Introduction to Web Engineering SENG2050/6050

Web Engineering

Web Application

- A website where the users' input affects the state of the business
- A system that utilises W3C standards and technologies to deliver web-specific resources to clients (typically) through a web browser
- Traditionally run as a client-server application where the client (a web browser) provides the user interface

Web Engineering

- *"Web engineering is the establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high-quality Web-based systems and applications."* – Murugesan et al. (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.325.9486&rep=rep1&type=pdf)
- An extension of Software Engineering to web applications.
- The process of applying repeatable best practices from software engineering to the development of large complex Web-based systems.

Web Engineering

- Application development on the Web remains largely *ad hoc*.
  - Spontaneous, one-time events
  - Individual experience
  - Little or no documentation for code/design
  - Short timeframes
- Short-term savings lead to long-term problems in operation, maintenance, usability, etc.

Web Engineering

- Root causes of poor design
  - Development as an authoring activity
  - Development is “easy”
  - Techniques that *should not* be used are misapplied.
  - Techniques that *should* be used are *not*.
- Particularly alarming given…
  - Most projects are now Web-based
  - More “mission-critical” apps moving to the Web

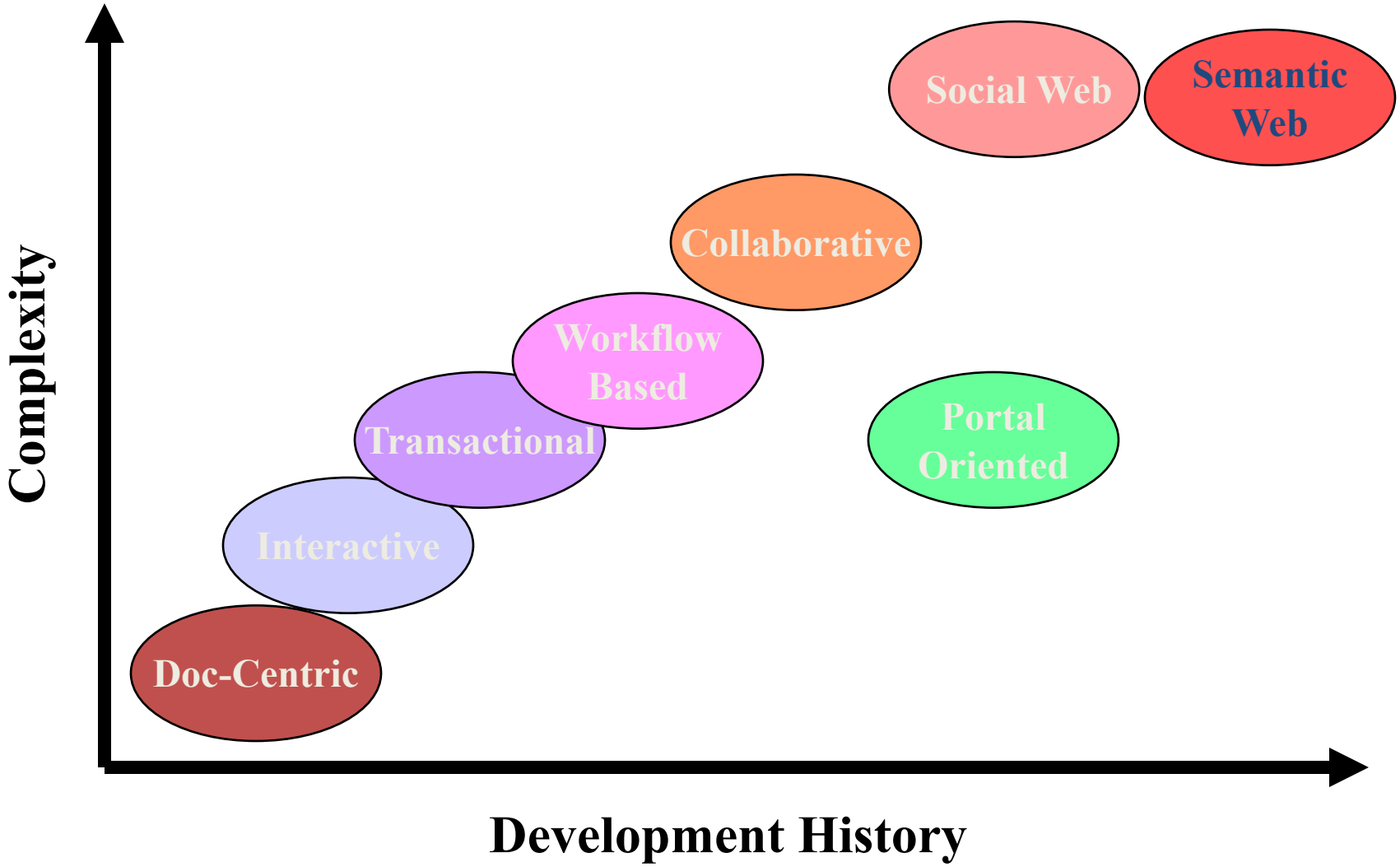
Web Engineering

- Top project pitfalls (Cutter, 2000)
  - 84% - Failure to meet business objectives
  - 79% - Project schedule delays
  - 63% - Budget overrun
  - 53% - Lack of functionality
  - 52% - Poor quality
- Chaos Report (The Standish Group, 2014)
  https://www.projectsmart.co.uk/white-papers/chaos-report.pdf
  - 31.1% - Projects will be cancelled before they are finished
  - 52.7% - Projects will cost 189% of their original estimates
  - 16.2% - Projects that are completed on-time and on-budget

Web Engineering

- The solution:
  - Clearly defined goals & objectives
  - Systematic, phased development
  - Careful planning
  - Iterative & continuous auditing of the entire process

Categories of Web Applications



Semantic Web

- Berners-Lee: Information on the Web should be readable to machines, as well as humans.
- Using metadata and ontologies to facilitate a "Web of Data"
  - Like a global database
- Is the Semantic Web even possible?
  - Open question

Characteristics of Web Apps

- How do Web applications differ from traditional applications?
- 3 dimensions
  - Development
  - Deployment
  - Usage

Characteristics - Development

- Multidisciplinary development team
- Parallelism:
  - CSS, JavaScript, HTML, and business logic (Java) can all be developed at the same time
- Flexible:
  - Ease of deployment means it's easy to include end users in the development process
    - Constantly changing requirements

Characteristics - Deployment

- Deployment happens in a single place
  - Prototypes can be deployed easily
  - Easy to send a URL to an end user / tester
    - Feedback can be provided throughout the project's development
  - Fixes/changes can be demonstrated almost instantly (once developed)
- Often no control over the client software that is used
  - Client may be using out-of-date or non-compliant web browser

Characteristics - Usage

- Diverse set of users
  - You may not know the background knowledge of all users
  - Users could be situated in different physical locations
    - Different time zones – technical support needs to adapt
- Users can link to (or bookmark) parts of a web application
  - No explicit single point-of-entry
  - Ensure the user doesn't end up in an invalid location
- Users can use navigation other than the application's provided navigation
  - Browser's back/forward/refresh buttons

Traditional Software Engineering

- Client is usually a (large) company
- Client has expert knowledge of the problem – main challenge is communicating such knowledge
- Budget (time and money) known – usually from medium to long term project
- Requirements usually defined in terms of processes – “a system to perform process X”
- End users are (employees of) the client

Web Engineering

- Web Engineering differs from Software Engineering in that…
  - Less well-defined requirements
  - Shorter delivery time
  - Users are (potential) customers of the client
  - A **MUCH** more diverse user base
  - Requirements defined in terms of user types – “a system to satisfy the needs of user Y”

Web Engineering

- User-centric Development
  - Identify types of users
  - Find out what they want/need – ask them!
- Pareto's 80/20 Rule
  - 80% of your user population will use 20% of the features
  - This can be even more extreme for Web Application

Agile Development

- Break development into small iterations
  - Short timeframes (1-4 weeks)
- Constant communication with stakeholders
  - Stakeholder = anyone who has an interest in the product
  - Demonstrate current work
  - Get feedback
  - Integrate feedback into future iterations
- Adapt to changes rather than predict them
  - Requirements **WILL** change – expect and respond

Agile Development

- Web engineering lends itself to an agile software development lifecycle. Why?
  - Ease of deployment
    - we can easily show the customer what we have
    - We don’t need to install the software for them
  - Ease of feedback
    - Because we can show the customer what we have, they can (should) give us timely feedback
  - Ease of update
    - Again because of how simple it is to deploy a web application, we can easily show the users the changes they have requested

Agile Development

- Basic workflow:
  1. Identify requirements
  2. Design solution
  3. Implement solution
  4. Test & evaluate
  5. Repeat
- Stakeholders have input at every stage

Agile Variations

- Scrum
- Test-Driven Development (TDD)
- Behaviour-Driven Development (BDD)
- Extreme Programming (XP)
- Kanban
- Etc.

Scrum

- One of the more common variations
- Roles:
  - Product Owner:
    - Represents the business stakeholders
    - Talks to the end users
    - Talks to the developers
  - Development Team
  - Scrum Master:
    - Responsible for keeping the development team on track

Scrum

- Elements:
  - Task – a manageable unit of work
  - User Story – some requested functionality. Often written as a narrative:

    As a [role]

    I want [feature]

    So that [benefit]
  - Backlog – ordered list of user stories and tasks still to be completed
  - Sprint – unit of development time (usually 2-4 weeks). At the end of each sprint there should be a deliverable product.
  - Daily Scrum – short (15 mins) stand-up meeting/progress report

Scrum

- Workflow:
  1. Gather requirements (in the form of user stories)
  2. Break into tasks where possible and priorities
  3. Estimate amount of work needed for each task/story
  4. Plan next sprint
  5. Start sprint
     a) Daily scrum
     b) Develop
     c) Repeat step 5 until the sprint is complete (never add items to a sprint, in some circumstances you can remove them)
  6. End of sprint review
     - What went well
     - What didn't
  7. Test and evaluate (can also be done as part of step 5)
     - Involve product owner and end users where necessary
  8. Rinse and repeat.

Test-Driven Development (TDD)

- Write unit tests first
  - A unit test is a piece of code that tests that unit of code (usually a single method) functions as expected
- Then write the code to pass the tests

- Write just enough test for the test to fail.
- Write just enough code to pass the test.
- Refactor.

Test-Driven Development (TDD)

- Workflow:
  1. Write the test case.
  2. Run tests.
  3. Write enough code to pass the test.
  4. Run the tests.
  5. Refactor.
     - Restructure code
  6. Run the tests.

Test-Driven Development (TDD)

- Advantages:
  - Makes your intention clear
    - You understand the requirements
    - Tests document your codes intentions
  - Guaranteed code coverage
  - Confidence
    - Safe to refactor
    - Safe to add new features
    - Future proof
  - Catch defects early

Test-Driven Development (TDD)

- Disadvantages:
  - Time Consuming (but so is bug fixing).
  - Lots of tests that sometimes don’t test the right thing!

Behaviour-Driven Development (BDD)

- A variation on TDD
- Workflow:
  1. Define user stories
     - Through meetings with client
  2. Create tests that ensures the user story functions as expected (not just a unit of code but all the units that make up the user story)
  3. Write code that fulfils the test, and therefore fulfils the user story

QUESTIONS??