CSC301

Software Processes - Organizing a software development team

Announcements

- A2 has been released
  - If there are any issues (e.g., your repo is empty), please let Adam know as soon as possible.
  - Due Feb 2 @ 23:59
- Team Project Deliverable 1 has been released
  - Due Feb 9 @ 23:59

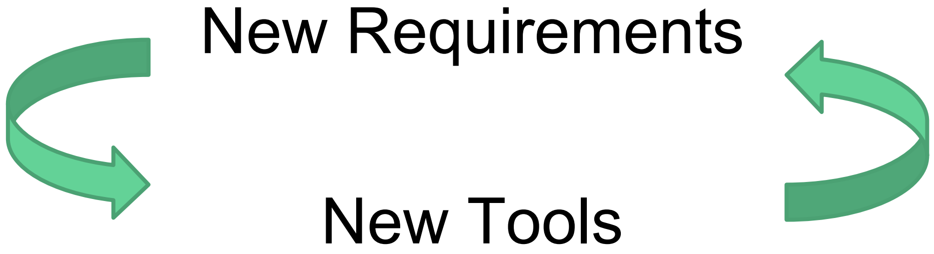
Announcements

- Team registration is now closed
  - Project signup sheet captures the assignment of people to teams and teams to tutorials.
- In this week's tutorial, your TA will be
  - Going over project requirements
  - Helping you pick good project ideas
  - **Attendance will be taken at some point during the tutorial**
- We will create your team repos by the end of next week
  - We'll give you some time to finalize your teams
  - If you don't have a team by the end of the week, we will assign you to a team (more or less arbitrarily)

OK, let’s start …

First Two Weeks

- Started talking about collaboration
- Focused on version control, and noticed a cycle:



This Week

- Collaboration among people
- Outside of the source code
- Goals are simple:
  - Get things right
  - Be efficient

Goals

- Be **effective** - Get things right
  - Identify target users and their problem(s)
  - Solve the right problem(s) in the best possible way
  - Deliver the most value to the user
- Be **efficient**
  - Deliver fast
  - Build easily maintainable systems
  - Adapt to changes quickly

Q: Can you think of a way to quantify these goals?

Be Effective + Be Efficient

- More difficult than it sounds
- Especially with uncertainty
    - Unknown requirements
    - Changing requirements
    - External factors (e.g., new technologies, competition)
- Requires a plan ...

What do we mean by a plan?

- Many informal “definitions”
  For example:
  - Structured set of activities, used (by a team) to develop software systems
  - A team’s standards, practices and conventions
- Let’s see another informal “definition” …

What do we mean by a plan?

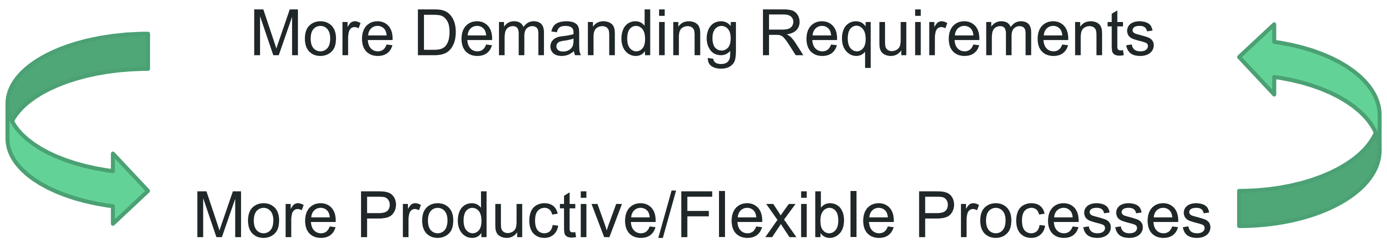
- Roles
  - What are the responsibilities of different team members?
  - Who fills these roles?
- Events
  - When/where/how do team members communicate?
  - When/how do we write/review/release code?
- Artifacts
  - Documents, diagrams,
  - Videos, images, audio recordings,
  - Pull requests, issues, code, etc.

Plan = Software Process

- Many synonyms:
  - Software Development Process
  - Software Development Methodology
  - Software Development Life Cycle

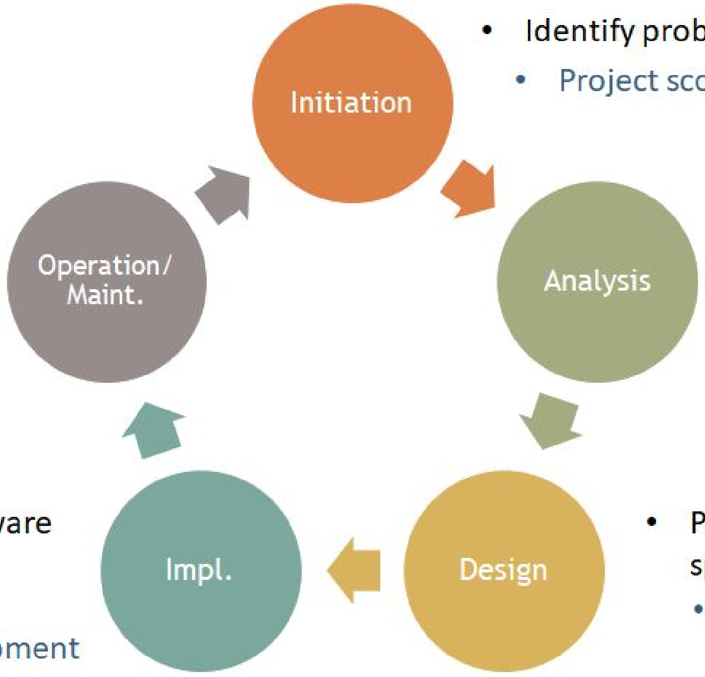
Software Process

- Like version control tools, software processes keep evolving


More Demanding Requirements
More Productive/Flexible Processes

- Let’s take a quick chronological tour of some popular software processes …

Software Development Life Cycle - Phases



- Identify problem or opportunity
  - Project scope, goals

- Analyze and understand problem
- Identify solution requirements, expectations, and priorities
  - Talk to stakeholders and identify their needs, wants.
  - Multiple elicitation techniques

- Produce technical blueprint and specification for a solution
  - Consider many factors: cost, perf., interop., usability, maintain., compliance, ...

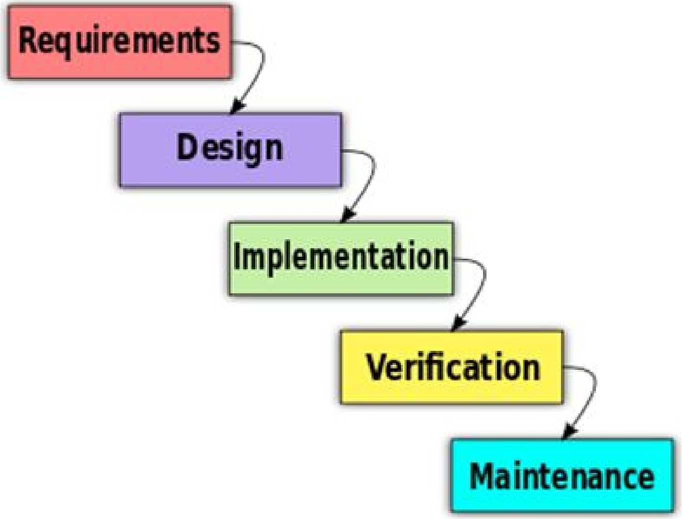
- Produce solution (hardware software)
  - Build vs. Buy
  - Outsourcing, development

- Run and maintain system
  - On premise vs. cloud
  - Monitoring, etc.

Waterfall

The full project (in some cases, that means multiple years) is divided into a sequence of phases (the number and names of phases vary depending on the source):

1. Requirements
2. Design
3. Implementation
4. (Integration)
5. Verification
6. (Deployment)
7. Maintenance



Waterfall

- 1970’s and 80’s
- Linear model
- Phases do not overlap
- A phase must be successfully completed before the next one can start
- **Cannot go back**

Waterfall

- Originates in the Construction & Manufacturing industries
- Suitable when “reverting” is costly (or impossible)
- Today it is considered as an anti-pattern (i.e., a bad practice) in most software industries
    - Where would Waterfall might still be used these days?
- A detailed description of the process

Waterfall

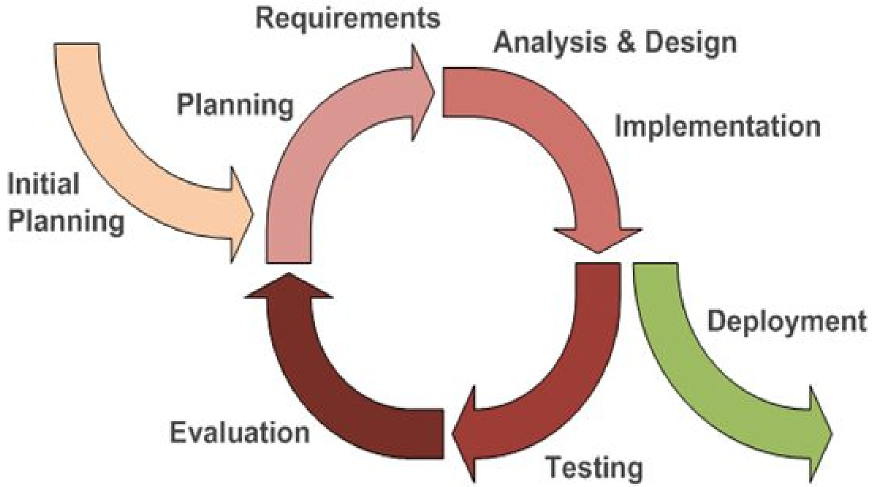
- Some arguments for it:
    - The later a bug is found, the more costly it is to fix
        - E.g., by spending a long time on Requirements and Design early on, we discover problems/bugs early on
    - Requirements & design phases produce documentation, which preserves knowledge
    - The predictions made by the detailed waterfall plan are largely science fiction, but one learns a lot about the problem space
    - Easy to understand

Waterfall

- Some arguments against it:
  - Not suitable for changing requirements
  - Many decisions are hard to get right, especially if they must be made before you’ve written a single line of code
  - Limited stakeholder involvement
  - Very limited feedback and possibility for revision

Iterative and Incremental Processes

- Build the product incrementally in short iterations
- In each iteration, the sequence of phases is
  - Planning
  - Design & Implementation
  - Testing
  - Evaluation/Review
    - Both from development and use



Iterative and Incremental Process

- Arguments for it:
    - Shorter iterations = More opportunities to adjust and correct/improve (**feedback!**)
        - We can release at the end of an iteration. Therefore, we can collect feedback from users frequently - increased user involvement
        - Less costly to adapt
    - "Fixes" one of the problems with Prototyping: we are developing the product, not just prototypes

Iterative and Incremental Process

- Arguments against it:
  - Unclear long-term vision
    - Architecture issues may arise
  - Might be inappropriate for specific industries
    - E.g.: Microprocessor manufacturers will most likely choose something closer to Waterfall

The Trend

- With time, software teams
  - Become more flexible and adaptive to changing requirements
  - Collect user feedback more frequently
  - Release code more frequently
- And then came the term Agile …

The Agile Manifesto

- The Agile Manifesto (2001) is a high-level, general description of a certain *type* of
  - Software process
  - Team culture
- The highlights are:
  - Promote collaborative culture within your team(s)
  - Focus on deliverables (i.e., working software that can be shipped to a customer)
  - Collect feedback from users
  - Be adaptive! Plans change all the time
- *Agile* is a big buzzword in the industry.
  - You need to separate the hype from the actual ideas & insights behind the "Agile movement"

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

The Agile Manifesto

- Keep in mind, it was written in 2001.
- In practice, most modern teams follow most/many Agile principles anyway
    - Make sense for software development
    - Help us get things right and do it efficiently
    - Proven to work well

Agile Processes

- Later in the course, we will review a few popular Agile processes
    - XP
    - Scrum
    - Kanban
- We will see the trend: processes are becoming lighter and more realistic
- For now, let’s stop talking about the process (being efficient), and start talking about the product (getting things right) …