CMPE 220

Class 16 – Developing System Software

Software Engineering Considerations

Types of Software

- Applications (specializations)
    - Financial
    - Scientific / Medical
    - Data Science / Big Data / Statistics / Modeling
    - Educational
    - Games
- Web Applications
- Software Development Tools
- Servers (Web, Database, Email, FTP, etc)
- Operating System Software
- Embedded Systems (applications and systems)

Which is Hardest?

- Each has its own challenges – but the skills are different
  - Some people find multi-threaded, asynchronous coding to be particularly difficult
- System software engineers are paid slightly more
- Median Salaries (US Department of Labor, 2019)
  - Application software developer: $103,620
  - System software developer: $110,000

Types of Software Development

	Applications	Web Applications	Software Development Tools	Servers (FTP, Web, Database, Email)	Operating Systems	Embedded Systems
Knowledge of application	YES	YES				YES
Knowledge of hardware			YES		YES	YES
Knowledge of standards		MAYBE	YES	YES	MAYBE	
Performance Sensitive		YES		YES	YES	YES
Memory Sensitive					YES	YES
Process Management		MAYBE		YES	YES	

Software Development Processes

	Applications	Web Applications	Software Development Tools	Servers (FTP, Web, DB, Email)	Operating Systems	Embedded Systems
Formal Specifications	VARIES	VARIES	YES	YES	YES	VARIES
Languages	ANY	PHP, JavaScript, Ruby, Python	ANY	C, C++, JAVA	C, C++, JAVA, Assembly	Assembly, C, C++, Python, Ada, Rust, Go
Release cycles	RAPID	VERY RAPID	SLOW	MODERATE	SLOW	SLOW
Testing (Risk)	VARIES	VARIES	EXTENSIVE	EXTENSIVE	EXTENSIVE	VARIES

Software Development Life Cycle (SDLC)

- A software development life cycle (SDLC) model is a conceptual framework describing all activities in a software development project from planning to maintenance.
- Major Models
  - None
  - Waterfall (traditional)
  - Spiral
  - Agile / Scrum
  - DevOps

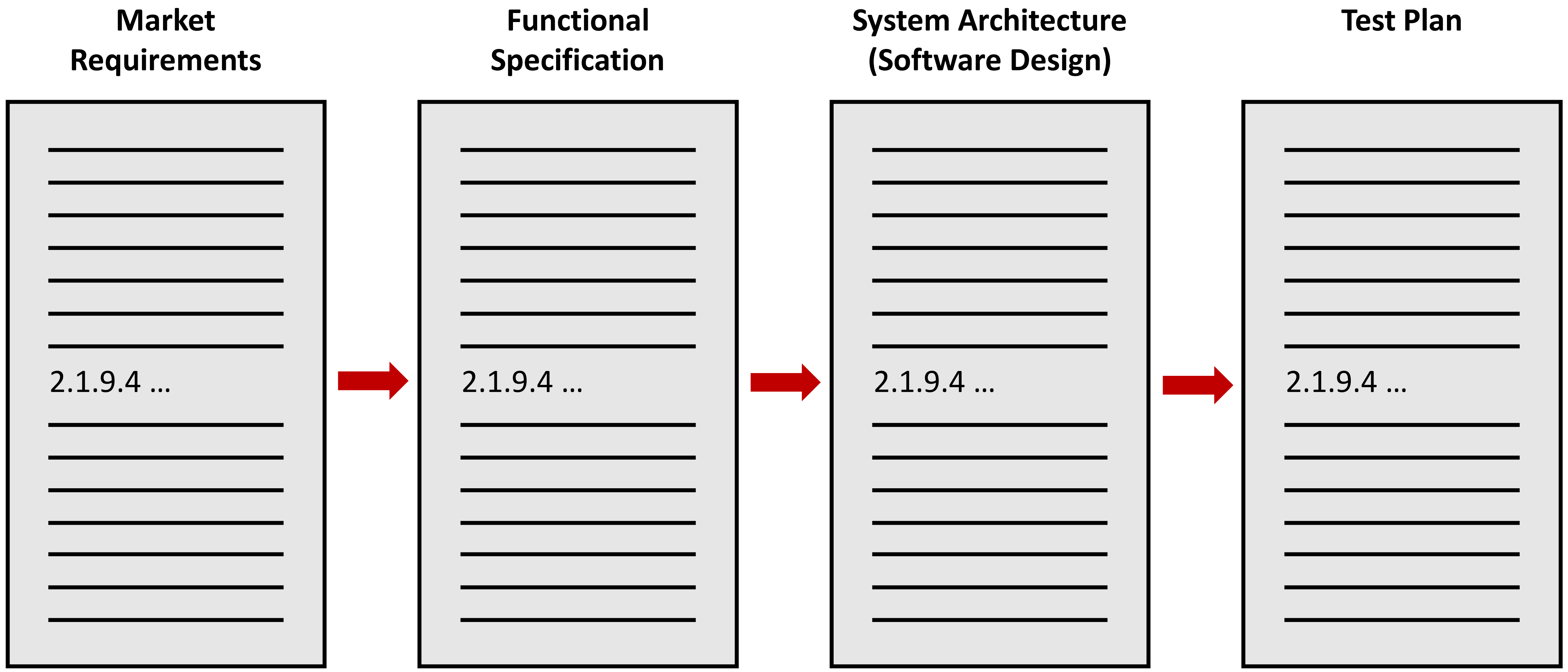
SDLC: None

- 1940s to mid-1950s
- Programs were small
- Features were limited
- Few programmers per project
- The entire project could be planned and tracked on a notepad

SDLC: Waterfall (traditional)

- First formal description of an SDLC - 1956
- First use of the term Waterfall -1976
- US Department of Defense standardizes Waterfall model – 1985
- Phases are linear:
  - Gather Requirements
  - Write Functional Specification
  - Write Internal Design
  - Develop Code
  - Test
  - Release
  - Maintain

Waterfall Documentation



Waterfall Pros and Cons

<u>Pros</u>

- Easy to understand
- Encourages large up-front investment in requirements analysis and planning
- Allows separation of teams
- Lends itself to tracking
  - Each requirement -> feature -> implementation -> test
- Well suited for delivery contracts
- Well suited for legal compliance

Waterfall Pros and Cons

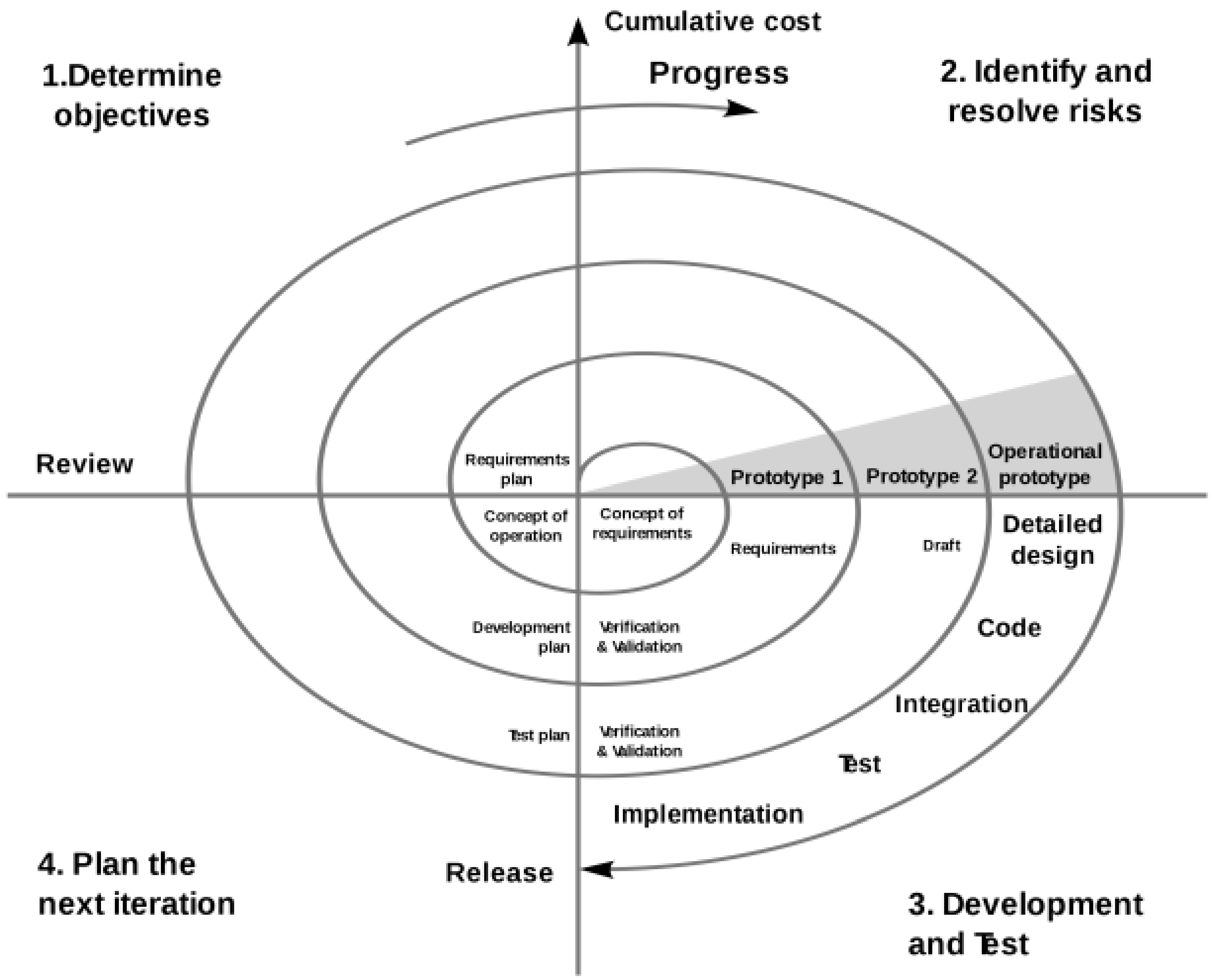
Cons

- Requirements may be wrong
- Functional Specifications may dictate poor design
- Little communication between teams
- Changes are difficult
- Testing is late in the cycle
- Tracking can be slow and expensive
- Release cycles are typically long
- Overly restrictive – and may be ignored

SDLC: Spiral

- First described by Barry Boehm - 1986
- A series of short waterfalls
- Overall project is loosely defined
- Project plan calls for a series of phases; each phase presents an opportunity to refine and correct the previous phase

SDLC: Spiral


Cumulative cost
Progress
1.Determine objectives
2. Identify and resolve risks
Review
Requirements plan
Concept of operation
Concept of requirements
Prototype 1
Prototype 2
Operational prototype
Requirements
Draft
Detailed design
Development plan
Verification & Validation
Code
Integration
Test plan
Verification & Validation
Test
Implementation
Release
4. Plan the next iteration
3. Development and Test

Spiral Pros and Cons

Pros

- All of the Pros of the traditional Waterfall model
- Catches incorrect requirements earlier
- Changes are easier
- Testing is earlier in the cycle
- Well suited to longer projects

Spiral Pros and Cons

Cons

- Slower and more expensive than traditional waterfall
- More difficult to track
- Harder to tie to delivery contracts

SDLC: Rapid Application Development (RAD)

- Use existing components and tools to develop and release a “prototype” very quickly
- May fill competitive gaps
- May be used to get user feedback
- May be a throw-away effort

RAD Pros and Cons

Pros

- Fast!
- Takes advantage of existing tools and components
- Mitigates risk

RAD Pros and Cons

Cons

- Requires experienced, skilled developers
- Expensive (throw-away)
- May commit organization to support of throw-away code

SDLC: Agile

Steps to Agile Development

- Mostly streamlining the process
- Rapid Application Development (RAD) – 1991
- Dynamic Systems Development Methodology (DSDM) – 1994
- Scrum – 1995
- Extreme Programming (XP) – 1996
- The "Agile Manifesto" – 2001
  - Written by 17 software developers at a resort in Snowbird, Utah

SDLC: Agile

- Continuous stakeholder involvement
- Short, overlapping define -> implement -> test cycles (sprints)
- Team approach: stakeholders, designers, developers, testers work together throughout the project
- Continuous communication
- Frameworks
  - Scrum – an organizational framework for managing agile development
  - Kanban – a visual workflow management system allows all team members to visualize project status

Agile Pros and Cons

<u>Pros</u>

- High degree of stakeholder satisfaction
- Easy to make changes
- Bugs are found early
- Allows frequent releases to roll out new features and bug fixes

Agile Pros and Cons

Cons

- Difficult to manage
- Difficult to scale
- Encourages feature creep…. Projects can easily “go off the rails”
- May be expensive (ties up teams, frequent meetings)
- Poorly suited for delivery contracts
- “Continuous change” is hard on:
  - People who write documentation & training materials
  - Support staff
  - Users!

SDLC: DevOps

- DevOps coined by Belgian developer Patrick Debois – 2009
- A methodology that integrates development, deployment, and ongoing operations (IT) – continuous deployment
- The development cycle doesn’t end when the product is released
- Emphasis on automated delivery tools

DevOps Pros and Cons

<u>Pros</u>

- All advantages of Agile:
  - High degree of stakeholder satisfaction
  - Easy to make changes
  - Bugs are found early
  - Allows frequent releases to roll out new features and bug fixes
- Significant improvements in operations

DevOps Pros and Cons

Cons

- All Cons of Agile:
  - Difficult to manage
  - Encourages feature creep…. Projects can easily “go off the rails”
  - May be expensive (ties up teams, frequent meetings)
  - Poorly suited for delivery contracts
  - “Continuous change” is hard
- Requires automated testing and deployment tools

DevOps: A Partial Solution

Unserved Communities

- Technical support
- Tech Writers
- Marketing
- Customers

Additional Wrinkles

Lean Software Development (LSD)

- An Agile variant designed to reduce wasted effort
- Also called MVP, or Minimum Viable Product strategy
- Release a bare-minimum version of the product (or feature). Learn what users like and don’t like, or want to see added, then iterate based on feedback.
- Anything that does not contribute to the MVP is eliminated.

Lean History

- Based on techniques developed at Japanese auto-maker Toyota in the 1960s
- Developed by Taiichi Ohno, the Toyota Production System (TPS) was aimed at minimizing waste
- Popularized as a development methodology in the book *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*, by James Womack and Danial Jones (2003)

Developer Testing

Traditional model

- Developers write code, quality assurance (QA) tests the code
- Developers typically perform basic unit/functional testing
- QA is responsible for overall system testing

Traditional Testing Model: Problems

- Little incentive for developers to create high-quality code
- Duplication of effort (developer tests and QA tests)
- Bugs are found late in the release cycle
- Multiple fix & test cycles required
- Potential for disastrous schedule slips!

Developer Testing

- Developers and development team is responsible for:
  - Creating and submitting unit/functional tests
  - Automation frameworks allow easy “regression tests”
    - Commercial / proprietary tools
    - Make!
  - Static Code Analysis
    - “Lint”
    - Coding standards
    - Specific security or performance issues
  - Source Code Control: Check-in “gates”
- Development team is responsible for:
  - Integration testing and submitting tests

Risks / Cons of Developer Testing

- Expensive and time time-consuming
- Developers are not expert testers
- Morale issues

Software Development Processes: Fit

- The best "fit" for developing a particular type of software is a matter of opinion, and based on specific circumstances. The table below is intended as a basis for discussion.

	Applications	Web Applications	Software Development Tools	Servers (FTP, Web, DB, Email)	Operating Systems	Embedded Systems
Waterfall	VARIES	NO	YES	YES	YES	YES
Spiral	VARIES	NO	YES	YES	YES	YES
Agile	VARIES	YES	NO	YES	NO	NO
DevOps	VARIES	YES	NO	YES	YES	NO