Software Development
Life Cycle

Lecture 6
GSL Peru 2014
Housekeeping

- Friday’s are not optional.
Announcements
SDLC

Software Development Life Cycle
Software Development Life Cycle

CONCEPTION
INITIATION
ANALYSIS
DESIGN
CONSTRUCTION
TESTING
DEPLOYMENT

Waterfall Model VS Agile

CONCEPTION
INITIATION
ANALYSIS
DESIGN
CONSTRUCTION
TESTING
DEPLOYMENT

Source: sdlc.wc
Waterfall Model

- Sequential design process
  - Scheduled stages of development in strict order
- Does not accommodate changes to requirements during project
- Integration done at the end
Breakdown of Process

❖ Step 1: Requirements
  ➢ Description of product/system behavior
  ➢ Includes the use cases/interactions between the users and product
  ➢ Establishes what the product should and should not do (functional and non-functional)
Requirements Documents

- Business Requirements Document (BRD)
- Marketing Requirements Document (MRD)
- Functional Requirements Document (FRD)
- Product Requirements Document (PRD)
Engineering Requirements Documents

- User Interface Requirements Document (UIRD)
- Interface Requirements Document
- Technical Requirements Document (TRD)
- Design Requirements Document
- Engineering Requirements Document
- Development Requirements Document
Breakdown of Process

❖ Step 2: Design
  ➢ Create the specification of the software architecture
  ➢ Low-level algorithm design and high-level architecture design
  ➢ Things to consider with software design:
    ■ Compatibility
    ■ Extensibility
    ■ Maintainability
    ■ Modularity
    ■ Performance
    ■ Scalability
Design Documents

- Data Design
- Architecture Design
- Interface Design
- Procedure Design
Breakdown of Process

❖ Step 3: Implementation

➢ Constructing or coding based on the design from step 2, resulting in software
➢ Keep in mind how to create good software (future lecture material)
➢ Take into account user-knowledge
  ■ Product is for the users--implement a product suitable for their needs!
Breakdown of Process

❖ Step 4: Verification
➢ Consists of testing and debugging
➢ Don’t rely on Quality Assurance (QA) to find all the product defects!
➢ Developers should test their code.
➢ QA will not have time to test all of the features on all of the platform. The hours required to test manually will be 100+ years depending on complexity
➢ QA uses 80/20 rule to test.
Breakdown of Process

❖ Step 5: Maintenance
  ➢ Future modifications to remove issues, improve performance, etc.
  ➢ Have a method for users to report bugs or request modifications
  ➢ If defects are found here, it is very time consuming and disruptive to fix.
Additional Facts

- Design up front model - need to know all the facts ahead of time. Cannot learn as you go.
- Can fall apart when all the facts are not available ahead of time or when requirements change.
- Requirements defect that is left until construction or maintenance will cost 50-200 times as much to fix as at requirements stage.
- More than source code for documentation.
Agile Model

- Assess direction through development process
- Continuous replanning
- Iterative and incremental
  - Repetition of work cycles and product yield analysis

Source: agilemethodology.org, sdc.net.au
12 Principles of Agile Manifesto

1. Customer Satisfaction from rapid delivery of useful software
2. Welcome to changes in requirements, even later in the development process
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business-side and developers

Beck, Kent 2001
12 Principles of Agile Manifesto

5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the measure of progress
8. Sustainable development at a constant pace
9. Continuous attention to technical excellence and good software

Beck, Kent 2001
12 Principles of Agile Manifesto

10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Self-organizing teams
12. Regular adaptation to changing circumstances
Additional Facts

- Short, adaptive cycles
- Criticized for code focus and lack of documentation
- Inefficient in large organization
- Adapted to processes outside of software
Scrum

- Focus on common goal
- Flexible, quick delivery
- Requirements can change ("requirements churn")
Roles

- **Product Owner** - represent stakeholders, creates backlog items from user stories
- **Development Team** - responsible for producing potential shippable increments
- **Scrum Master** - enforcer of the scrum rules and removes obstacles from the team to deliver the product goals
Sprints

- Basic unit of development time
- “timeboxed” effort - scope based on time
- Duration is fixed from 2 weeks to 1 month
- Product must be in working condition at the end of the sprint. i.e. integrated, fully tested, end-user documented, and potentially shippable
Meetings

● Sprint planning meeting

● Daily Scrum meeting (status)
  ○ 15 mins/standing
  ○ same location/same time
  ○ development status
    ■ What have you done since yesterday?
    ■ What are you planning to do today?
    ■ Any impediments/stumbling blocks?

● End meeting
  ○ Sprint Review
  ○ Sprint Retrospective
Other terms

- Product backlog
- Sprint backlog
- Product increment
- Burn down chart
Overview

❖ Flexible and holistic product development strategy
❖ Recognizes that customers can change their mind about what they want or need
  ➢ Focus is on quick delivery and responses to change
  ➢ Empirical feedback
  ➢ Team self-management
Extreme Programming (XP)

- XP teams performed analysis, design, coding, and testing every day
- Test-driven development
- Short iterations provided structure
  - Iteration started with planning, ended with product demo

**The XP Lifecycle**

Source: James Shore 2007
Pair Programming

What is Pair Programming?

Pair programming is an agile software development technique where two programmers work together at one workstation. One programmer writes code (driver) while the other reviews each line (observer). Both programmers switch roles frequently.
Pair Programming

Benefits

Looking at benefits in the following categories:

❖ Economics
❖ Design Quality
❖ Satisfaction
❖ Learning
❖ Team Building & Communication
Pair Programming Benefits

Economics

- Takes about 15% more time than working individually, but defects are 15% less
- Costs and quality assurance affect expenses
  - Reduce defects in program => expenses decrease
- Example: IBM spent $250 million repairing and fixing 30,000 customer-reported issues
  - Defects could have been reduced with pair programming
Pair Programming Benefits

Design Quality

❖ Greater potential for more diverse solutions
  ➢ Programmers bring different prior backgrounds and experiences
  ➢ Programmers have different perspectives of the problems presented
  ➢ Programmers have different functional roles
    ■ Coding vs. Reviewing

❖ Chances of selecting a poor method decrease with two programmers rather than one
Pair Programming Benefits

Satisfaction

❖ Online surveys show…

➢ 96% of pair programmers enjoy pair programming more than working alone
➢ 95% of pair programmers are more confident in their code when working together
Pair Programming Benefits

Learning

❖ Constant sharing of knowledge between the programmers
   ➢ Tips on coding rules
   ➢ Tips on design skills

❖ Providing feedback increases knowledge for the reviewer as well, not only the coder
   ➢ Programmer becomes more aware of monitoring code
Pair Programming Benefits

Team Building & Communication

❖ Programmers in a team naturally share problems and solutions quicker with pair programming
   ➢ Time is saved
   ➢ Hidden agendas amongst team members are avoided

❖ Communication is made easier
   ➢ Information flow in the team increases
Waterfall vs. Agile Model

Pros and Cons

Pros
❖ Waterfall Model
➢ Strong documentation
➢ Clients know what to expect
➢ Meticulous records => easier to improve in future
❖ Agile Model
➢ Changes can be made after initial planning
➢ Testing and feedback at the end of each run
➢ Product can be launched at the end of any cycle

Cons
❖ Waterfall Model
➢ Initial requirements that can’t be changed
➢ Testing only at the end
➢ Doesn’t take into account client’s evolving needs
❖ Agile Model
➢ Project can easily become a constant run of code cycles
   ■ Delayed & over-budget
➢ No initial definite plan can result in very different end product
Waterfall vs. Agile Model

When to use Which

Waterfall Model
❖ Clear picture of what the final product should be
❖ Clients don’t have ability to change project scope after project has begun
❖ Definition, not speed, is key to success

Agile Model
❖ Unclear picture of what the final product should be
❖ Rapid production is of importance
❖ Clients can change the scope of project
❖ Product is for an industry with rapidly changing standards
Resources

- Wikipedia