INTERNET & WEB APPLICATION DEVELOPMENT SWE 444

Fall Semester 2008-2009 (081)

Module 6: Web Engineering Fundamentals

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives/Outline

- Objectives
  - Understand the role of web engineering
  - Learn a systematic process for web applications development
- Outline
  - Introduction
  - Requirements Analysis
  - Web Modeling
  - Web Design ans Architectures
  - Web Accessibility

Resources

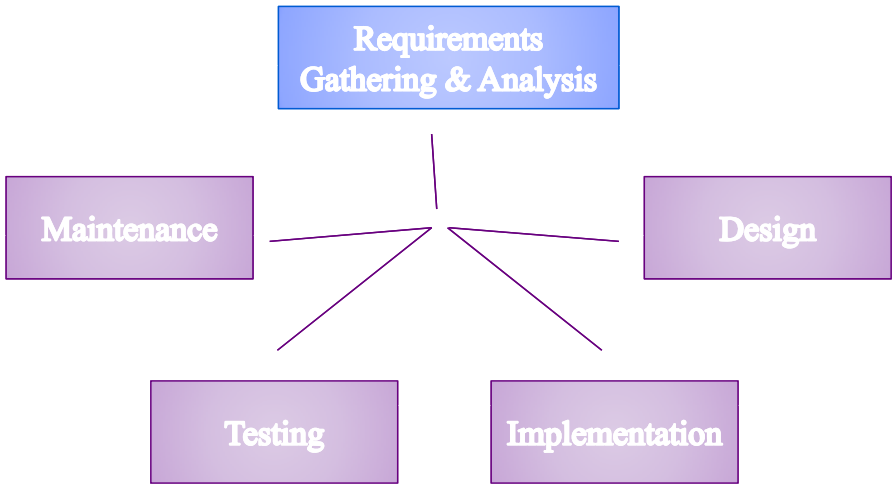
- Books
  - Roger S. Pressman, David Lowe (2009). *Web Engineering: A Practitioner's Approach*, McGraw-Hill. http://highered.mcgraw-hill.com/sites/0073523291/
  - Roger Pressman (2005). *Software Engineering: A Practitioner's Approach*, 6/e, McGraw-Hill Higher Education. Chapters 16-20. http://highered.mcgraw-hill.com/sites/0072853182/information_center_view0/
  - G. Kappel, B. Pröll, S. Reich, and W. Retschitzegger (eds), *Web Engineering - The Discipline of Systematic Development of Web Applications*, John Wiley & Sons, 2006. http://www.web-engineering.at/eng/
- Online material
  - INFSCI 2955: Web Engineering
  - Department of Information Science and Telecommunications, University of Pittsburgh http://www.sis.pitt.edu/~jgrady/

6.2 REQUIREMENTS ENGINEERING

Outline

- Introduction to Requirements Engineering
- Fundamentals
- Specifics in Web Engineering
- Principles
- Adapting traditional Requirements Engineering to Web applications

Summary – Web Engineering



Introduction

- Requirements Engineering (RE)
  - the principles, methods, & tools for eliciting, describing, validating, and managing project goals and needs.
- Given the complexity of Web apps, RE is a critical initial stage, but often poorly executed.
- What are the consequences?
  - Inadequate software architectures
  - "Unforeseen" problems
    - Budget overruns
    - Production delays
    - "That's not what I asked for"
  - Low user acceptance

Why Define Requirements?

- The authors build their case:
  - Bell & Thayer (1976) – Requirements don't define themselves.
  - Boehm (1981) – Removal of mistakes post hoc is up to *200 times* more costly.
  - The Standish Group (1994) – 30% of project fail before completion & almost half do not meet customer requirements
    - Unclear objectives, unrealistic schedules & expectations, poor user participation

Fundamentals of RE

- Identify and involve (if possible) the *stakeholders*
  - Those that directly influence the requirements
  - E.g. customers, users, developers
- What are their expectations?
  - May be misaligned or in conflict.
  - May be too narrowly focused or unrealistic.
- Already, one can see RE as more of an art than a science.

Fundamentals of RE (cont.)

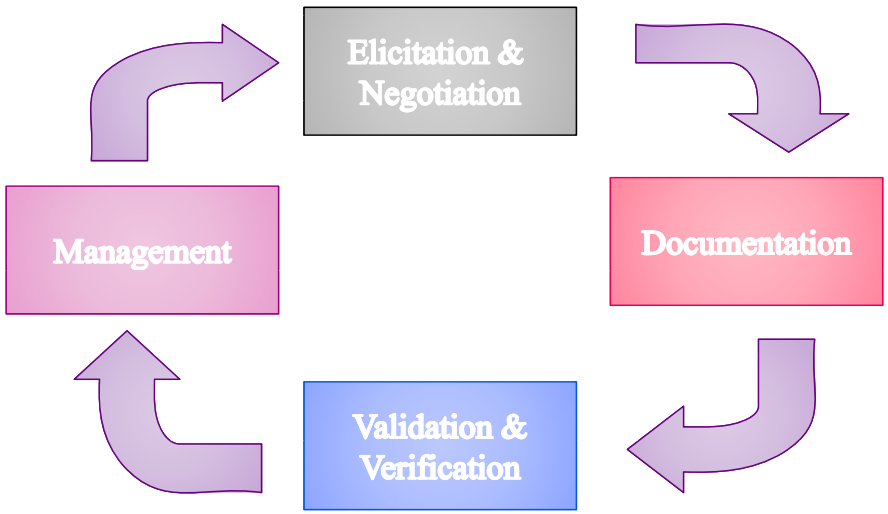
- IEEE 601.12 definition of *requirement:*
  - 1) Solves a user's problem
  - 2) Must be met or possessed by the system to satisfy a formal agreement
  - 3) Documented representation of conditions in 1 and 2
- Keys to requirement definition:
  - Negotiation
  - Scenario-based discovery
  - Clear definition of context and constraints

Fundamentals of RE (cont.)

- Objectives, objectives, objectives
  - Advertising
  - Customer service
  - Business transactions
- Audience, audience, audience
  - The designer is not the audience
  - Audience segmentation
  - User interviews and testing
- What about the Competition?
  - Other web sites
  - Other forms of advertising and transactions

Example: SIS Website

Summary - RE Activities

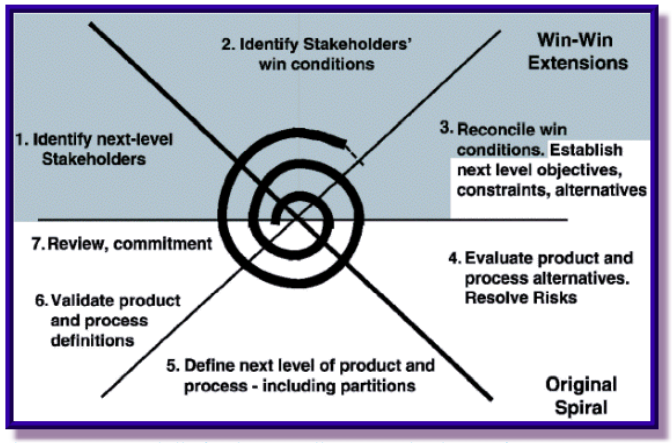


Specifics in Web Engineering

- Is RE for the Web really that different than RE for conventional software?
- Some would argue "no", but many aspects of Web applications suggest otherwise
- Distinguishing characteristics
  - Multidisciplinary
  - Unavailability of stakeholders
  - Rapidly changing requirements & constraints
  - Unpredictable operational environment
  - Integration of legacy systems
    - Constrained by existing system and budget
  - Quality aspects
    - User interface quality
    - Content quality
  - Developer inexperience
  - Firm delivery dates

Principles for RE

- Inspired by the win-win spiral model (Boehm, 1996) and favored for large, expensive, and complicated projects.



Source: http://www.stsc.hill.af.mil/Crosstalk/2001/12/boehm3.gif
More details: http://sunset.usc.edu/research/WINWIN/winwinspiral.html, http://sunset.usc.edu/~aegyed/publications/Using_the_WinWin_Spiral_Model-A_Case_Study.pdf

Principles for RE (cont.)

- Understanding the system context
  - Web apps are always a component of a larger entity
  - Why do we need the system?
  - How will people use it?
- Involving the stakeholders
  - Get all groups involved.
  - Balance – one group's gain should not come at the expense of another.
  - Repeat the process of identifying, understanding and negotiating.

Principles for RE (cont.)

- Iteratively define requirements
  - Requirements need to be consistent with other system aspects (UI, content, test cases)
  - Start with key requirements at a high level; basis for:
    - Feasible architectures
    - Key system use cases
    - Initial plans for the project
  - As the project progresses, requirements can become more concrete.

Principles for RE (cont.)

- Focusing on the System Architecture
  - The "solution space" – existing technologies & legacy systems – defines the "problem space."
  - The architecture *must* be considered in the elicitation stage.
  - Refine requirements and architecture iteratively with increasing level of detail.

Principles for RE (cont.)

- Risk Orientation
  - Risk management is at the heart of the analysis process.
  - What are the greatest risks?
    - Integration issues w/ legacy systems
    - Expected vs. actual system quality
    - Inexperience of developers
  - How to mitigate risks?
    - Prototyping (avoid IKIWISI)
    - Show changes to customer iteratively
    - Integrate existing systems sooner than later

Adapting RE to Web Applications

- There isn't one single "right way" to RE among the many methods, techniques, tools, etc. available.
- For your Web application project, ask the following questions:
  - What are the critical requirements?
  - How should requirements be documented?
  - What tools should be use, if any?

Adapting – Requirement Types

- Taxonomies (e.g. IEEE 830) exist that describe *functional* and *non-functional* requirements.
  - *Functional* – describes the capability's purpose (e.g., the ability to transfer money between user accounts.)
  - *Non-functional* – describes the capability's properties (e.g., the system can handle 1,000 concurrent users)

Adapting – Requirement Types

- Non-functional requirement types
  - Content
  - Quality (6 Types)
    - Functionality
    - Reliability
    - Usability
    - Efficiency
    - Maintainability
    - Portability

Adapting – Requirement Types

- Non-functional requirement types (continued)
  - System Environment
  - User Interface
    - Self-explanatory & intuitive
    - Usage-centered design
  - Evolution
  - Project Constraints

Adapting – Documentation

- 4 categories of notation
  - *Stories* – Plain-language scenarios; understandable to non-technical persons.
  - *Itemized Requirements* – Plain-language lists of requirements
  - *Formatted Requirements* – Accurately-defined, but allow for plain-language descriptions
    - Ex. Use case scenarios in UML
  - *Formal Specifications* – Expressed in formal syntax & semantics; rarely used in Web applications.

Adapting – Documentation

- So, what's best for a Web development project?
  - Low to medium accuracy is suitable for Web apps; formal specifications very rarely required.
  - Keep elicitation and management of requirements low.
  - Scalability is (most likely) important.
  - Formatted requirements (i.e. use cases) are heavily used.

Adapting – Tools

- Requirements Elicitation
  - EasyWinWin (the author's software)
  - Book: *Getting to Yes: Negotiating an Agreement Without Giving in* by Fisher, Ury, Patton (1994)
- Requirements Validation
  - Online feedback (Web surveys)
- Requirements Management
  - Database system – traceability, versioning

Challenges with Stakeholders

- McConnell (1996)
  - Users don't know what they want.
  - Lack of commitment.
  - Ever-expanding requirements.
  - Communication delays.
  - Users don't take part in reviews.
  - Users don't understand the technology.
  - Users don't understand the process.

Challenges with Developers

- Users and engineers/developers speak different "languages".
- The tendency to "shoe-horn" the requirements into an existing model
  - Saves time for developers, but results may not meet user's needs.
- Engineers & developers are also asked to do RE, but sometimes lack people skills and domain knowledge

Q & A

?