Technical debt and other challenges

A mini-lecture series

CSE498 Collaborative Design (W) - Secure and Efficient C++ Software Development

02/17/2025

Kira Chan

https://cse.msu.edu/~chanken1/

Technical Debt

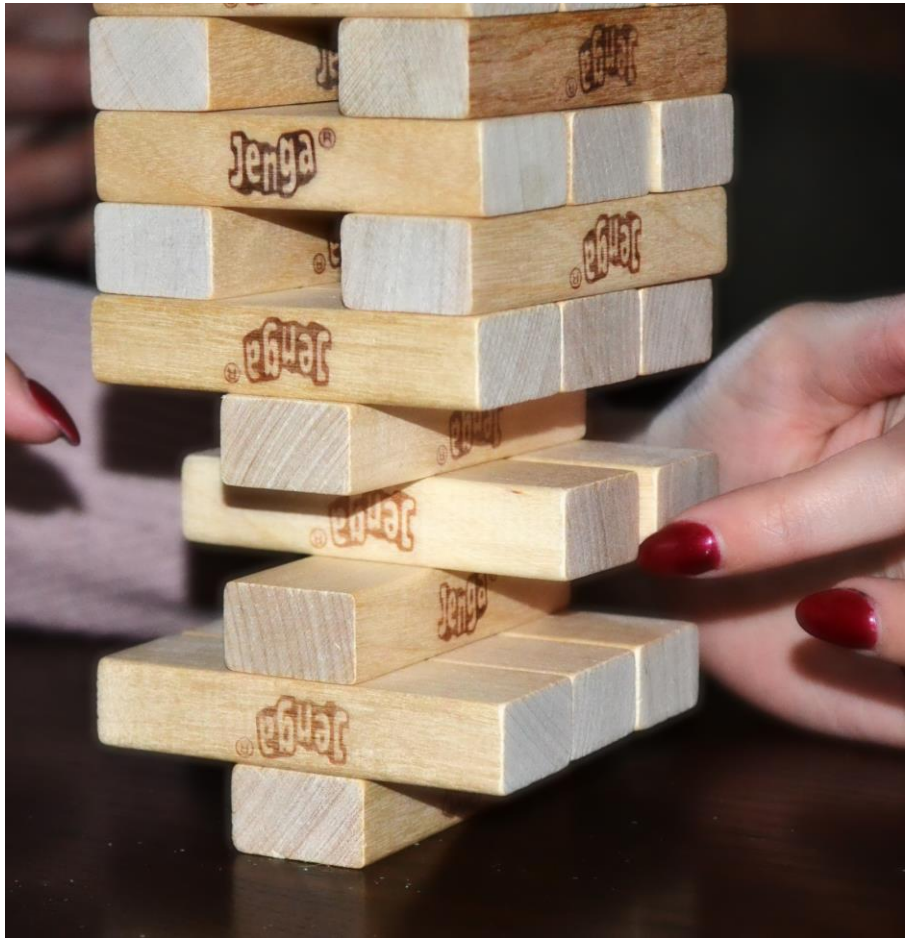
- Technical debt refers to the implied future cost resulting from immediate solutions
- “Punting the problem for future me”
- Example:
  - Hard coding
  - Outdated libraries or deprecated dependencies
  - Lack of documentation
  - Lack of testing
  - Deferred updates
  - TODOs, FIXMEs

Causes

- Business and management pressure
  - Prototyping
- Knowledge gap
- Development process challenges
  - Lack of requirements (incomplete)
  - Conflicting requirements
- Changing requirements
- Security****

Problems that arises

- Maintenance after deployment
- Adding new features (think of a Jenga tower)
- New developers or switching teams
- User experiences
- Speed



Legacy code

- How many of you have worked directly with an “ancient” programming language
  - Fortran
  - Lisp
  - COBOL
  - BASIC
  - Pascal
- Easier to think of them as input output black-box transformation and write a wrapper for them

Complexity and difficulties

- Essential complexity
  - Inherent to the problem itself, and thus cannot be truly eliminated
- Accidental complexity
  - Introduced accidentally as we are trying to solve the underlying problem
- Fred Brooks article: No Silver Bullet - Essence and Accident in Software Engineering
  - One of the most well known and cited articles

Person of the Day
Fred Brooks

- Landmark contributions to computer architecture, operating systems, and software engineering
- Invented the interrupt signal
- Started the Computer Science department at University of North Carolina at Chapel Hill