CEN 4020
Software Engineering

About Me

Education, Teaching, Interests

BAs in Computer Science and Applied Mathematics (Franklin College) in 2015

MSCS in Computer Science (USF) in 2017

PhD in Computer Science and Engineering (USF) in 2020

Teaching Assistant for 16 courses

Assistant Professor of Instruction at USF since 2020

Teaching *Advanced Database for IT, Software Engineering* in Summer 2022

Hobbies include soccer, tennis, guitar

Dissertation: Sentiment Analysis in Peer Review

MyLexicon: 3.49/4.00 (B+)
The team provided a condensed essay without any references. The graphics while very eye-catching, were not referenced in the text so I am unsure where they came from. However, the team provided an informative essay that broke down scripting in debugging nicely. In their [project title], the team provided the upsides of using automated testing in debugging by providing examples that included the fact that since automated scripts did not require human interaction to complete that they could be run overnight to reduce the cost of supervision and to optimize the time one had to run tests. Their 'Application example' section provided an excellent resource for an automated testing application in Selenium due to its use across multiple languages.

SentiWordNet: 3.08/4.00 (B)
The team provided a condensed essay without any references. The graphics while very eye-catching, were not referenced in the text so I am unsure where they came from. However, the team provided an informative essay that broke down scripting in debugging nicely. In their [project title], the team provided the upsides of using automated testing in debugging by providing examples that included the fact that since automated scripts did not require human interaction to complete that they could be run overnight to reduce the cost of supervision and to optimize the time one had to run tests. Their 'Application example' section provided an excellent resource for an automated testing application in Selenium due to its use across multiple languages.

ANEW: 3.53/4.00 (A-)
The team provided a condensed essay without any references. The graphics while very eye-catching, were not referenced in the text so I am unsure where they came from. However, the team provided an informative essay that broke down scripting in debugging nicely. In their [project title], the team provided the upsides of using automated testing in debugging by providing examples that included the fact that since automated scripts did not require human interaction to complete that they could be run overnight to reduce the cost of supervision and to optimize the time one had to run tests. Their 'Application example' section provided an excellent resource for an automated testing application in Selenium due to its use across multiple languages.

Teaching Assistant

- Zihe Ye

Syllabus

Syllabus Highlights

- Be sure you thoroughly read and understand the key information from the syllabus:
  - **Academic Integrity of Students**
  - **Piazza: Preferred Means of Communication. It is required to sign up now.**
  - **Regrading and Late Work Policies**
  - **Group Work Policy**
- Book – VERY useful. Provides much more than we'll be able to cover in class. Helpful figures. Reference manual for future projects/work
- SDS accommodations requests must be sent this week so I can prepare assignments.
- Please write down assignment due dates. Any conflicts must be emailed to me this week.
- FDA – Quiz on canvas. Must be completed today.

Semester Project

- This semester you will be using the knowledge you learn to create a complete website to manage conference paper submissions.
- Groups of 3. May not work alone, same team whole semester.
- The groups in Canvas are first-come, first-served: 'People' tab, then 'Project Groups' tab. Do not create your own “student group”.
- Please don’t send me an individual message with your project question – post to Piazza.

Software Engineering: Principles and Practices (Third Edition)

Chapter 2 Software Engineering

QUOTE:

“Indeed, the woes of Software Engineering are not due to lack of tools or proper management, but largely due to lack of sufficient technical competence.”

Niklaus Wirth (Programming Language Designer and Author)

What is software engineering?

- The creative, disciplined, and quantifiable application of engineering principles to the analysis, design, implementation, and maintenance of small, medium, and large software systems

What is software engineering?

- Benefits
  - Software product improvement
  - Software project improvement
  - Software process improvement

What is software engineering?

- Goal - High-quality software systems that deliver exactly what the end user needs in an economical manner

Titles of the first issue of Transactions on Software Engineering

- Specification Techniques for Data Abstractions
- On Preventing Programming Languages from Interfering with Programming
- Towards a Programming Apprentice
- Testing Large Software with Automated Software Evaluation Systems
- An Interactive Program Verification System
- A Synthesizer of Inductive Assertions
- Proving Loop Programs
- Concurrent Software Fault Detection
- Analytic Models for Rollback and Recovery Strategies in Data Base Systems
- A Machine and Configuration Independent Fortran: Portable Fortran
- Some Experience with Automated Aids to the Design of Large-Scale Reliable Software

Five major objectives of the SWEBOK project

- To promote a consistent view of the field of software engineering worldwide
- To clarify the place and boundaries of software engineering with respect to other disciplines such as computer science, project management, computer engineering, and mathematics
- To characterize the contents of the software engineering discipline
- To provide topical access to the software engineering body of knowledge
- To provide a foundation for curriculum development and for individual certification and licensing materials

Software Engineering Body of Knowledge

- Software Requirements
- Software Design
- Software Construction
- Software Testing
- Software Maintenance
- Software Configuration Management
- Software Engineering Management
- Software Engineering Process
- Software Engineering Models and Methods
- Software Quality
- Software Engineering Professional Practice
- Software Engineering Economics
- Etc.

Figure 1-1. Traditional hardware failure curve

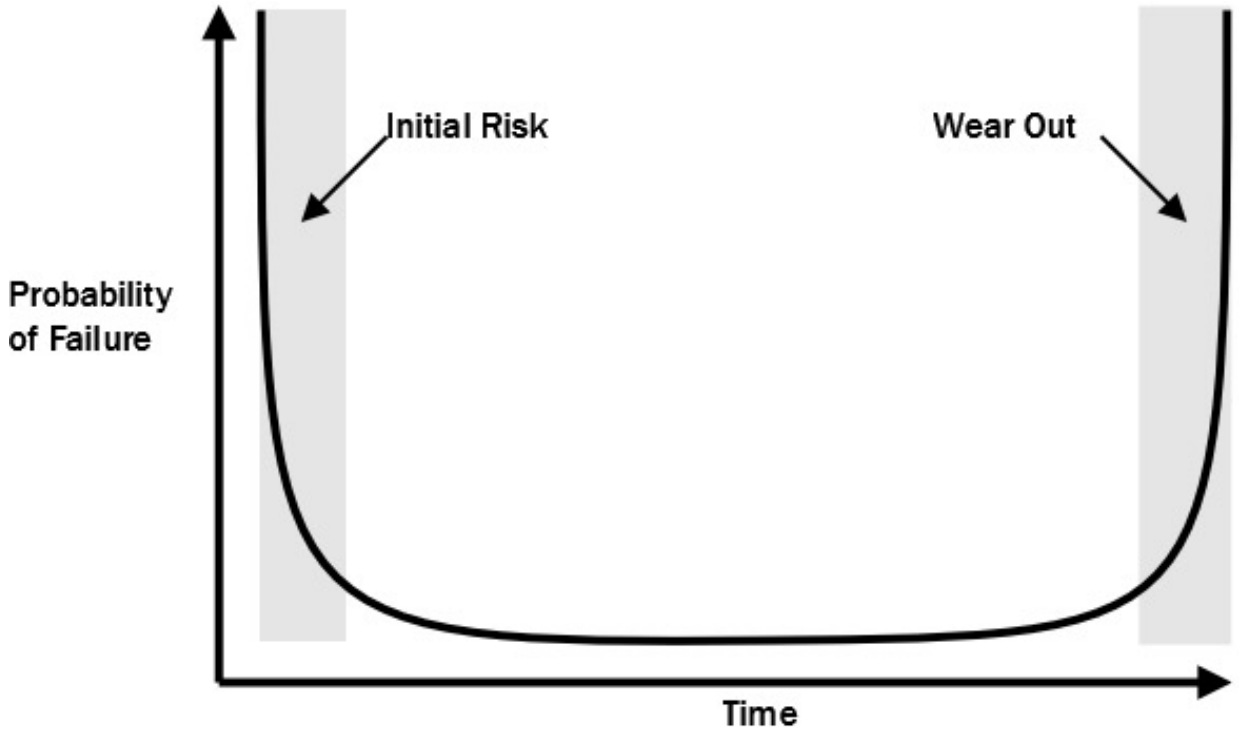


Figure 1-2. Idealistic software failure curve

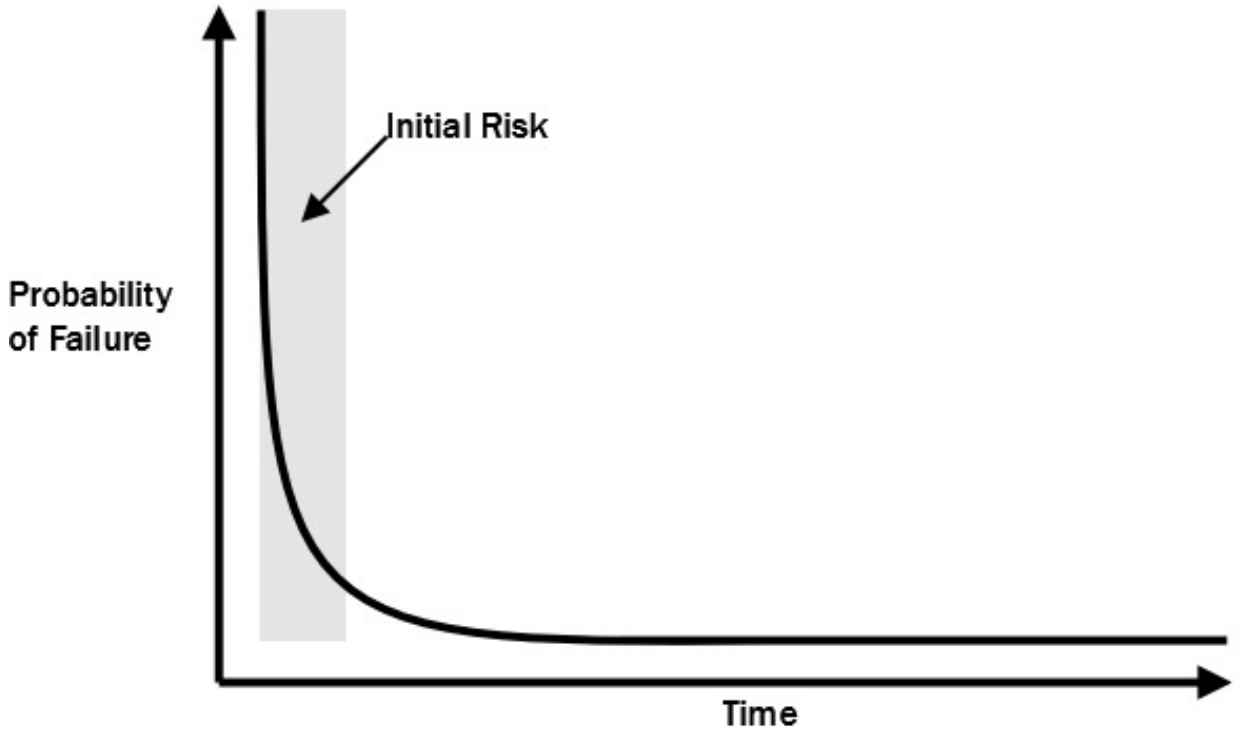
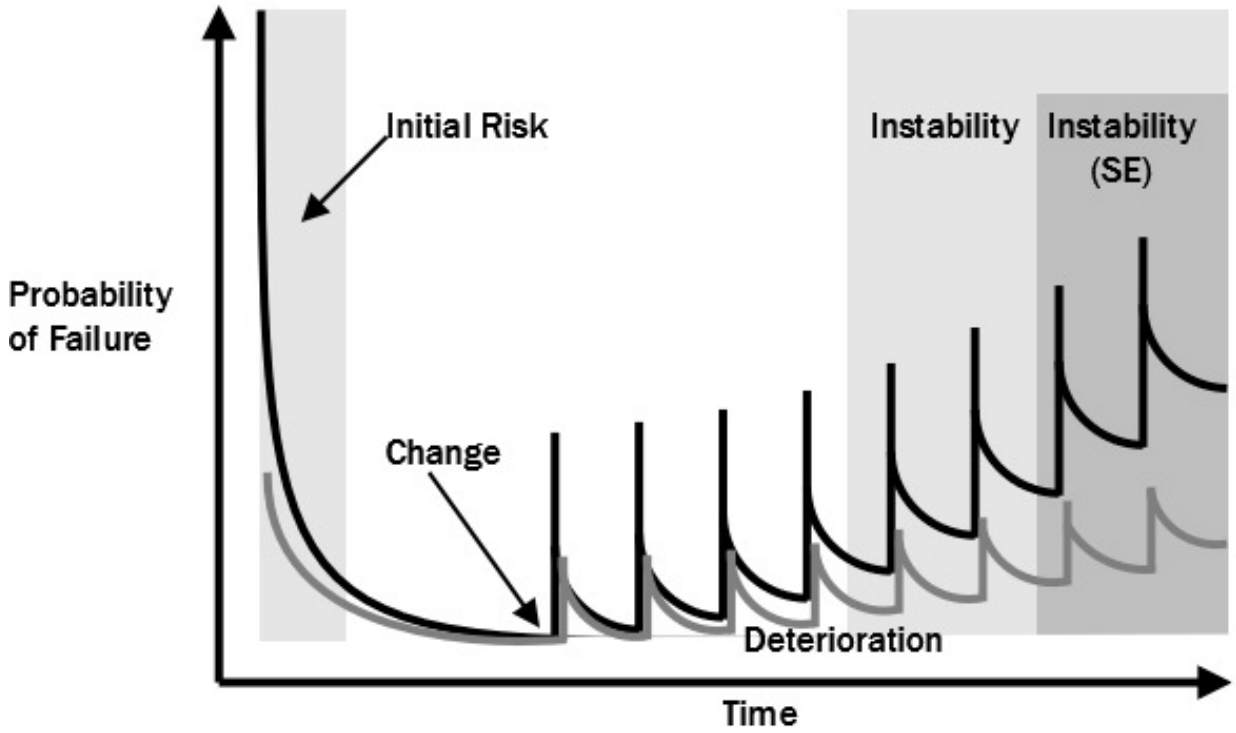


Figure 2-1. Realistic software failure curve without and with software engineering applied



Software Engineering: Principles and Practices (Third Edition)

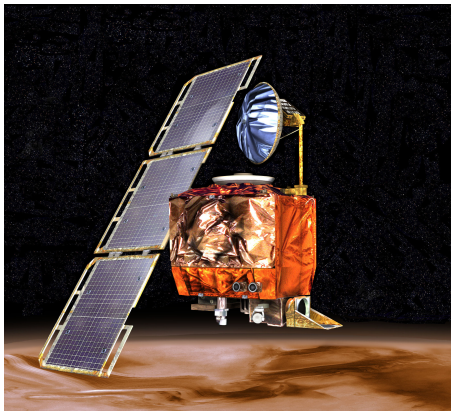
Chapter 2 Software Engineering

Most software professionals know that

- Software projects are difficult to estimate in terms of effort and cost
- Software projects are difficult to schedule and track
- Software requirements are difficult to capture
- Software systems are difficult and expensive to develop
- Software systems are difficult and expensive to test
- Software systems are difficult and expensive to maintain

Software defects that have caused tremendous damage

- A cancer radiation therapy machine that miscalculated proper dosages of radiation exposing several patients to harmful, and in some cases fatal, levels of radiation
- A worm that was permitted to infect millions of computers, delete files, and change registry entries resulting in billions of dollars of damage
- A satellite that crashed into Mars because of the faulty analysis of requirements causing millions of dollars of damage
- The Denver airport baggage system: https://www.youtube.com/watch?v=xx8f4x6C_KY
- Find short synopses of some of these catastrophes at www.devtopics.com/20-famous-software-disasters



Disciplines that share boundaries with software engineering

- Computer engineering
- Computer science
- General Management
- Mathematics
- Project Management
- Quality management
- Systems engineering

Conditions under which a computing professional can be called a software engineer

- Practices software engineering for his or her employer
- Limits his or her practice of software engineering to the software products his or her employer develops
- Does not use the software engineering title outside of his or her employer's organization
- Does not claim to be qualified to offer software engineering services to another party

Examples of certifications

- ASQ Certified Software Quality Engineer (CSQE)
- IEEE Computer Society Certified Software Development Associate (CSDA)
- IEEE Computer Society Certified Software Development Professional (CSDP)

The four Cs of certification

- Current (i.e., to stay current with industry best practices)
- Connected (i.e., to network with the brightest minds in software engineering)
- Career-minded (i.e., to position yourself for that next promotion or opportunity)
- Committed (i.e., to advance the software engineering profession)

Software engineers shall adhere to the following eight principles

- Public. Software engineers shall act consistently with the public interest.
- Client and employer. Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.
- Product. Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
- Judgment. Software engineers shall maintain integrity and independence in their professional judgment.
- Management. Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
- Profession. Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
- Colleagues. Software engineers shall be fair to and supportive of their colleagues.
- Self. Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession”

Activity

- In small groups, select one of the general knowledge areas of the software engineering body of knowledge (SWEBOK) presented in the book. Then, prepare a one paragraph report that describes how a solid understanding of that general knowledge area should contribute to flattening of the realistic software failure curve shown in figure 2-1. Post with all participating teammates' names to Piazza as a **note**.

QUOTE:

“The system is that there is no system. That doesn't mean we don't have process. Apple is a very disciplined company, and we have great processes. But that's not what it's about. Process makes you more efficient.”

Steve Jobs (Co-Founder of Apple)

What are software activities and tasks?

- System development life cycle (SDLC) activities
  - Analysis
  - Design
  - Implementation
  - Maintenance
- Each activity contains a set of related tasks.
- Each task usually results in the production of a software deliverable.
- The activities and tasks require the ability to apply an appropriate problem-solving technique.
- The activities and tasks are combined and sequenced to ensure the progress and eventual completion of a software project.
- The way that a software development organization chooses to combine and sequence the activities and tasks for a given project determines that project's software process.

Table 3-3. Relationships between Polya activities, software activities, and software tasks

Polya Activity	**Software Activity**	**Software Task**
Understanding the Problem	Analysis	Problem identification
		Problem analysis
		Scope definition
		Requirements identification
		Decision analysis
Devising a Plan	Design	Network design
		Database design
		Process design
		Input design
		User-interface design
		Output design
Carrying Out the Plan	Implementation	Construction
		Testing
		Data migration
		System installation
		System changeover
Looking Back	Maintenance	Corrective maintenance
		Perfective maintenance
		Adaptive maintenance
		Preventive maintenance
		Refactorative maintenance

Table 3-1. Some of the problem-solving techniques available today (Part 1)

Technique	Description	Examples
Abstraction	Solving a problem by solving it in a model first and then applying that solution to the actual problem	Prototyping and network, data, and process modeling
Analogy	Solving a problem by adapting a solution that has solved a similar problem in the past	Locating and making use of similar coding solutions
Brainstorming	Solving a problem by spontaneously generating ideas for solving the problem, evaluating the generated ideas, and then applying the best idea for solving the problem	Requirements generation during Joint Application Development
Divide-and-Conquer	Solving a problem by breaking it down into smaller, more easily solvable problems and then assembling the solutions to those problems to form the overall solution	Decomposing a proposed software system into its functional components and then decomposing those functional components into their constituent procedures, functions, and/or methods
Drilling Down	Solving a problem by focusing on increasingly detailed parts of the problem until the source of the problem is located and the problem can be solved	Debugging an end-user reported logic error in a computer program
GROW Model	Solving a problem by stating a goal, evaluating the current reality with respect to that goal, identifying the obstacles that are impeding progress toward the goal and the options available for working around those obstacles, and then choosing the way forward	Managing a quickly approaching project deadline

Table 3-2. Some of the problem-solving techniques available today (Part 2)

Technique	Description	Examples
Hypothesis Testing	Solving a problem by assuming a solution and then trying to prove (or disprove) that solution	Writing and testing code
Means-Ends Analysis	Solving a problem by choosing an action at each step of the problem-solving process that moves the process closer to a solution	Scheduling a project
Research	Solving a problem by locating and analyzing information that is helpful in solving the problem	Identifying, analyzing, and comparing candidate solutions to an existing information problem
Reverse Brainstorming	Solving a problem by changing the focus from how to solve the problem to how to cause the problem or how to make the problem worse and then addressing those issues	Improving the change control process
Root Cause Analysis	Solving a problem by clearly defining the problem, identifying its root causes and resulting effects, and then correcting the root causes of the problem	Performing a cause-effect analysis to determine the root cause of end-user frustration
Trial-and-Error	Solving a problem by testing possible solutions to it until the right one is found	Design prototyping a system feature