CS 307: Software Engineering

Lecture 2: Software Quality, Software Life Cycles, and Scrum

Prof. Jeff Turkstra
Reminders

- Team assignments due **TODAY**
  - Even if it’s “assign me to a team”
  - Email jeff@purdue.edu

- Consultation meetings
  - Monday, August 28, 8:30am – 4:30pm
  - LWSN 3102
  - After teams are assigned, we’ll point you to a site for scheduling
Grades

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>65%</td>
</tr>
<tr>
<td>Midterm Exam</td>
<td>25%</td>
</tr>
<tr>
<td>Quizzes and Homework</td>
<td>10%</td>
</tr>
</tbody>
</table>

Grading issues will be addressed as they occur. Not at the end of the semester.
Homework 1

- Posted on the website
- Due in two weeks
- Discuss Tuesday
  - Look it over this weekend
- 15 Points
National Society of Professional Engineers

“to hold paramount the safety, health, and welfare of the public.”
Lecture 02

- Software Quality
- Software Life Cycles
- Scrum
Quality

- Conformance to requirements
  - Crosby
- Fitness for use
  - Japan
- I know it when I see it
  - Guaspari
- Value to someone
  - Weinberg
Software quality attributes

- For product operation
- For product revision
- For product transition
Operation attributes

- Correctness
- Efficiency
- Integrity
- Reliability
- Usability
Correctness

The degree to which a program satisfies the user’s requirements.
Efficiency

The amount of computing resources, time and space, required to perform a user-defined function or task
Integrity

How well is the software and data protected from security breaches and installation errors?
Reliability

To what degree can the system be expected to perform its intended function without failure in the user’s environment?
Usability

How much effort do the users have to spend learning to use the system efficiently?
Revision attributes

- Flexibility
- Maintainability
- Testability
Flexibility

How much effort will be required to enhance the system?
Maintainability

- How much effort will be required to locate and repair defects in the system?
Testability

How much effort will be required to test the structure and functionality of the system?
Transition attributes

- Interoperability
- Portability
- Reusability
Interoperability

How much effort will be required to link this program or system to another?
Portability

How much effort will it take to transfer a program or system from one machine to another?
Reusability

To what extent can the design, or system modules be used in other applications. How much effort will it take to reuse them?
Quality is everyone’s business

- Customer and end users
- Developers
- Maintainers
Quality is free

“What costs money are the unquality things – all the actions that involve not doing jobs right the first time.”

“An erroneous assumption is that quality is an intangible and therefore not measurable. In fact, quality is precisely measurable by the oldest and most respected of measurements: cold hard cash”
Is quality free?

“Well, maybe. However, it takes a lot of time and effort to achieve.”  

Crosby
Quality and Stakeholders

**Customer:**
solves problems at an acceptable cost in terms of money paid and resources used

**User:**
easy to learn; efficient to use; helps get work done

**Developer:**
easy to design; easy to maintain; easy to reuse its parts

**Development Manager:**
sells more and pleases customers while costing less
Software qualities can conflict

- Increasing efficiency may impact interoperability
  - Or portability, or maintainability, or ...
- Increasing usability may impact efficiency
- Increasing integrity may impact efficiency
- etc
Objectives

- Setting objectives for quality is a key engineering activity
  - You then design to meet the objectives
- Optimizing is always necessary
  - E.g., obtain the highest possible reliability using a fixed budget
Internal quality

- Characterizes aspects of the design of the software
- Impacts external quality attributes
- For example
  - Amount of commenting of the code
  - Complexity of the code
  - Use of well-understood software patterns
Software Development Cycle

- Begins with decision to develop software product
- Ends when the software is delivered
- Typically includes
  - Requirements phase
  - Design phase
  - Implementation phase
  - Test phase
  - Installation and checkout phase
Software Life Cycle

- Software Development Cycle
- And
  - Operation and maintenance phase
  - Retirement phase
Process models

“The primary functions of a software process model are to determine the order of the stages involved in software development and evolution and to establish the transition criteria for progressing from one stage to the next.” – Boehm
Process model questions

- What should we do next?
- When should we start doing it?
- How long should we continue to do it?
- How will we know when we are done?
- How well does it handle change?
- How are decisions made?
Why?

- To define the project activities
- For consistency
- To set up meaningful management checkpoints
Observations

- Life cycle is there to help
- The life cycle is a roadmap
- It is not the vehicle
  - People must still manage the project and carry out the tasks
- Software methodologies focus on how to navigate through the phases
  - Also deal with physical representation
Observations

- Software methodologies focus on how to navigate through the various phases of a life-cycle model.
- They also deal with the physical representation of the various phase products.
- Software process models provide the guidance on the order in which the major events in the life-cycle are carried out.
Models

- Code and Fix
- Stagewise and Waterfall
- Prototyping
- Evolutionary
- Spiral
- Agile/Scrum
Code and fix

Code

Fix

Product
Code and fix model

- Code a little
- Fix a lot
- Repeat until your time, money, people, and customers run out
- Good luck
- It’s time to update your resume
Waterfall model

System Engineering

System Analysis

System Design

Implement the System

Test the System

Maintain the System

© 2017 Dr. Jeffrey A. Turkstra
System engineering activities

- Business considerations
- Technical considerations
- Manufacturing considerations
- People
- Environmental considerations
- Legal considerations
- Develop an installation plan
System analysis

- This is the **what** part
- Identify the real system requirements
- Hopefully **work with** and involve the **user**
- Develop an **acceptance test plan**
System design

- This is the **how** part
- Identify the system specifications
  - Hardware
  - Software
- Develop the **system integration test** plan
System implementation

- Build the system
- Test the parts
- Carefully complete all system documentation
System test

- Carry out system integration test plan
- Carry out the acceptance test plan
- Carry out system installation plan
- Establish system regression test library
System maintenance

- Manage change
- Fix defects
- Port the system
- Enhance the system
- Retire the system
Weaknesses of classic waterfall

- Difficult to measure real progress and manage change
Tends to push much of the real analysis and design feedback into the maintenance phase
- Document driven up front
- Form seems to be more important than content
- Feedback paths are not used very much
Assumes an unrealistic sequential progression through the cycle
Top-down approach works as long as the number of levels between top and bottom are small. Too much focus on justifying work and meeting milestones and not enough focus on doing the work. This model is not able to respond to problems (change requests) in an efficient and timely manner.
When should it be used?

- For systems which are well understood?
- In environments with a good software development process?
- Small start up projects?
- When a high degree of risk and a great amount of accountability is required?
As a teaching tool to introduce students, in an orderly manner, to the activities required for a successful software project?

Maybe never?
Prototyping

1. Determine Requirements
2. Build a prototype
3. Do a Partial design
4. Evaluate the system
5. Engineer the product

Be sure to throw away the prototype.
Why prototype?

- When the user is not very computer literate
- When the user is unable to completely pre-specify the needed system requirements
- When there is little algorithmic detail
- Whenever you are not sure your design will work
Evolutionary development models

- Can become a modern version of code and fix
- Product evolves over time as the real requirements become known
- Assumes the customers will tell us what they like and dislike about each incremental release
  - Fix and enhance the product in next go around
Spiral model

- Incremental, risk-oriented
- Four main phases

Pros:
- Reduces risk of failure
- Functionality can be added later
- Software produced early
- Early feedback

Cons:
- Risk analysis requires expertise
- Complex
The Right Way (TM)

- There is no single best process model for developing high quality software systems
The following slides are based on a presentation created by Kevin Schenk, BS in Computer Science, 2012
Roles

- Product owner
- Scrum master
- Development team
Product owner

- Defines the features of the product
- Decides on release date and content
- Prioritizes features
- Adjusts features and priority every iteration, as needed
- Accepts or rejects work results
Scrum “master”

- Represents management to the project
- Responsible for ensuring adherence to scrum practices
- Removes impediments
- Ensures that the team is fully functional
- Shields team from external interference
Development team

- Members are assumed cross-functional
- Not always (usually?) possible
  - Programmers, UI designers, testers, business analysts, etc...
- Only title for members is "developer"
  - May have specialized skills, but accountability belongs to the team
- No sub-teams (testing, etc)
- "Optimal" size 3-9 members
Events

- Sprint
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective
Sprint

■ “Heart” of Scrum
■ Usually one month or less
  ▪ Consistent duration throughout development effort
■ “Done,” usable, and potentially-releasable product increment is created
■ New sprint starts immediately after previous ends
No changes are made that impact the Goal(s)
Team composition remains constant
Quality attributes do not change
Scope may be clarified/re-negotiated
Sprint planning meeting

- Work to be performed in upcoming sprint is planned
- Suggested eight hours for a one-month sprint
  - Shorter sprint, shorter meeting
- Two parts
  - What will be delivered?
  - How will the work get done?
Deliverables

- Product owner presents a list of items to complete
- Team develops a *sprint goal*, considering:
  - Product backlog
  - Latest product increment
  - Capacity of the team
  - Past performance
Sprint backlog

- Selected product backlog items, along with The Plan.
- Organized by team
- Estimate time to complete each item
- Assign tasks to individual members
Daily scrum meeting

- “Stand-up meeting”
- Fifteen minutes for team to “synchronize” and plan for the next 24 hours
- Held same time, same place
- Three questions
  - What was accomplished since the last meeting?
  - What will be done before the next meeting?
  - What obstacles are in the way?
Sprint review meeting

- Development team presents accomplishments
- Form of a demo of new features or progress
- Informal
- Entire team participates
Retrospective

- **Opportunity to reflect and plan for improvements during the next sprint**

- **Process**
  - Think about how the last sprint went
    - People, relationships, process, and tools
  - Identify major items that went well and potential improvements
  - Create a plan for implementing improvements
Retrospective questions

- What went well during the last sprint?
- What didn’t go well during the last sprint?
- How should the team improve for the next sprint?
Artifacts

- Project charter
- Product backlog
- Sprint backlog
- Burn down chart
Project charter

- Problem statement: short and succinct (one or two sentences)
- Project objectives: what the project will achieve
- Stakeholders: persons who will be actively involved with the project
  - Project sponsor, types of users, etc
- Deliverables: major results or services that will be produced
  - Specific things the software will do
Product backlog

- Ordered list of everything that might be needed in the product
  - Source of requirements
- Product owner is responsible for the backlog
  - Content, availability, ordering
- Never complete
- Lists all features, functions, requirements, enhancements, and fixes that need to be made
### Format

<table>
<thead>
<tr>
<th>Backlog Item</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a guest, I want to make a reservation.</td>
<td></td>
</tr>
<tr>
<td>As a guest, I want to cancel a reservation.</td>
<td>5</td>
</tr>
<tr>
<td>As a guest, I want to change the dates of a reservation.</td>
<td>3</td>
</tr>
<tr>
<td>As an admin, I want to change the availability of dates at my hotel.</td>
<td>8</td>
</tr>
<tr>
<td>As a developer, I want to improve exception handling.</td>
<td>8</td>
</tr>
</tbody>
</table>

- **Backlog items are usually of the form:**
  - As a ____ , I want to ____ (so that I can ____).
- **Product backlog items are sometimes called “user stories”**
Sprint backlog

- Team selects items from the product backlog and commit to completing them
- Identifies tasks associated with each item and estimates hours to complete

As a travel planner, I would like to see the reviews of each hotel.

Program the Back-End (8 hours)
Program the Front-End (4 hours)
Write Test Cases (4 hours)
Make Database Changes (2 hours)
Update Dependent Pages (3 hours)
# Sample sprint backlog

<table>
<thead>
<tr>
<th>Backlog Tasks</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thur</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program the Back-End</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program the Front-End</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Write Test Cases</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Make Database Changes</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update Dependent Pages</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>
Burn down chart

- Graphical representation of work left to do vs. time left in which to do it
Scrum

Product Tasks → Sprint Tasks → Sprint → Working increment of software

- 24 h
- 30 days
Criticisms

- Assumes you can estimate task duration
- Many meetings
  - Expensive
  - Disrupt flow
- Interchangeable cogs
- Regulatory environments?
- Difficult to collaborate with outside groups
  - How does marketing know what to advertise?
Agile assumes development will be ongoing

- If it doesn’t get done, it doesn’t get done
Sprints

- We will have three sprints:
  - Tue, Sep 19 – Fri, Oct 6
  - Thu, Oct 12 – Fri, Nov 11
  - Tue, Nov 7 – Fri, Dec 1

- Sprint planning meeting should take from one to two hours

- “Daily” scrum meeting should be held 2-3 times per week
  - Same time and place
Sprint review meeting

- Held with a project coordinator on or before the following dates:
  - Thu/Fri October 5-6
  - Thu/Fri November 2-3
  - Thu/Fri November 30, December 1

- Final Project Presentations
  - Dead week
  - See course website
Questions?