-GAME PHASE

- The pre-game phase includes two sub-phases:

1. **Planning:**
   - Definition of the system being developed
   - A Product Backlog list is created containing all the requirements that are currently known
   - The requirements are prioritized, and the effort needed for their implementation is estimated
   - The product Backlog list is constantly updated with new and more detailed items, as well as with more accurate estimations and new priority orders
   - Planning also includes the definition of the project team, tools and other resources, risk assessment and controlling issues, training needs and verification management approval

PRE-GAME PHASE

2. Architecture

- The high-level design of the system including the architecture is planned based on the current items in the Product Backlog
- In case of an enhancement to an existing system, the changes needed for implementing the Backlog items are identified along with the problems they may cause
- A design review meeting is held to go over the proposals for the implementation and decisions are made on the basis of this review

DEVELOPMENT (GAME) PHASE

- This phase is treated as a "black box" where the unpredictable is expected
- The system is developed in Sprints
  - Sprints are iterative cycles where the functionality is developed or enhanced to produce new increments.
  - Each Sprint includes the traditional phases of software development: requirements, analysis, design, evolution and delivery phases.
  - One Sprint is planned to last from one week to one month.

POST-GAME PHASE

- This phase is entered when an agreement has been made such as the requirements are completed.
- In this case, no more items and issues can be found, nor can any new ones be invented.
- The system is now ready for the release and the preparation for this is done during the post-game phase, including the tasks such as the integration, system testing and documentation.

ROLES AND RESPONSIBILITIES

Scrum Master

- Scrum Master is responsible for ensuring that the project is carried through according to the practices, values, and rules of Scrum and that it progresses as planned.
- Scrum Master interacts with the project team as well as with the customer and the management during the project.
- They have specific duties to both the Product Owner and the Development Team
- The Scrum Master's responsibilities to the Product Owner include, finding techniques to manage the backlog. Helping the scrum team to understand the need for a clear and concise backlog. Ensuring the product owner knows how to prioritize the backlog to get maximum value. And facilitating scrum events.
- The Scrum Master's duties to the development team include coaching the team to self-organize. Removing development roadblocks. And facilitating scrum events (Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective)

ROLES AND RESPONSIBILITIES

❑ Scrum Team

- Scrum development teams are small. Ideally, they are larger than THREE and less than NINE. The Scrum Master and Product Owner are not included in that number, unless they are contributing to development.
- Scrum Team is the project team that has the authority to decide on the necessary actions and to organize itself in order to achieve the goals of each Sprint.
- The scrum team is involved, for example, in effort estimation, creating the Sprint Backlog, reviewing the product Backlog list and suggesting impediments that need to be removed from the project.
- There are also no sub-teams in the Scrum development team.
- Developers may have a specialized skill within the team. However, accountability belongs to the entire team

ROLES AND RESPONSIBILITIES

- Product Owner
  - Product Owner is officially responsible for the project, managing, controlling, and making visible the Product Backlog list.
  - Product owner is responsible for defining and prioritizing requirements for the product backlog.
  - He is selected by the Scrum Master, the customer, and the management.
  - He makes the final decisions of the tasks related to product Backlog.
  - Only the product owner can make changes to the requirements on the product backlog.

ROLES AND RESPONSIBILITIES

- Customer
  - Customer participates in the tasks related to product Backlog items for the system being developed or enhanced.
- Management
  - Management is in charge of final decision making, along with the agreements, standards, and conventions to be followed in the project.
  - Management also participates in the setting of goals and requirements.

SCRUM PRACTICES

❑ Product Backlog

- Defines the work to be done in the project A prioritized and constantly updated list of business and technical requirements for the system being built or enhanced
- Include features, functions, defects, bug fixes, requested enhancements and technology upgrades
- Suggestions from the spring review that will change the Sprint goal will go in the Backlog and can be implemented in the next Sprint if that's what the Product Owner decides.

❑ Effort Estimation

- Effort estimation is an iterative process, in which the Backlog item estimates are focused on a more accurate level when more information is available on a certain Product Backlog item.
- The Product Owner together with the Scrum Team(s) are responsible for performing the effort estimation.

SCRUM PRACTICES

❑ Sprint Backlog

- Sprint Backlog is the starting point for each Sprint. It is a list of Product Backlog items selected to be implemented in the next Sprint.
- The items are selected by the Scrum Team together with the Scrum Master and the Product Owner in the Sprint Planning meeting, on the basis of the prioritized items and goals set for the Sprint.
- Unlike the Product Backlog, the Sprint Backlog is stable until the Sprint (i.e., 30 days) is completed. When all the items in the Sprint Backlog are completed, a new iteration of the system is delivered.

SCRUM PRACTICES

- Sprint
  - A Sprint is a development phase of a specific amount of time, in which a working prototype is delivered to the client. No changes can be made during sprint process time period
  - The Sprint goal is the big picture view of what is going to get done that Sprint.
  - The working tools of the team are the four SCRUM events (Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective).
  - At the end of each Sprint, the client is delivered a working prototype.
  - These Sprints are also time boxed, but more strictly enforced. A Sprint lasts one month or less, typically 1 or 2 weeks. Once your Scrum Team specifies the length of your Sprints, they must consistently stay that length for the entire development. Sprints cannot be made longer or shorter once decided.

SCRUM PRACTICES

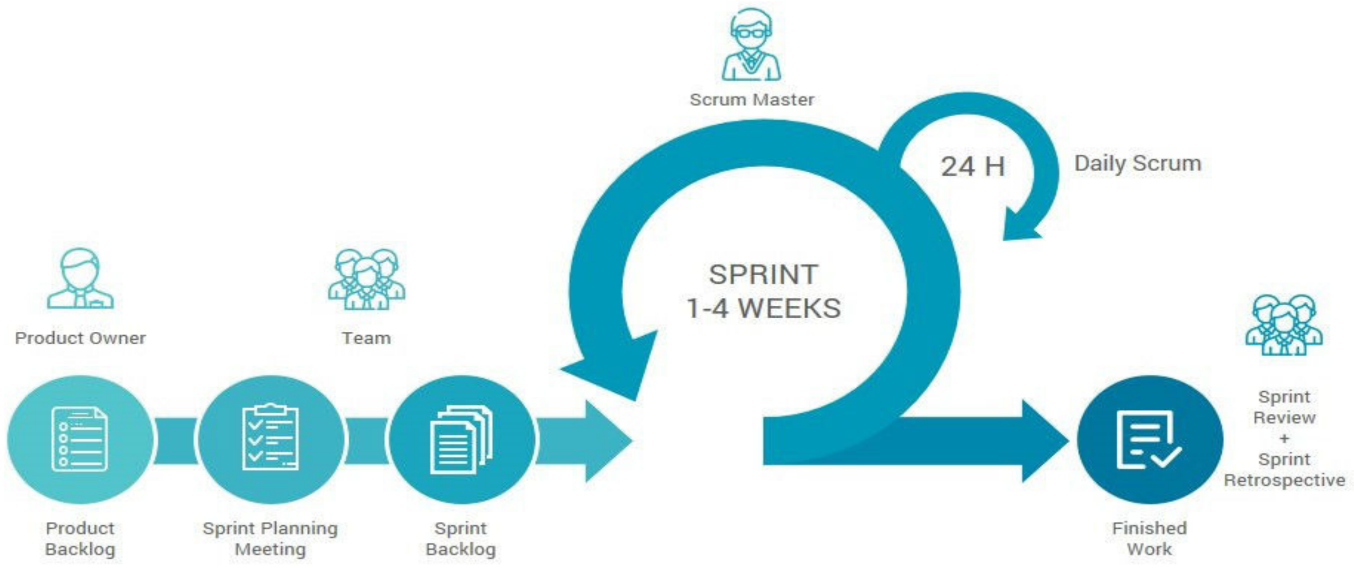
- Sprint Planning meeting
  - A Sprint Planning Meeting is a two-phase meeting organized by the Scrum Master.
  - First Phase: The Scrum Master, Management, Product Owner, and Scrum Team participate in the first phase of the meeting to decide upon the goals and the functionality of the next Sprint.
  - Second Phase: The second phase of the meeting is held by the Scrum Master and the Scrum Team focusing on how the product increment is implemented during the Sprint.

SCRUM PRACTICES

- Daily Scrum meeting
  - Daily Scrum meetings are organized to keep track of the progress of the Scrum Team continuously and they also serve as planning meetings: what has been done since the last meeting and what is to be done before the next one.
  - Also, problems and other variable matters are discussed and controlled in this short (approximately 15 minutes) meeting held daily. Any deficiencies or impediments in the systems development process or engineering practices are looked for, identified and removed to improve the process. The Scrum Master conducts the Scrum meetings. Besides the Scrum team also the management, for example, can participate in the meeting.

SCRUM PRACTICES

- Sprint Review meeting
  - On the last day of the Sprint, the Scrum Team and the Scrum Master present the results (i.e., working product increment) of the Sprint to the management, customers, users, and the Product Owner in an informal meeting for their feedback.
  - The participants assess the product increment and make the decision about the following activities.
  - The review meeting may bring out new Backlog items and even change the direction of the system being built.



SCRUM CASE - I

- You have been hired by a company that say they have tried to implement the Extreme Programming methodology exactly. You walk into the workplace and see a maze of cubicles. As you walk around the workspace and peek into cubicles, you see pairs of programmers working together. You pop into a cubicle and ask the developers where you can find the client. One developer hand you a business card that has the client's phone number on it. He said you can leave a message there and the client will get back to you. You go into the next cubicle and ask the two programmers working there when the next release is due. They tell you that releases are due every 2nd Friday. They have a release coming up this Friday. You ask these programmers what they are working on. They explain that they are the database team, and they exclusively work on maintaining and building the database. You realize that there are many areas of Extreme Programming that they are not following properly. What are some areas of development you are going to need to change in order to have Extreme Programming running precisely in this office?

(A) The workspace (B) Pair Programming (C) Client Availability

(D) Small Frequent Releases (E) Developer Versatility

SCRUM CASE – I (SOLUTION)

- Case I Options:

  (A) The workspace (B) Pair Programming (C) Client Availability

  (D) Small Frequent Releases (E) Developer Versatility

- Note: An Extreme Programming workspace environment is open and encourages collaboration. A maze of cubicles does not align with that practice. The client should always also be available . A hard-to-reach client does not encourage effective communication. Finally, developers should move around and not specialize in a feature. Having two developers exclusively working on the database does not comply with extreme programming. Therefore A, C, and E are the correct answers.

SCRUM CASE - 2

- You are the software product manager who has been hired to work on a running app for a major athletic apparel company. In your first meeting with your development team and Morgan, the representative from the company, the development team decided to use Scrum. Penny was appointed as the product owner. Your team is halfway into development, and you get an email from the CEO of the company. He is requesting pre made running playlists users can listen to when they run. You know that your development team had already talked to Penny about this feature and they decided as a team that they wouldn't implement this feature in favor of another one. Following the practices of Scrum, what should you do?

  (A) Send him an email back saying "no, you get no say in this project
  (B) Tell your development team to add the feature to the backlog
  (C) Tell your development team to develop the feature for this iteration
  (D) Send him back an email asking for all feature requests to come through Morgan

SCRUM CASE - 2 (SOLUTION)

- CASE – 2 Options:

  (A) Send him an email back saying "no, you get no say in this project
  (B) Tell your development team to add the feature to the backlog
  (C) Tell your development team to develop the feature for this iteration
  (D) Send him back an email asking for all feature requests to come through Morgan

- Note: In Scrum, all decisions about the product must come from the Product Owner. Does that mean that you reject the CEO of the company? No, that's just bad business sense. Your best bet is to send a polite email to the CEO with a copy to the Product Owner asking that all feature requests come through the Product Owner (from a single source of contact)

SCRUM CASE - 3

- You are at a conference on Scrum Practices. You have been talking to several other software product managers about their development teams. The first SPM you talk to is named Kelsey. She describes her team as having 4 programmers, 2 testers, and 1 user interface designer. Next you meet Jim. He says his current team consists of 20 developers. Next, you meet Billy. He simply says he has a very talented team of 7 developers. Then you run into an old friend, Carry. She says her team has 9 developers. 5 are on the development team and 4 are on the testing team. Finally, you meet a new software product manager, Hillary. She says her team consists of just 3 developers and they subcontract out their testing. Which of these software product managers, if any, have a true scrum development team?

(A) Kelsy (B) Jim (C) Billy (D) Carry (E) Hillary

SCRUM CASE - 3 (SOLUTION)

- Case 3 Options:

  (A) Kelsy (B) Jim (C) Billy (D) Carry (E) Hillary

- Note: These are all examples of development teams that you will commonly see. However, only Billy has a true Scrum development team. There are no titles or sub-teams. And the team is cross functional and an appropriate size. Kelsey's team had titles. Jim's team was too large. Carrie's team had sub-teams. And Hilary's team is not cross functional because they subcontract out the testing. Therefore, C is the only correct answer.

SCRUM CASE - 4

- Let's say your team is developing a mobile app that allows users to search for food and drink specials at local restaurants. The 'sprint goal' for this iteration is to set up the user interface. This is everything that the end-user will see and interact with. Halfway through development for the iteration, the client sends you an email that they've changed their mind and they want the main colors of the application to be blue and white instead of red and black. This is a quick fix for the development team. The client also requests that they want the application to now support ads that popup on the screen. Which of these changes can be made in the current iteration?

  (A) The color change (B) Popup ads

  (C) Both of them (D) Both of them

SCRUM CASE – 4 (SOLUTION)

- Case 4 Options:

  (A) The color change (B) Popup ads

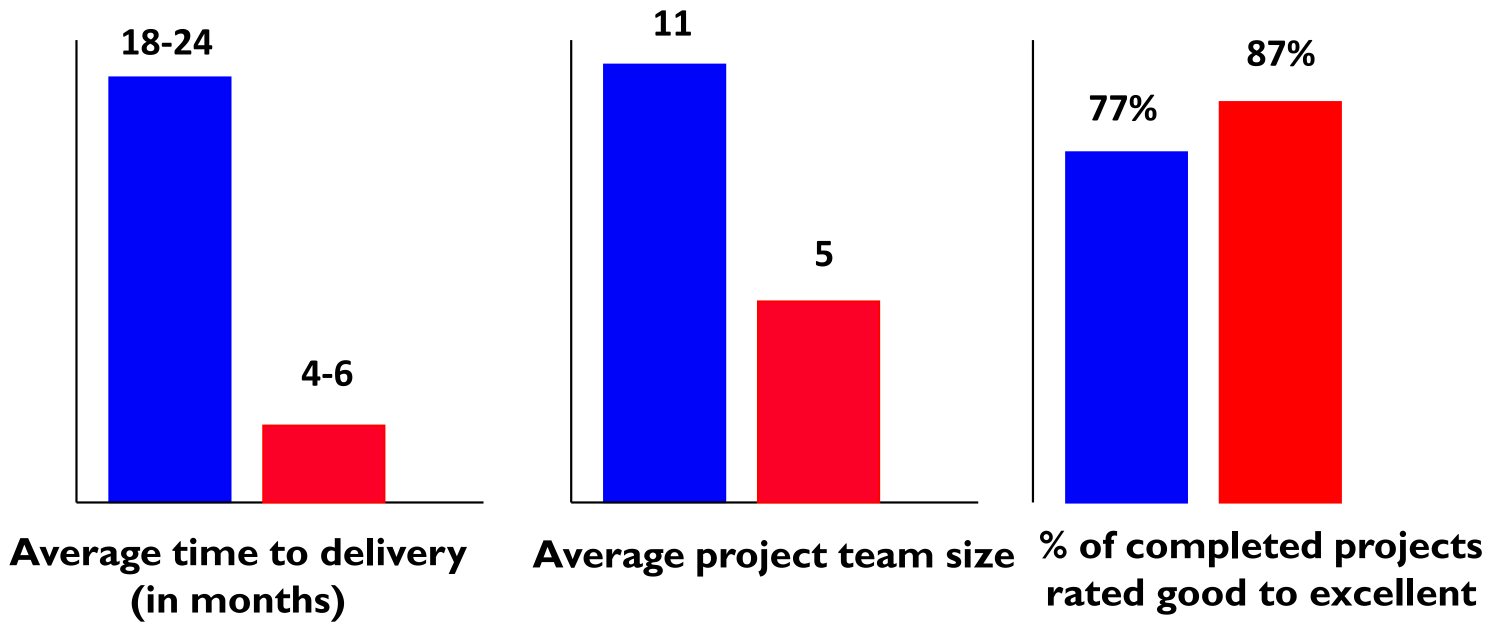
  (C) Both of them (D) Both of them

- Note: Changing the color of the application is a minor change that doesn't affect the sprint goal. So, that change could be implemented mid-sprint. Adding popup ads will take more development and does not fit with the sprint goal. This should be added to the product backlog and revisited at a future sprint. Therefore, A is the correct answer.

DSDM

- The Dynamic Systems Development Method (DSDM) is a public domain Rapid Application Development method which has been developed through capturing the experience of a large group of vendor and user organisations. It is now considered to be the UK's de-facto standard for RAD.
- The key to DSDM is to deliver what business needs when it needs
  - Achieved by using the various techniques in the framework and flexing requirements
  - The aim is always to address the current and imminent needs of the business rather than to attack all the perceived possibilities

TRADITIONAL METHOD VS. DSDM

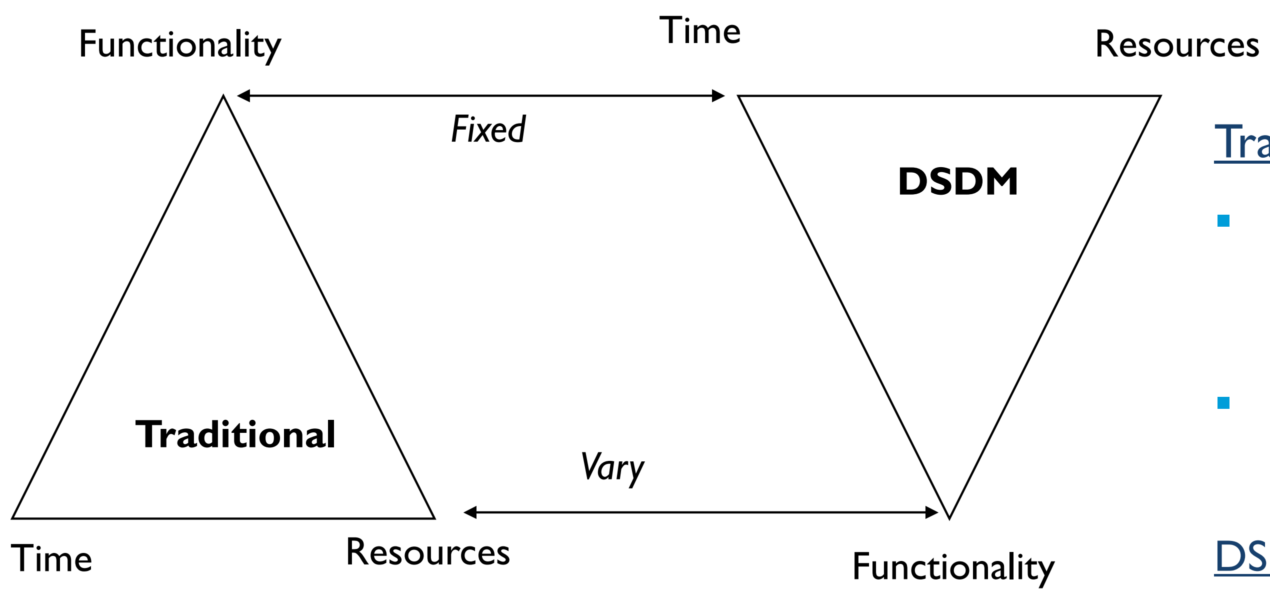


🟦 Using traditional approaches

🟥 Using DSDM

Source: British Airways IM Department, Newcastle

DIFFERENCE BETWEEN TRADITIONAL DEVELOPMENT VS. DSDM



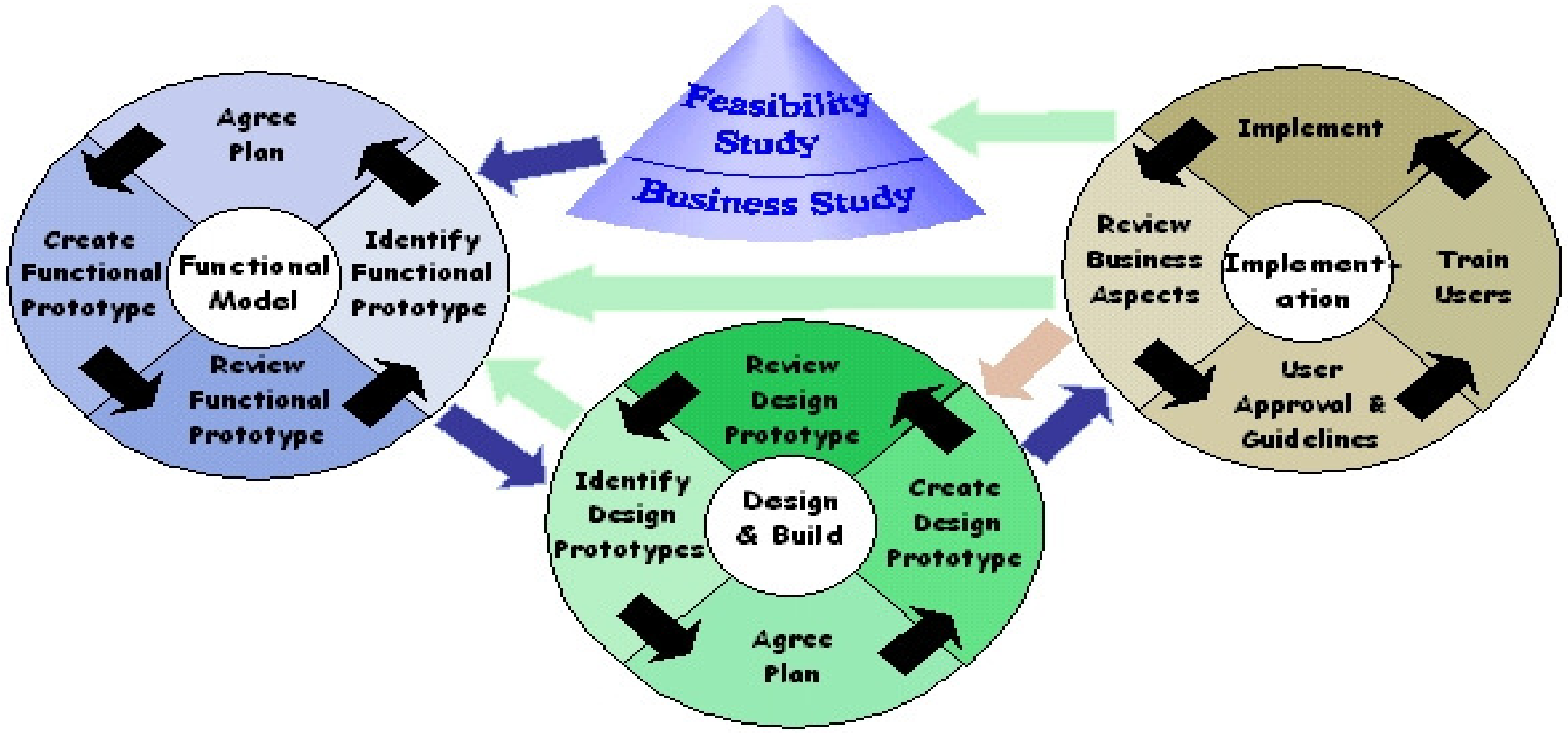
<u>Traditional Method</u>

- Functional/requirements are fixed (i.e., need well understood requirements)
- Time & resources can varies

<u>DSDM/Agile Methods</u>

- Functional/requirement varies
- Time & resources are fixed

DSDM PROCESS VIEW



DSDM PROCESS

Activity	Sub activity	Description
Study	**Feasibility Study**	Stage where the suitability of DSDM is assessed. Judging by the type of project, organizational and people issues, the decision is made, whether to use DSDM or not. Therefore, it will generate a FEASIBILITY REPORT, a FEASIBILITY PROTOTYPE, and a GLOBAL OUTLINE PLAN which includes a DEVELOPMENT PLAN and a RISK LOG.
	Business Study	Stage where the essential characteristics of business and technology are analyzed. Approach to organize workshops, where a sufficient number of the customer's experts are gathered to be able to consider all relevant facts of the system, and to be able to agree on development priorities. In this stage, a PRIORITIZED REQUIREMENTS LIST, a BUSINESS AREA DEFINITION, a SYSTEM ARCHITECTURE DEFINITION, and an OUTLINE PROTOTYPING PLAN are developed.

DSDM PROCESS

Activity	Sub activity	Description
Functional Model Iteration	Identify functional prototype	Determine the functionalities to be implemented in the prototype that results from this iteration. In this sub-stage, a FUNCTIONAL MODEL is developed according to the deliverables result of business study stage.
	Agree schedule	Agree on how and when to develop these functionalities.
	Create functional prototype	Develop the FUNCTIONAL PROTOTYPE, according to the agreed schedule and FUNCTIONAL MODEL.
	Review functional prototype	Check correctness of the developed prototype. This can be done via testing by end-user and/or reviewing documentation. The deliverable is a FUNCTIONAL PROTOTYPING REVIEW DOCUMENT.

DSDM PROCESS

Activity	Sub activity	Description
Design and Build Iteration	Identify design prototype	Identify functional and **non-functiona**l requirements that need to be in the tested system. And based on these identifications, an IMPLEMENTATION STRATEGY is involved. If there is a TEST RECORD from the previous iteration, then it will be also used to determine the IMPLEMENTATION STRATEGY.
	Agree schedule	Agree on how and when to realize these requirements.
	Create design prototype	Create a system (DESIGN PROTOTYPE) that can safely be handed to end-users for daily use, also for testing purposes.
	Review design prototype	Check the correctness of the designed system. Again testing and reviewing are the main techniques used. An USER DOCUMENTATION and a TEST RECORD will be developed.

DSDM PROCESS

Activity	Sub activity	Description
Implementation	User approval and guidelines	End users approve the tested system (APPROVAL) for implementation and guidelines with respect to the implementation and use of the system are created.
	Train users	Train future end user in the use of the system. TRAINED USER POPULATION is the deliverable of this sub-stage.
	Implement	Implement the tested system at the location of the end users, called as DELIVERED SYSTEM.
	Review business	Review the impact of the implemented system on the business, a central issue will be whether the system meets the goals set at the beginning of the project. Depending on this the project goes to the next stage, the post-project or loops back to one of the preceding stages for further development. This review is will be documented in a PROJECT REVIEW DOCUMENT.

DSDM TECHNIQUES: FLEXIBILITY

- A fundamental assumption of DSDM is that nothing is built perfectly first time
- 80:20 Rule: assumes that a usable and useful 80% of the proposed system can be produced in 20% of the time it would take to produce the total system.
- In "traditional" development practice, a lot of time is spent in getting from the 80% solution to the total solution, with the assumption that no step ever needs to be revisited. The result is either projects that are delivered late and over budget or projects that fail to meet the business needs since time is not spent reworking the requirements.
- DSDM assumes that all previous steps can be revisited as part of its iterative approach. Therefore, the current step need be completed only enough to move to the next step, since it can be finished in a later iteration.
- RAD has fixed time (90 days) to deliver the first working product whether the project duration is 6 months or 1 year where 80:20 rule gives a range according to the project duration

DSDM TECHNIQUES: TIMEBOXING

- Without effective timeboxing, prototyping teams can lose their focus and run out of control.
- Timeboxing works by concentrating on when a business objective will be met as opposed to the tasks which contribute to its delivery.
- **Timeboxing Basics**
  - Time between start and end of an activity
  - DSDM uses nested timeboxes, giving a series of fixed deadlines
  - Ideally 2 - 4 weeks in length
  - Objective is to have easiest 80% produced in each timebox
  - Remaining 20% potentially carried forward subsequent timeboxes
  - Focus on the essentials
  - Helps in estimating and providing resources

DSDM TECHNIQUES: MOSCOW RULES

MoSCoW rules formalised in DSDM version 3

IMPORTANCE

- **Must have** – fundamental to project success
- **Should have** – important but project does not rely on
- **Could have** – left out without impacting on project

URGENCEY

- **Want to have but Won't have** this time for those valuable requirements that can wait till later development takes place; in other words, the Waiting List.
- And some of the requirements have prerequisite or dependency on other requirements.

DSDM TECHNIQUES: PROTOTYPING

Prototypes are necessary in DSDM because

- Facilitated workshops define the high-level requirements and strategy
- Prototypes provide the mechanism through which users can ensure that the detail of the requirements is correct
- Demonstration of a prototype broadens the users' awareness of the possibilities and assists them in giving feedback to the developers
- Speeds up the development process and increases confidence that the right solution will be delivered

DSDM TECHNIQUES: FACILITATED WORKSHOPS

- Purpose is to produce clear outcomes that have been reached by consensus

- **Participants**
  - Workshop sponsor
  - Participants (development team)
  - Scribes and Observers (record)
  - Prototypers
  - Facilitator (help a group of people understand their common objectives and assists them to plan how to achieve these objectives)

- **Advantages of Workshops**
  - Speed
  - Involvement /ownership
  - Productivity
  - Consensus
  - Quality of decisions
  - Overall perspective / synergy (cooperation)

WHAT IS FDD?

- Original Creator: Jeff De Luca
  - Singapore in late 1997
- FDD evolved from an actual project
  - Bank Loan Automation
  - Luca was Project manager
  - 50-member developer team

- Feature Driven Development (FDD)
- FDD is an agile software development process
- FDD uses a short-iteration model
- FDD combines key advantages of other popular agile approaches along with other industry-recognized best practices
- FDD was created to easily scale to much larger projects and teams

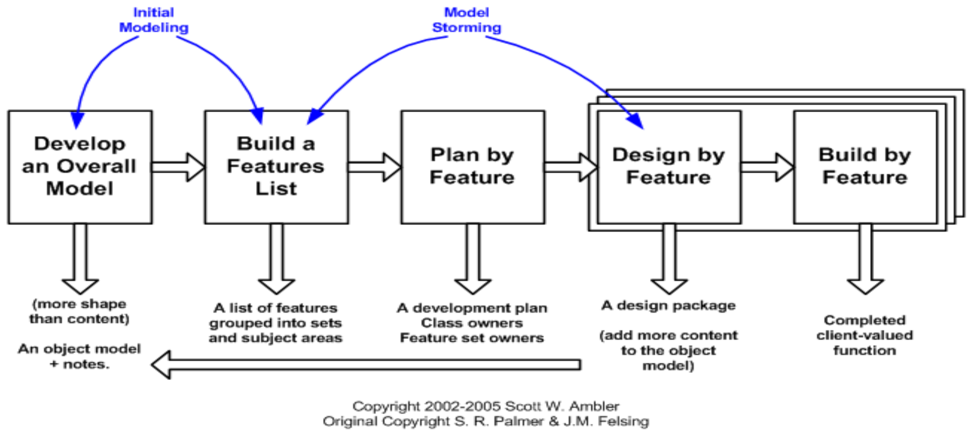
WHAT IS A FEATURE?

- FDD delivers the system feature by feature
- Feature is a small function expressed in client-valued terms which presents the customer requirements to be developed in software using small iteration
- Features are to be "small" in the sense they will take no more than two weeks to complete Features that appear to take longer are to be broken up into a set of smaller features. Two weeks is the maximum, most features take less time (1 - 5 days)
- Feature naming template:

  <action> the <result> <by|for|of|to> a(n) <object>

- Examples: Calculate the total of a sale
  Validate the password of a user
  Authorize the sales transaction of a customer

FDD PROCESS


Initial Modeling
Model Storming
Develop an Overall Model
Build a Features List
Plan by Feature
Design by Feature
Build by Feature
(more shape than content)
An object model + notes.
A list of features grouped into sets and subject areas
A development plan Class owners Feature set owners
A design package
(add more content to the object model)
Completed client-valued function
Copyright 2002-2005 Scott W. Ambler
Original Copyright S. R. Palmer & J.M. Felsing

- ❑ Project wide upfront planning and design activities:
  - Process #1: Develop an Overall Model
  - Process #2: Build a Features List
  - Process #3: Plan By Feature
  - Goal: not to design the system in its entirety but instead is to do just enough initial design that you are able to build on
- ❑ Deliver the system feature by feature:
  - Process #4: Design By Feature
  - Process #5: Build By Feature
  - Goal: Deliver real, completed, client-valued function as often as possible

FDD PROCESS

- Process #1: Develop an Overall Model
  - Form a modeling team
  - Domain walk-through
  - Build High-level object model
  - Record Notes
  - Goal - for team members to gain a good, shared understanding of the problem domain and build a foundation

- Process #2: Build a Features List
  - All Features are organized in a three-level hierarchy :

Domain Subject Area – Garments Industry
Business Activity – Operation Mgt.
Features – List of features

FDD PROCESS

❑ Process #3: Plan by Feature

- Construct initial schedule
  - Formed on level of individual features
    - Prioritize by business value
    - Also consider dependencies, difficulty, and risks
- Assign responsibilities to team members
  - Determine Class Owners
  - Assign feature sets to chief programmers

FDD PROCESS

- Process #4: Design by Feature
- Form Feature Teams
- Team members collaborate on the full low-level analysis and design
- Certain features may require teams to bring in domain experts
- Teams need to update the model artifact to support their changes

Feature Team

- Chief Programmers pick teams based on the current feature in development
- Chief Programmers lead picked team (usually 3 to 5 people)
- Upon completion of the current feature the team disbands
- Each team will concurrently work on their own independent iteration
- Possible to be on multiple teams at once

FDD PROCESS

- Process #5: Build by Feature
  - Implement designed feature
  - Test feature
    - Unit-level
    - Feature-level
  - Mandated Code Inspections (formal review with checklist)
  - Integrate with regular build

FDD PRACTICES

- Mandated Code Inspections for Two Main Reasons
  - Research has shown that when it is done properly, inspections find more bugs as well as different types of bugs than any other form of testing.
  - It is also a great learning experience
- Reporting
  - FDD emphasizes the ability to provide accurate, meaningful, and timely progress information to all stakeholders within and outside the project
  - Feature Milestones

FDD PRACTICES

- **Class ownership**
- Class (feature) assigned to specific developer
- Class owner responsible for all changes in implementing new features
- Collective Ownership
  - Any developer can modify any artifact at any time
- Advantages of Class Ownership are:
  - Someone responsible for integrity of each class
  - Each class will have an expert available
  - Class owners can make changes much quicker
  - Easily lends to notion of code ownership (XP)

FDD ROLES

- FDD Primary Roles

Project Manager	Chief Architect
Class Owners	Domain Experts
	Chief Programmers

- FDD Supporting Roles

Language Guru (shared vocabulary)	
Toolsmith (making tools for application)	
Tester	
Technical Writer (documentation)	

REFERENCES

- R.S. Pressman & Associates, Inc. (2010). *Software Engineering:A Practitioner's Approach.*
- Kelly, J. C., Sherif, J. S., & Hops, J. (1992).An analysis of defect densities found during software inspections. *Journal of Systems and Software*, *17*(2), 111-117.
- Bhandari, I., Halliday, M. J., Chaar, J., Chillarege, R., Jones, K., Atkinson, J. S., & Yonezawa, M. (1994). In-process improvement through defect data interpretation. *IBM Systems Journal*, *33*(1), 182-214.
- "Lean Software Development - MSDN - Microsoft." 2015. 21 Jun. 2016 <https://msdn.microsoft.com/en-us/library/hh533841(v=vs.120).aspx>
- The Agile Unified Process (AUP) Home Page - Scott Ambler." 2005. 21 Jun. 2016 <http://www.ambysoft.com/unifiedprocess/agileUP.html>
- "How to Manage the "7 Wastes" of Agile Software Development ..." 2013. 21 Jun. 2016 <https://www.scrumalliance.org/community/articles/2013/september/how-to-manage-the-7-wastes%E2%80%9D-of-agile-software-deve>
- "A Practical Guide to Seven Agile Methodologies, Part 1 - DevX." 2006. 21 Jun. 2016 <http://www.devx.com/architect/Article/32761>