Software Engineering Overview

Zohra Hemani
GWU CS Class of 2001
Group & Project Manager, Northrop Grumman
Agenda

- What is Software Engineering?
- Process Areas
- Software Development Methodologies
  - Waterfall
  - Spiral
  - Agile
- Class Exercises
- Lessons Learned
- Q&A
What is Software Engineering?

- When you hear Software Engineering, what comes to mind?

*It is not just programming.*

*Cartoon source: [http://wwwx.cs.unc.edu/~pozefsky/COMPS23_S08/]({http://wwwx.cs.unc.edu/~pozefsky/COMPS23_S08/})*
Software Engineering is...

- Software engineering (SE) is concerned with developing and maintaining software systems that behave reliably and efficiently, are affordable to develop and maintain, and satisfy all the requirements that customers have defined for them. (ACM.org)
- More than just programming...
  - Getting & understanding customer requirements
  - Designing, developing & testing software that meets user needs
  - Follow a process to attain high quality end product
  - Team consists of a large # of stakeholders...Customer, Users, Manager, Architect, Sub-area leads, Developers, Integrators, Testers, Configuration Manager, Quality Assurance, Upper Management, etc.

http://computingcareers.acm.org/?page_id=12
sunset.usc.edu/~neno/cs589_2003/Week1.ppt

Basic tension of software engineering
   better, cheaper, faster — pick any two!
What happens when a project is short-staffed? You end up working in other domain areas. Just because you are software engineers doesn’t mean you won’t have to deal with hardware or won’t have to do technical writing.
Process Areas

- Integrated Project Management
  - Establish and manage a project and all its stakeholders by a set of plans and processes
- Requirements Development and Management
  - Analyze customer needs, develop requirements and manage changes
- Architecture & Design
  - Create high level design/plan of how the solution will be implemented
- Development
  - Programming/coding of the solution; includes detailed design and unit testing
- Integration
  - Bring the various pieces of the system together such as COTS, GOTS, custom components etc.; includes software deployment
- Verification
  - Test to evaluate if the product has been developed correctly and meet the specified requirements (Did we build the system right?)

PM: Project Management Plan, Project Schedule, etc.
Development: Make sure to do your design and plan out/figure out what you got to do before you start coding. Design & Code Reviews are important. Peer or pair reviews
Integration: But it works on my machine;
Verification: Do dry runs and keep track of them

Why are the various areas important and what could go wrong without them?
Examples:

Potential Risk: HW may not come in time
Risk: difference between risks and issues. Issues are risks that have actualized. Example: waiting for a computer to do development on and don’t know when the computer will come...what risk mitigation plan can you develop.

CM: from source control and baselines to change requests to locked cabinet of assets (avoid things walking away or people developing on different versions of sw); configuration control board; make sure baseline matches requirements (e.g. right versions of COTS software)

QA: following process such as doing design reviews; also helps in doing causal analysis
System: setting appropriate permissions, not requiring admin privileges to run your sw (least access); This way if there was a vulnerability, the complete system would not be compromised.

Validation example: does the product fulfill its intended purpose
Mini Class Exercise
Software Development Lifecycle Models

- Waterfall
- Agile
- Spiral
Development consists of implementation & integration. linear and sequential

Source: http://delivery.acm.org/10.1145/1770000/1764814/p8-ruparelia.pdf?key1=1764814&key2=3698244821&coll=GUIDE&dl=GUIDE&CFID=104417691&CFTOKEN=54415256
Customer doesn’t see the product till test time...
Document driven...

## Waterfall Model Highlights

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Specific phases &amp; their defined start &amp; end points</td>
<td>• What is on paper is not always what the customer wants – does not lend well to changing requirements</td>
</tr>
<tr>
<td>• Easy to track progress in phases due to milestones</td>
<td>• Only good for 'stable' projects otherwise risk of inaccurate time &amp; cost estimates</td>
</tr>
<tr>
<td>• Focus on having defined requirements ahead of time reduces risk for scope creep</td>
<td>• Jumping back &amp; forth between phases is not feasible</td>
</tr>
<tr>
<td>• Design before development encourages higher quality</td>
<td>• Longer time to deliver product to user &amp; get feedback</td>
</tr>
</tbody>
</table>
Agile Model

Start → Waterfall → Spiral 0 → Initial Requirements

Spiral 1 → Spiral 2 → Agile → Actual User Needs

Scoop Cost? → Change Requests
Agile Model Highlights

- Permit change to happen, don’t try to control change
- Opportunity to stop, assess, and re-plan at frequent intervals during the development process
- Iterative development - Short cycles (1-4 weeks) of focused development on completing a specific set of functionality
- Revisit interim progress with the stakeholders to ensure successful completion of the requirements (helps with ill-defined or unknown requirements) – need customer buy-in
- Continuous build and test during the development
- Not Process Free – highly structured environments

Agile Alliance (www.agilealliance.org)
   A non-profit organization promotes agile development

Pros:
- Requirements churn is managed (not avoided)
- Customers see progress regularly and have an opportunity to provide input

Cons:
- Need full customer buy-in to be successful
- Hard to see the forest for the trees

One approach for requirements: User Stories
Agile Methods (Scrum)

- An iterative and incremental project delivery framework
- Allows us to rapidly & repeatedly inspect actual working software
- Every one week to a month anyone can see real working software and decide to release it as is or continue to enhance for another iteration
- Business sets the priorities. Our teams self manage to determine the best way to deliver the highest priority features
- Allows us to adjust to changing requirements in a managed fashion
- Small, Self-organizing Teams

Other Methods:
Extreme Programming (XP)
Adaptive Software Development (ASD)
Dynamic System Development Method (DSDM)
Feature Driven Development
Pig Roles: The Pigs are the ones committed to the project in the Scrum process—they are the ones with “their bacon on the line” and performing the actual work of the project.

Roles: Product Owner, ScrumMaster, Team

Artifacts: Product Backlog, Sprint Backlog, Estimations, and Burndown Charts

Daily stand-ups
3-week sprints
Consolidated/information documentation
Variety of tools
Formalized ‘done’ checklist
Class Exercise – Waterfall (2 Teams)

- Assign roles to represent each step of the waterfall approach
  - Project Lead
  - Systems Engineer (Requirements)
  - Architect (high level design)
  - 2 Developers
  - 1 integrator
  - Tester
- Go through the steps using scenario – draw a house
- Discuss results

Project Goal: drawing of a house
Time frame: 20 minutes

Roles: Project Lead, Sys Eng (requirements), Architect, 2 developers (gets assigned differences parts by the PM), 1 integrator, tester, customer (me)

Watch out for:
- Distractions
- Scope Creep
- Schedule delays (customer availability, equipment deliveries etc)
- Team members leaving
Class Exercise – Agile (1 Team)

- Assign the various roles:
  - Product Owner
  - Scrum Master
  - Architect (high level design)
  - 2 Developers
  - 1 integrator
  - SE & Tester (1 person)
- Go through the steps using scenario – draw a house
- Discuss results

Project Goal: drawing of a house
Time frame: 20 minutes

Roles: Project Lead, Sys Eng (requirements), Architect, 2 developers (gets assigned differences parts by the PM), 1 integrator, tester, product owner, scrum master (? Me?)

Watch out for:
-disengaged customer
-How is the team member leaving handled in this case?

Scrum Master:
Wikipedia: primary job is to remove impediments to the ability of the team to deliver the sprint goal/deliverables. The ScrumMaster is not the leader of the team (as the team is self-organizing) but acts as a buffer between the team and any distracting influences. The ScrumMaster ensures that the Scrum process is used as intended. The ScrumMaster is the enforcer of rules. A key part of the ScrumMaster’s role is to protect the team and keep them focused on the tasks in hand.
Real-world Lessons Learned

- It always takes longer than you think
- There is no such thing as complete requirements
- There is such a thing as unclear, useless requirements
- “…but it works on my machine…”
- Developers are not the most important people on the project
- Stay current: stay abreast of new technologies
- Expand your scope & learn about other areas such as system integration

Cartoon sources:
http://www.cartoonstock.com/newscartoons/cartoonists/twi/lowres/twin660l.jpg
http://cnx.org/content/m32188/latest/
The longer a fault exists in software
  the more costly it is to detect and correct
  the less likely it is to be properly corrected

Up to 70% of all faults detected in large-scale software projects are introduced in requirements and design
  detecting the causes of those faults early may reduce their resulting costs by a factor of 100 or more
Questions?
Backup
Mini waterfalls.

i. Determine objectives and system requirements – plan; determine constraints and alternatives
ii. Evaluate alternatives, and identify and resolve risks. Develop preliminary design. Initial prototype may be developed to clear uncertainty in requirements/resolve risks.
iii. Develop and test next level product. First full prototype.
iv. Plan the next iteration/phase. Test and show the product to customer to get feedback.

A little time is initially spent in each phase followed by several iterations over all four phases.

Source: http://delivery.acm.org/10.1145/1770000/1764814/p8-ruparelia.pdf?key1=1764814&key2=3698244821&coll=GUIDE&dl=GUIDE&CFID=104417691&CFTOKEN=54415256
## Spiral Model Highlights

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Thorough risk analysis</td>
<td>• Relies on understanding of complete requirements at the beginning</td>
</tr>
<tr>
<td>• Initial version of software is produced sooner than in waterfall approach</td>
<td>• Big design still happens upfront</td>
</tr>
<tr>
<td>• Provides an iterative approach</td>
<td></td>
</tr>
</tbody>
</table>