Some Important Announcements

- Cancel mid term on Jan 9 but will be a quiz due to midterm and final are 1.5 weeks away.
- There will be an industry talk on Jan 9 at 5:30
Software Engineering talk

- **Title:** Lean or Agile software engineering process: an industry perspective
- **By:** Keith Hanson, CEO, Twin Engine Labs
- **Time & Place:** 5:30pm, Jan 9 2014, Bogard Hall
Towards creating survivable architecture

View of Software Engineering

User Needs/Inputs – Specification or Requirement

Technical: Design/Analysis Implementation

Project/Product Management: Manufacturing/Quality Control

Marketing

Sale

Typical SE life cycle process

• User Inputs -> Requirements and Specification
  • Design & Analysis -> Analysis & Design documents
  • Implementation -> Programming, integration, Tools -> Code or system
  • Testing -> Test Plan & Test results
  • Manufacturing, installation/Deployment and Quality Control -> Change Management & Configuration Management.

• A classic water model.. Not good..why?
• the above SE workflow can be divided into smaller iterations
Recap: requirements, analysis & design

Requirements and Specifications

- User Inputs/expectations are translated into agreement documents among users/customers and various stakeholders in SE lifecycle

- Can be legal documents between client and supplier

- How do we know whether the software product will meet the expectation?
Requirements and Specifications

- Functional Requirements
  - Tangible Needs
  - E.g. your order processing system, online store with shopping cart.
- Non-Functional Requirements
  - Performance (how well your system can perform, # transaction)
  - Reliability (how long your system can run w/o failure or what is uptime?)

- How do we know whether the software product will meet the expectation?

What are requirements?

- “What customers or users expect from the system”
- Two types
  - Functional Requirements
    • Features (more tangible)
  - Non-functional requirements
    • Reliability and performance (equally if not more)
**Capture requirement**

- Reach agreement on system context
  - provided by customers
  - Vision statement (e.g. from marketing/product team)
  - Survey or research
- Communication.
- Articulation.
- Clarity.
- Come up with Abstractions of a given problem domain
- **Arrive at actions representing/involving the abstractions (USE-CASES)**

**Requirements definition vs specifications**

- Requirements definition
  - A statement in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers
- Requirements specification
  - A structured document setting out detailed descriptions of the system services. Written as a **contract** between client and contractor
- Software specification
  - A detailed software description which can serve as a basis for a design or implementation. Written for **developers**
Requirements readers

- Requirements definition
  - Client managers
  - System end-users
  - Client engineers
  - Contractor managers
  - System architects

- Requirements specification
  - System end-users
  - Client engineers
  - System architects
  - Software developers

- Software specification
  - Client engineers (perhaps)
  - System architects
  - Software developers

Use case process & notation

- Identify actors
- Brain-storm actions that will lead to features/promises to customers
- Refine use-cases and add exception cases
- eg. A doctor clinic

Towards creating survivable architecture
Our previous homework

1. follows the req workflow guideline and works for ATM systems
   – Brainstorm Actors
   – Finding use cases
   – For online students, please submit your requirement documents based on this template for ATM (online Banking)
     – Requirement document template:
       http://www.latech.edu/~box/ase/srs_template.doc

HW Requirements
Use Cases for ATM homework

ATM

1. Login/Swipe Card
2. Enter Pin
3. Check Account

Use Cases (continued)

ATM

1. withdraw
2. deposit
3. transfer
HW on requirement

5.1 Log in
Actor: caretaker, doctor, system administrator
Goal: Authenticate a user
Input: The user must input his credentials
Output: The system responds by authenticating the user
Main Scenario: The user enters his credentials correctly and is successfully authenticated
Pre-condition: The user is not yet authenticated
Steps:
- the user clicks the "login" button on the home page and the login form is displayed
- the user introduces his username and password and clicks "login"
- if the username and password are correct, the personal homepage is displayed
Post-condition: The user is authenticated and redirected to his homepage
Exceptional Scenario: non-existent user name
If the user name does not exist, an appropriate message should be displayed
Exceptional Scenario: wrong password
If the password is incorrect, the system must prompt the user to reenter his password

5.2 Log out

USE CASE types

- Main flow of events
- Exceptional flow of events
Recap: The Analysis Workflow

Purpose of Analysis & Design

- To transform the requirements into a design of the system to-be.
- To evolve a robust architecture for the system.

An Architecture document & later design documents (for each subsystem, modules)

- To adapt the design to match the implementation environment, designing it for performance.
What is system architecture?

- A blueprint for software system
- Includes
  - High Level Software components/subsystem
  - High Level Hardware components/subsystem
  - Relationship/connectivity
  - Data Models (optional)

Define Candidate Architecture

- Create an initial sketch of the architecture of the system
- Identify analysis classes from the architecturally significant use cases
- Update the use-case realizations with analysis class interactions
Analysis Artifacts (UML)

Robustness Diagram:
Usually contains attributes, not operations.

- Boundary Classes
- Control Classes
- Entity Classes

Example of Robustness Diagram: Login usecase
Sequence DIAGRAM (in design workflow)

Actor1

Home page
  login()
  display()
  enter user and password()
  display()

Login page
  display()
  validate user login()

Account

Recap: Design Workflow
GOAL OF DESIGN WORKFLOW

EXPANSIONS OF THE ANALYSIS MODEL

- MOSTLY PHYSICAL
- MORE DETAILED
- SHOWS THE DECISIONS
- Capture how internal working of the system at the object-level

ARTIFACTS

- CLASS DESIGN & various static & dynamic structure
- USE CASE REALIZATION-DESIGN
- DESIGN MODEL
- DEPLOYMENT MODEL
DESIGN MODEL

• Class Design
  – Class Diagram

• Use Case Realization-Design
  – sequence diagram or
  – collaboration diagram
  – Statechart diagram
  – Activity diagram

Example: class diagram

Superclass
(parent)

Subclasses

GroundVehicle
weight
licenseNumber
register()

Person
owner
0..*
1

Generalization

Car
size

Truck
tonnage
getTax()

Trailer
Activity Diagram (flow chart)

Example for a statechart diagram of an order class
Sequence DIAGRAM

Actor1

Actor1

Home page

Login page

Account

Actor1

display()

display()

enter user and password()

validate user login()

display()

display()

display()

display()

display()

COLLABORATION DIAGRAM

Customer

Account

Home page

Login Page

1. click login

2. display

3. enter ID and password

4. click

5. validate login (user, password)

6. display
architectural design

• Come up with a system layout
  – Subsystems -> component/interface
• Top down vs bottom up

Deployment Diagram
Deployment

- A static view of hardware configuration and software components that run on those systems
  - Hardware
  - The software on the hardware
  - The middleware connecting the disparate machines to one another.
- Notation:
  - 3-D box represents a node, either software or hardware
  - HW node can be signified with << stereotype >>
  - Connections between nodes are represented with a line, with optional << stereotype >>
  - Nodes can reside within a node

Sample of Deployment Diagram (2)
Summary on the recap

- User input -> requirements (usecases in UML)
- Use case oriented analysis, design & development as well as testing
- To ensure that customers’ expectations are met based on use case and quality assurance process

What are main activities in these workflows?

- Requirement
- Analysis
- Design
Implementation Work Flow

Introduction

• Fundamental goal is to build a working version of system.
• Craft out the Architecture
• Implement the Design in terms of components.
• Plan System Integrations
• Implement Design classes and subsystems
• Unit test components
• Integrate Components.
Implementation Workflow Activities and Workers

Architect & Developers → Architectural Implementation

System Integrator → Integrate System

Component Engineer

Implement a Subsystem/component → Perform a Test

Implement a Class

Artifacts

• Implementation Model
  - Component
  - Implementation subsystem
  - Interface

• Architecture Description
  - View of Implementation Model

• Integration Build Plan
Implementation

- Describes how elements of design model are implemented in terms of components such as source code, executable, etc.
- Write programs, write or acquire framework or components (IDE, J2EE or AMPP MVC etc)
- Describe how components are organized using the structuring and modularization mechanisms of the implementation environment and programming languages
- Hierarchy of Implementation (sub) Systems containing Components and Interfaces

Component

- Physical packaging of model elements
- Standard stereotypes which the UML assigns
  - <<executable>> -- a program that may run on a node
  - <<library>> -- static or dynamic library
  - <<file>> -- static or dynamic library
  - <<table>> -- a database table
  - <<document>> -- a document
  - technology specific (<<ActiveX>>, <<Applet>>, <<DLL>>, <<CORBA Component>>, etc.)
Implementation of subsystem or components

- Organize implementation model artifacts into more manageable pieces
- Manifested by a “packaging mechanism” in implementation environment
  - Component packages
  - Interfaces
- Trace one-to-one with design subsystems
  - same dependencies to other subsystems, interfaces
  - provides same interfaces
    - maps one-to-one with which components or other implementation subsystems within the subsystem provide the interfaces of the subsystem
Interface

- As in design, interfaces define the operations implemented by components and implementation subsystems.
- A component that realizes (and thus provides) an interface must implement all the operations defined by the interface.

---

Component implementation with build tools & version control

- Build tools
  - Unix make
  - Ant: XML based build tool
  - Organize implementation into modules
    - source
    - Document
    - Output (binary)
    - Package

- Version control or change Management
  - Help manage every change to your implementation or any artifact without any lost
Towards creating survivable architecture

Architectural Description

• implementation model
• Decomposition of implementation model into subsystems, their interfaces, and dependencies between them
• Key components
  – trace to architecturally significant design classes
  – executable components
  – components that are general, central, or implement generic design mechanisms that many other components depend on

Integration Build Plan

• Describes the build (SW product) that will occur with In a given iteration.
• Build plan for each increment
  – list of functionality: use cases and/or scenarios
  – list of implementation subsystems and components
• Test each build, including regression tests
• Configuration management with the build plan
WORKERS

- ARCHITECT
- COMPONENT ENGINEER or Developer
- SYSTEM INTEGRATOR

ARCHITECT

- Outlines the implementation model.
- Ensure the completeness and correctness.
- Mapping the executable components onto nodes within the deployment model.
- Include the components from the implementation model and the updated contents of deployment model.
Component Engineer/Developer

- Is responsible for the source code of the components and subsystems.
- For coding the interfaces associated with components and subsystems.
- Is responsible for unit testing of his or her components.

Unit Testing

- Performed by a developer
- Specification testing
- Note that black box testing is not unit testing
- Structural testing or white box testing
  - control flow graph
  - Method flow graph
  - Class flow graph
System Integrator

- Is responsible for designing integration build plan and performing incremental integration & perhaps regression test.

Implementation Workflow Activities and Workers

- Architect
- System Integrator
- Component Engineer

Flow:
- Architectural Implementation
- Integrate System
- Implement a Class
- Perform a Unit Test
The Test Workflow

CSC 532: Advanced Software Engineering

Main Goals

• To ensure that the system offers a high degree of quality before it’s delivered to customers
• Result: Test Model
• Use-case oriented testing
• Quality Gate
  – Entrance Criteria
  – Exit Criteria
Artifacts

- Test Case
  - Black-box testing-use cases
  - White-box testing-use case realizations-design
  - Integration testing-system level
- Test Procedure
  How to perform

Sample of test cases

d) Testing Cases & Testing Procedure

Test Case 1: Register
- Profile: Admin/User/Patient
- Test Data: Registrar
- Enhancement: EKG Service
- Pre-requisite: User should be accessible to the sign up form.
  Test Description: Steps:
  1) Users should fill the form with correct details.
  2) Users should provide the password and agree to the terms and conditions to create an account.
- Expected Result: User’s account should be created on clicking the register button and should be able to sign in with username and password provided while registering.
- Actual Result:
- Comments:
- Tested By:

Test Case 2: Login
- Profile: Admin/User/Patient
- Test Data: Login
- Enhancement: EKG Service
- Pre-requisite: User should have a login user id and password
  Test Description: Steps:
  1) User should be able to provide his login detail.
  2) User detail provided to login the application should be validated.
  3) User access profile should be based on his privileges.
Artifacts

- Test Components/tools
  Piece of code that automates all or parts of one or more test procedures
- Test Model – test cases, procedures and test components
- Test Plan - Describes plans including the resources
- Defect - Refers to any problem
- Test rest & Evaluation
  Evaluation of the results of a set of tests based on quality criteria

Sample of test reports/results

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Description</th>
<th>Release no</th>
<th>p/f</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Registration sunny day</td>
<td>1</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Registration with illegal data</td>
<td>1</td>
<td>F</td>
<td>It allows illegal data. Reported in TR001</td>
</tr>
<tr>
<td>3</td>
<td>Login</td>
<td>1</td>
<td>P</td>
<td></td>
</tr>
</tbody>
</table>
Sample of Entrance Criteria for system test

• All code must be passed unit test at 98%
• Code must be ensured no memory leak
• IT test must be passed
• All code must be IT passed in the change control

Sample of Exit Criteria for system test

• There must be 95% SVpass for a total test cases in the current release.
• No severity 1 opened change requests
• All code must be IT passed in the change control
• Mean Time To Failure must be greater than 2 days (non-functional requirements)
Workers

• Test Engineer/SVV
  - Planning Integration, system and regression testing
  - Selecting and describing test cases and corresponding test procedures
  - Ensuring the completeness and correctness
  - Evaluating the results of testing

• Component Engineer
  - Building one or more test components

• Integration Tester
  – Integration testing on each build of the system

• System Tester
  – System testing on each build of the system
Activities

• Plan Test
  - Developing a test plan for the given iteration
  
  Two Basic Strategies
  - conformance-directed testing (feature sufficiency)
  - fault-directed testing (fault efficiency)

• Execute test plans

• Important but most ignored
  – Performance and Reliability testing
  – E.g stress testing

Design Test

- Designing the various levels of tests
- Procedures to carry out those tests
- The test engineer(s) should design integration and system test cases with an eye toward reuse
- Reuse of test cases within test procedures

All of the test cases go into the test model
Activities (continued)

• Implement Test Tool
  - creating test components that automate test procedures (Component engineer)

• Perform Integration Test
  - performing manual and automated integration testing for each build of the system
  - integration testers (defects)

• Perform System Test
  - system tester (results of integration testing satisfy the quality goals)

• Evaluate Test
  - evaluating results of integration and system (use-cases)
  - comparing to appropriate quality goals
Key Measure

• Completeness of testing or Coverage
• Reliability
• Performance

Test Workflow Activities

Plan Test → Design Test → Implement Test

Evaluate Test → Perform Integration Test → Perform System Test
How to make decision: Readiness of your project

- Requirement & Test Plan
- Objective and Criteria
  - Entrance Criteria
  - Exit Criteria
- Quality Model and Measurement (Software Metrics)

Failure/Defect Severity Classification—System

- Example by Capability Impact excerpted from Musa 1998 book

<table>
<thead>
<tr>
<th>System capability impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic service interruption</td>
</tr>
<tr>
<td>Service degradation</td>
</tr>
<tr>
<td>Inconvenience, correction not deferrable</td>
</tr>
<tr>
<td>Minor tolerable effects, correction deferrable</td>
</tr>
</tbody>
</table>
Example of Acceptance Chart by FIO

RUP First half summary
End chapter Questions/Review

- What are main activities in this workflow?
- Describe what are necessary tools or processes in this workflow and how they can help growing maturity of, improving quality of the SW product?