Web Application Development

Software Development Methodologies

Software Development Stages

- Software Development follows a number of stages which need to be managed.
- There are various approaches to managing the software development process.
- The best way of managing this process is usually dependent on a number of factors:
  - Size of Software Application
  - Number of People involved
  - Culture (Organizational)
  - Tradition versus New
  - Time
  - Money

A typical Approach

- A requirement or requirements are identified for a software application. This can be a new or existing software application.
- Some kind of initial analyses is done on the requirements and these are usually documented. For example, a business systems document.
- Technical and business users/developers will get together (potentially with software vendor(s)), usually have a workshop, and discuss the requirements.
- The output of this meeting will result in an outline proposal, which will include a summary of how the requirements can be met from a technical perspective.

A typical Approach continued …

- The proposal will also include some costs and an estimate as to how longs it will take to deliver software to meet the requirements.
- The proposal will provide a good indicator as to how viable meeting the proposals is.
- At this point decisions makers will decide whether or not to give the go ahead to commence a project to deliver the software.

A typical Approach continued …

- If given the go ahead, workshops will commence with all interested parties – Business, Technical, Users, Vendors.
- The workshops discuss the requirements in details and produce a functional specification for review.
- Some of the estimates for time and costs are revised at this stage.
- Functional Specification is sent for review.
- The functional specification is revised if necessary, depending on review feedback, and signed off by the interested parties.

A typical Approach continued …

- Once the functional specification is signed off, this is used to commence the design. The design is carried out by technical architects and/or software developers.
- The design is documented in a technical specification, reviewed, maybe revised, signed off and handed over to a software developer.
- The developer will then write the code to meet the design.

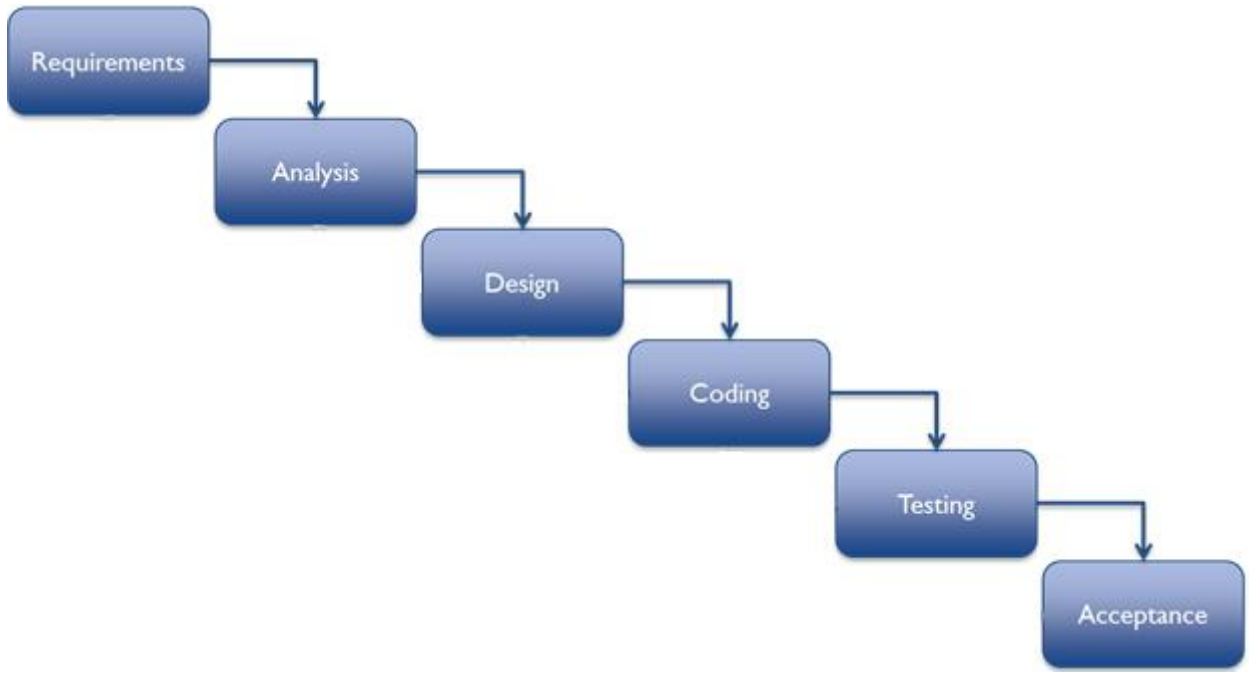
A typical Approach continued …

- The developer will do some unit testing on the code once they have completed their task to ensure nothing from the specification was missed.
- Test cases will be created from the specification and handed over to testers to ensure they can be executed and completed without issue.
- When testing is complete and signed off, the software is made available for release.

Software Development Stages

- The approach outlined on the previous slides could be categorised into the following stages:
  - Requirements
  - Analysis
  - Design
  - Coding
  - Testing
  - Acceptance

Software Development Stages



Software Development Life Cycle (SDLC)

- The process described is viewed on as a traditional approach to building software.
- It is referred to as the Waterfall Model.
- The approach is linear as in each stage is well defined and commences once the previous stage is complete.
- This approach is around since the 1950's, but was refined and referred to as the Waterfall model in the 1970's.
- It was and continues to be used for software development, and has a number of advantages.

Waterfall Advantages

- Easy to understand
- Provides Structure
- Milestones are well understood
- Requirements stability
- Easy time management

Waterfall Disadvantages

- The disadvantages are garnered from various sources of criticisms of the waterfall model.
  - It is rigid and inflexible.
  - Real projects rarely follow the sequential flow that the model proposes
  - At the beginning of most projects there is often a great deal of uncertainty about requirements and goals, and it is therefore difficult for customers to identify these criteria on a detailed level. The model does not accommodate this natural uncertainty very well.
  - Developing a system using the waterfall model can be a long, painstaking process that does not yield a working version of the system until late in the process.

Waterfall Criticism

- Knowing all the requirements fully in advance is the main problem with the waterfall approach. During the course of design and development it can become obvious that some requirements need to be amended, new requirements added and others dropped.
- The waterfall model does not provide the flexibility to accommodate this.
- An example, based on old research but still relevant to emphasize the point:
  - Research showed that 30% of requirements on Microsoft projects were not known in advance but emerged once development was underway.
- Waterfall model can lead to projects failing.
- It is not suitable to more modern applications development – mobile or web, and it more suitable to the traditional type of software which is complex and large in scale.

Waterfall Criticism

- Some would argue the criticism is unjustified. These problems arise as it is not properly managed or requirements not understood. Basically blame any criticism of the model on the people involved.
- Over the years, if you students pursue careers in various roles within software development you will form your own views based on your own experiences.

Waterfall Contributes to Failure

- One of the criticisms of the waterfall model is that it contributes to projects failing.
- Some of the reasons for failure are described by the following symptoms:
  - Project not delivered on time and within budget.
  - Repeatably missing project milestones.
  - Project cancellation prior to delivery.
  - Application out of date by the time of delivery.
  - Staff burnout, resulting in people leaving and new staff struggling to take up the baton.

Alternative Approach

- To address the criticism of the Linear and Rigid Waterfall Model, as some see it, alternative approaches to this model were devised.
- These approaches still recognise the stages, but attempt to address the rigidness, and other problems which can occur with the waterfall model.
- One approach was to allow for some flexibility within the model, so that the model is still largely adhered to, while another approach is to provide not only flexibility but peoples approach to using such a model.

Alternative Approach

- We will look at these approaches in the subsequent slides, which are:
  - Iterative
  - Agile

Quiz

- What stage comes after design?
- What stage comes before testing?
- What are the advantages of the waterfall approach?
- What are the disadvantages of the waterfall approach?

Iterative

- A **process** for arriving at a decision or a desired result by repeating rounds of analysis or a cycle of operations. The objective is to bring the desired decision or result closer to discovery with each repetition (**iteration**).
- It is about breaking down the work into more manageable chunks. Still follows the waterfall methodology and is managed in a similar way.
- For example, a system design might be broken down into 20 requirements. On each iteration, 4 requirements are delivered. After the fifth iteration, all requirements are delivered.

Iterative

- A lot of the work in defining the requirements would be required before the requirements can be split up for progression into different iterations.
- From Design phase on, the iteration typically applies, but on next iteration some analysis will be required.

Iterative


Vision
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Continue
Iteration Detail
Implementation & Developer Testing
Design & Analysis
QA / Acceptance Testing
Detailed Requirements
Evaluation / Prioritization
(Deployment)

Iterative Advantages

- Generates working software quickly and early during the software life cycle.
- More flexible – less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Easier to manage risk because risky pieces are identified and handled during its iteration.
- Each iteration is an easily managed milestone.

Iterative Disadvantages

- Each phase of an iteration is rigid and do not overlap each other.
- Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

Agile

- Agile is a more flexible approach which aims to focus on all stages at the same time.
- Where the waterfall model was widely used for more traditional software systems it did not necessarily suit for more recent technologies, such as web and mobile, unless the development was more large scale and/or complex.
- The previous point is a general comment. Waterfall model is still used for software systems on more recent technologies and in a number of cases quite successfully.
- Agile is considered a more lightweight looser approach which moves project control from managers to the developers and end users (collaboration approach).
- There are many agile approaches available or defined. Some of them are listed below:
  - Scrum
  - Adaptive Software Development
  - Lean Development
  - Extreme Programming (XP)
  - Agile Modelling (AM)
  - Feature-driven Development (FDD)
- Which one is chosen can depend largely on the software requirements, people involved and company culture.
- Some of the methodologies largely focus on project management and collaborative practices.
- While for others the focus is on software implementation practices.

Agile Project Management Methodologies

- The following methodologies listed on the previous slide provide a focus on project management and collaboration practices:
  - Scrum
  - Adaptive Software Development
  - Lean Development

- The following methodologies listed on the previous slide provide a focus on software implementation practices:
  - Extreme Programming (XP)
  - Agile Modelling (AM)
  - Feature-driven Development (FDD)

Note: This categorization of the methodologies is a broad perception. Arguments would exist for categorizing some, if not all, in both category. It largely depends on how these models are adapted.

Agile

- We will look mainly at XP as it gives a good idea in general of an agile approach.
- We will also briefly describe SCRUM for comparison purposes.
- First, lets look at some of the general characteristics of Agile.

Characteristics of Agile

- Simple planning and design
- Short Development Iterations
- Frequent releases
- Frequent customer feedback
- Tacit knowledge (Shared amongst team members, not formally communicated or documented)
- Development is test driven resulting in test cases getting produced before and during development rather than the traditional approach of producing test cases after the event.

Characteristics of Agile

- Change of requirements or new requirements are accommodated.
- Teams are self organizing (Less need for managers).
- Release debriefs which allow project teams to reflect on what went well and what didn’t allowing teams to learn and improve for the next development iteration.
- Smaller team sizes working short development iterations improves communication and productivity.

Agile Problems

- Micro management: Project managers can’t let go, and get involved in everything. This leads to micro management of tasks and issues. Also, the number of meetings team members have to attend increases.
- The user community needs to be actively involved throughout the project. While this involvement is a positive for the project, it is demanding on the time of the staff and can add project delay.
- Communication and coordination skills take centre stage in project development. If team members are lacking ability in this area then this can become an issue.

Agile Problems

- Informal requests for improvement after each phase may lead to confusion – a controlled mechanism for handling substantive requests needs to be developed.
- The iterative model can lead to scope creep, since the user feedback following each phase may lead to increased customer demands. As users see the system develop, they may realize the potential of other system capabilities which would enhance their work.

Extreme Programming (XP)

- User Stories: A series of statements from the customer describing how they would like to interact with the system reduces complexity.
- Planning Game: Just plan enough to get started and after each release review the plan and modify for next release.
- Short Iterations: Frequent releases, typically every 3-6 months, ensures regular feedback from the customer to the developer allowing for improvements and better understanding of what is required. Each iteration is referred to as a phase or time box.
- Test Driven Development: Preparing tests before and during development allows for changes in requirements and helps to identify defects early. This helps to ensure good quality.

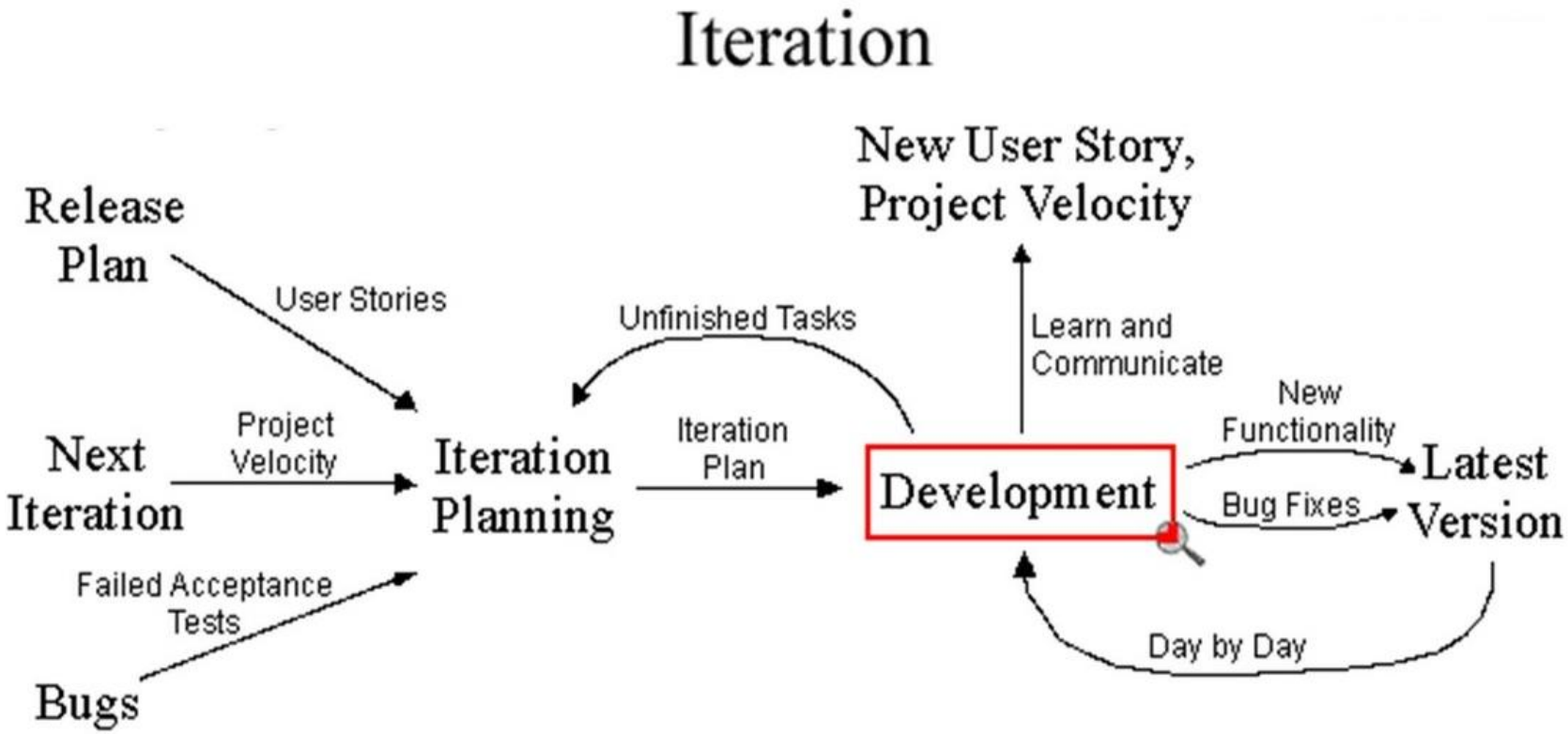
Extreme Programming (XP)

- Refactoring: Continually enhancing the design approach of applications helps to simplify the development by reducing redundancy and complexity.
- Pair Programming: Two developers working together at one terminal increases productivity, improves the quality of output, promotes learning and creates joint ownership of the code. The last point is important especially when maintaining complex code you are not relying on just one developer.
- Continuous Integration: Scripts of code are under version control (e.g. GitHub, SVN, Mercurial).

Extreme Programming (XP)

- Collective Code Ownership: The whole team owns the code. Pair programming promotes this.
- Onsite Customer: Provides quick feedback on progress and allows any misunderstandings the developers have about the requirements to be resolved quickly.
- Sustainable Pace: Avoid working late nights. When planning, estimates are based on 40 hour weeks working at a consistent pace.
- System Metaphors: It is important to use metaphors that everyone in the team understands. Managers, customers and developers need to understand each other.

XP Diagram


Iteration
New User Story,
Project Velocity
Release
Plan
User Stories
Unfinished Tasks
Learn and
Communicate
New
Functionality
Project
Velocity
Iteration
Plan
Next
Iteration
Iteration
Planning
Development
Latest
Version
Bug Fixes
Failed Acceptance
Tests
Day by Day
Bugs

Agile Conclusion

- All the methodologies are fairly adaptive so can be mixed and matched. Usually an organisation will adapt certain practices from a number of methodologies.
- The success of adapting a methodology to ensure a successful build, typically measured on time and cost, largely depends on the people involved in the project.
- Basically how it is managed, and adapted by team members. Everyone needs to contribute and go along with it, whether they agree or not, for it to have any chance of being a success.
- Agile manifesto: (http://agilemanifesto.org/)

XP Strengths

- It’s a move away from the bureaucratic approach of controlled management and documentation to coaching and knowledge sharing through informal communication.
- Decision making is placed in the hands of the developers and end users instead of managers.
- Emphasis is on delivering the software and not documentation.
- Promotes technical excellence. This is achieved through test driven approach, simple design, refactoring and collaboration. This ensures the quality of the application.
- End users will defend the application from criticism.

XP Weaknesses

- Difficult to have end users permanently assigned to development team. They normally have other duties that require their time as well.
- Management are fearful of XP. They don’t have the control they normally would, as with the waterfall approach for example. They can sometimes find it difficult to understand what is going on. They have to trust others. Of course this means less work for them but if a control freak by nature then this can be hard to accept and lead to them interfering.
- Can be difficult to apply and manage when the team of developers is large, usually more than 10 people.
- Can be difficult to implement if team members are not all in one location.
- Pair programming can be difficult to implement. Depends on the personalities.

Note: All these weaknesses can be addressed as the practices can be modified to accommodate any of the weaknesses associated with XP.

SCRUM

- Focus of SCRUM is on the management of the project.
- It is an agile, iterative, incremental developing method which assumes that changes and chaos exist through entire life-circle of the project and attempts to solve these problems.
- Scrum is designed to add energy, focus, clarity and transparency to project teams development software systems.
- It allows team to operate in close proximity to foster rapid system evolution.

Sprints

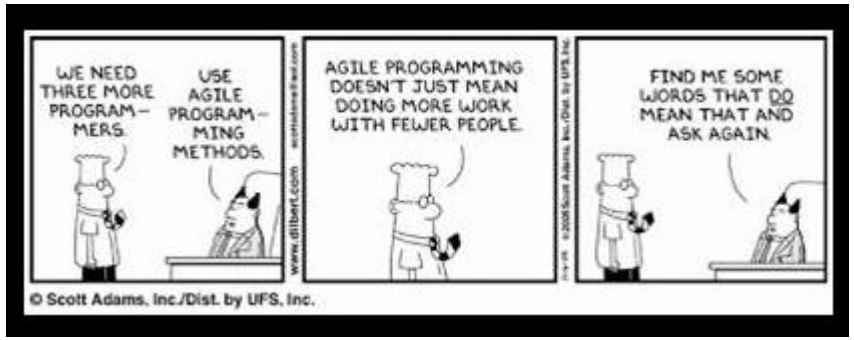
- In Scrum, work is structured in cycles of work called sprints.
- These are iterations of work that are typically two to four weeks in duration.
- During each sprint, teams pull from a prioritized list of customer requirements, called user stories, so that the features that are developed first are of the highest value to the customer.
- At the end of each sprint, a potentially shippable product is delivered.
- SCRUM is a methodology which can be adapted at any stage within a project lifecycle. For example, if the development phase is in trouble then the SCRUM model can be applied to help complete the development. Of course, this could apply to other models as well, but the simplicity of the SCRUM model makes it more suitable for such a scenario.

SCRUM Diagram

SCRUM Versus XP

- Scrum is more high level, focusing on the management of the project (e.g. the requirements or features are managed) rather than specifying or defining engineering practice such as pair programming or test driven development
- The length of an iteration in XP is usually 1-3 weeks whereas, in Scrum sprints are 1-4 weeks
- Once sprint (or iteration in XP) starts, customer cannot change the requirements, in other words the customer will have to wait until the sprint (or iteration in XP) finishes. In XP however, requirements can change anytime
- In XP features are developed in a strict order whereas in Scrum, the team is free to choose the features to be developed. Sequence does not matter
- Both XP and Scrum define the role of a coach, In Scrum it is called Scrum Master and requires (or strongly recommended) certification, whereas, XP defines the role of coach quite informally and the role may float between members of the team.

Note: Scrum and XP are often used together: Scrum defines the framework and XP defines the engineering practices and they fit together nicely.


WE NEED THREE MORE PROGRAM-MERS.
USE AGILE PROGRAM-MING METHODS.
AGILE PROGRAMMING DOESN'T JUST MEAN DOING MORE WORK WITH FEWER PEOPLE.
FIND ME SOME WORDS THAT DO MEAN THAT AND ASK AGAIN.
www.dilbert.com
© Scott Adams, Inc./Dist. by UFS, Inc.