Web Engineering: Introduction

The University of Aizu
Quarter 2, AY 2018

Outline

- Administration issues
- Course goal
- Tentative Schedule
- Bibliography
- Motivation
- Introduction to Web engineering

Administration

- Course home page:
  http://web-int.u-aizu.ac.jp/~vkluev/courses/webengineering/
  - TSI: to upload your solutions, to get personal score, etc. http://cseng.u-aizu.ac.jp:4080/WE18STD5/tsi/
    http://cseng.u-aizu.ac.jp:4080/WE18STD6/tsi/
- Instructors:
  - Prof. Vitaly Klyuev
  - E-mail: vkluev@u-aizu.ac.jp
  - Office: 342C
- Class materials and assignments will be put on the Web every Friday
- Please print them and take to our class
- Basic for grades:
  - Assignments 40%,
  - Quizzes during lectures 25%,
  - Final examination 35%.

Course Goal

- The main goal of the course is to provide students with a solid understanding of a pragmatic process for engineering Web-based systems and applications.

Prerequisites

- You should recover materials from the following courses
  - Java Programming 1
    - object-oriented concepts
  - Java Programming 2
    - Internet programming
  - Software Engineering 1
    - Software engineering process
  - Web Programming
    - Web technologies

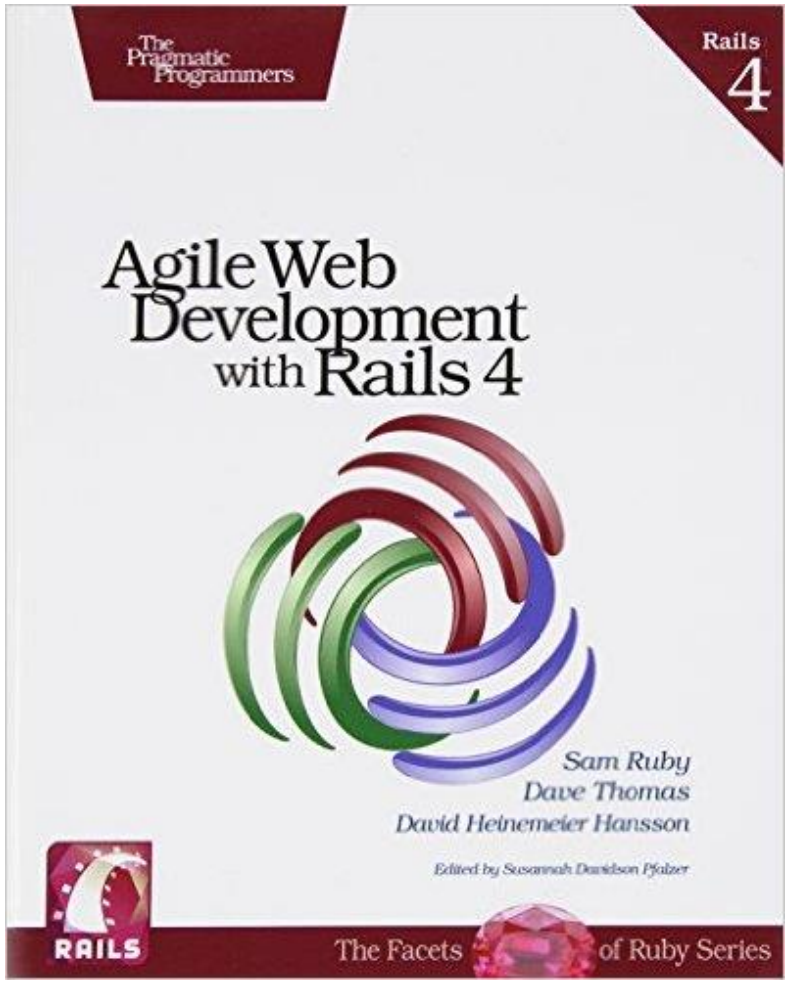
Tentative Schedule

- Introduction
- A Web engineering process
- Communication and planning
- Hello Rails!
- Ruby from other Languages
- Web application architecture
- Building an Application
- Finding your way around Rails
- Tasks: Validation, Testing, Enhancing, etc.
- Universal Design for Web Applications: Google's Approach pps

Literature

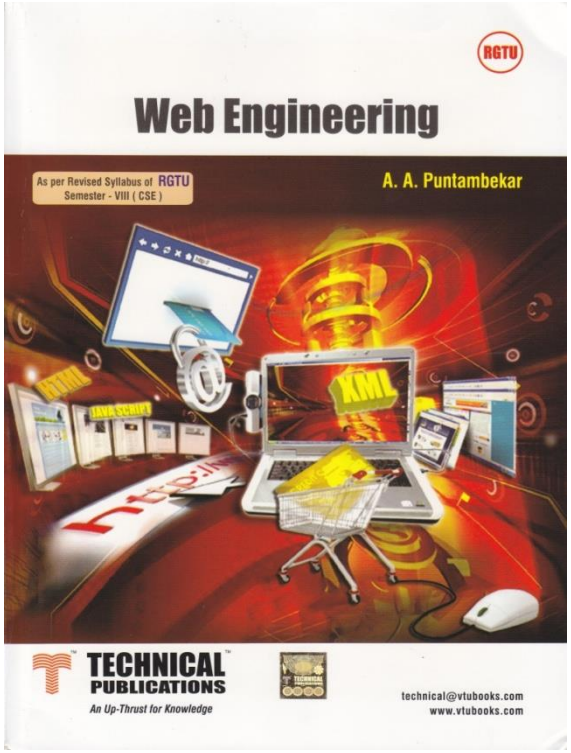
- Agile Web Development with Rails 4 (1st edition) by Sam Ruby, Dave Thomas and Devid Hansson, The Pragmatic Bookshelf, 2013.
- Web resources:

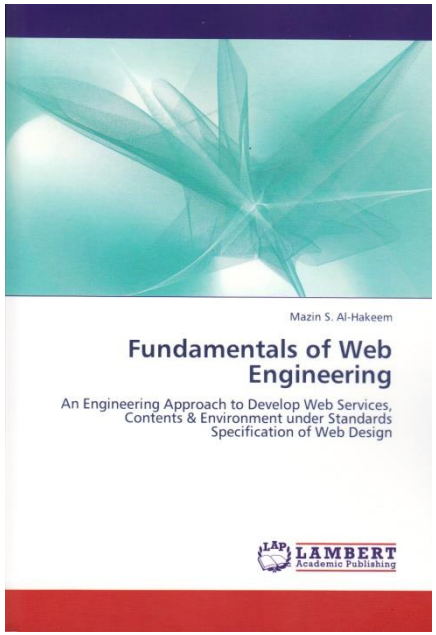
http://www.buildingwebapps.com/



Literature

- Puntambekar A.A., Web Engineering, Technical Publications, 2012.
- Mazin S. Al-Hakeem, Fundamentals of Web Engineering: An Engineering Approach to Develop Web Services, Contents and Environment under Standards Specification of Web Design, Lambert Academic Publishing, 2012.





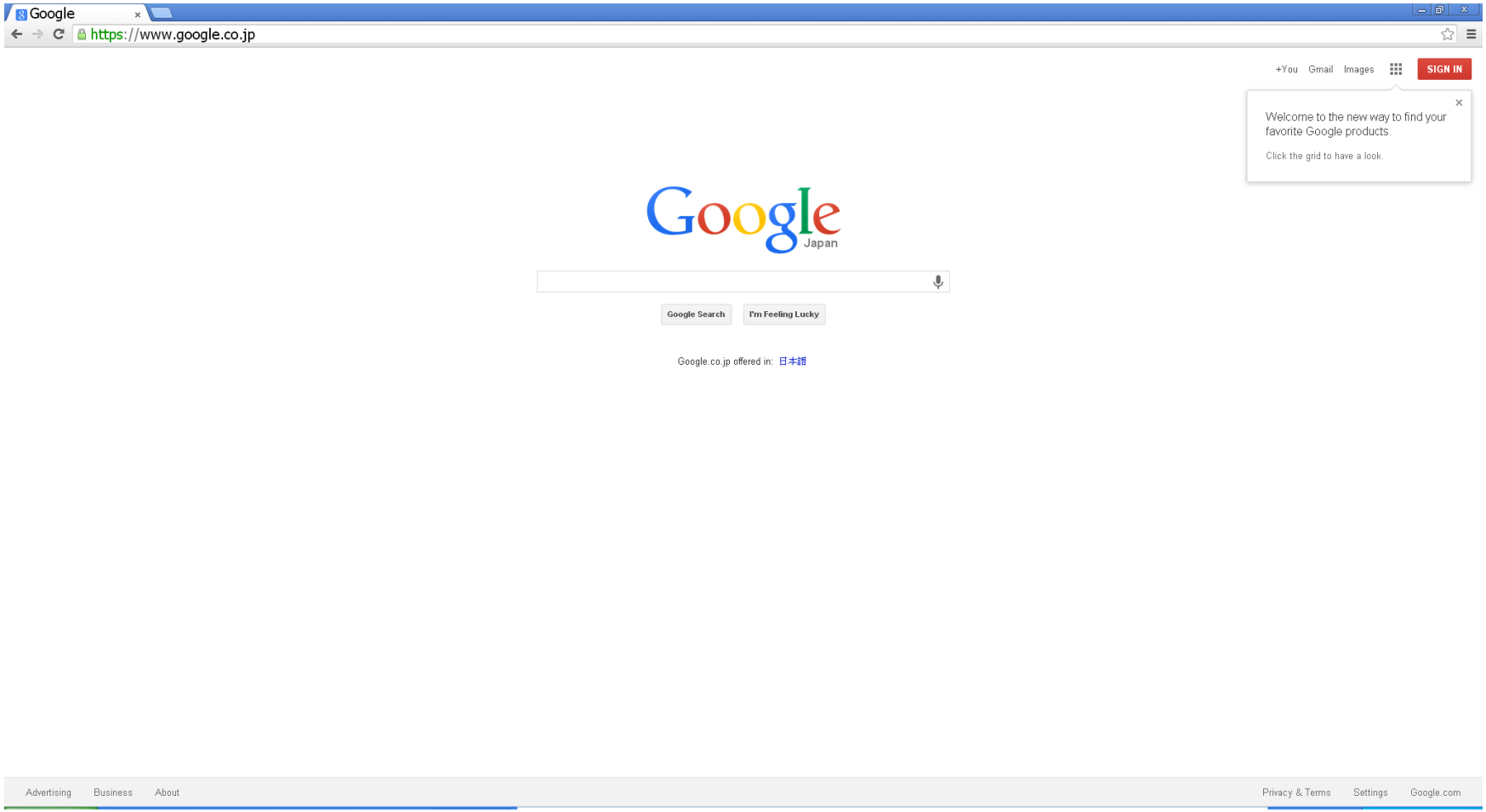
Literature

- In our class, we will use materials of the world leading universities in computer science
  - Paderborn University, Germany
  - University of Pittsburg, USA
  - University of Innsbruck, Austria

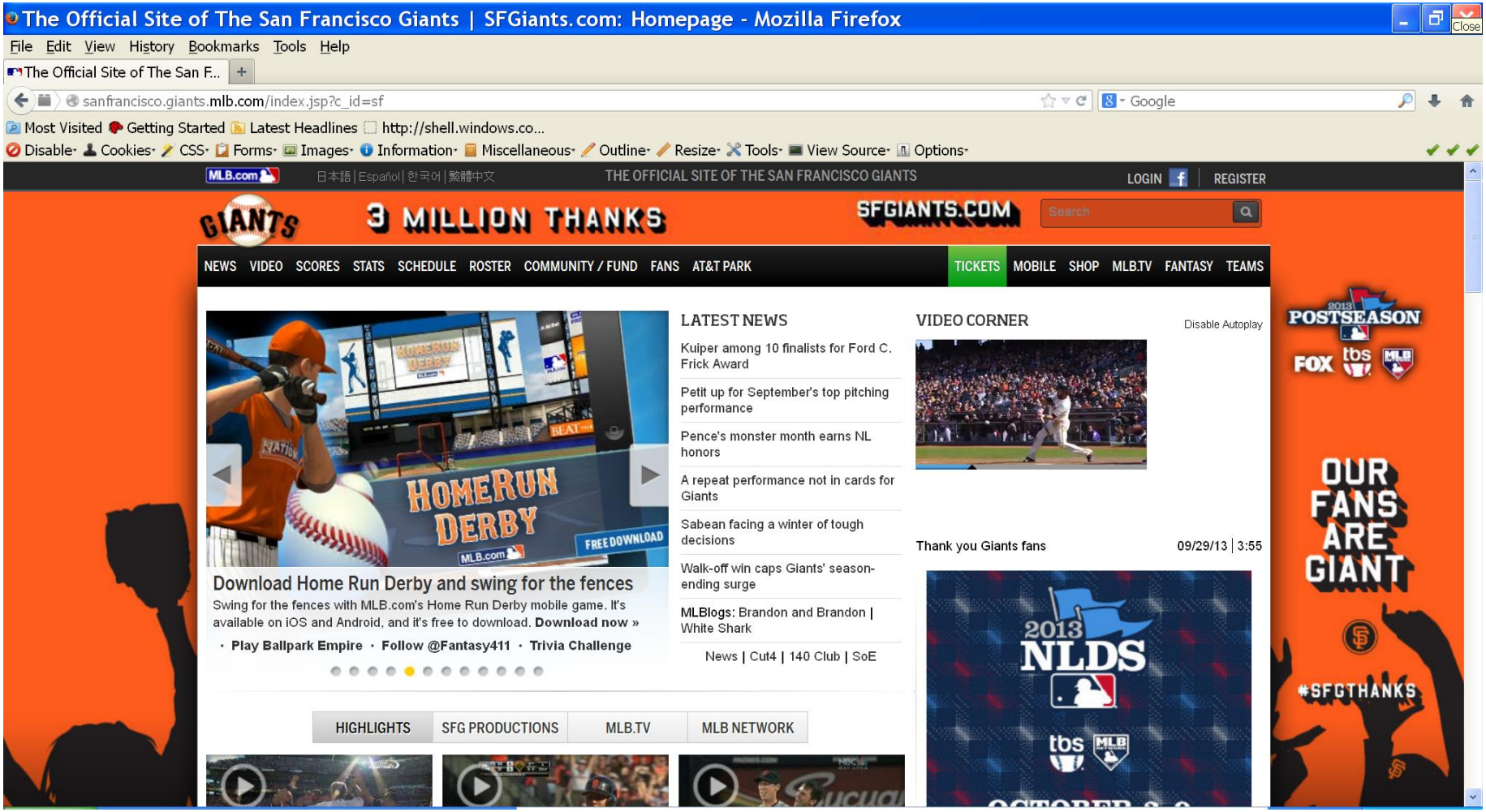
Motivation

- The World Wide Web is omnipresent!
- Why?
  - global and permanent availability
  - comfortable and uniform access
  - anyone can produce and publish contents
- Business (small, medium and big) goes on the Web
- After graduation, you will definitely deal with creating Web applications.

Example: Search Engine



Example: Information

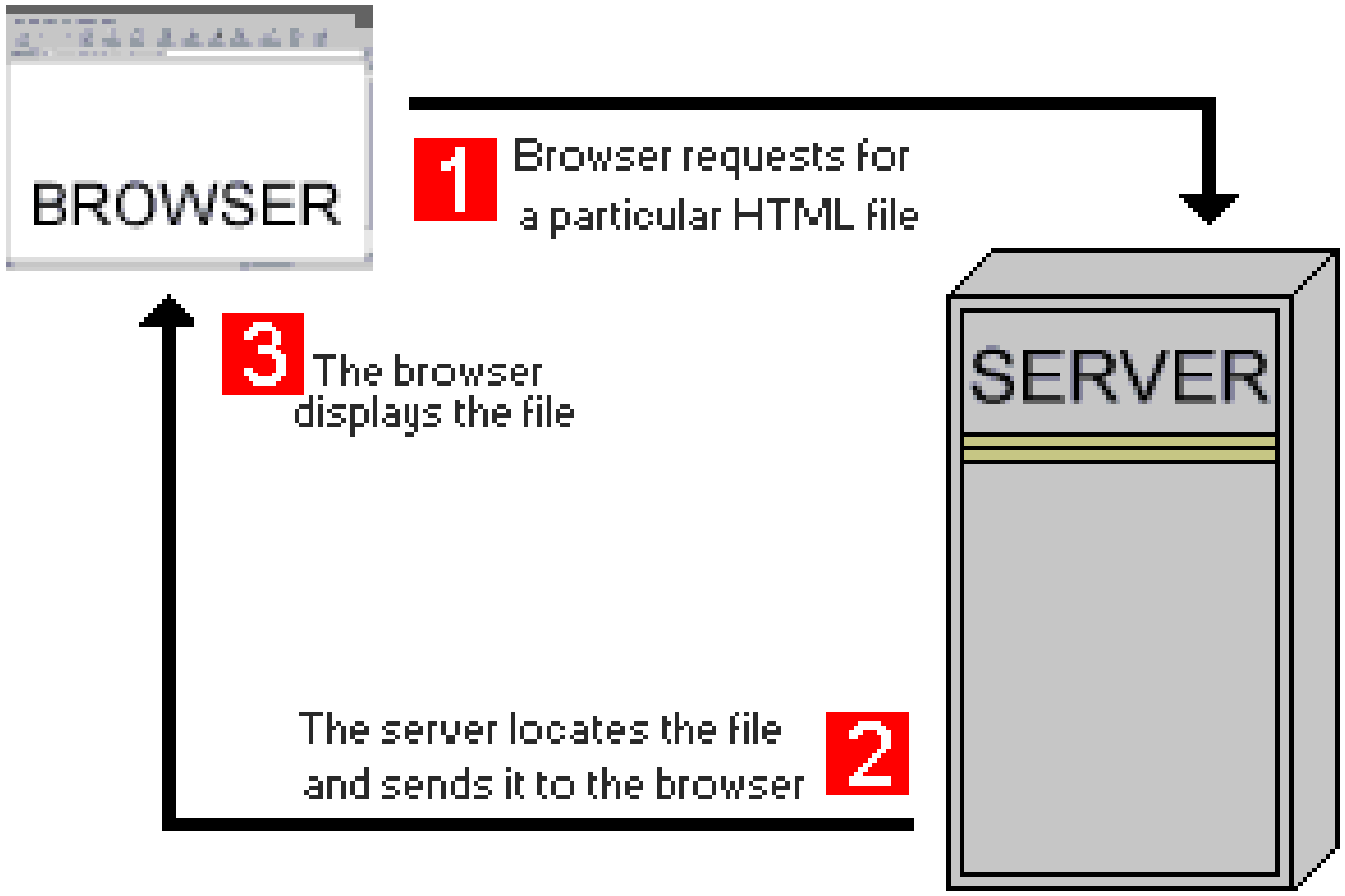


Example: www.japanrail.com/

Basic paradigms

- Hypertext + Internet
- Hypertext: textual documents together with the ability to interconnect documents by links between them as part of the document contents
- HTML: HyperText Markup Language
- HTTP: HyperText Transfer Protocol

Conceptual Architecture



History of the Web

- 1969: ARPA (Advanced Research Projects Agency)
  - First small network: Stanford Research Institute, UCLA, UC Santa Barbara, Univ. of Utah
  - TCP (Transmission Control Protocol)
  - IP (Internet Protocol)
- 1972: Telnet protocol
- 1973: SMTP (Simple Mail Transfer Protocol)
- 1973: FTP (File Transfer Protocol)
- 1989: T. Berners-Lee et al.: Word Wide Web
- 1994: W3C (World Wide Web Consortium)
- 1996: HTTP (HyperText Transfer Protocol)

World Wide Web Consortium (W3C)

- International consortium where member organizations, a full-time staff, and the public work together to develop Web standards
- http://www.w3.org
- W3C's mission:
  - to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web.

Web-Based Systems

- In the early days of the Web, we built systems using informality, urgency, intuition, and art
  - *Informality* leads to an easy work environment—one in which you can do your own thing.
  - *Urgency* leads to action and rapid decision making.
  - *Intuition* is an intangible quality that enables you to "feel" your way through complex situations.
  - *Art* leads to aesthetic form and function—to something that pleases those who encounter it.
- Problem is— this approach can and often does lead to problems.

Why does the old-school Approach lead to Problems?

- As WebApps become larger and more complex,
  - Informality remains, but some degree of requirements gathering and planning are necessary
  - Urgency remains, but it must be tempered by a recognition that decisions may have broad consequences
  - Intuition remains, but it must be augmented by proven management and technical patterns
  - Art remains, but it must be complemented with solid design
- Bottom line—we must adapt the old-school approach to the realities of the current Internet

WebApps

- The term *Web application* (WebApp) encompasses:
  - everything from a simple Web page that might help a consumer compute an automobile lease payment to a comprehensive website that provides complete travel services for business people and vacationers.
  - Included within this category are complete websites, specialized functionality within websites, and information-processing applications that reside on the Internet or on an Intranet or Extranet.

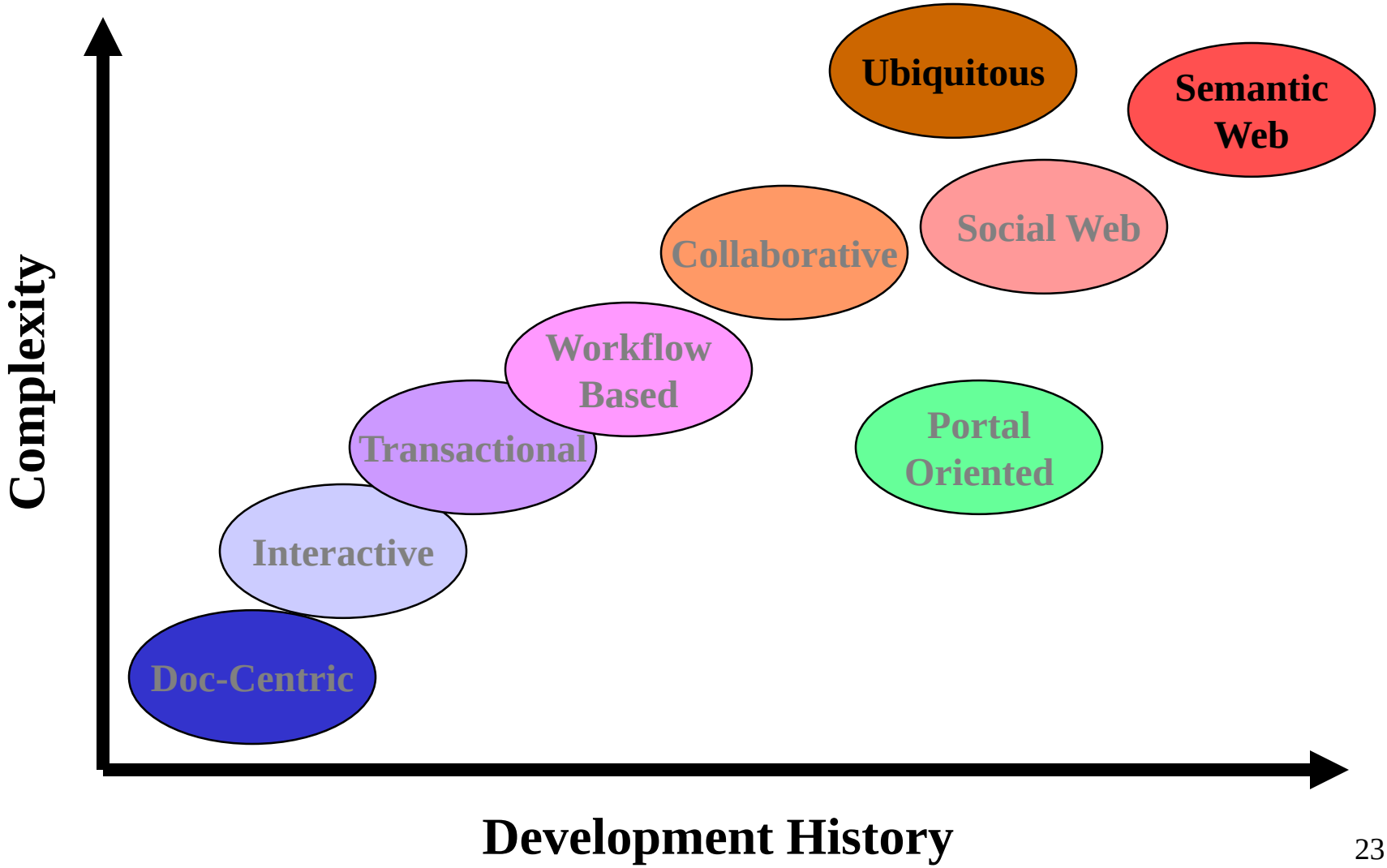
WebApps Attributes

- Network intensiveness: Every WebApp resides on a network and serves the needs of clients.
- Concurrency: A large number of users may access the WebApp at one time.
- Unpredictable load: The number of users of the WebApp may change unpredictable.
- Performance: If a WebApp user has to wait too long, he may decide to go elsewhere.
- Availability: Users of popular WebApps demand access on a "24/7/365" basis.
- Data driven: The key function of many WebApps is to use hypermedia to present text, graphics, audio, and video content.

WebApps Attributes

- Content sensitive: The quality and aesthetic nature of content remains an important factor of the quality of a WebApp.
- Continuous evolution: Content of WebApps may be updated on minute-by-minute schedule.
- Immediacy: The strong need to get software to market quickly.
- Security: To protect sensetive content and provide secure data transmission.
- Aesthetics: When an a WebApp is designed to market or sell progucts or ideas or provide services, aesthetics may equal to technical design.

Categories of Web Applications



Document-Centric Web sites

- Precursors to Web applications
- Static HTML documents
- Manual updates
- Pros
  - Simple, stable, short response times
- Cons
  - High management costs for frequent updates & large collections
  - More prone to inconsistent/redundant info

Interactive & Transactional

- The Common Gateway Interface
  - http://hoohoo.ncsa.uiuc.edu/cgi/interface.html
- Simple interactivity
- Dynamic page creation
- Content updates -> Transactions
  - Decentralized
  - Database connectivity
  - Increased complexity

Workflow-Based Applications

- Designed to handle business processes across departments, organizations & enterprises
- Business logic defines the structure
- The role of Web services
  - Interoperability
  - Loosely-coupled
  - Standards-based
- Examples: B2B (Business-to-business: commerce transactions between businesses) & e-Government (http://ec.europa.eu/idabc/)
- High complexity; autonomous entities

Collaborative & Social Web

- Unstructured, cooperative environments
- Interpersonal communication is paramount
- Classic example: Wikis
- The Social Web
  - Anonymity traditionally characterized WWW
  - Moving towards *communities of interest*
  - *Examples:* Blogs, collaborative filtering systems, social bookmarking (e.g., del.icio.us)
  - Integration with other forms of web applications (e..g, NetFlix)

Portal-Oriented

- Single points-of-entry to heterogeneous information
  - Yahoo!, AOL.com, my.pitt.edu
- Specialized portals
  - Business portals (e.g., employee intranet)
  - Marketplace portals (horizontal & vertical)
  - Community portals (targeted groups)

Ubiquitous

- Customized services delivered anywhere via multiple devices
- HCI (human-computer interaction) is critical
  - Limitations of devices (screen size, bandwidth?)
  - Context of use
- Still an emerging field; most devices have single focus:
  - Personalization
  - Location-aware
  - Multi-platform delivery

Semantic Web

- Berners-Lee: Information on the Web should be readable to machines, as well as humans.
- Using metadata and ontologies to facilitate knowledge management across the WWW.
  - Ontology is a formal domain model that represents the objects in the domain, attributes of those objects, and relationships between objects.

Web Engineering

- We define Web Engineering this way:
  - *an agile, yet disciplined framework for building industry-quality Web Applications.*
- We must understand the meaning of:
  - Agile
  - Disciplined framework
  - Industry quality

Why Agility?

- Business strategies and rules change rapidly
- Management demands are very quick and changeable (even when such demands are completely unreasonable)
- Stakeholders (anyone who has a take in the successful outcome of the project: business managers, end users, Web engineers, support people) often don't understand the consequences of the Web and keep changing their mind even as they demand rapid delivery
- An agile approach (to support changes intelligently) helps cope with this fluidity and uncertainty.

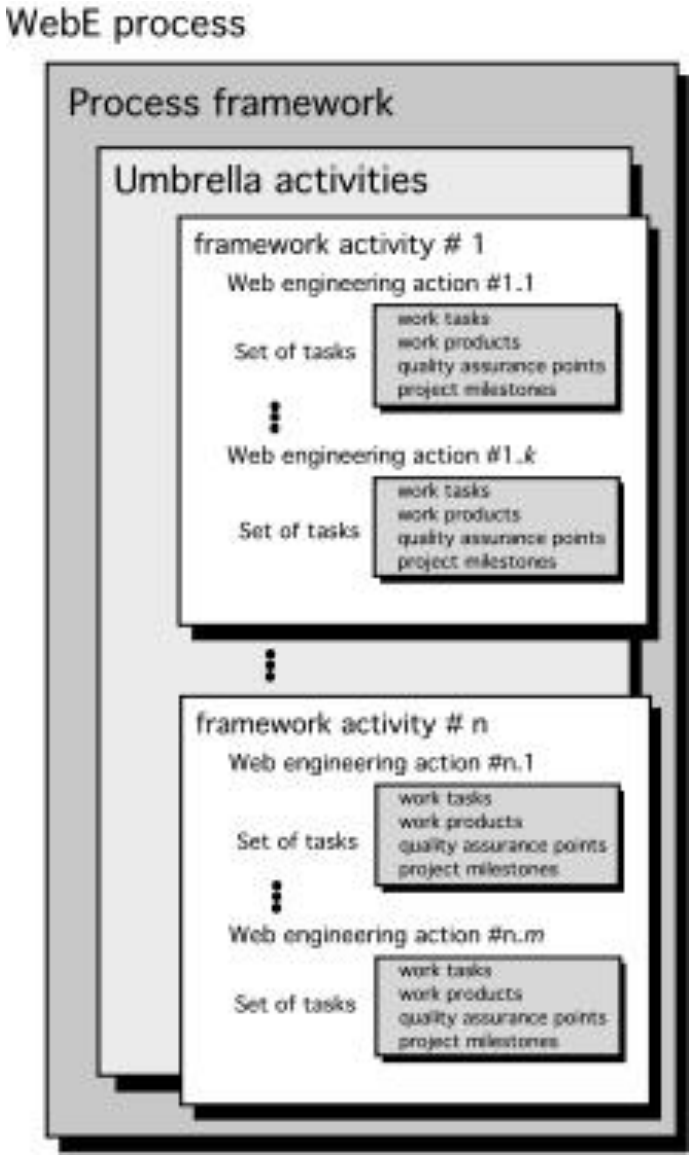
What is an Agile Process?

- Agile Web engineering combines a philosophy and a set of development guidelines. The philosophy encourages:
  - customer satisfaction
  - early incremental delivery of the WebApp
  - small, highly motivated project teams
  - informal methods
  - minimal work products
  - overall development simplicity.
- An agile process stresses delivery over analysis and design (although these activities are not discouraged), and active and continuous communication between developers and customers.

What is a WebE Framework?

- A framework is a set of activities that will *always* be performed for *every* Web engineering project – though the nature of the activities might vary to suit the project.
- Each framework activity is composed of a set of actions
- Actions encompass
  - work tasks
  - work products
  - quality assurance points, and
  - project milestones
- A framework also has a set of "umbrella activities"

A Generic Framework



The WebE Framework: Activities

- **Communication.** Involves heavy interaction and collaboration with the customer (and other stakeholders) and encompasses requirements gathering and other related activities.
- **Planning.** Establishes an incremental plan for the WebE work.
- **Modeling.** Encompasses the creation of models that assist the developer and the customer to better understand WebApp requirements and the design
- **Construction.** Combines both the generation of HTML, XML, Java, and similar code with testing that is required to uncover errors in the code.
- **Deployment.** Delivers a WebApp increment to the customer who evaluates it and provides feedback based on the evaluation.

Adapting the Framework

- Adapt to the problem, to the project, to the team, and to the organizational culture
  - And continue to adapt throughout the project as circumstances change!

Underlying Agility Principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness continuous change for the customer's competitive advantage.
- Deliver working software increments frequently, from as often as every few days to every few months, with a preference to the shorter timescales.

Underlying Agility Principles

- Business people and developers must work together daily throughout the project.
- Build projects around motivated people. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Underlying Agility Principles

- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.

Underlying Agility Principles

- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The Influence of Software Engineering

- Quality: foster a continuous process improvement culture
- Process: the glue that holds the technology layers together
- Methods: provide the technical how-to's
- Tools: support for the process and the methods

WebE Methods

- Communication methods: To support communication between Web engineers and all other stakeholders
- Requirements analysis methods: To provide a basis for understanding the content delivered by a WebApp, functions provided for the end users, etc.
- Design methods: To address WebApp content, application and information architecture, interface design and navigational structure
- Construction methods: To apply tools and technology to the creation of WebApp content and functionality
- Testing methods: To incorporate a wide array of testing techniques that address component-level and architectural issues.

What about Tools and Technology?

- Tools and technology are very important, but they'll work well only if they're used within the context of an agile framework for Web engineering and in conjunction with proven methods for understanding the problem, designing a solution, and testing it thoroughly.

WebE Best Practices

- Take the time to understand business needs and product objectives, even if the details of the WebApp are vague.
- Describe how users will interact with the WebApp using a scenario-based approach.
- *Always* develop a project plan, even if it's very brief.
- Spend some time modeling what it is that you're going to build.
- Review the models for consistency and quality.
- Use tools and technology that enable you to construct the system with as many reusable components as possible.
- Don't reinvent when you can reuse.
- Don't rely on early users to debug the WebApp—design and use comprehensive tests before releasing the system.