CSC301

Product management in an Agile environment

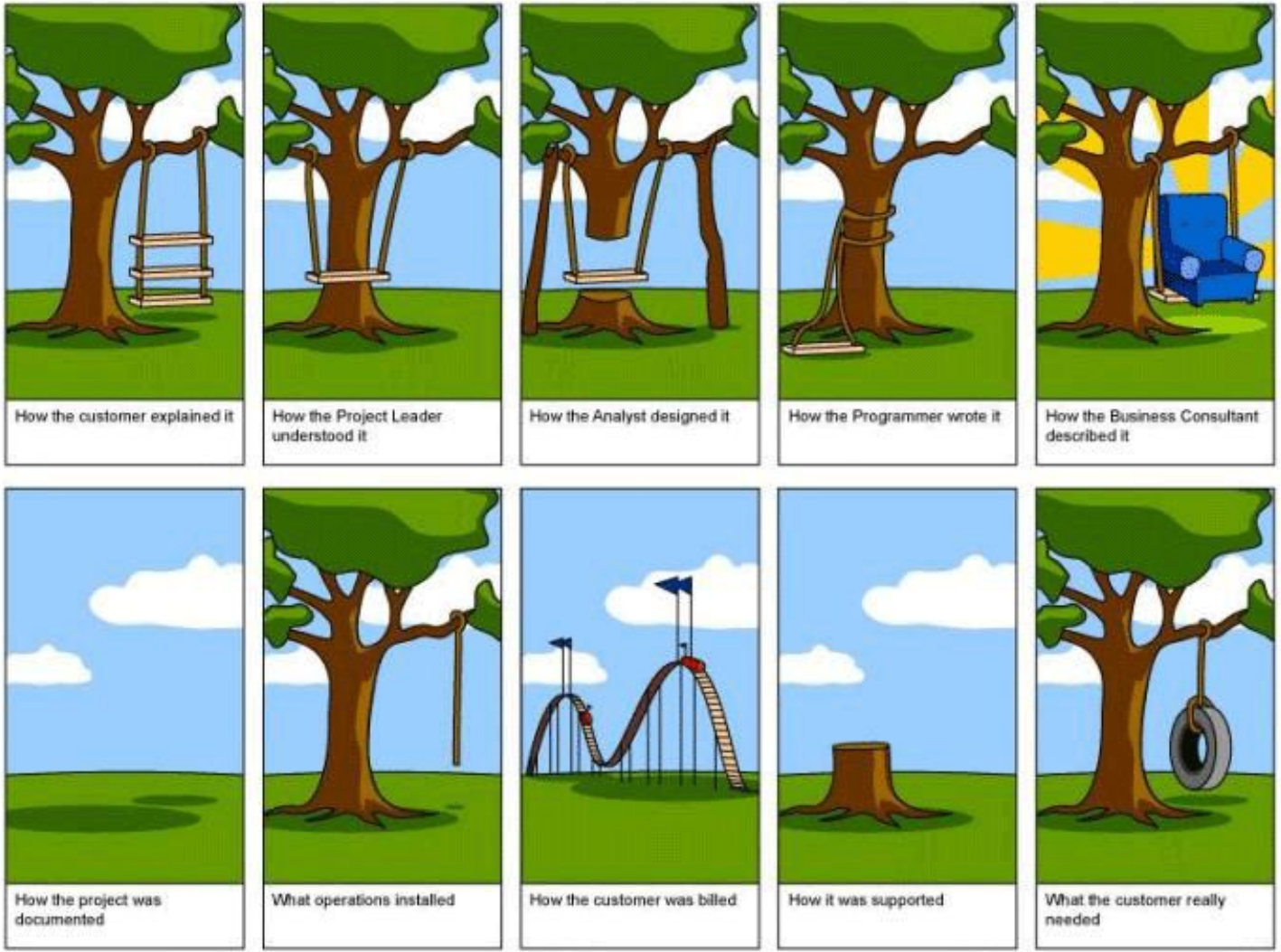
Last week, we talked about software processes (aka project management).

This week, we'll talk about product management ...

Project vs. Product

Project Management	Product Management
How to organize your team?	What should the team build?
Delegating tasks, time estimation, scheduling, tracking performance, etc.	Scoping, prioritizing, collecting and analyzing feedback, etc.
Requires understanding of your team members, their abilities and working habits.	Requires understanding of your users and their needs.
Requires organizational skills	Requires domain expertise.

Lack of project and/or product management leads to jokes like this one ➜



Agile Product Management

- Agile approach:
  - Iterative, ongoing process
  - Work in short cycles to allow frequent delivery
  - Focus on the user!
- Each iteration involves roughly the following steps:
  - Planning
    - Articulate goals and success metrics
    - Make some decisions
  - Doing work
    - Ex: build (and test) software, perform market research, etc.
  - Reviewing outcomes
    - Collect feedback and evaluate decisions in retrospect

Example 1

- Planning:
  - Goal: Get visitors to our promotion page.
  - Metric: Number of visitors who see the promotion / Total number of visitors to the site
  - Decision: Add a big, red button at top-right corner of the home page.
- Work:
  - Add the button
  - Make sure we can track which pages are visited by each visitor
  - Optional: Allow for A/B testing (i.e. some users see the button, some don’t)
- Review:
  - Check the percentage of users who saw the promotion.
  - Are we doing better than we did last week?
  - If we did A/B testing, is there correlation between seeing the red button and visiting the promotion page?

Example 2

Consider a hypothetical automated assignment-submission system at a university (e.g. MarkUs, Blackboard, the tools we use in this course, etc.)

- Goal: Minimize the need for human intervention.
- Metric: Percentage of assignments that require human intervention.
- Decision: When a student submits an assignment, validate the file names and present a friendly message in case of an error.

Example 2, Cont’d

- Work:
  - Option 1 - Build the feature
    - Instructor can specify which files/folders are required
    - If a submission is missing a required file, reject it and show a friendly error message.
  - Option 2 - Try to *validate the hypothesis**, without building anything
    - Collect data from past courses.
    - Out of those assignments that required human intervention, how many had an issue with missing required files?

** In this case, the hypothesis is that submissions with invalid file names cause the need for human intervention.*

Example 2, Cont’d

- Review:
  - Option 1 - Count how many invalid submissions were detected.
  - Option 2 - Estimate how many submissions such a feature will detect (which will allow you to decide whether it is worth the cost/effort of implementation).

Notice that, in both cases, collecting data might be tricky:

1. “Running experiments” is not so easy.
2. Historical data is usually not tracked.

Pragmatic Product Management

- Use intuition to come up with ideas.
- Use a scientific method to validate intuition:
  - Instrument (i.e set things up), experiment, measure & conclude.
- Use tools to collect and analyze feedback.
  - Various analytics tools
  - Product management tools
  - Custom, in-house tools
- Gradually, it should become easier to ask (the right) quantitative questions.
  - And get statistically significant answers

Product Management

- Once you have a product and active users, it’s easier to articulate goals:
  - User actions and feedback drive decisions
  - Business goals drives decisions
  - Easier for all team members to understand the overall vision
- But, at the beginning, things are much less clear and structured ...

Product Definition

- How do we get from an initial idea to a working product (or even a prototype)?
- Once again … incrementally
- Start with high-level concepts, and gradually create more detailed plans
  - Goals
  - Requirements
  - Design

In the next few slides, we will try to clarify how we distinguish between the three stages, and review some of the tools/techniques/standards that we use along the way.

Step 1 - Goals

- High-level, concise English description
  - Ignore most technical details
- Focus on:
  - What is the **objective** of your product?
  - Who are your **users**?
  - Why would they use the product? That is, what **value** will your product offer to its users?
- The focus is the problem and users, not the product.
- Plan first, don’t build it just because you can!
- Borrow techniques from the marketing world ...

Objectives

- Articulating objectives can change the priorities and full nature of your product.
- For example, here are different objectives for a “TTC app”:
  - Provide ETA of next vehicle(s) to nearby station(s).
    - Accuracy of vehicle data is a high priority
  - Help tourists and visitors explore the city and its attractions using the TTC.
    - Internationalization is a high priority
    - A good database of attractions & events is a high priority
  - Help commuters plan trips that combine driving and TTC
    - Accurate traffic reports are a high priority
    - Parking (at TTC stations) information might be a high priority as well.

Notice: We only talk about the problem we want to solve. We don’t say what the product is, and we definitely don’t mention any technical details at this point.

Personas

- A tool/technique used for identifying users/customers.
- Popularized by marketers
- Goals:
  - Identify and understand our users
  - Remember the details of our “user profiles”
  - Efficiently discuss our archetypal users (by using names, instead of full descriptions).
- The idea is simple:
  - Develop a character (name, picture, background story, etc.)
  - The character represents an archetypal user
  - A named character is much easier to remember (and to use in a conversation) than a list of characteristics and attributes.

Personas

- Here a few examples:
  - Dan The Commuter Student
  - Suburban (Family man) Joe
  - Shirley & Mike (retired culture vultures)
- And more online resources:
  - A nice video tutorial on developing personas
  - An article about the origin of personas
  - And a nice blog post.

Step 2 - Requirements

- What is needed in order to achieve our goals?
  - High-level description of the main concepts of our system and the interaction between them.
- Artifacts can be
  - Structured English documents

    For example: User stories, Use cases
  - Diagrams

    For example: UML use case diagrams

User Stories

- A tool/technique used for specifying high-level requirements.
  - Specified from the user's perspective.
  - Follow a pattern: As *<role>*, I want *<action>*, so that *<benefit>*

- For example:

  - As an **instructor**, I want to **specify which files/folders are required** for a given assignment, so that **students know what they need to submit**.

  - As a **student**, I want to see a **clear error message if something is wrong** with my submission, so that I can **fix it and not lose marks**.

Use Cases

- Another convention/standard for specifying requirements
- More verbose and detailed compared to user stories
  - Identifier, name, description
  - Pre & post conditions
  - Basic and alternative courses of action
- Serve the same purpose as user stories, but they are not the same
- Here is yet another nice example of a hypothetical use case versus user story

Use Case Diagrams

- Use case diagrams are part of the the UML standard
- Visually connect between use cases and related *actors*
  - Which user(s) participate in a certain use case?
  - Which external/internal application(s) participate in a certain use case?
- Optionally, organize further using system boundary boxes and/or packages.
- Can provide a high-level view of the dependency graph.

Step 3 - Design

- Implementation details
  - Need to “invent” them
  - We’re “getting closer” to code
- Typically written by the dev team
- Artifacts can be:
  - CRC cards - Highlight the important classes, their responsibilities and the dependencies among these classes.
  - Class Diagrams - Part of the UML standard. More detailed than CRC
  - Mockups & Wireframes - There are many cool tools out there.
  - Storyboards, English descriptions, interfaces in code, etc.

Agile Planning, Summary

- Planning artifacts can be very useful:
  - It’s easy to forget why you are building something. Planning artifacts help remember.
  - Once again, traceability is important.

    Utopia … Could trace every:
    - Requirement to a goal,
    - Design feature to a requirement,
    - Line of code to a design feature,
    - Bug fix to issue.
- But, we still want to be agile and focus on deliverables not documentation.

Agile Planning, Summary

- There is no silver bullet!
  - Different teams, projects and points in time may require different tools and approaches.
- Constantly *evaluate* which new features offer users the most value.
  - Better to deploy minimalist features sooner than rich feature sets later.
  - Failing fast is better than failing slowly!
- First release is no exception:
  - *Minimal Viable Product* mantra of startups

Minimal Viable Product (MVP)

- Concept guiding modern startup thinking
- So you have an idea?
  - What value proposition does it offer which users?
  - What user pain does it alleviate?
  - What problem will they “hire” your product to solve?

    If you have a few minutes, watch Clayton Christensen, Understanding the Job (but please don’t start drinking milkshakes for breakfast).
- Isolate minimal set of features
  - Maximize return on risk/cost

Why MVP?

- If you are right, and close to minimal set:
  - You get to market as quickly as possible, for the least cost
- If you are wrong, you “fail fast”
  - Collected feedback ⇒ Which features were lacking (or wrong)
  - Could be a chance to *pivot*
- Investors look for intelligently chosen MVP
  - Watch: Steve Blank, No business plan survives first customer contact
- Past experience … Waterfall plans frequently led to multi-year, multi-million dollar, startup trainwrecks.

MVP and Your Team Project

- Officially, you are asked to build a prototype, not a product.
- But, the MVP concepts still apply - You want to produce maximum value with minimum cost/effort.
- The main challenge: Scoping
  - Deciding what to build and what not to build.
  - Focusing on the important features.
  - Being realistic with your plans.
- The first few weeks are all about planning.
  - And producing useful artifacts as part of the process.