Exercise Sheet 1

Software Engineering as a Process
**Topics we’ll cover ...**

- Software Engineering Basics
- Agile/Scrum Software Development
- User-Centered Design (UI design / User Experience)
- Requirements Gathering & Analysis
- Estimation & Planning
- Risk Management
- Testing (Code Review, Unit, System, Usability, Performance)
- Source Code Management
- Software Design & Maintenance

**Overview ...**

**First couple of weeks**
- Basic ideas in Software Engineering & Agile/Scrum
- Gain an understanding of project needs (requirements)
- Next week (Wed) is “meet your sponsor” CEDE event

**Rest of first half of course (to week 8)**
- Work towards developing your “project plan”
- Emphasis on requirements & estimation/planning

**Second half of course (to week 15)**
- Work on software prototype (working demo)
- Emphasis on testing, software design, project management
Project Plan Overview (see template)

1. **Project Overview** (summary, objectives, deliverables, related, …)
2. Requirements
3. **Design Considerations** (Initial UI design, architecture)
4. Project Risks
5. Initial Product Release Plan
6. Maintenance Considerations
7. Project Management Considerations

Major Features Checklist (see template)

**Minimal Viable (Useful) Product (MVP)**

- the minimal set of features that make the product “viable”
- for us, we (mostly) are interested in the minimal *useful* product

**The Major Features Checklist**

- list of the main areas of functionality, etc., needed for your MVP
- you should view these as a contract/commitment
  - features must be approved by your sponsor and advisor
  - team evaluated on completion and quality of the features
- features can change, but changes must be documented and approved
Weekly Tasksheets (see templates)

You will be doing 2-week “sprints” in this class
- **Basic idea:** every 2 weeks something to show to sponsor
- 2 week iterations on your software product with **sponsor feedback**

You will break each sprint into weekly tasks ("mini" sprints)
- **Basic idea:** meet with advisor weekly on goals / tasks / progress
- Each week, fill out the task sheet as you go:
  - Initial goals and what you plan to finish (demo) at end of week
  - Initial tasks, initial estimate, actual time, and completion status
  - Summary of week with ideas for improvement (reflection)

Exercise Sheet 2

Part 1: Major Feature Checklist
- Copy and share the template
- Name it CPSC-<#>-feature-checklist (e.g., CPSC-01-feature-checklist)

Part 2: Weekly Tasksheet
- Copy and share the template
- Name it CPSC-<#>-tasksheet (e.g., CPSC-01-tasksheet)
- As a team, start filling it out
**Create Good Task Descriptions**

- Should be **clear** what the task is
- Should be obvious when task is “**done**” (testable)
  - “Research node.js” not a good task description
  - “Do tutorial foo on node.js” or “Build a simple webpage using node.js” much better
- Describe what **complete** means for each task
- **Estimate** how long it will take in hours (... more later)

**Product vs Process vs Project**

A **product** is the software or other deliverables being built (artifacts)

A **process** defines steps to follow to develop the product

A **project** follows a process to develop (and deliver) the product

Characteristics of all projects:

- **temporary** … definite start and end
- **measurable** … e.g., how much has been finished
- **progressively elaborated** … learn more as you go
What is project vs product success?

Projects have (business, external) objectives

- desired outcomes of the client/company creating the product
- e.g., increase profits, decrease costs, attract investors, etc.
- a successful project achieves the business objectives

A successful product

1. **users**: product does what is needed / is useful
2. **client**: helps the company achieve the business objectives

Typically 2 doesn’t happen unless 1 happens

Successful vs Challenged vs Failed

A successful project

- develops the “correct” product on time and within budget

Two notions of “correctness” in software engineering

1. **Verification**: was it built correctly? (to specification)
2. **Validation**: was it the correct thing to build? (to users needs)

A challenged project

- late, over budget, or partially incorrect (partially useful)

A failed project

- canceled or not adopted (not useful)
How well do Software Projects do?

<table>
<thead>
<tr>
<th>MODERN RESOLUTION FOR ALL PROJECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>SUCCESSFUL</td>
</tr>
<tr>
<td>CHALLENGED</td>
</tr>
<tr>
<td>FAILED</td>
</tr>
</tbody>
</table>

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011 - 2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

* Standish Group, 2015