Today’s Outline

• Quick recap
  • Software Engineering
  • Project Proposals

• Software development lifecycles (SDLC)
  • What and why are they needed
  • Recurring themes
  • Popular models and their tradeoffs
Software Engineering is ...

“An **engineering discipline** concerned with all aspects of **software production** from the early stages of system specification [requirements] through to maintaining [evolving] the system after it has gone into use.” — Ian Sommerville

Software Engineering tasks include:

- Requirements engineering
- Specification writing and documentation
- Architecture and design
- Programming
- Testing and debugging
- Deploying, operating, evaluating, refactoring and evolving
- Planning, teamwork and communication
CSE 403 Projects work as learning tools

We are here
Project Proposals and Lifecycles
Assignment 1 – Project Proposals

An elevator pitch is a brief, persuasive speech that you use to spark interest in a product, project or idea, or in yourself. An elevator pitch is short, about the time you spend in an elevator, hence the name.

A foolproof elevator pitch template

1. Introduce yourself
2. Present the problem
3. Present your solution
4. Share your value proposition
5. Add a call to action

https://asana.com/resources/elevator-pitch-examples

You have 2-3 minutes for your project pitch to the class - this is a good example of how it could flow
Try pitching your project, or yourself, to your neighbor

<table>
<thead>
<tr>
<th>Introduce yourself</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Present the problem</td>
<td></td>
</tr>
<tr>
<td>Present your solution (This is your lucky day!)</td>
<td></td>
</tr>
<tr>
<td>Share your value proposition</td>
<td></td>
</tr>
<tr>
<td>Add a call to action</td>
<td></td>
</tr>
</tbody>
</table>
Lifecycles: Here’s the challenge

Problem Specification

???

Source Code Solution
One solution: Code and fix

Specification (maybe)
SDLC: Code and fix

Pros:
- Little or no overhead - just dive in and develop, and see progress quickly
- Applicable *sometimes* for small projects, short-lived prototypes, and/or small teams

Cons:
- <Over to you>
Let’s look at data*

Thrashing = doing a lot of work but not making progress towards the goal

Imagine recoding something again and again and it’s still not right for purpose
Let’s look at data*

Project with early attention to SDLC process
Is a more structured SDLC necessary?

It’s used to establish an order – provide a model - in which software project events occur from project conception to project delivery

• It forces us to think of the “big picture” and follow steps so that we reach it without glaring deficiencies

• Without it we may make decisions that are individually on target but collectively misdirected

• It allows us to organize and coordinate our work as a team

• It allows us to track progress and risks, and adjust as necessary
Recurring themes in SDLCs

A SDLC defines how to produce software through a series of stages

Goals of each stage

• Define a clear set of actions to perform
• Produce tangible (trackable) items
• Allow for work revision
• Plan actions to perform in the next stage

Common stages

• Requirements
• Design
• Implementation
• Testing
• Release
• Maintenance

Key question — how to combine the stages, in what order, and why
Today’s Outline

• Quick recap
  • Software Engineering
  • Project Proposals

• Software development lifecycles (SDLC)
  • What and why are they needed
  • Recurring themes
  • **Popular models and their tradeoffs**
    • Waterfall model
    • Evolutionary prototyping
    • Spiral model
    • Staged delivery
    • Agile (XP, Scrum)

We are here!

All have the same goal – deliver high quality software, on time, meeting the customers needs
SDLC: Waterfall model

- Top-down approach
- Sequential, non-overlapping activities and steps
- Each step is signed off on and then frozen
- Most steps result in a final document
SDLC: Waterfall model

- Top-down approach
- Sequential, non-overlapping activities and steps
- Each step is signed off on and then frozen
- Most steps result in a final document

Conceptually very clean, but what’s missing?
SDLC: Waterfall model

- Top-down approach
- Sequential, non-overlapping activities and steps
- Each step is signed off on and then frozen
- Most steps result in a final document

In what context would it work well?
Likely parts of their SDLC is waterfall-like due to the upfront and regulated requirements.
Pros:
- Simple to understand
- Promotes common dialogue
- Highly regulated deliverables

Cons:
- Hard to do all the planning upfront
- Inflexible – changes are expensive
- Test and integration come late – fixes are expensive
- Final product may not match the customer’s needs
SDLC: Prototyping

• Problem domain or requirements not well defined or understood
• Create small implementations of requirements that are least understood
• Requirements are “explored” before the product is fully developed
• Developers (and customers) gain experience when developing the product
• Prototype can evolve to the real product or can serve to be a learning tool only
SDLC: Prototyping

- Problem domain or requirements not well defined or understood
- Create small implementations of requirements that are least understood
- Requirements are “explored” before the product is fully developed
- Developers (and customers) gain experience when developing the product
- Prototype can evolve to the real product or can serve to be a learning tool only

In what context would it work well?
UI prototyping is popular

https://internetdevels.com/blog/what-is-website-prototype-how-build-website-prototype

https://learn.microsoft.com/en-us/power-bi/fundamentals/desktop-what-is-desktop
SDLC: Prototyping pros and cons

Pros:
- Client involvement and early feedback
- Improves requirements and specifications
- Reduces risk of developing the “wrong” product

Cons:
- Time/cost for developing may be high
- Hard to commit what will be delivered and when
- May end up evolving a poor choice (limit thinking holistically)
SDLC: Spiral Model

- Incremental/iterative model (combines waterfall and prototyping)
- Iterations called spirals
- Repeat these activities:
  - Determine objectives (reqs)
  - Risk analysis
  - Develop and test
  - Plan
- Phased reduction of risks (address high risks early)

Boehm, *Spiral Development: Experience, Principles, and Refinements*
SDLC: Spiral Model

- Incremental/iterative model (combines the waterfall model and prototyping)
- Iterations called spirals
  - Phased reduction of risks (address high risks early)
  - Phased reduction of risks (address high risks early)
  - Phased reduction of risks (address high risks early)

In what context would it work well?

Boehm, *Spiral Development: Experience, Principles, and Refinements*
SDLC: Spiral Model pros and cons

Pros:
- Early indication of unforeseen problems
- Allows for changes
- The risk reduces as costs increase

Cons:
- More complex to run
- Requires proper risk assessment
- Requires more planning and experienced management

Boehm, *Spiral Development: Experience, Principles, and Refinements*
**SDLC: Lots of variants** - Staged Delivery

- Combines waterfall, spiral, scrum
- Waterfall-like planning upfront then spiral/scrum-like short release cycles
  - Pros: ?
  - Cons: ?

---

McConnell: [https://stevemcconnell.com/](https://stevemcconnell.com/)
SDLC: Staged Delivery pros and cons

• **Pros:**
  - Can ship at the end of any release cycle
  - Intermediate deliveries show progress, satisfy customers, and lead to feedback
  - Problems are visible early

• **Cons:**
  - Requires tight coordination
  - Product must be decomposable
  - Extra releases cause overhead
Thoughts on which SDLC to use?

Crossroads Request for Proposal (RFP) No. 511017

All proposals are due by 2:30 p.m. Mountain Time on Monday.

Interested parties are invited to submit a proposal for one (1) of the Crossroads supercomputer system. The subcontract in support of the New Mexico Alliance for Computing at Extreme Scales is composed of the following NNSA High Performance Computing Systems:

- Los Alamos National Laboratory (LANL)
- Sandia National Laboratories (SNL)

Interested parties are advised to monitor this website for potential amendments and other Crossroads RFP information updates. Administrators may notify interested parties of updated Crossroads RFP information via e-mail; however, there is no obligation to do so.

It is the responsibility of all interested parties to monitor this Crossroads RFP information.

Interested parties must submit in writing all communications regarding the Crossroads RFP (questions, comments, etc.) to the Contract Administrator.

Crossroads 2021 Technical Requirements Document

LA-UR-18-25993
SAND2018-73660

Crossroads 2021: Technical Requirements

1 INTRODUCTION
1.1 SCHEDULE

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract 8923321SCN000001. LA-UR-18-25993. Approved for public release; distribution is unlimited.

Stay tuned for more!

• Truly, there is no end, but we’ll move to the more recent SDLC next week
• Questions on the traditional models?