Advanced Software Engineering

Dr. Cheng

Overview of Software Engineering and Development Processes

CSE870

1

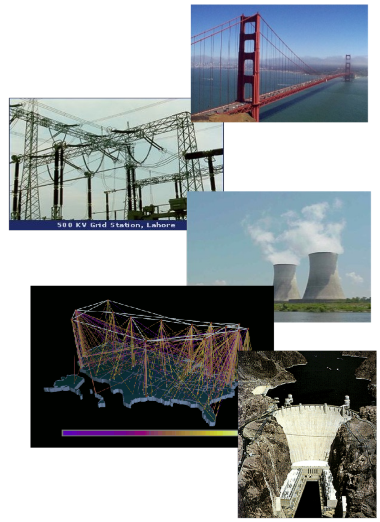
FYI

- Professor in Computer Science and Engineering
- **Here at MSU for > 20 years**
  - Trusted Systems Lab (formerly Software Engineering and Network Systems (SENS) Lab
  - BEACON: NSF Science and Technology Center ("Evolution in Action")
  - MSU Sociomobility REU
- **Research and Instruction areas:**
  - High-assurance systems
  - Model-driven engineering
  - Autonomic (self-adaptive) systems
  - Automotive Cybersecurity
  - Evolutionary-based computing
  - Recently, also working in following areas:
    - **Trusted AI/Machine Learning**
    - Model-Driven Engineering for Sustainable Systems (e.g., smart grid)
  - Work extensively with industrial collaborators (e.g., Ford, GM, Aerospace Corp., Continental Automotive, Motorola, Dataspeed, Groundspeed, BAE Systems, Siemens); NASA
  - International collaborations (sustainability, uncertainty interaction, SE4AI)

2

High-Assurance Autonomic Computing

- **Autonomic computing [2001]:** Promises self-managed and long-running systems with limited human guidance.
- Systems must continue to ***operate correctly*** during exceptional situations, upgrades, and evolution under ***uncertain*** conditions
- Need for assurance
  - hardware component failures
  - network outages
  - software faults
  - security attacks



4

New Scale

High-Assurance Cyberphysical Systems

Intelligent Transportation and Vehicle Systems

Requires increasingly complex systems

- Thousands of platforms, sensors, decision nodes, complex systems
- Connected through heterogeneous wired and wireless networks.

5

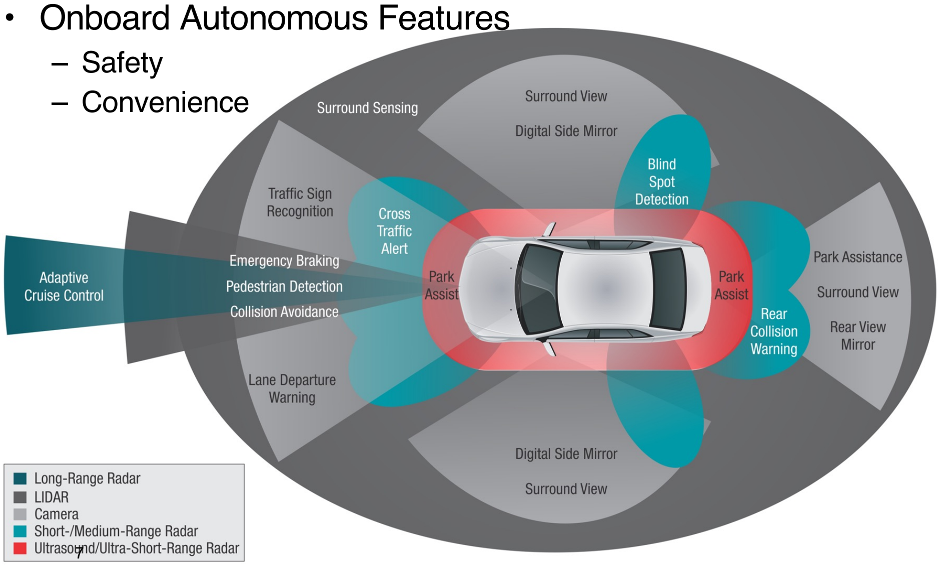
The future…

6

6

Now…
Advanced Driver-Assistance Systems

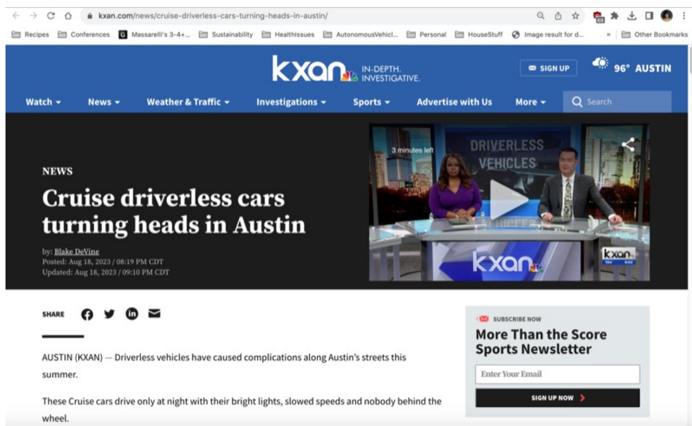
- Onboard Autonomous Features
  - Safety
  - Convenience



7

7

Reality Check

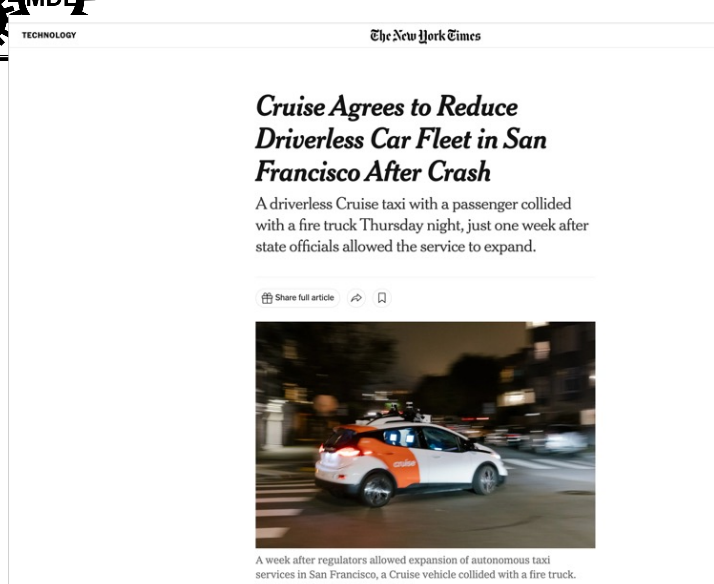


- https://www.kxan.com/news/cruise-driverless-cars-turning-heads-in-austin/

8

8

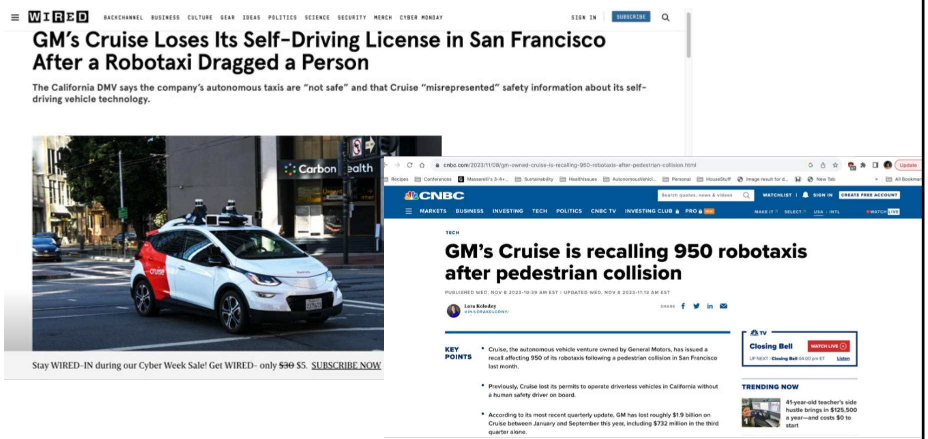
Reality Check



NYTimes, Aug. 18, 2023

9

Three years later…


GM's Cruise Loses Its Self-Driving License in San Francisco After a Robotaxi Dragged a Person

The California DMV says the company's autonomous taxis are "not safe" and that Cruise "misrepresented" safety information about its self-driving vehicle technology.

Stay WIRED-IN during our Cyber Week Sale! Get WIRED- only ~~$30~~ $5. SUBSCRIBE NOW

TECH

GM's Cruise is recalling 950 robotaxis after pedestrian collision

PUBLISHED WED, NOV 8 2023-10:39 AM EST | UPDATED WED, NOV 8 2023-11:13 AM EST

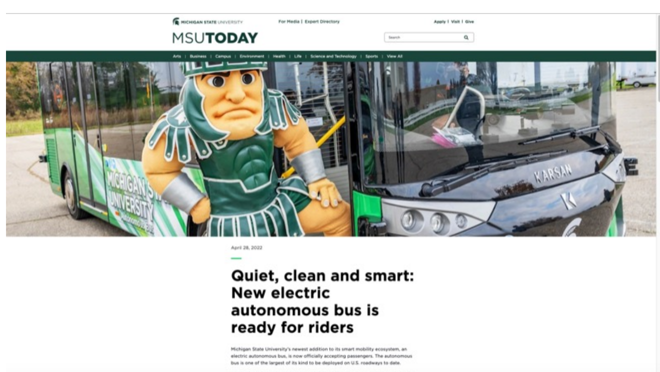
Lora Kolodny

KEY POINTS

- Cruise, the autonomous vehicle venture owned by General Motors, has issued a recall affecting 950 of its robotaxis following a pedestrian collision in San Francisco last month.
- Previously, Cruise lost its permits to operate driverless vehicles in California without a human safety driver on board.
- According to its most recent quarterly update, GM has lost roughly $1.9 billion on Cruise between January and September this year, including $732 million in the third quarter alone.

10

A little closer to home


MSU TODAY

April 28, 2022

Quiet, clean and smart: New electric autonomous bus is ready for riders

Michigan State University's newest addition to its smart mobility ecosystem, an electric autonomous bus, is now officially accepting passengers. The autonomous bus is one of the largest of its kind to be deployed on U.S. roadways to date.

- https://youtu.be/TuY6SUPd5Vk?si=gqVJDPE0_DCsNg22

11

11

Objectives of this course

Introduce ***industrial-strength*** software development:

- formal processes/artifacts for planning, specifying, designing, implementing, and verifying
- Individual and team-based development
- life-cycle issues and “umbrella” activities

Introduce key foundations underlying these activities

- E.g., requirements engineering
- E.g., software modeling
- E.g., *assurance*

12

What is Software Engineering?

- Systematic approach for developing software
- Methods and techniques to develop and maintain quality software to solve problems.

(Software Engineering: Methods and Management, Pfleeger, 1990)

- Study of the *principles* and *methodologies* for developing and maintaining software systems.

(``Perspectives on Software Engineering,'' Zelkowitz, 1978)

CSE870: Advanced Software Engineering (Cheng): Intro to Software Engineering 13

13

What is Software Engineering?

- *Practical* application of scientific knowledge in the design and construction of computer programs and the associated *documentation* required to develop, operate, and maintain them.

(``Software Engineering,'' Boehm, 1976)

- Deals with establishment of *sound engineering principles and methods* in order to *economically* obtain software that is reliable and works on real machines.

(``Software Engineering,'' Bauer, 1972)

Questions addressed by Software Engineering

- How do we ensure the quality of the software that we produce?
- How do we meet growing demand and still maintain budget control?
- How do we avoid disastrous time delays?

Why apply Software Engineering to Systems?

- Provide an understandable process for system development.
- Develop systems and software that are maintainable and easily changed.
- Develop robust software and system.
- Allow the process of creating computing-based systems to be repeatable and manageable.

16

Objectives of Course

- Provide exposure to leading-edge topics
  - Emphasize model-driven engineering
  - Emphasize requirements and design
  - Emphasize assurance of computing-based systems
- Provide hands-on experience to reinforce concepts
  - Homework assignments
  - Modeling and specification assignments
- Synthesize several topics into mini-projects
  - Programming/design Project with written component
  - Prepare presentation materials for lay audience.
- Overarching application theme: assurance for onboard automotive systems

17

Tentative Topics

- Requirements Engineering
- Model-driven engineering (UML)
- Architectural Styles
- Design Patterns
- Security
- Testing
- (Search-based Software Engineering)
- (Interplay between SE and ML)

Administrative Info

- Background Survey
- In-class activities
- Tentative Evaluation Mechanisms:

Exams (2)	60 %
In-class participation; Homework/Design Exercises	15%
Mini-Project(s)	25%

Assignments

- Syllabus: on course website:
  - https://www.cse.msu.edu/~cse870
- Homework #1: Due Jan. 22 (before class)
  - See Class website for details
  - Submit via D2L
- See AI Use policy on website

Administrative Activities

- Background Survey
- In-class activities

https://forms.gle/FG1baoQzHw7SYEM7A

PAUSE

Historical Perspective

- **1940s:** computers invented
- **1950s:** assembly language, Fortran
- **1960s:** COBOL, ALGOL, PL/1, operating systems

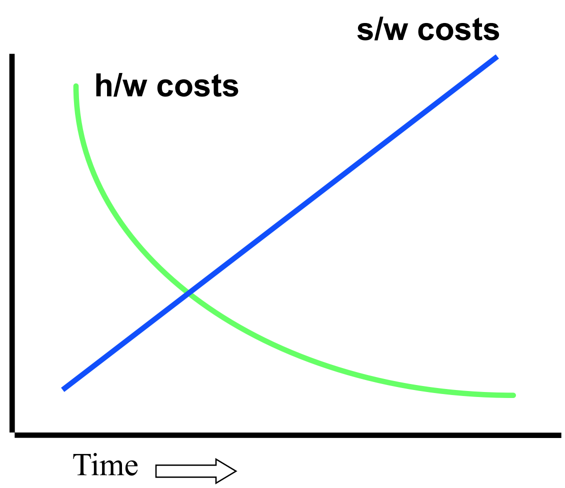
  1969: First conference on Software Eng
- **1970s:** multi-user systems, databases, structured programming

Historical Perspective (cont.)

- **1980s:** networking, personal computing, embedded systems, parallel architectures
- **1990s**: information superhighway, distributed systems, OO in widespread use**.**
- **2000s:** virtual reality, voice recognition, video conferencing, global computing, pervasive computing...
- **2010s:** EMRs, autonomous vehicles, new security awareness, ...

24

Hardware Costs vs Software Costs (% of overall costs)



25

Why is software so expensive?

- Hardware has made great advances
- But, software has made great advances ...
- We do the least understood tasks in software.
  - When task is simple & understood, encode it in hardware
  - Why?
- Demand more and more of software
  - Consider your cell phone

26

Size of programs continues to grow

- **Trivial:** 1 month, 1 programmer, 500 LOC,
  - Intro programming assignments
- **Very small:** 4 months, 1 programmer, 2000 LOC
  - Course project
- **Small:** 2 years, 3 programmers, 50K LOC
  - Nuclear power plant, pace maker
- **Medium:** 3 years, 10s of programmers, 100K LOC
  - Optimizing compiler

27

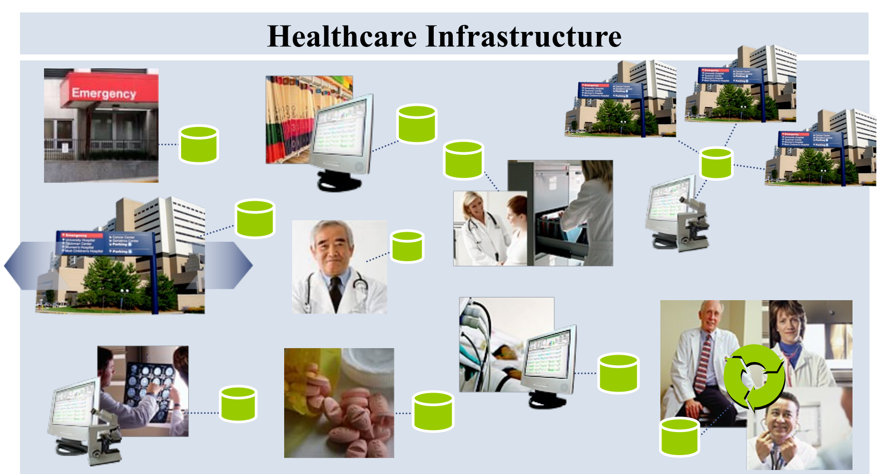
Size of programs continues to grow

- **Large**: 5 years, 100s of programmers, 1M LOC
  - MS Word, Excel
- **Very large:** 10 years, 1000s of programmers, 10M LOC
  - Air traffic control,
  - Telecommunications, space shuttle
- **Very, Very Large:** 15+ years, 1000s programmers, 35M LOC
  - W2K
- **Ultra-Large Scale:** ? years, ? developers distributed,
  - 1000s of sensors, decision units,
  - heterogeneous platforms, decentralized control
  - Intelligent transportation systems; healthcare systems

28

New Scale

Ultra-Large Scale SW-Intensive Systems (IOT/SOS)



Software Engineering Institute | Carnegie Mellon

Ultra-Large-Scale Systems
Linda Northrop, ICSE 2007

29

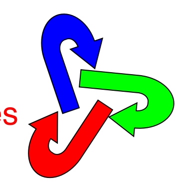
29

New Scale

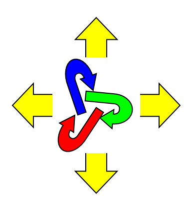
Intelligent Transportation and Vehicle Systems

The ULS Ecosystem

- Key elements:
  - Computing devices
  - Business and organizational policies
  - Environment (including people)



- Forces:
  - Competition for resources
  - Unexpected environmental changes
  - Decentralized control
  - Demand for assurance



Context: ''*Sufficient*'' System Health

High-level Objective:

- ***How to design a safe adaptive system with incomplete information and evolving environmental conditions***

- **Execution environment**
  - **How to model environment**
  - **How to effectively monitor changing conditions**
  - **Adaptive monitoring**
- Decision-making for dynamic adaptation
  - Decentralized control
  - Assurance guarantees (functional and non-functional constraints)
- Adaptation mechanisms:
  - Application level
  - Middleware level

32

What's the problem?

- Software cannot be built fast enough to keep up with
  - H/W advances
  - Rising expectations
  - Feature explosion
- Increasing need for high reliability software

33

What's the problem?

- Software is difficult to maintain
  "aging software"
- Difficult to estimate software costs and schedules
- Too many projects fail
  - Arianne Missile
  - Denver Airport Baggage System
  - Therac

34

NOTAMs

- Notice to Air Missions (est. 1947)
  - Essential pre-flight info for pilots, dispatchers
  - Weather on routes,
  - Runway/taxiways
  - Closed airspaces
- Jan 10-11, 2023-- SW-based failure:
  - Impact: 1600 cancelled flights; 9K delayed
  - Database corrupted

35

Crowdstrike Incident

- Crowdstrike: Onboard cybersecurity protection software (monitor/block)
  - 8.5M PCs
  - Customers: Google, Amazon, airlines, healthcare, banking, F500
- Incident:
  - Update to security SW (new template)
  - Contained logic error (”out of bounds error” – see Report)
- Widespread outage affecting Microsoft Windows
  - Due to Crowdstrike update
  - Affecting PCs running AWS and Azure platforms
  - 11:30 am. EDT, July 19, 2024
- Impacted 8.5M Windows devices
  - Costs: ~$1B (initial assessment) to $1.94B (just healthcare losses); $5.4B (Fortune 500 companies)[Parametrix Insurance]

36

36

Crowdstrike Timeline

- July 19, 2024:
  - 11:30 am. EDT : Initial Alert
  - 07:30 pm EDT: Update with impacts
- July 20 2024:
  - Guidance for outage and remediation
  - Logic error
  - Impact: Cyber threat actors leverage outage to execute malicious activity (e.g., phishing).
- July 21, 2024: 9:45 am EDT
  - Release recovery tool (uses USB drive to boot/repair systems)
- July 24, 2024: 12:00 pm EDT:
  - Release instructional video to remediate
  - Release list of domains impersonating CrowdStrike brand
- July 26, 2024: 12:30 pm EDT
  - Counter Adversary Operations: lists reports of malicious cyber activity (exploiting their outage)
- August 6, 2024:
  - Root Cause Analysis (RCA) report

37

37

Recovery from WinPE

1. Insert the USB key into an impacted device.
2. Reboot the device.
3. During restart, press F12 (or follow manufacturer-specific instructions for booting to BIOS).
4. From the BIOS boot menu, choose Boot from USB and continue.
5. The tool will run.
6. If BitLocker is enabled, the user will be prompted for the BitLocker recovery key including the dashes. The recovery key options are provided here. For third-party device encryption solutions, follow any steps provided by the vendor to gain access to the drive.
7. The tool will run the issue-remediation scripts as recommended by CrowdStrike.
8. Once complete, remove the USB drive and reboot the device normally.

Using Safe Boot Media

To repair an impacted device without using the BitLocker recovery key and if you have access to the local administrator account:

1. Insert the USB key into an impacted device.
2. Reboot the device.
3. During restart, press F12 (or follow manufacturer-specific instructions for booting to BIOS).
4. From the BIOS boot menu, choose Boot from USB and continue.
5. The tool runs.
6. The following message appears: "This tool will configure this machine to boot in safe mode. WARNING: In some cases you may need to enter a BitLocker recovery key after running."
7. Press any key to continue.
8. The following message appears: "Your PC is configured to boot to Safe Mode now."
9. Press any key to continue.
10. The machine reboots into safe mode.
11. The user runs repair.cmd from the root of the media/USB drive. The script will run the remediation steps as recommended by CrowdStrike.
12. The following message appears: "This tool will remove impacted files and restore normal boot configuration. WARNING: You may need BitLocker recovery key in some cases. WARNING: This script must be run in an elevated command prompt."
13. Press any key to continue.
14. The user repair will run and the normal boot flow will be restored.
15. Once successful, the user will see the following message: “Success. System will now reboot.”
16. Press any key to continue. The device will reboot normally.

"A perfect storm…"

- Customers: 29,000
  - Google, Amazon, Microsoft, Major Airlines, hospitals
  - PC-based users
- #1 security software used by Enterprise-based users
- Software Update: (improve security)
  - Logic error → system crash
  - "out of bounds" error!
- Required systems admin-level expertise to fix problem
  - Needed to fix one machine at a time (including reboot after fix)
- Impact:
  - 8.5M devices impacted
  - $5.4B in damages across multiple sectors
- Is this system safety critical?

40

Why is software engineering needed?

- To predict time, effort, and cost
- To improve software quality
- To improve maintainability
- To meet increasing demands
- To lower software costs
- To successfully build large, complex software systems
- To facilitate group effort in developing software

41

Software Engineering Phases

- Definition: What?
- Development: How?
- Maintenance: Managing change
- Umbrella Activities: Throughout lifecycle

Definition

- Requirements definition and analysis
  - Developer must understand
    - Application domain
    - Required functionality
    - Required performance
    - User interface

Definition (cont.)

- Project planning
  - Allocate resources
  - Estimate costs
  - Define work tasks
  - Define schedule
- System analysis
  - Allocate system resources to
    - Hardware
    - Software
    - Users

Development

- Software design
  - User interface design
  - High-level design
    - Define modular components
    - Define major data structures
  - Detailed design
    - Define algorithms and procedural detail

Development (cont.)

- Coding
  - Develop code for each module
  - Unit testing
- Integration
  - Combine modules
  - System testing

46

Maintenance

- Correction - Fix software defects
- Adaptation - Accommodate changes
  - New hardware
  - New company policies
- Enhancement - Add functionality
- Prevention - Make more maintainable

47

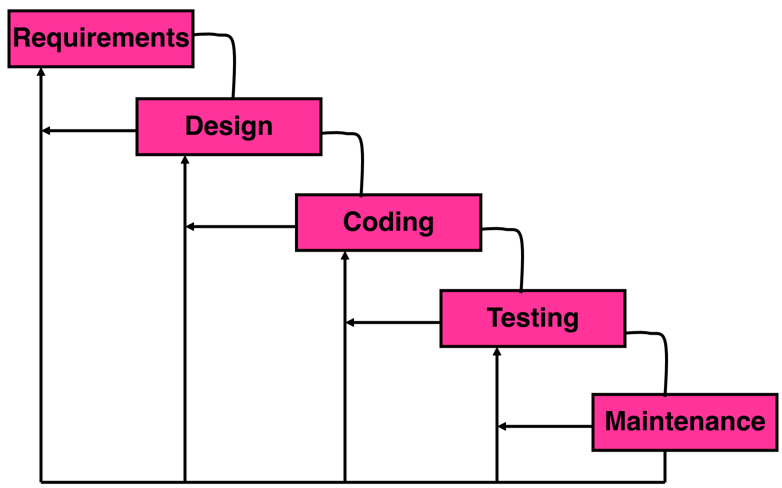
Umbrella Activities

- Reviews - assure quality
- Documentation - improve maintainability
- Version control - track changes
- Configuration management - integrity of collection of components

Development Process

- Step-by-step procedure to develop software
- Typically involves the major phases:
  - analysis
  - design
  - coding
  - testing

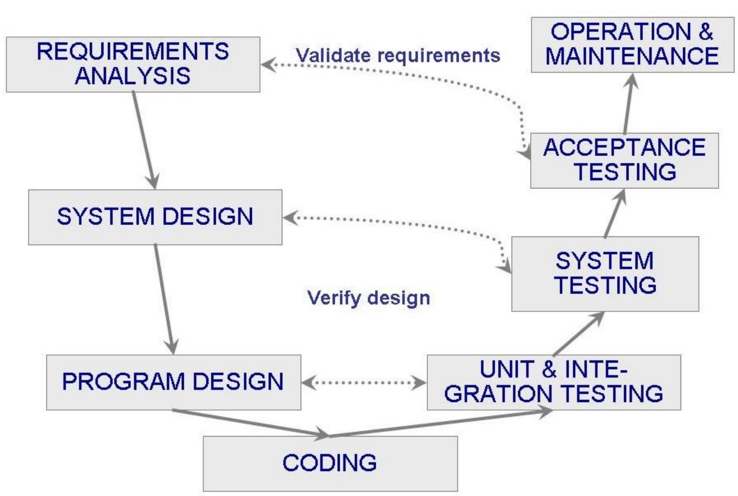
Waterfall Process Model



CSE870: Advanced Software Engineering (Cheng): Intro to Software Engineering 50

50

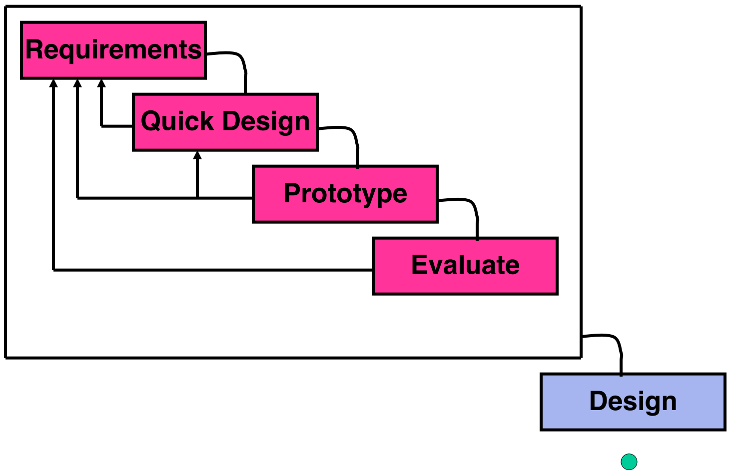
Software Process Models: V Model



Pfleeger and Atlee, Software Engineering: Theory and Practice (edited by B. Cheng) Chapter 2.

51

Software Process Models: Prototyping Process Model



52

When to use prototyping?

- Help the customer pin down the requirements
  - Concrete model to “test out”
  - Often done via the user interface
- Explore alternative solutions to a troublesome component
  - e.g., determine if an approach gives acceptable performance
- Improve morale
  - Partially running system provides visibility into a project

53

Software Process Models: Spiral Model

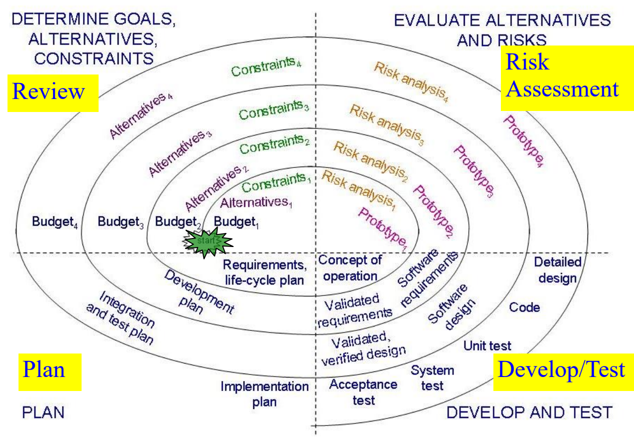


Figure 2.10 the spiral model.

Software Process Models: Agile Methods

- Emphasis on flexibility in producing software quickly and capably
- Agile manifesto
  - Value individuals and interactions over process and tools
  - Prefer to invest time in producing working software rather than in producing comprehensive documentation
  - Focus on customer collaboration rather than contract negotiation
  - Concentrate on responding to change rather than on creating a plan and then following it

Software Process Models
Agile Methods: Extreme Programming

- Emphasis on four characteristics of agility
  - Communication: continual interchange between customers and developers
  - Simplicity: select the simplest design or implementation
  - Courage: commitment to delivering functionality early and often
  - Feedback: loops built into the various activities during the development process

Software Process Models
Agile Methods: Twelve Facets of XP

- The planning game (customer defines value)
- Small release
- Metaphor (common vision, common names)
- Simple design
- Writing tests first
- Refactoring
- Pair programming
- Collective ownership
- Continuous integration (small increments)
- Sustainable pace (40 hours/week)
- On-site customer
- Coding standard

Software Process Models

Sidebar: When Extreme is Too Extreme?

- Extreme programming's practices are interdependent
  - A vulnerability if one of them is modified
- Requirements expressed as a set of test cases must be passed by the software
  - System passes the tests but is not what the customer is paying for
- Refactoring issue
  - Difficult to rework a system without degrading its architecture

60

Process Models

- Idealized views of the process
- Different models are often used for different subprocesses
  - may use spiral model for overall development
    - prototyping for a particularly complex component
    - waterfall model for other components

61

Capability Maturity Model

- Level 1: Initial
  - ad hoc
  - success depends on people
- Level 2: Repeatable
  - track cost, schedule, functionality
- Level 3: Defined
  - use standardized processes
- Level 4: Managed
  - collect detailed metrics
- Level 5: Optimizing
  - continuous process improvement
  - "built-in" process improvement

Software Engineering Institute:
http://www.sei.cmu.edu/cmm/

Why is software development so difficult?

- Communication
  - Between customer and developer
    - Poor problem definition is largest cause of failed software projects
  - Within development team
    - More people = more communication
    - New programmers need training
- Project characteristics
  - Novelty
  - Changing requirements
    - 5 x cost during development
    - up to 100 x cost during maintenance
  - Hardware/software configuration
  - Security requirements
  - Real time requirements
  - Reliability requirements

Why is software development difficult? (cont.)

- Personnel characteristics
  - Ability
  - Prior experience
  - Communication skills
  - Team cooperation
  - Training
- Facilities and resources
  - Identification
  - Acquisition
- Management issues
  - Realistic goals
  - Cost estimation
  - Scheduling
  - Resource allocation
  - Quality assurance
  - Version control
  - Contracts

Summary

- Software lifecycle consists of
  - Definition (what)
  - Development (how)
  - Maintenance (change)
- Different process models concentrate on different aspects
  - Waterfall model: maintainability
  - Prototype model: clarifying requirements
  - Spiral model: identifying risk
- Maintenance costs much more than development

Bottom Line

- U.S. software is a major part of our societal infrastructure
  - Costs upwards of $200 billion/year
- Need to
  - Improve software quality
  - Reduce software costs/risks

In-class activity

- In-class Poll:

https://forms.gle/Vt6Q2JqBhB4jSs2UA