Faculty of Computer Science, Dalhousie University *25-Feb-2025*

DGIN 5201 — Digital Transformation

Lecture 13: Lec 13: Course Project Planning

Location: LSC C236 Instructor: Vlado Keselj and FE. Bordeleau
Time: 13:05–14:25

2 About Course Project: Choosing Topic

Plan for Today Class

- Discussion about the project:
  - Technical and Business requirements
  - Guest speaker: Tapajyoti Das
- Emerging topics: AI and Deep Learning (time allowing)

Technical Project Requirements

- Learning objectives and goals
- Choosing project topic
- Presenting a case for your project
  - preparing project specification
- Design and implementation choices
- Planing development

Project Learning Objectives

- Strenghen innovation and startup culture
- Learn to build a functioning three-tier system
- Learn fast Web-based prototyping
- Understand fundamentals of web technology
- Explain trade-offs between web-oriented programming languages and frameworks

Course Project Goals

- A simulation of practical software development, particularly in a start-up environment; this means:
  - building a substantial and useful system
  - working in a team (ideally 3–4 members)
  - having (hopefully) real users
- Building a complete three-tier system
  1. User interface (presentation tier)
  2. Processing logic ("business" logic, control tier)
  3. Database (persistent data, data access tier)
- Related but different than Model-View-Controller design pattern

How to Choose a Project Topic?

- Look at the examples of existing systems
- Find something interesting to you
- We will try to provide project ideas
- Define appropriate scope
- Discuss with other team members

Look at the Existing Systems

- There are many examples of web applications and services:
  - Google, Facebook, Twitter, Amazon, Instagram, LinkedIn, ...
  - email, chat, search, collaboration, maps, mobile apps, ...
  - standalone applications are okay, but may be challenging to recruit beta testers
- You should aim at something of this form, but much simpler and feasible for a term project

Find Interesting Topic to you

- The application should be useful to you, to start with
- A quote by Paul Graham, co-founder of Y Combinator (www.paulgraham.com):

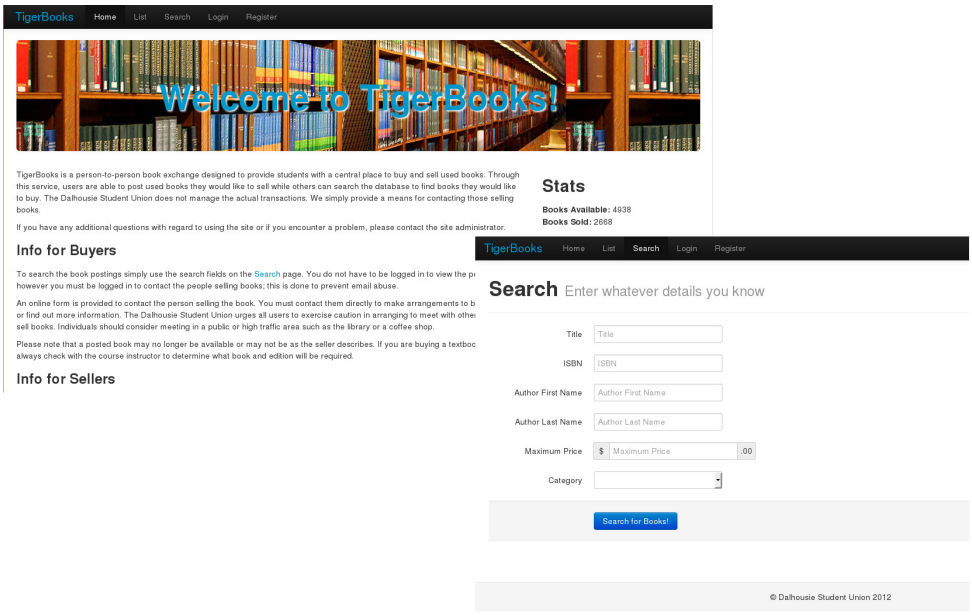
  "The way to get startup ideas is not to think of startup ideas. It's to look for problems, preferably problems you have yourself.
  The very best startup ideas tend to have three things in common: they're something the founders themselves want, that they themselves can build, and that few others realize are worth doing...

Define Appropriate Scope

- By the end of term (probably sooner), you need to have an MVP (Minimal Viable Product)
- There is not much time by the end of term, and you have other courses, so your product must be much simpler than typical interesting examples out there

A (Former) Local Example

- Dal TigerBooks



Presenting Strong Case for Your Project

Different approaches to think about your project:

1. Elevator speech: How would you make a case for your project to an investor during a 60sec elevator ride
2. 1-page ad: what would you put there; what information you would put on your landing web page
3. 5–7 slides for a 5–10min presentation of the project
4. Demo: what case scenarios you would include in a short demo to a client
5. Business plan: value proposition, competitive environment, revenue model and similar
6. Job interview: how would you talk about the project to a potential employer

Project Calendar Overview

```
2025    Mo Tu We Th Fr Sa Su
Jan     .6  7  8  9 10 11 12 (w1) Rapid Prototyping
        13 14 15 16 17 18 19 (w2)
        20 21 22 23 24 25 26 (w3)
Feb     27 28 29 30 31  1  2 (w4) Disruptive Innovation
         3  4  5  6  7  8  9 (w5)
        10 11 12 13 14 15 16 (w6)
        17 18 19 20 21 22 23 (study break)
Mar     24 25 26 27 28  1  2 (w7)  Team meetings
         3  4  5  6  7  8  9 (w8)  Project spec.
        10 11 12 13 14 15 16 (w9)  Team meetings
        17 18 19 20 21 22 23 (w10) Team meetings
        24 25 26 27 28 29 30 (w11) Project Demos
Apr     31  1  2  3  4  5  6 (w12) Final Presentations
        .7  8  9 10 11 12 13       Report and code
```

A2 Git Submission, Step 1: FCS GitLab Login

A2 Git Submission, Step 2: FCS GitLab Repository

A2 Git Submission, Step 3: Push and Existing Folder instructions

A2 Git Submission, Step 4: Login to timberlea and cd to directory

A2 Git Submission, Step 5: Create git subdirectory and rsync

A2 Git Submission, Step 6: Rsync all directories e1 to e8

A2 Git Submission, Step 7: cd to git directory

A2 Git Submission, Step 8: Follow Git Instructions

A2 Git Submission, Step 9: Follow Git Instructions

A2 Git Submission, Step 10: Final GitLab Repository check over Web

Some Project Technical Notes

- Many choices to approach creating a working MVP
- Our `timberlea` exercises can be a good starting point
- If you are interested and familiar with other platforms, tools, and software; you can use them

A Baseline Implementation

- Assume diverse background knowledge and levels
- Baseline Implementation:
  - login to `timberlea.cs.dal.ca` using CSID
  - work with a shell; e.g., bash, basic Unix commands
  - use of a plain-tex editor: emacs, vi, vscode, or similar
  - use of HTML, scripting languages, JavaScript, CSS
  - choose a back-end language (PHP, Python, Perl, ...)
  - plain files for persistant data, database

Relevant Topics and Tools: 3 Tier Implementation

- Tier 1: User Interface: Standalone app vs. Web client
- Interface Tier 1 $\Leftrightarrow$ Tier 2; "Wire" format
  - formatted text, JSON, REST, XML, SOAP, ...
- Tier 2: Business logic: Perl, PHP, Python, etc.
- Interface Tier 2 $\Leftrightarrow$ Tier 3; "Plumbing"
  - TCP/IP, authentication, ...
- Tier 3: Databases; Persistent data
  - Flat files, MySQL, SQLite, Postgres, MongoDB, Redis, ...
- Server solutions: timberlea, etc.

Interfaces and their Importance

- Interfaces are defined at the boundaries of system components
- Contracts about format of inputs and outputs and semantics
- Interfaces hide implementation: implementation of one component can be changed without affecting the rest of the system

- Examples: change database, change user interface

Design Choice Examples

- user interface
  - Web browser, desktop, mobile phone, tablet, API...
  - HTML/CSS/LESS, Javascript, Bootstrap, Jquery...
- languages
  - Perl, PHP, Python, Javascript, Ruby, Java, C, C++...
- server: `timberlea`, own machine, Amazon AWS, Heroku...
- database
  - plain files, MySQL, SQLite, Postgres, MongoDB...
- information exchange formats
  - custom text format, JSON, XML, REST, ...
- frameworks
  - WordPress, Django, Mojolicious, Flask, Rails, GWT...
- development environments
  - emacs, vscode, Eclipse, XCode, Visual Studio, ...

"Make versus Buy" Choice

- You can use components and code from other systems; such as open source
- Overall project design must be yours and majority of the implementation should work
- Components from other parties must be clearly described in the documentation at the end
- It is okay to in large part re-implement an existing system, but there should inevitably be some original ideas about your system when compared to an existing system

Some Relevant Topics and Tools

- Toolkits: jQuery, Dojo, YUI, ...
- GUI tools
  - Swing, TkInter (Python), jQueryUI, Bootstrap...
- Web frameworks
  - GWT, Django, Flask, Zend, Rails, Cocoa, Express, WordPress, ...
- Development Environment
  - shell and tools (emacs, make, ...), Eclipse, Xcode, vscode, ...
- Repository
  - GitLab, Github, Git, SVN, Git, Mercurial, Bazaar, ...

Planning Process of System Development

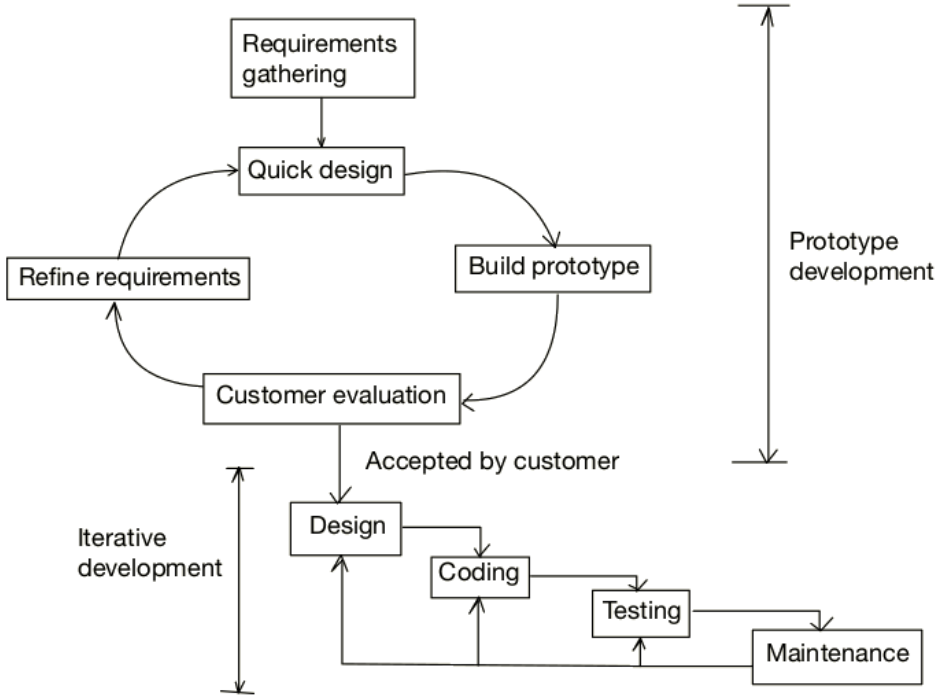
- One choice is the classic Waterfall model:
  Requirements, Design, Implementation, Testing, Maintenance
- or, closer to the industrial practice:
  Specifications
  Requirements
  Architectural design
  Detailed design
  Coding
  Integration

Testing
Delivery

Which Development Process to Use?

- Waterfall Model is overkill and not completely appropriate
  - follows the "big bang" model of development
- Rapid Prototyping Model is more appropriate
- However, it should not be a rapid "hacking" model
- Have a clear plan to try to make clear steps forward
- Keep a log with completed tasks, and what to do next
- Keep iterating working prototype, and after each iteration be able to declare success and walk away

Rapid Prototyping Model



A Less Formal Approach to Development Process

- Conceptual design
  - general description, sketches, scenarios, screenshots, rough diagrams
- Requirements Specification ("what")
  - precise ideas and requirements; understanding that once requirements are set it will be costly to change them
- Architectural Design ("how")
  - overall structure diagrams: components and connections, subsystems, interactions and interfaces, languages, systems, connectivity, data availability
- Implementation ("what by when")
  - make prototype and iterate, get real users asap, prepare tests as you go