Everything I Need to Know about Senior Design, I Learned in Software Engineering…

CSC 326

What is Senior Design?

- Capstone Course
  - A culmination of everything you have learned as a computer science undergrad.
- Official Description
  - Application of **software engineering principles** and **basic computer science** to the **total development** of a **software system**. Consideration of the **software system design process**, including **requirements** and **design** detail. **Development** and **evaluation** of a prototype accomplished through **design team activity**. Comprehensive **written and oral project report** is required.

Overview of Senior Design

- Work on teams to solve a computer science problem
  - Create requirements, design, implement, and test a software system
  - Continue development of project started in previous semesters
  - Research possible applications
  - …
- Go to https://sdc.csc.ncsu.edu/ and look through this semester's topics (Projects tab)

Objectives

- Technical
  - Negotiate, clarify, and document customer requirements
  - Apply knowledge of fundamental algorithms, PL concepts, and design patterns to determine an overall design
  - Implement
  - Test
  - Plan and monitor progress
- What are the processes and practices we use to accomplish these objectives?

Objectives (2)

- Communication
  - Audience-Sensitive oral technical presentation
  - Audience-Sensitive technical document
- Teaming
  - Work on a team to develop a software product
- What are the processes and practices we use to accomplish these objectives?

Your Project

- Every team will have a *different project*
- The teaching staff does NOT have a solution for these projects
  - Part of the work is to define your OWN solution
- Every team has a different project sponsor
  - Interact with your sponsor professionally
  - Meetings (what do meetings require?)
  - Presentations (audience appropriate)
- Log all activities…

Software vs. Developer

- There's a difference between what the software does and what you do
  - Requirements: What the software does
    - Use Cases
    - Formal Requirements
    - In terms of the system (personify the system)
  - Task Plans: What you do to define, design, implement, test, and maintain the requirements
    - Activities that you DO
    - Write the requirements
    - Design the system
    - Etc.

[Presentations Slides]

Oral Progress Reports

- Present progress to teaching staff and peers throughout the semester
  - Introduce project and team
  - Overview
  - Sponsor Background
  - Problem Summary & Proposed Solution
  - Project Goals & Benefits
  - Challenges & Risks
  - Requirements, Design, Implementation, Test
    - May include code snippets
  - Team Contributions
  - Resources Needed
  - Action Steps

Written Progress Reports

- Project Overview Document
  - Executive Summary
  - Sponsor Background
  - Problem Description
  - System Overview
  - Project Goals/Benefits
  - Project Risk Analysis
  - Development Methodology
  - Ethical Implications of Project
  - Preliminary Resources Needed
  - Preliminary High-level Design
- Iteration 1 Document
  - Iteration Definition
  - System Overview
  - System Requirements
  - Ethical Implications of Project
  - Security Considerations
  - Resources Needed
  - Design
  - Implementation
  - Testing
  - Interaction Retrospective
  - Task Plan
- Document Design
  - Quality of Writing
  - Report Improvement
- Team Effort
  - Collaboration
  - Authorship
  - Instructor's Observations
  - Team Progress
- Guides
  - Developer's Guide
  - User Guide
  - Install Guide

Sponsor Background & Problem Statement

- Go beyond the sponsor-supplied text
- Who is your sponsor and what is the nature of their business?
  - How does this tie to your project?
- What is the need or business case?
- What already exists? What will be extended?
- Quantify "better"

[Economics Slides]

Project Goals & Benefits

- What are you delivering?
- What are the goals?
- What is the system you will create or improve?
- What are the benefits to the sponsor?

- The goals/benefits must address needs from the problem statement

[Risk Slides]

Risks & Risk Mitigation

- Identify potential risks that can either:
  - Jeopardize the completion of the project
  - Limit or prevent the implemented solution from fulfilling its intended goal.
- Record how you plan to minimize these risks

- Examples
  - Ecological validity – works with mock data, but not production data
  - Time – needed resource unavailable during the project
  - Communication – challenges clarifying requirements
  - Technology – available to support the system needs – can you solve the problem given current technologies?

- Learning new technologies is not a risk unless something is blocking the team from learning a technology

[Process + Tools Slides]
[Planning + Estimating + Project Management Slides]
[Process Slides]
[Process 2 Slides]

Development Methodology

- Description of your software development process
- Justification for the process that you select
  - Nature of the project
  - How it can help mitigate project risks
- Agile and Plan-driven are paradigms, not methodologies.

13

13

[Requirements Intro Slides]
[Requirements Engineering Slides]
[Requirements Modeling Slides]
[Requirements Modeling – User Stories Slides]

System Requirements

- What must your ***system*** DO?
  - This is different than what YOU must do to implement the system.
- Overall View – from the user's perspective
  - Users/Personas
  - Glossary
- Detailed Requirements
  - Functional Requirements
  - Non-Functional Requirements
  - Constraints
- Justification – How do your requirements meet the sponsor's needs?
- Model as traditional, use cases, user stories, or for gaming projects as description of gameplay

14

Design

- Plan the structure of your system at a level where another team of developers could implement the system from your design
- System Architecture/High-Level Design
  - Global view of Software
  - Major system components and how they interact (label arrows)
  - Describe purpose and overall behavior
  - How does the high-level diagram meet the requirements
- Low-Level Design
  - Detailed module descriptions
  - More detailed class diagrams and sequence diagrams
  - Data (DB schema, E/R diagrams), algorithms, state diagrams, etc.
  - File formats
  - Data structures
  - APIs
  - GUI design – mockups and evolution
  - Meet requirements?

[Backend + Persistence + DB Slides]
[Backend Design Slides]
[Rest API Slides]
[Frontend Design Slides]
[Accessibility Slides]
[Architecture Slides]
[Secure Design Slides]

Design

- Also includes
  - Screen layouts
    - Prototypes and feedback from customer
  - Report formats
    - How you present data to the user
  - Database designs
    - Entity Relationship Diagrams
    - Table Design
    - Normal Form
  - File designs
    - How data are stored in files
  - Data design
    - Schemas, etc.

Design Elements

- Data
  - Overview of data in product
- Architecture
  - Overview of product being developed
- Interface
  - Information in and out of product and between product elements
  - Expose interface for client programs
- Component-Level/Procedural Level
  - Details for small units

Implementation

All implementation work from prior courses
[Secure Design Slides]
[Privacy Slides]
[Maintenance]
[CI/CD Slides]

- Say where you're at – Current status!
- Iteration Definition
  - Goals and dates of each iteration
  - Update with what was accomplished
- Security Considerations
  - How the system's design and implementation preserve confidentiality, integrity, availability, identification & authentication, accountability, and privacy
    - This may include how security would be factored into the system in future iterations
- Project Folder Structure
  - How is the project organized?
- Project Configuration/Settings
  - How is the project configured?
  - How to run the system with different parameters?
  - What should change for deployment?

[Testing Slides]

Testing

- Acceptance Test Plan
  - Tests that your customer would use to accept your product as “done”
  - Record actual results
  - Implications of testing results
- Unit Testing
  - Appropriate framework for your language
  - 70% statement coverage across testable areas
    - Negotiate testable areas
  - At a minimum, a scripting method is used to test
  - Implications of testing results
- Additional testing considerations (project-specific): usability testing, performance testing, etc.

Supplemental Guides: Project Installation & Delivery

- Installation Guide
  - Tasks required for installation
  - How to roll back if problems
  - Verification of successful installation
- Developer Guide
  - Where’s the code, main components, etc.?
  - How does a new developer get started with development?
  - Low-level design diagrams
  - Module relationships
  - What your code is currently like, not how you got there
    - How you designed your code and why is in the design document.
- User’s Guide
  - How do users interact with the application?

Key Ideas

- Documentation and modeling need to fit your specific product
  - Don't tie yourself to specific tools, they may not be the right tools for the job
- Justify your decisions
  - Have reasons for what you do
  - Document those reasons
- Organize your software artifacts appropriate to your system
- Be flexible and listen to constructive criticism
- Have fun!!!

Exam Overview

Empirical Software Engineering

© NC State Software Engineering Faculty

23

"Empirical"

Comes from the Greek word…

ἐμπειρία (empeiría)

….meaning ***Experience*.**

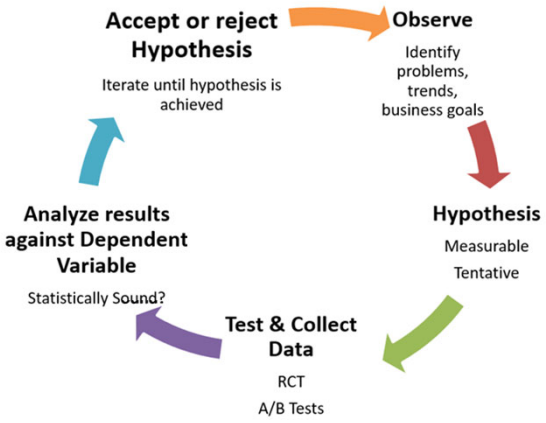
24

Learning goals:

- Different methods of investigation result in different types of information
- Our understanding of SE is based on the **scientific method**
- 'Pearls of wisdom' about SE:
  - Detecting bugs early $$ <<< detecting bugs later $$
  - Automated testing outperforms human testing.
  - Code review and pair programming make code better.

25

Scientific Method in SE



Hypothesis are often called **"Research Questions"** in the SE literature

img: uxdesign.com

26

Ways to measure things

Survey people

Converse with people (e.g., interviews)

Watch people (e.g., observational studies)

Watch what people watch (e.g., eye tracking)

Analyze a product (e.g., code, designs)

Analyze lots of products (e.g., many repos)

Watch inside people (e.g., brain scans)

… many more

27

What to measure

Software tools used

Who is speaking and for how long in a group

Defects per lines of code

Testing coverage

Libraries used

Cyclomatic complexity

Where the eyes focus

Brain activity

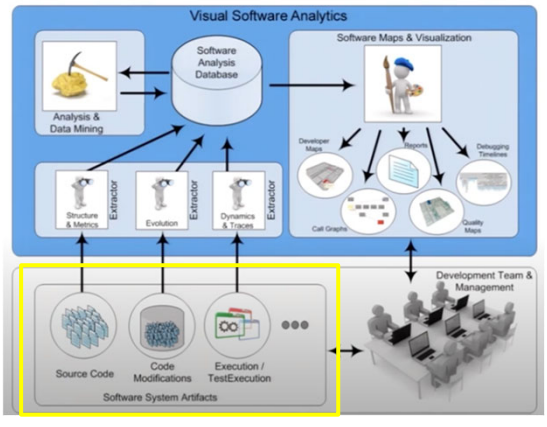
… many more

28

Teaming & AI in SE

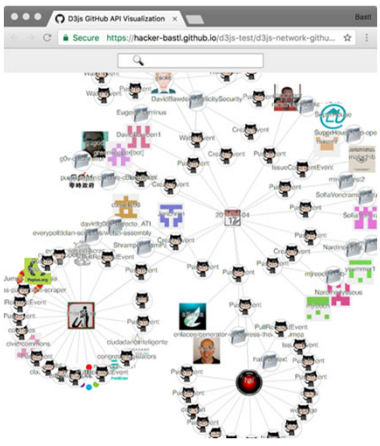
Mining Software Repositories

using datamining and graph search



img: TujsMUNKzyt.wfqk

How people code together….

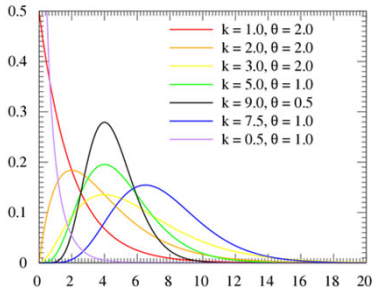


img: hackr-bastl

Analyzing data

Qualitative Analysis

- Grounded theory
- Open card sorting
- … many more



Quantitative Analysis

- Descriptive statistics (e.g., mean, median)
- Statistical tests (e.g., ANOVA, T-test)
- … many more

img:wikipedia

31