

Systems Development and Implementation

Lesson I

Copyright 2008 John Blommers

Topics

- **Part 1 Lecture**
 - Introductions, Overview and Goals
 - Systems Development Life Cycle
 - Software Development Methodologies
 - Extreme Programming
 - RAD and JAD
- **Part 2 Group Activities**
 - Class agrees on systems development tasks for the groups
 - 5-person team formation (lead, recorder, analyst, SME roles)
 - Groups go through a development process
 - Groups capture their work
 - Groups prepare 10 slides
 - Groups conduct a dry run
- **Part 3 Presentations and Assessment**
 - Team presentations 10 minutes max
 - Lessons learned from each presentation

Part I

Lecture

Introductions

- Your name
- Company
- Background
- Expertise (important for team exercises)
- Equipment (laptop, PDA ...)

Course Objectives

From the syllabus

- The System Development Life Cycle
- Tools for Systems Development
- Software Development Methodologies
- Case Analysis
- Conceptual Modeling
- Customizing Applications vs. Reengineering Business Processes
- Making Outsourcing Decisions
- Benefits and Pitfalls of Open Source vs. Proprietary Systems
- Usability Testing
- Prototyping
- Project Scheduling

Weekly Topics

Visit the course page:

[http://faculty.washington.edu/blommers/
syllabus_sys_dev_imp.htm](http://faculty.washington.edu/blommers/syllabus_sys_dev_imp.htm)

The SDLC

Phase		Activity	What you have when you're done
1	Identify	Identify unmet business requirements. (ex JAD)	Business feature requirements, justification, broad scope. "What"
2	Initiate	Scope out the system and project plan	Costs, staffing, timetable (Gantt), critical success factors, locations effected, participants. "Who, when, how much"
3	Analyze	Requirements analysis	Comparison of alternative solutions. <u>Testing</u> requirements. "How, why, where"
4	Logical Design	Architecture - blocks of functions - data movement	Functional details about data, inputs, outputs, models, diagrams, charts. "How, what"
5	Physical Design	Technical details	Technical specifications about networks, storage, compute power. "What, how"
6	Prototype	<u>Testing</u> trial with partial functionality. (ex RAD)	Lessons learned about functionality, scalability, usability, reliability, fine-tune implementation schedule. <u>Test</u> results. Go To steps 1-5 as needed to apply the lessons.
7	Implement	Real hardware, software	System is fully operational. Awards & accolades
8	Maintain	Support, repair, enhance	This never ends until the system is obsolete, cannot be upgraded, and is replaced

Software Development Methodologies

- JAD
- RAD
- Distributed large projects (ex Linux, Koffice)
- Smaller projects (ex Marine Aquarium)
- Extreme Programming

JAD

- Joint Application Development for fact-finding
- Executive sponsor, a champion
- Pre-meeting to set goals for the JAD
- Schedule sessions
- Participants (session lead, vendors, developers, business analysts, IT ...)
- The session lead should be a neutral party
- Kickoff session
- JAD sessions
- Finalization, scribe the design document (powerpoint slides or better)

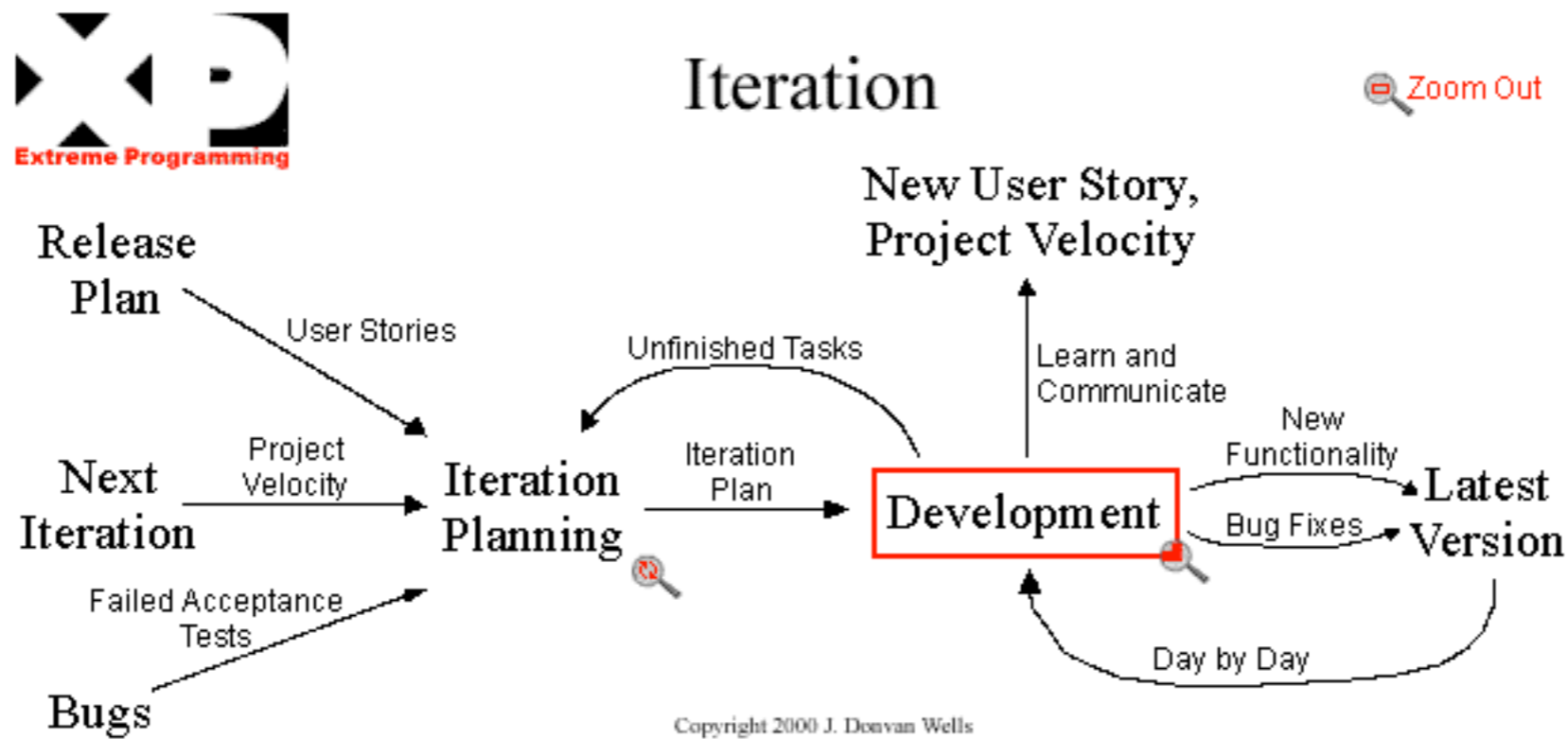
RAD

- Rapid Application Development
- Reduced SDLC with multiple iterations
- Provides “good enough” functionality first
- Increase features via downstream upgrades
- For example, create a Photoshop killer by writing a basic threaded image editor with a plug-in architecture, then rolling out plug-ins for the next few years.
- Not appropriate for complex mission critical systems

Extreme Programming

- Software engineering methodology
- Program tasks are divided up into modules
- 2-person teams develop code
- Risky project with dynamic requirements
- More adaptable than traditional methods
- <http://www.extremeprogramming.org/>

Extreme Programming



<http://www.extremeprogramming.org/map/project.html>

http://en.wikipedia.org/wiki/Extreme_Programming

Large Distributed Projects

- Build farms for operating systems
- Source Forge web site
- SubVersion (SVN)
- git (used for Linux kernel development)
- Distributed C Compiler (distcc)
- BitKeeper

Small Projects

```
#include <stdio.h>
main() {
printf("Hello world!\n");
}
```

- LAN workgroup file server + workstations
- Build on one workstation
- Concurrent Versioning System (CVS)

Project Compile Times

Apache	5 min
KDE	15 min
perl	15 min
Firefox	40 min
Xorg	48 min
gcc	2 hours
OpenOffice	5 hrs
Hello world!	0.116 seconds

<http://linuxreviews.org/gentoo/compiletimes/>

Examples

- Windows (1.0 ... 95 ..98 .. NT 2K XP Vista 2008)
- Adobe Photoshop (5.0 5.0...7.0, 7.01 ... CS2 CS3)
- Mac OS X (10.0, 10.1 10.2 10.3 10.4 10.5)
- Linux (smooth) (2.6.10.x 2.6.11 ...)
- Hello world (one time demo)

sp1 sp2 sp3

10.5 10.5.1 10.5.2

Part 2

Group Activity

Group Activity

- Workshop
- Class agrees on systems development tasks for the groups
- 5-person team formation (lead, recorder, analyst, SME roles)
- Groups go partly through a development process
 - SDLC early stages (repeated on next slide)
 - JAD session
 - S/W or other project chosen
 - **Objective is to collect requirements, pre-design docs**
- Groups capture their work (the scribe)
- Groups prepare 10 slides (10 slides x 1 slide/minute)
- Groups might conduct a dry run if time permits
- We may well run out of time today
- So we use time next week to finish up

SME = Subject Matter Expert

Phase		Activity	What you have when you're done
1	Identify	Identify unmet business requirements. (ex JAD)	Business feature requirements, justification, broad scope. "What"
2	Initiate	Scope out the system and project plan	Costs, staffing, timetable (Gantt), critical success factors, locations effected, participants. "Who, when, how much"
3	Analyze	Requirements analysis	Comparison of alternative solutions. <u>Testing</u> requirements. "How, why, where"
4	Logical Design	Architecture - blocks of functions - data movement	Functional details about data, inputs, outputs, models, diagrams, charts. "How, what"
5	Physical Design	Technical details	Technical specifications about networks, storage, compute power. "What, how"
6	Prototype	<u>Testing</u> trial with partial functionality. (ex RAD)	Lessons learned about functionality, scalability, usability, reliability, fine-tune implementation schedule. <u>Test</u> results. Go To steps 1-5 as needed to apply the lessons.
7	Implement	Real hardware, software	System is fully operational. Awards & accolades
8	Maintain	Support, repair, enhance	This never ends until the system is obsolete, cannot be upgraded, and is replaced

Some System Development Ideas

- We brainstorm this right in class
- Generate lots of system ideas first
- Each team picks one system to implement
- Try to avoid duplication (unique team systems)
- Starter ideas on the next slide
- This system may be used for the whole term
- But last term some students changed systems

Starter Ideas

1. Take apart a computer, catalog all the parts, and put it back together (forensics)
2. Develop a manufacturing plant for chairs
3. Develop the workflow for a standard computer forensic examination
4. Upgrade an existing SMB network to improve security (Pixel)
5. Design and build a low cost computer lab (LTSP, other)
6. Design and implement a Teledesic constellation (never built)
7. Upgrade an existing constellation such as Iridium
8. Migrate a fortune 100 company from IPv4 to IPv6
9. Spec, buy, and build an IPv6 testing lab
10. Spec design and build a 3-tier online bulletin board system (phpbb)
11. p2p application update system for a software product (BitTorrent)
12. Migrate a Windows shop to Linux (pick some migration)
13. Design and construct a new office complex for Apple Inc.
14. I want a tablet (?) a doctor can use to take patients notes in real time
15. We're going into the gaming console renting business
16. Let's set up a business to sell Linux computers
17. We're going to add email into our business
18. We're going to replace our old email system with a new one
19. We're going to create a hard drive repair and replace service
20. I want to develop and sell program to deblur people's photos
21. Let's develop the ideal PDA or mobile computer or a smartphone
22. We're to implement a Network Management System in our company
23. Building the One Laptop Per Child computer

Stuck Finding Requirements?

- Ask who, what, where, when, why how
- Probe other dimensions or areas
- Ask if there are requirements in these areas:
 - performance
 - security
 - scalability
 - manageability
 - maintainability
 - reliability
 - cost
 - supportability
 - expandability

More Requirements Areas

- performance
- security
- scalability
- manageability
- maintainability
- reliability
- cost
- risk
- time table
- supportability
- expandability
- power
- air
- light
- heat
- operations
- funding
- staffing
- quality
- legal
- education
- training
- organization
- environmental
- documentation
- interface
- storage
- transmission
- connectivity
- network
- hardware
- software
- apps
- servers
- drivers
- licenses
- support functions
- weight
- life span
- precision
- repeatability
- spectrum
- language
- skill

Decomposition

who, what,
where, when,
why, how

Brainstorm
to generate
more areas

Do we have testing requirements in these areas?

Test all aspects of operations

Part 3

Class Presentations and Assessment

Part 3

Presentations, Assessments

- Team presentations 10 minutes max
- Lessons learned from each presentation:
 - What went well?
 - What was hard?
 - Is the solution complete?
 - Did they understand the problem?
 - Is the solution too complex?
 - Is the solution correct?
 - Does the solution assume too much?
 - How confident the teams are in their work