

Information Systems Development Life Cycle (SDLC)

- Unit 2 -

Agenda

- Information System Development Lifecycle (SDLC) Models and Methods
- IT Auditor's Role
- In-Class Exercise
- Quiz
- Next week...

Information System Development Lifecycle Methods

Software development lifecycle (SDLC) models are formal management processes for guiding the development of information systems

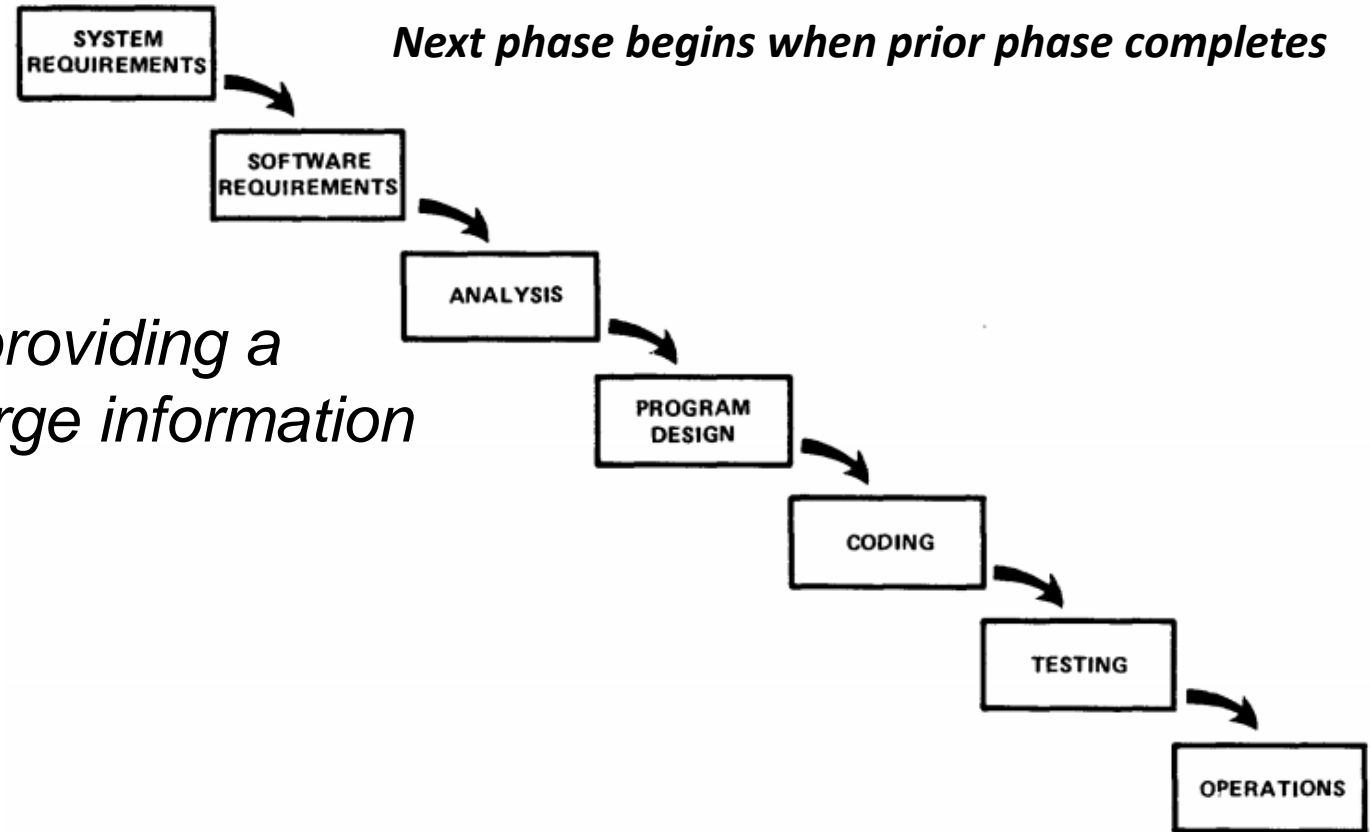
These include:

- **Waterfall Models** – 1970's and 1980's
 - Waterfall
 - Structured Systems Analysis and Design Method (SSADM)
- **Spiral and Iterative Models** – 1980's and 1990's
 - Structured Rapid Prototyping
 - Rapid Application development (RAD)
 - Agile Models
 - Rational Unified Model

If execution of the SDLC methodology is inadequate, however, the project may fail to meet business and user needs.

- *IS Auditor is responsible for verifying that the SDLC model is appropriate for the project's goals and is properly implemented*

Waterfall SDLC Model

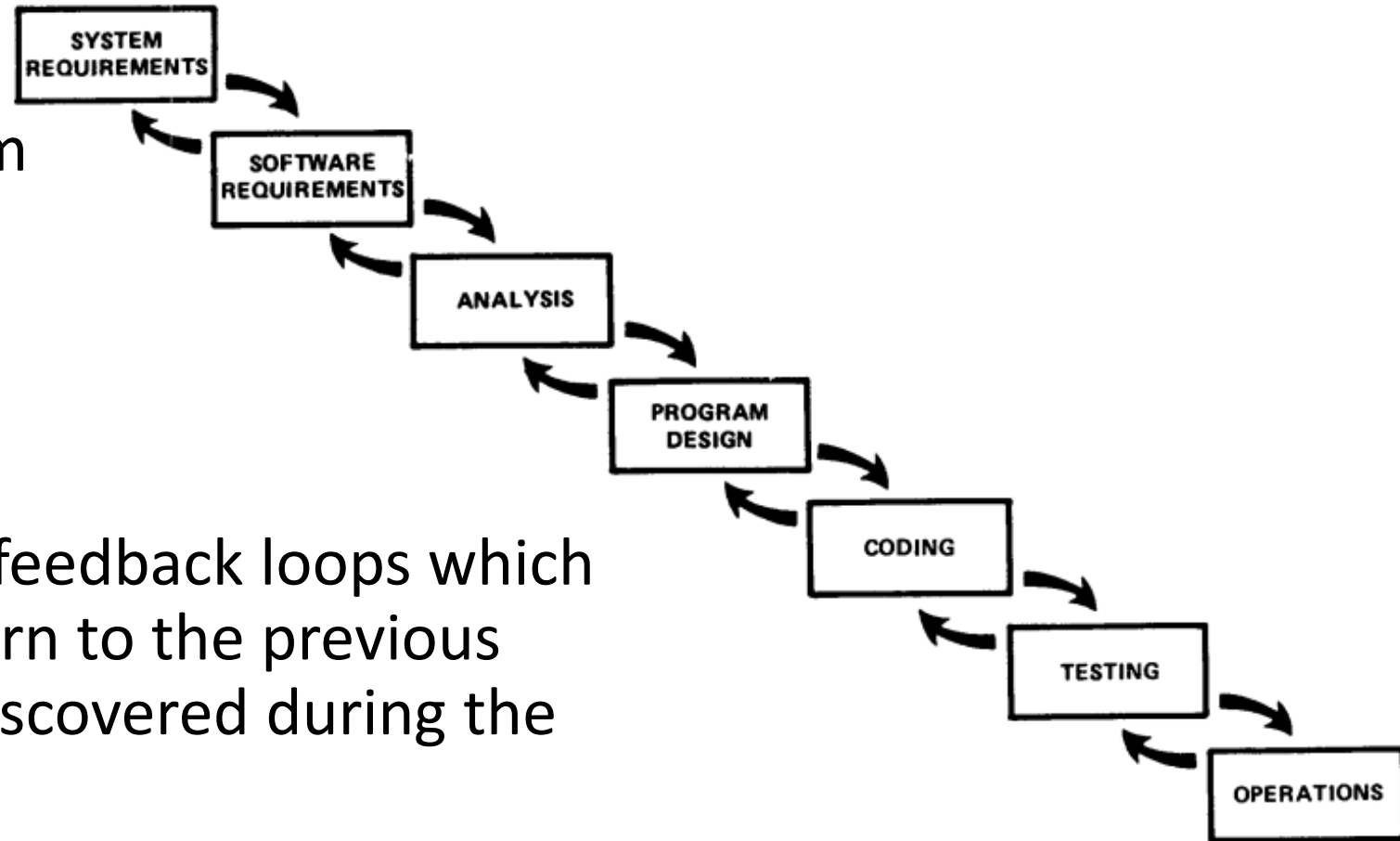


A seven-stage structured approach providing a systematic process for developing large information systems

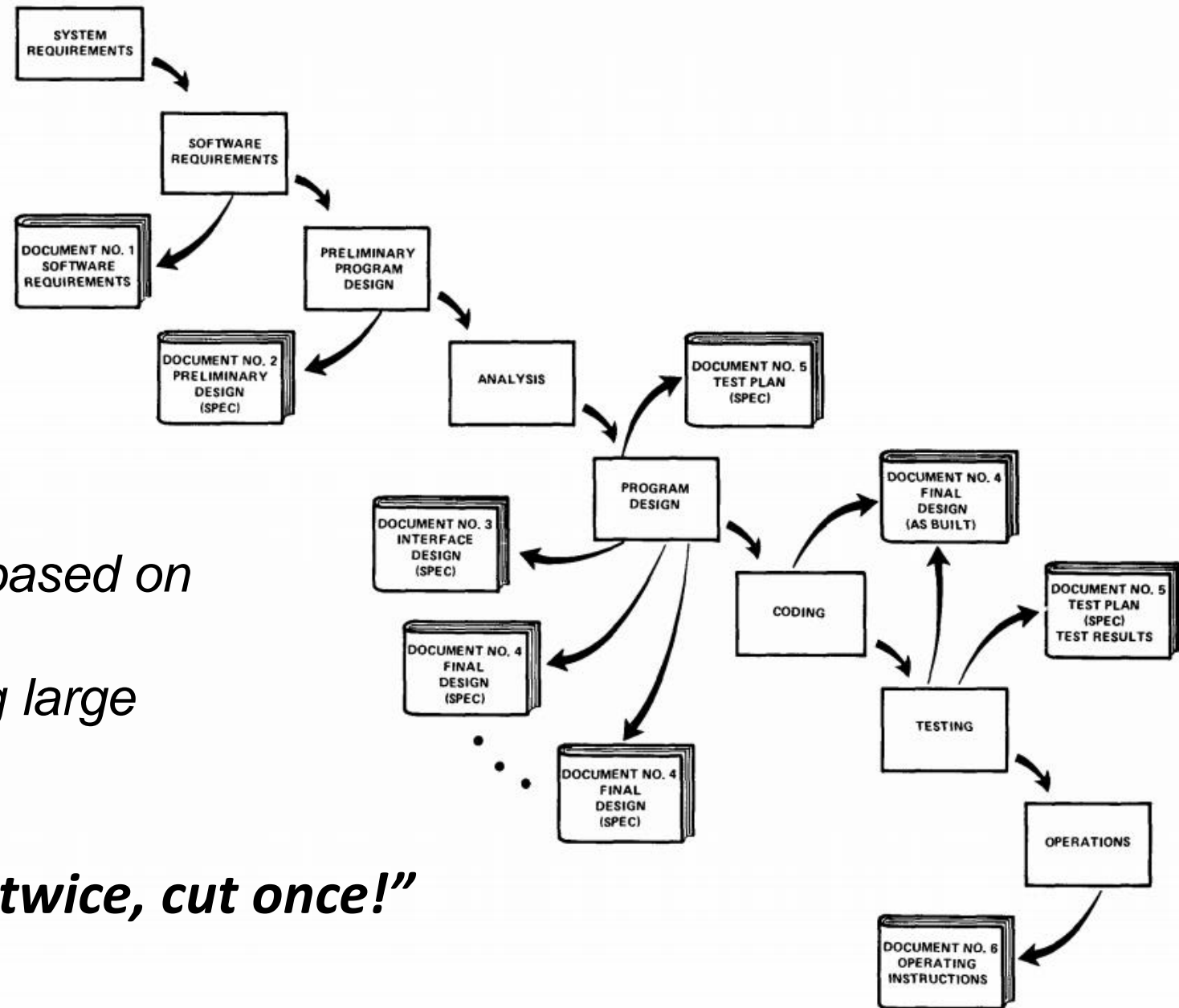
Royce, W.W. (1970) "Managing the Development of Large Software Systems" , in Proceedings of IEEE WESCON, pp.1-9

Waterfall SDLC Model

- As each information system development stage is completed, the project moves into the next phase
- Backward arrows indicate feedback loops which allow development to return to the previous phase to correct defects discovered during the subsequent phase



Waterfall Model



A structure systematic approach based on formal documentation providing a systematic process for developing large information systems

“Measure twice, cut once!”

Royce, W.W. (1970) “Managing the Development of Large Software Systems” , in Proceedings of IEEE WESCON, pp.1-9

Waterfall Model – Pros and Cons

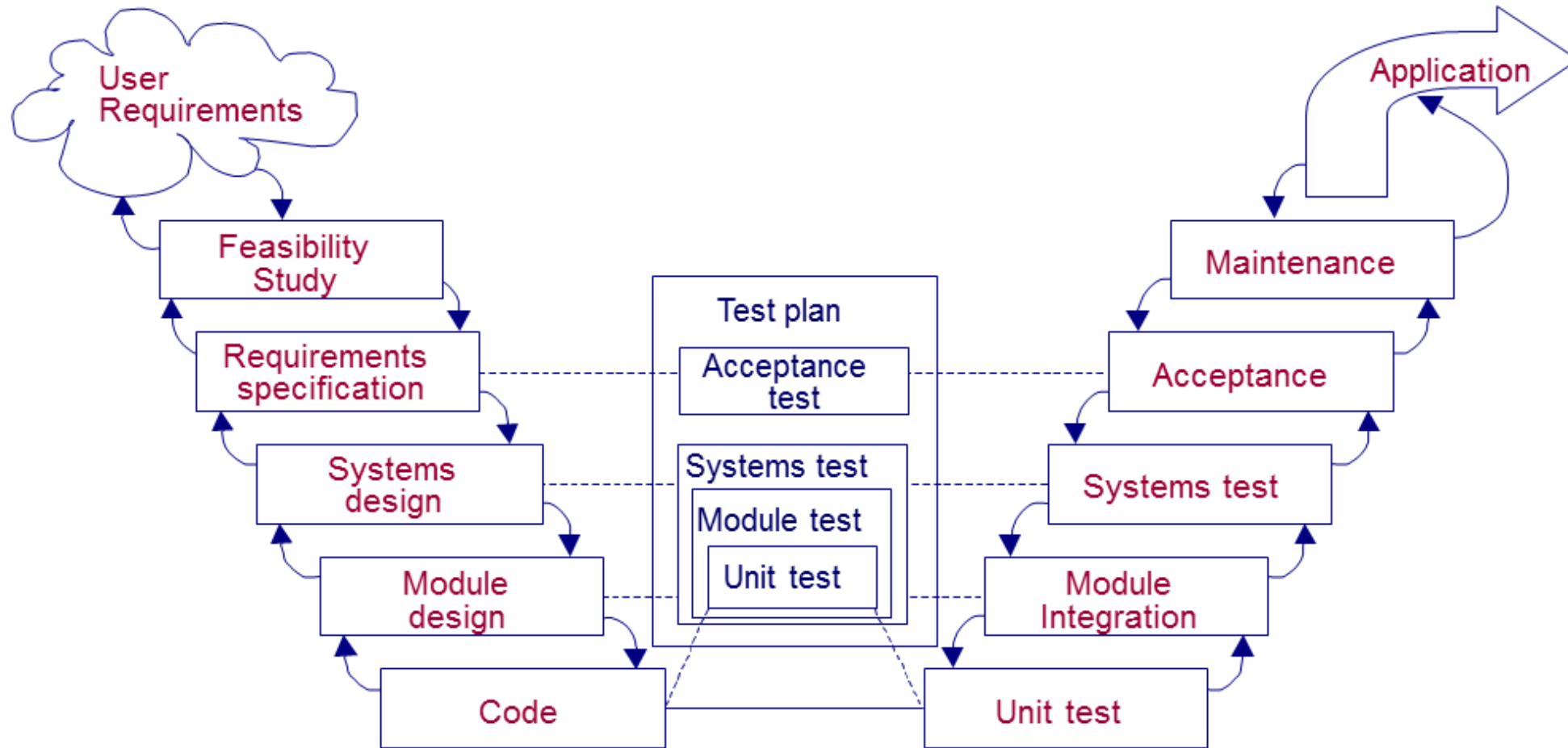
Pros

- Structured
- Worked well when requirements were well defined
- For large projects with enough time available

Cons

- Too much documentation
- Making changes becomes difficult during SDLC
- Poor speed to market
- Delayed implementation too long

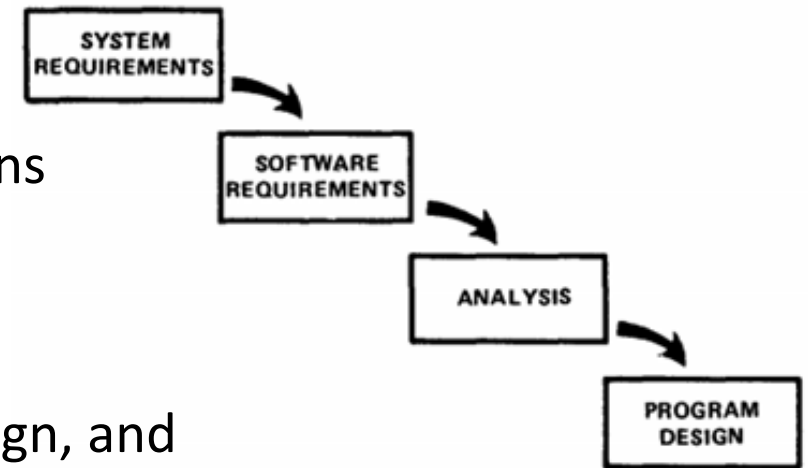
Waterfall Model with Verification and Validation



Morris - "The Management of Projects" 1994

History - Information System Development Methods

- 1970 – Waterfall Model
- **1980's – Structured Systems Analysis and Design Method (SSADM)**
 - A structured systems approach that applies the Waterfall Model to the analysis and design of information systems
 - Produced for the Central Computer and Telecommunications Agency, a UK government office concerned with the use of technology in government
 - Pinnacle of rigorous document-led approach to system design, and contrasts with more contemporary iterative methods
 - Built on different prior structured analysis and development methods, such as: structured design, structured method, and structured analysis



Structured Systems Analysis and Design Method (SSADM)

Techniques:

1. Logical data modeling

- Process of identifying, modeling and documenting the data requirements of the system being designed
 - Result is a **data model** containing:
 - **Entities** - things about which a business needs to record information
 - **Attributes** - facts about the entities
 - **Relationships** - associations between the entities

2. Data Flow Modeling

- Process of identifying, modeling and documenting how data moves around an information system
- Examines:
 - **Processes** (activities that transform data from one form to another)
 - **Data stores** (the holding areas for data)
 - **External entities** (what sends data into a system or receives data from a system),
 - **Data flows** (routes by which data can flow)

3. Entity Event Modeling

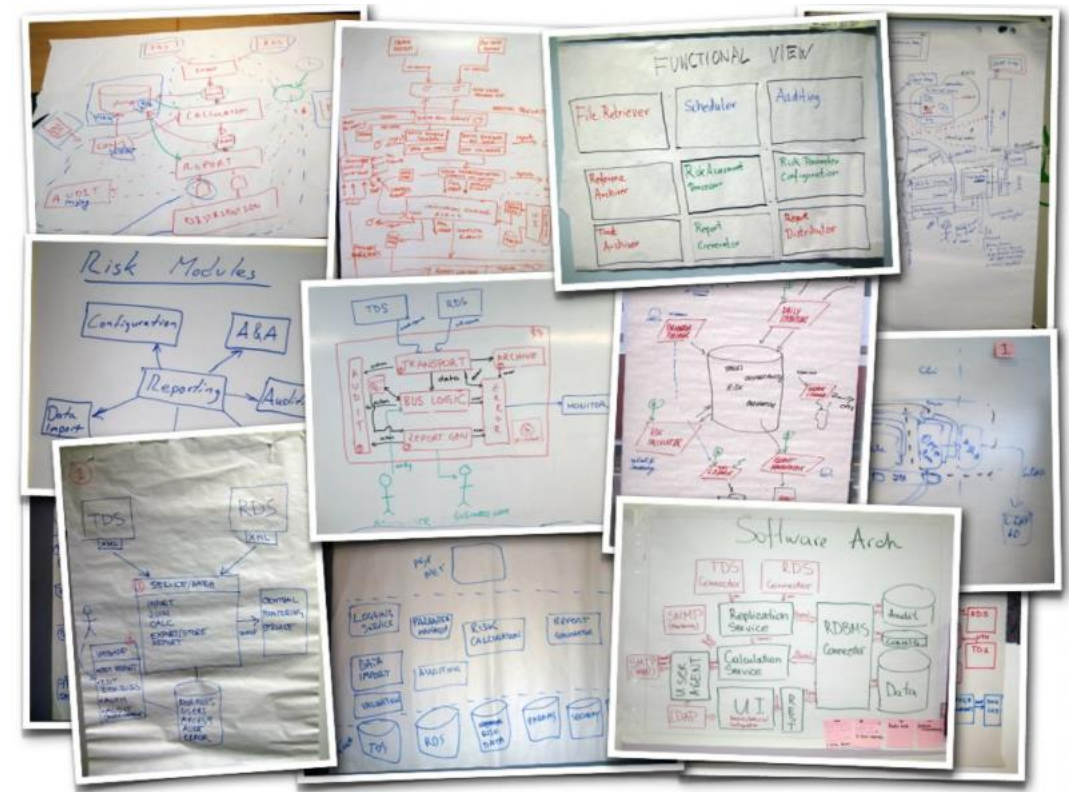
- A 2-part process:
 1. **Entity Behavior Modeling** - identifying, modeling and documenting the events that affect each entity and the sequence (or life history) in which these events occur
 2. **Event Modeling** - designing for each event the process to coordinate entity life histories

Structured Systems Analysis and Design Method (SSADM)

Stages:

The SSADM method involves the application of a sequence of analysis, documentation and design tasks concerned with the following:

0. Feasibility study
1. Investigation of current environment
2. Business system options
3. Requirements specification
4. Technical system options
5. Logical design
6. Physical design



Structured Systems Analysis and Design Method (SSADM)

Stage 0. Feasibility study

Investigation focuses on the goals and implications to determine if a project is feasible

- For very small scale projects this may not be necessary if scope of the project is easily understood
- In larger projects, feasibility may be done informally because there is no time for a formal study or because the project is a “must-have” and will have to be done one way or the other

Areas of consideration:

1. Technical – is the project technically possible?
2. Financial – can the business afford to carry out the project?
3. Organizational – will the new system be compatible with existing practices?
4. Ethical – is the impact of the new system socially acceptable?

To answer these questions, the feasibility study is effectively a condensed version of a fully blown systems analysis and design

- Users and requirements are analyzed to some extent, business options are identified, some details of the technical implementation may be included.

Feasibility study product is a formal feasibility study document

Contains preliminary models that have been constructed, details of rejected options and the reasons for their rejection

https://en.wikipedia.org/wiki/Structured_systems_analysis_and_design_method

Structured Systems Analysis and Design Method (SSADM)

Stage 1. Investigation of current environment

Recognizes that in almost all cases there is some form of current system even if it is entirely composed of people and paper

Based on a combination of interviewing employees, circulating questionnaires, observations and existing documentation

Product of this stage is the analyst's full understanding of the system in its "As Is" condition at the start of the project

Structured Systems Analysis and Design Method (SSADM)

Stage 2. Business system options

The analyst leverages an understanding of the existing system to decide on the overall design of the new system and develop a set of business system options

- These are different ways in which the new system could be produced varying from doing nothing to throwing out the old system entirely and building an entirely new one
- The analyst may hold a brainstorming session so that as many and various ideas as possible are generated

Ideas are collected and options presented to the user.

Options:

- Degree of automation
- Boundary between the system and the users
- Distribution of the system, for example, is it centralized to one office or spread out across several?
- Cost/Benefit
- Impact of the new system

Where necessary, the option will be documented with a logical data structure and high-level data-flow diagram

Users and analyst together choose a single business option

- This may be one of the ones already defined or may be a synthesis of different aspects of the existing options

Product of this stage is the single selected business option together with all the outputs of the feasibility stage

Structured Systems Analysis and Design Method (SSADM)

Stage 3. Requirements specification

The analyst develops a **full logical specification** of what the new system must do

- *Does not say how the system will be implemented, but rather describes what the system will do*
- *Must be free from error, ambiguity and inconsistency*

To produce the logical specification, the analyst builds logical models for both:

- **Data-Flow Diagrams (DFDs)**
- **Logical Data Model (LDM)**
 - Consisting of the Logical Data Structure (referred to in other methods as **Entity Relationship** diagrams) and full descriptions of the data and its relationships

These are used to produce:

- **Functional specifications** for every function which the users will require of the system
- Entity Life-Histories (ELHs) which describe all events through the life of an entity,
- Effect Correspondence Diagrams (ECDs) which describe how each event interacts with all relevant entities

These are continually matched against the requirements and where necessary – requirements are added to and completed

Product of this stage is a complete **requirements specification** document made up of:

- **Data catalogue**
- **Requirements catalogue**
- **Processing specification** which in turn is made up of:
 - user role/function matrix
 - function definitions
 - required logical data model
 - entity life-histories
 - effect correspondence diagrams

https://en.wikipedia.org/wiki/Structured_systems_analysis_and_design_method

Structured Systems Analysis and Design Method (SSADM)

Stage 4. Technical system options

This stage is the first step towards a physical implementation of the new system

Like Business System Options (Stage 2), this stage generates, considers, and narrows down a large number of options for implementing the new system to two or three to present to the user from which the final option is chosen or synthesized

However, the considerations are quite different, being:

- Hardware architectures
- Software to use
- Cost to implement
- Staff required
- Physical limitations of the space occupied by the system
- Distribution and networks which may be require
- Overall format of the human computer interface

All of these aspects must also conform to any constraints imposed by the business such as regulations, available money, standardization of hardware and software, integration with enterprise architecture

Product of this stage is a chosen technical system option

Structured Systems Analysis and Design Method (SSADM)

Stage 5. Logical design

Concentrates on requirements for the human computer interface

Specifies the main methods users will interact with the system, in terms of:

- Menu and command structures
- User dialogues

Analyzes effects of events in

- Updating the system
- Making inquiries about the data

Both of these use the events, function descriptions and effect correspondence diagrams produced in Stage 3 to determine precisely how to update and read data in a consistent and secure way

Product of this stage is the logical design, made up of:

- Data catalogue
- Required logical data structure
- Logical process model – includes dialogues and model for the update and inquiry processes

Structured Systems Analysis and Design Method (SSADM)

Stage 6. Physical design

Final stage where all the logical specifications of the system are converted to descriptions of the system in terms of real hardware and software

This is a very technical stage:

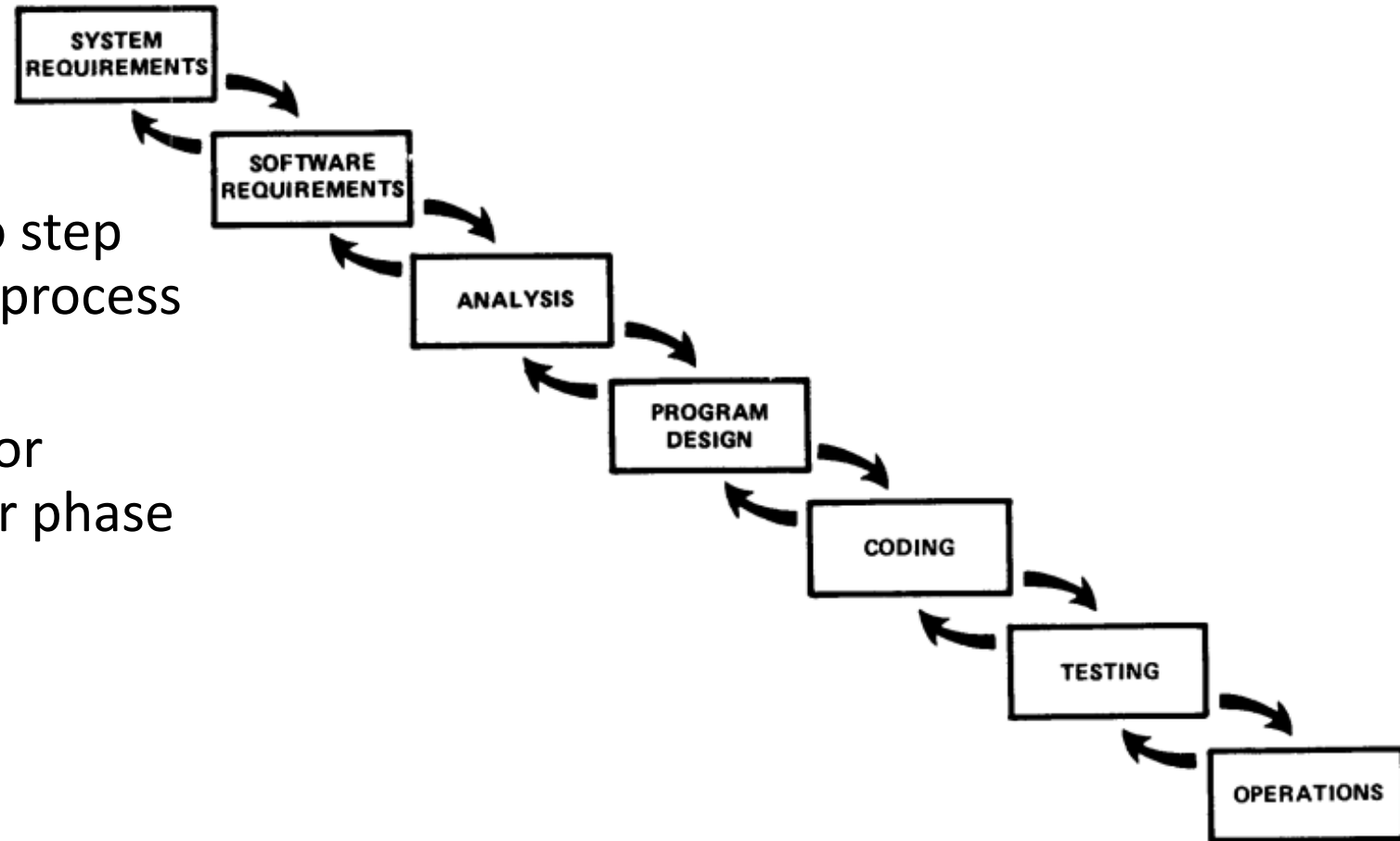
- Logical data structure is converted into a physical database structures
- Exact structure of functions and how they are implemented is specified
- Physical data structures are optimized to meet size and performance requirements

Product of this stage is a complete Physical Design which tells software engineers how to build the system in specific details of hardware and software and to the appropriate standards

SSADM's Waterfall SDLC Model

Major criticism

- Model allows developers to step back only one phase in the process
- Does not make provisions for discovery of errors at a later phase in the development cycle

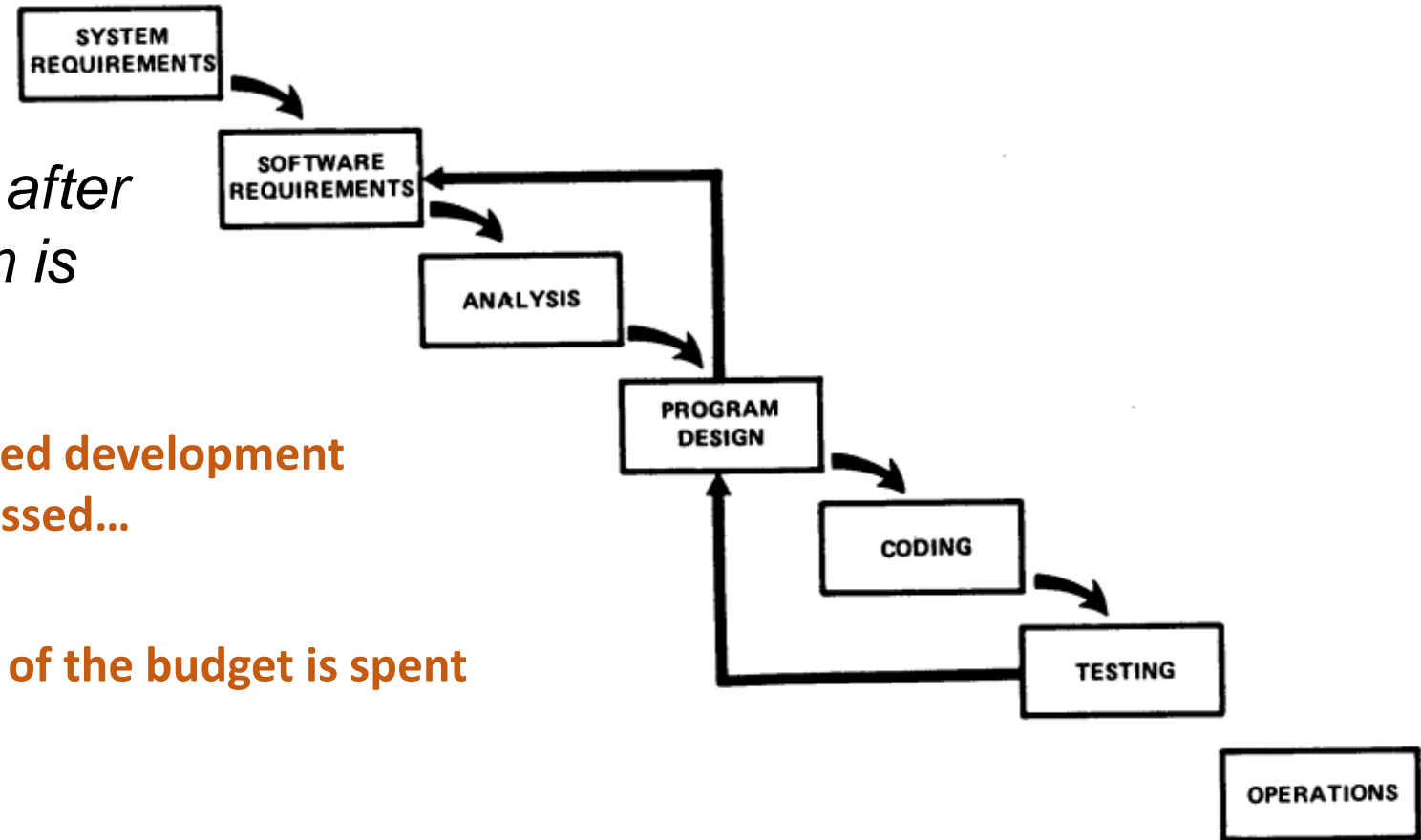


Problems with Waterfall and SSDM IS Development

Software design not tested until after coding of the information system is complete

...after most of the scheduled development project milestones have passed...

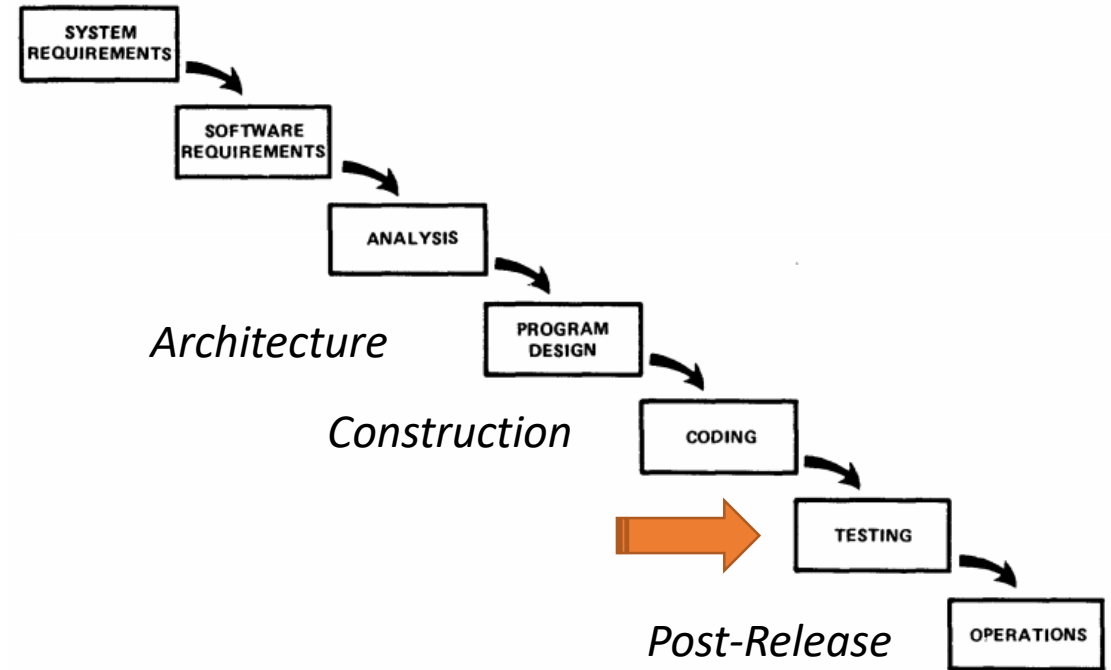
...after most of the budget is spent



What are the implications of not finding a “high-impact” problem until the system is in production?

Problems with Waterfall and SSDM IS Development

Waterfall model delays evaluation of the information system until testing...



Time Introduced	Time Detected				
	Requirements	Architecture	Construction	System Test	Post-Release
Requirements	1	3	5–10	10	10–100
Architecture	—	1	10	15	25–100
Construction	—	—	1	10	10–25

Relative Cost of Fixing Defects

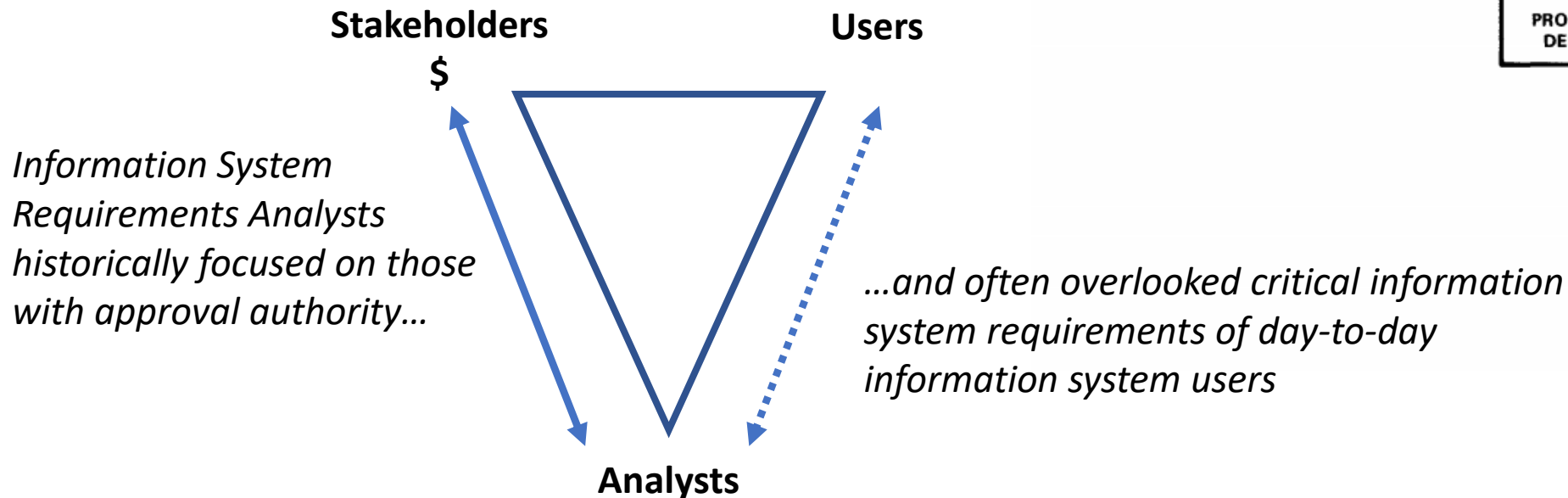
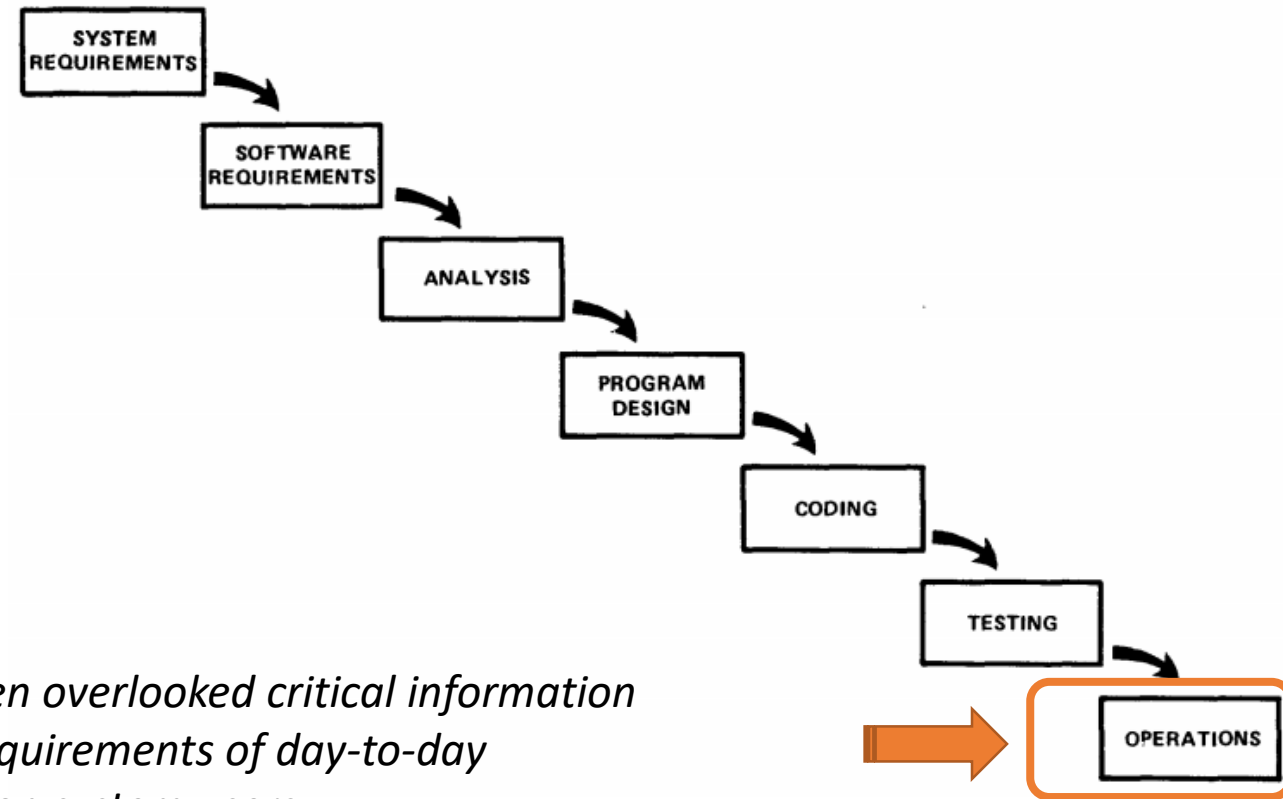
Based on When Introduced and Detected

McConnell, S. (2004) *Code Complete*, Second Edition, Microsoft Press

An architecture defect that would cost \$1,000 to fix during Program Design may cost \$15,000 to fix during system test phase

Problems with Waterfall and SSDM IS Development

Waterfall model assumes all requirements are completely known and accurately specified up front



What are the implications of not finding a “high-impact” problem until the system is in production?

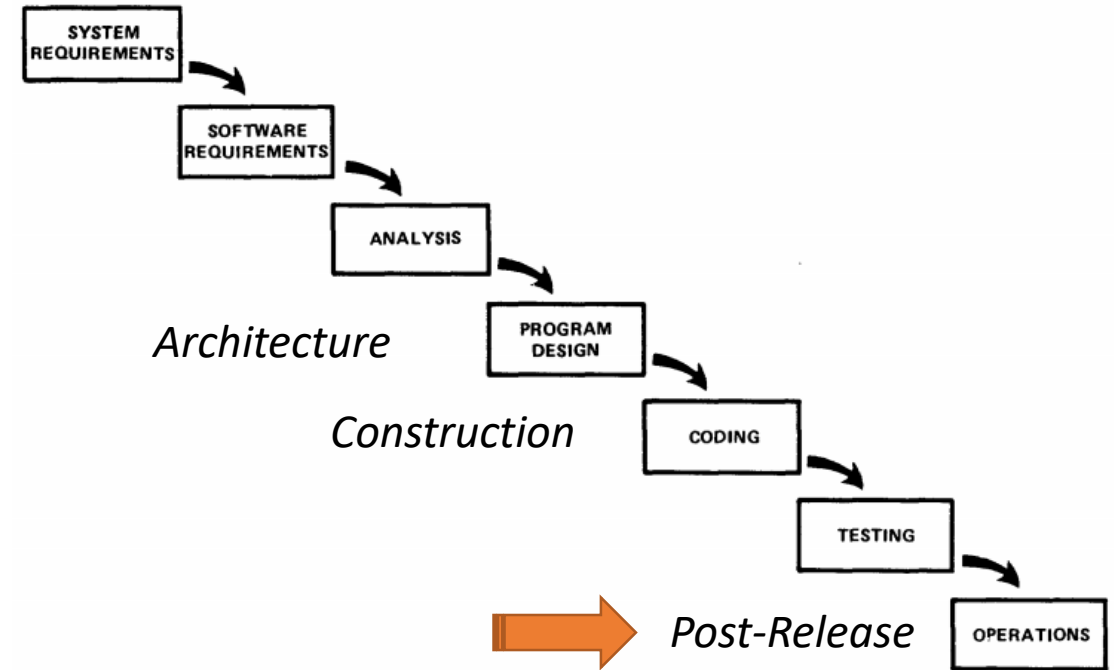
Problems with Waterfall and SSDM IS Development

An overlooked or misunderstood user requirement costing \$1,000 to fix when requirements were determined can cost \$10,000 to fix in Testing or as much as \$100,000 to fix in Post-Release operations

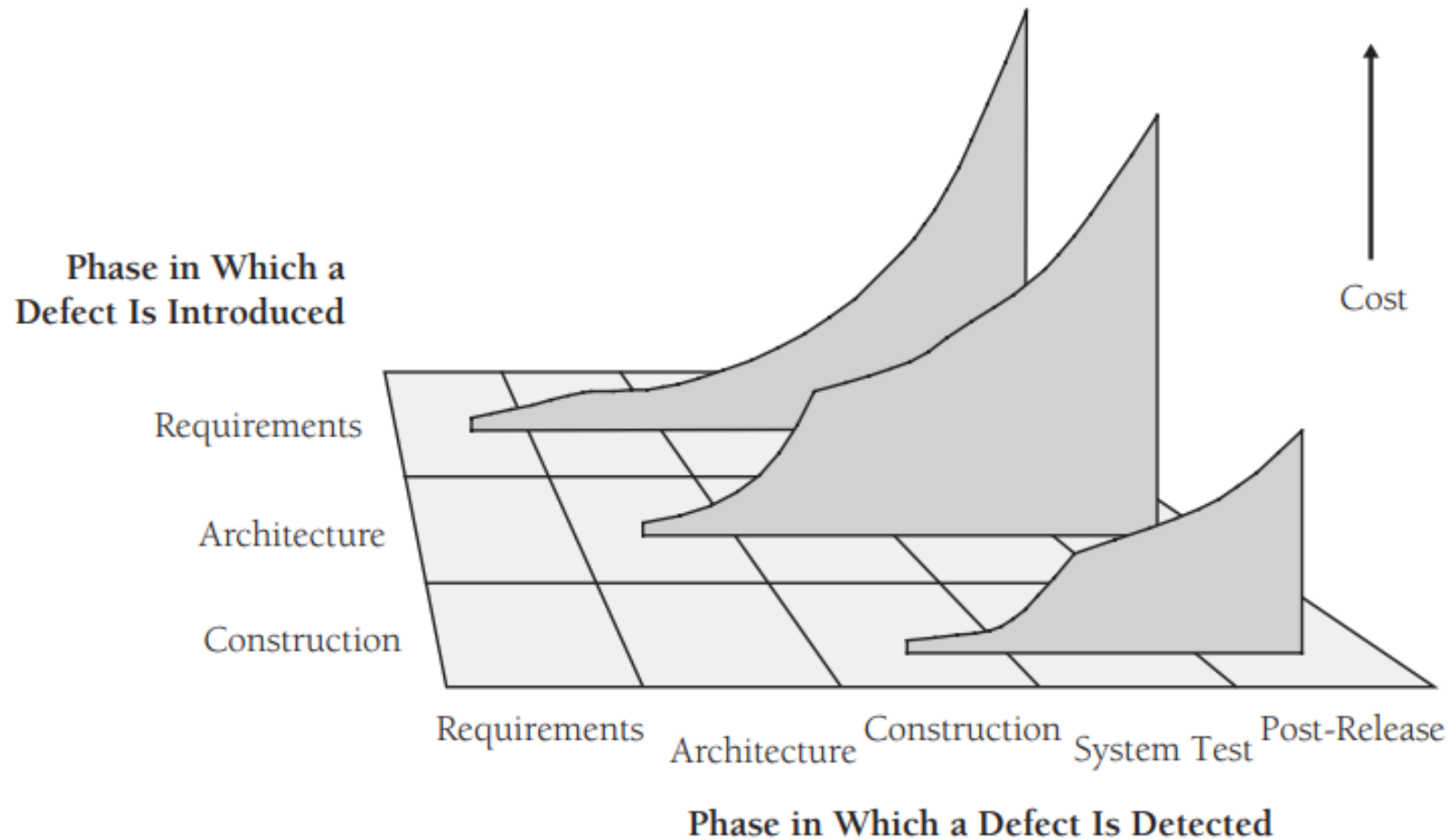
Time Introduced	Time Detected				
	Requirements	Architecture	Construction	System Test	Post-Release
Requirements	1	3	5-10	10	10-100
Architecture	—	1	10	15	25-100
Construction	—	—	1	10	10-25

Relative Cost of Fixing Defects Based on When Introduced and Detected

McConnell, S. (2004) Code Complete, Second Edition, Microsoft Press



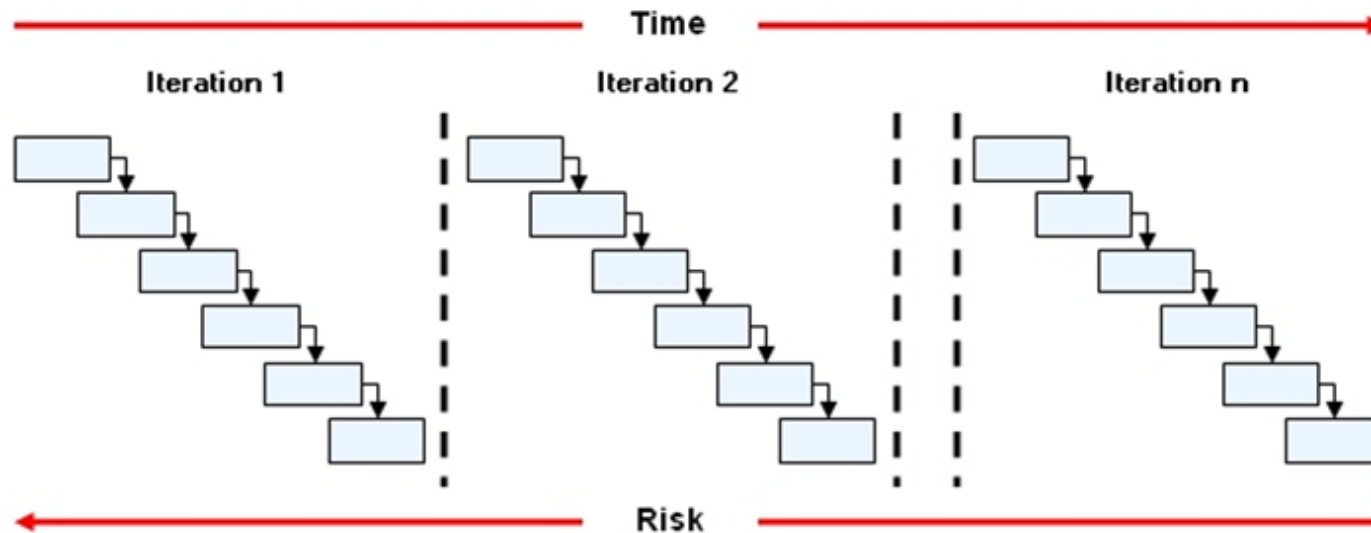
Problems with Waterfall and SSDM IS Development



McConnell, S. (2004) Code Complete, Second Edition, Microsoft Press

Iterative and Spiral IS Development Methods

Alternative Software Development Life Cycle (SDLC) models allow for multiple iterations of a waterfall-style approach



These encapsulate a number of iterations of another model (i.e. waterfall model)

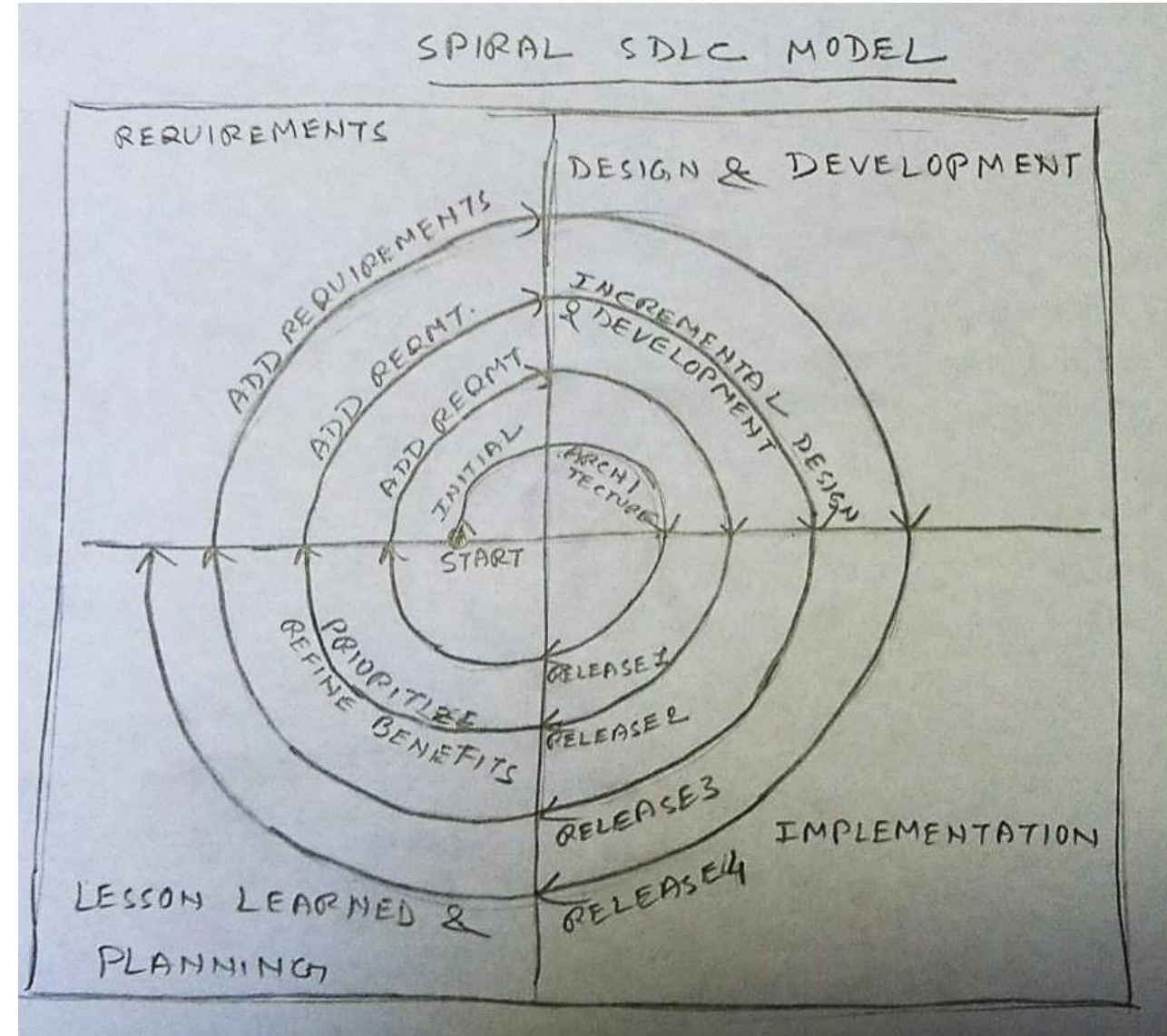
...and are known as metamodels or "model of models"

Spiral SDLC

Each “loop” of the spiral results in development of new information system prototype release

System developers apply the entire waterfall process to develop each prototype

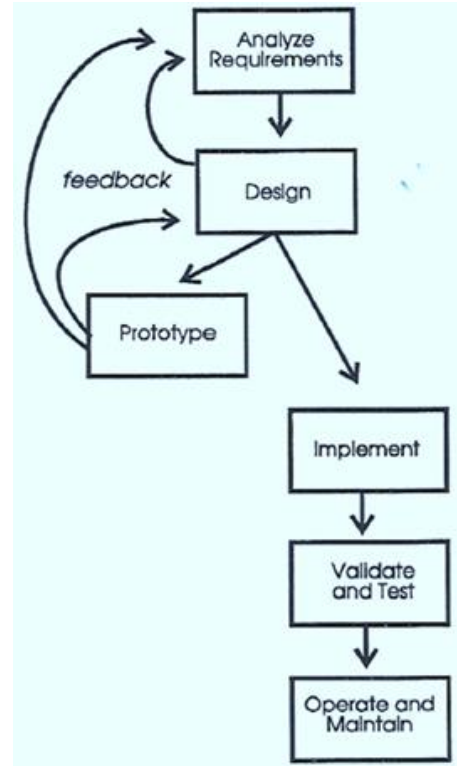
- Incrementally working toward a complete system that incorporates and validates all functional requirements
- Solved the major criticism of the Waterfall model
 - Allows developers to return to planning stages as changing customer requirements and technical demands necessitate evolution of the system



Rapid Application Development (RAD)

RAD approaches to software development put less emphasis on planning and more emphasis on an adaptive process

- Prototypes are often used in addition to or sometimes even in place of design specifications
- RAD is especially well suited for (although not limited to) developing software that is driven by user interface requirements



Prototypes' advantage over traditional specifications:

Risk reduction (Key Benefit of RAD Approach)

- Prototype testing the most difficult potential parts of the system early on in the life-cycle provides valuable information about feasibility of a design and can prevent the team from pursuing solutions that turn out to be too complex, time consuming and expensive to implement

Better way to get requirements

- Users are better at using and reacting to prototypes than at creating specifications
- Most users give much more useful feedback when they can experience a prototype of the running system rather than abstractly define what that system should be
- Prototypes can be usable and can evolve into the completed product

Rapid Application Development (RAD)

4 Phase approach:

1. Requirements planning phase

- Combines elements of system planning and systems analysis phases of SDLC
- Users, managers, and IT staff members discuss and agree on business needs, project scope, constraints, and system requirements
- Ends when team obtains management authorization to continue

2. User design phase

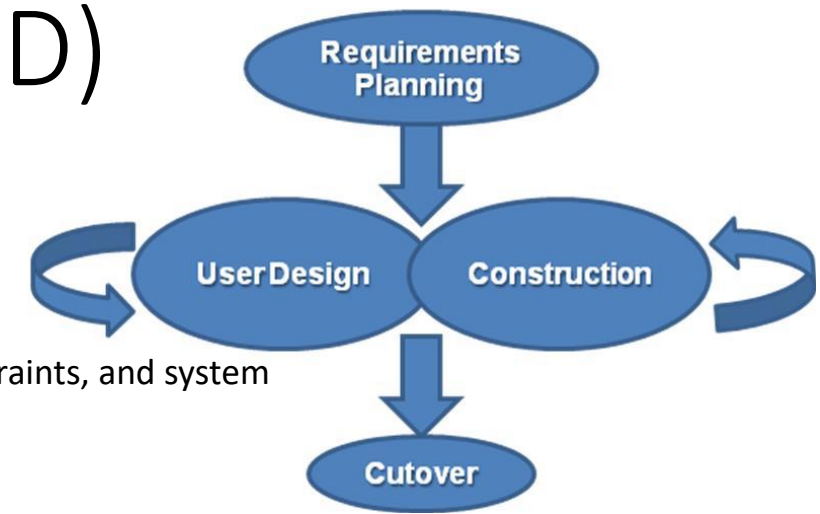
- Users work with systems analysts to develop models and prototypes representing all system processes, inputs, and outputs
- Typically uses Joint Application Development (JAD) technique and CASE tools to translate user needs into working models
- User Design is a continuous interactive process that allows users to understand, modify, and eventually approve a working model of the system that meets their needs

3. Construction phase

- Focuses on application development task but in RAD users continue to participate and can still suggest changes or improvements as actual screens or reports are developed
- Tasks are coding, unit-integration and system testing

4. Cutover phase

- Includes data conversion, testing, changeover to new system, and user training
- Compared with traditional methods, the entire process is compressed
- As a result, the new system is built, delivered, and placed in operation much sooner



Agile SDLC

Considered antidote to downside of rigid SDLC models of the past

- Emphasizes the needs of the customer
- Quickly develops new software functionality to meet customer needs in an iterative fashion

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

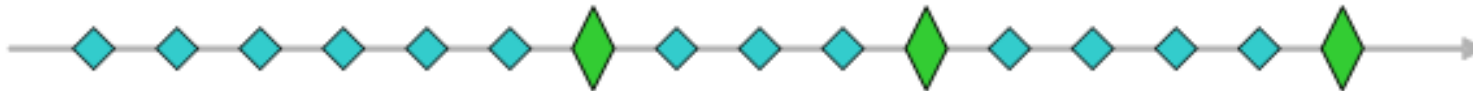
Agile SDLC


<http://agilemanifesto.org/>


Has many variants, including:

- Scrum
- Agile Unified Process (AUP)
- Dynamic Systems Development Model (DSDM)
- Extreme Programming (XP)

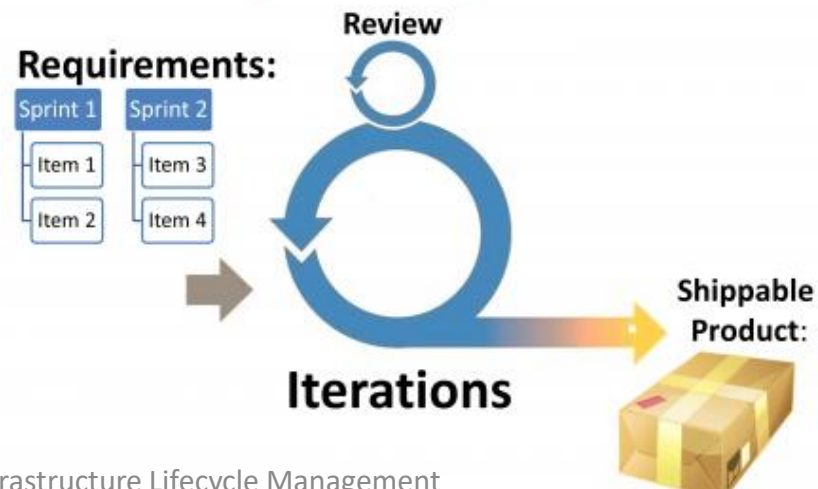
Project Timeline



 A Development Release iteration results in a deployment to the Stage/QA area

 A Production Release iteration results in a deployment to the Production area

Agile Projects:



We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

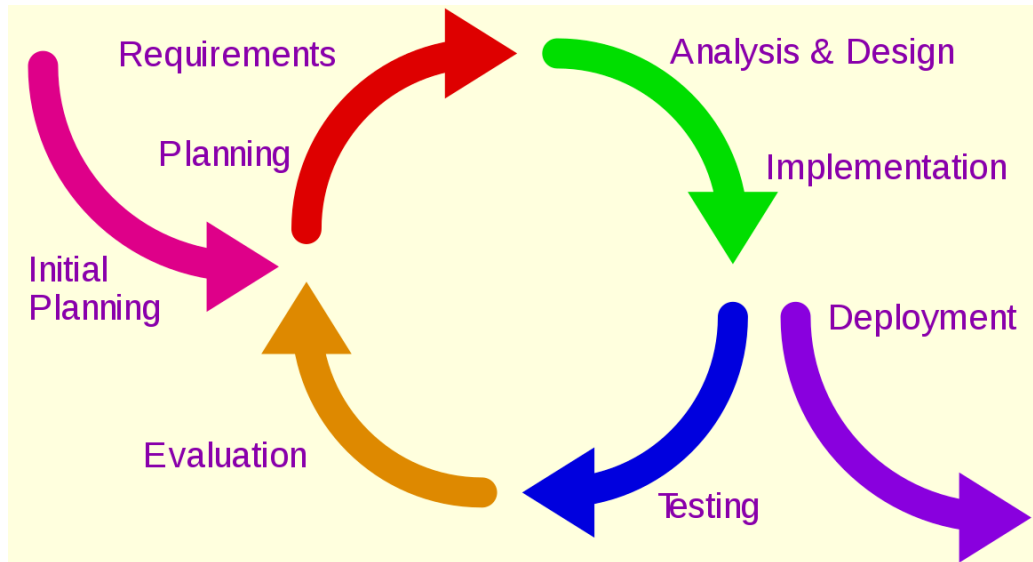
Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Spiral, Iterative and Agile SDLC methods

- Prototyping-Evolutionary Development
- Rapid Application Development (RAD)
- Agile Development



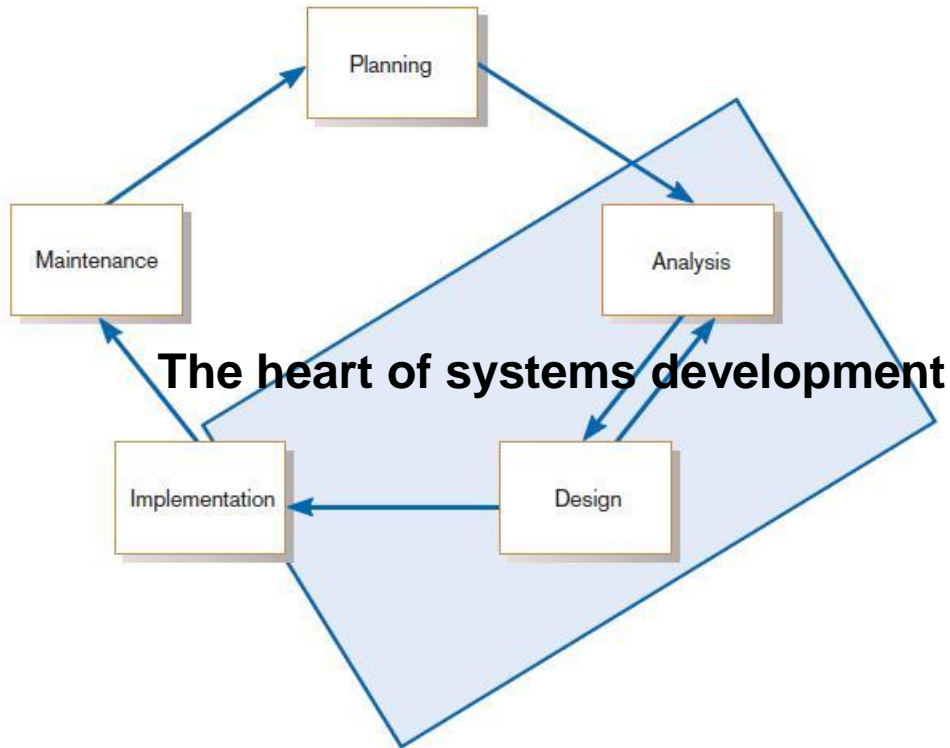
Address difficulty of obtaining complete accurate requirements up front by recognizing users are:

- Imperfect at imagining what is possible
- Not good at explaining what they need
- Very good at criticizing what they see

Rapidly creates and evolves IS versions

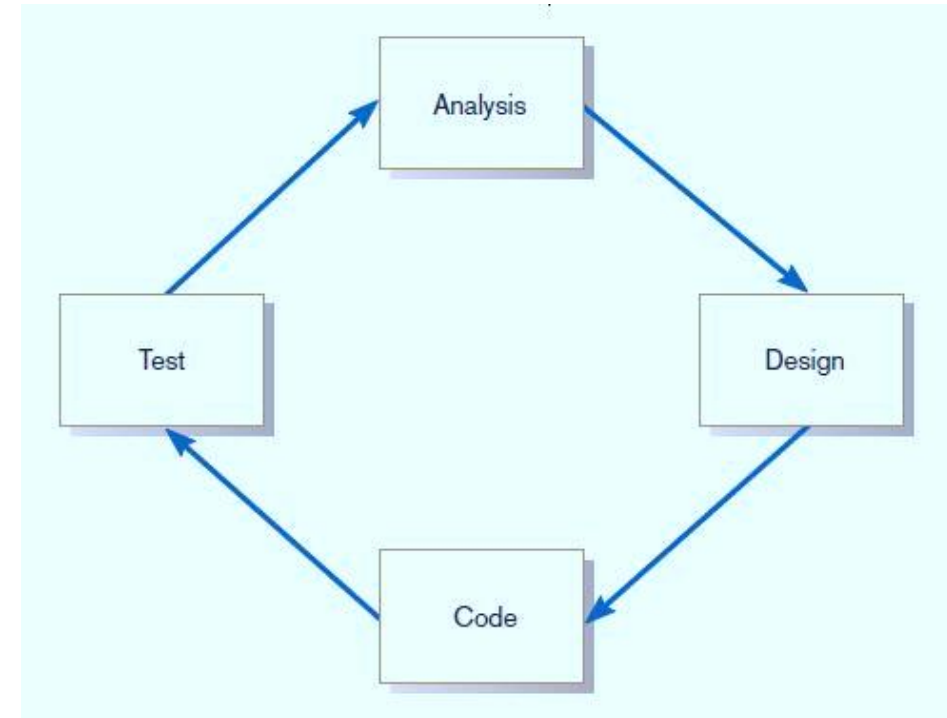
- Quickly delivers incremental progress
- Enables mid-course corrections
- Engages users' understanding and buy-in
- Gains user feedback and engagement in process and system improvement

Modern Systems Analysis and Design Textbook's SDLC Approach



The heart of systems development

Analysis–design–code–test loop



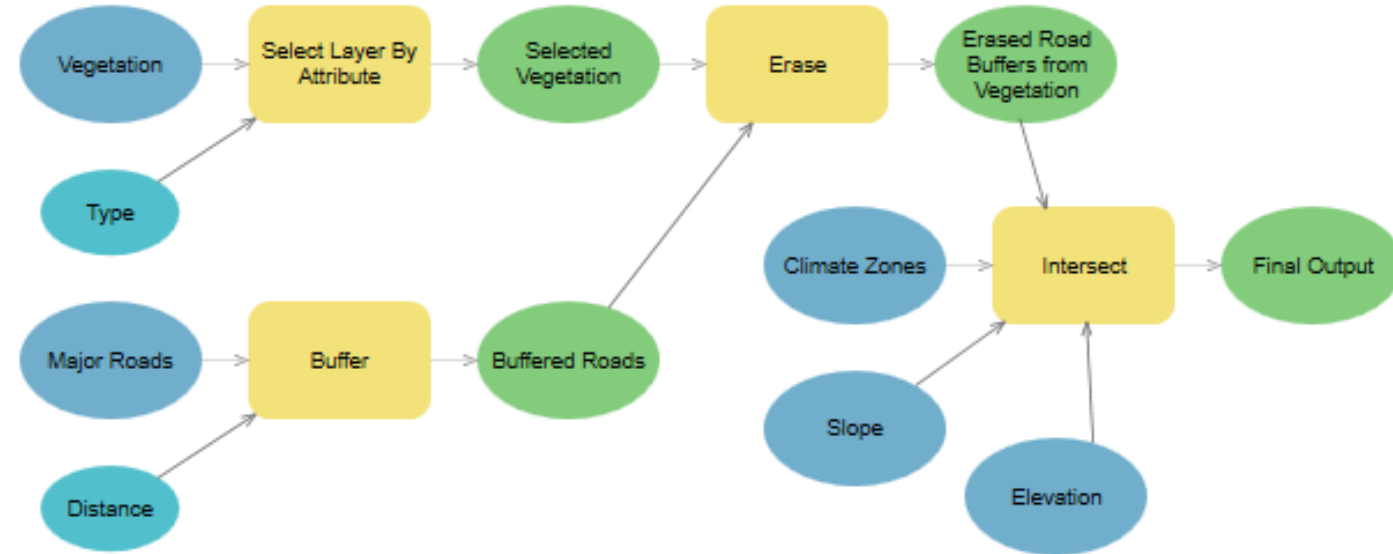
Combines analysis, design, and implementation into a single iterative process of activities

Valacich, J.S. and George, J.F. (2017), Modern Systems Analysis and Design

Computer-Aided Software Engineering (CASE) Tools

Software application programs used to automate SDLC activities

- Used by software project managers, analysts and engineers to develop software system
- Include:
 - Requirements Analysis tools
 - Software Design tools
 - Software Code Generation tools
 - Database Management tools
 - Documentation tools



Geoprocessing model built with
ESRI ModelBuilder CASE tool

Software Capability Maturity Model SW-CMM

Created by Software Engineering Institute (SEI) at Carnegie Mellon University

Organizations engaged in software development move through a sequence of maturity phases:

Level 1: Initial

Hardworking people charging ahead in a disorganized fashion. Little or no defined software development process

Level 2: Repeatable

Basic lifecycle management processes introduced. Code reuse and repeatable results expected from similar projects.

Key process areas: Requirements management, Project planning and tracking, oversight, Subcontract management, Quality assurance, and Configuration Management

Level 3: Defined

Software developers are trained and development projects take place within a standardized process model and follow formal documented software development processes

Level 4: Managed

Quantitative measures provide a detailed understanding of the development and qualitative processes

Level 5: Optimizing

Continuous improvement processes are based on feedback from one phase reaching previous phase to improve future results.

Key process areas include: Defect prevention, Technology change management and Process change management

IT Auditor's Role in Information System Development

Two alternative approaches

1. Review end-stage deliverables throughout development process, without becoming part of the process
 - Auditor reviews each stage's deliverables to ensure:
 - i. What was planned from the previous stage has been accomplished and the planning of the next stage has been refined appropriately
 - ii. Planning of the next stage has been refined appropriately
2. Internal control consultant, becoming part of systems development process.
 - Auditor provides ongoing proactive recommendations by participating in selected project-management meetings including: risk-assessment, systems-design, development, and systems delivery meetings
 - Auditor's independence may be compromised, but this is mitigated by another auditor who should find a system with well-designed controls incorporated

IT Auditor's Role in Information System Development

Produce and provide formal audit reports to the appropriate business managers including:

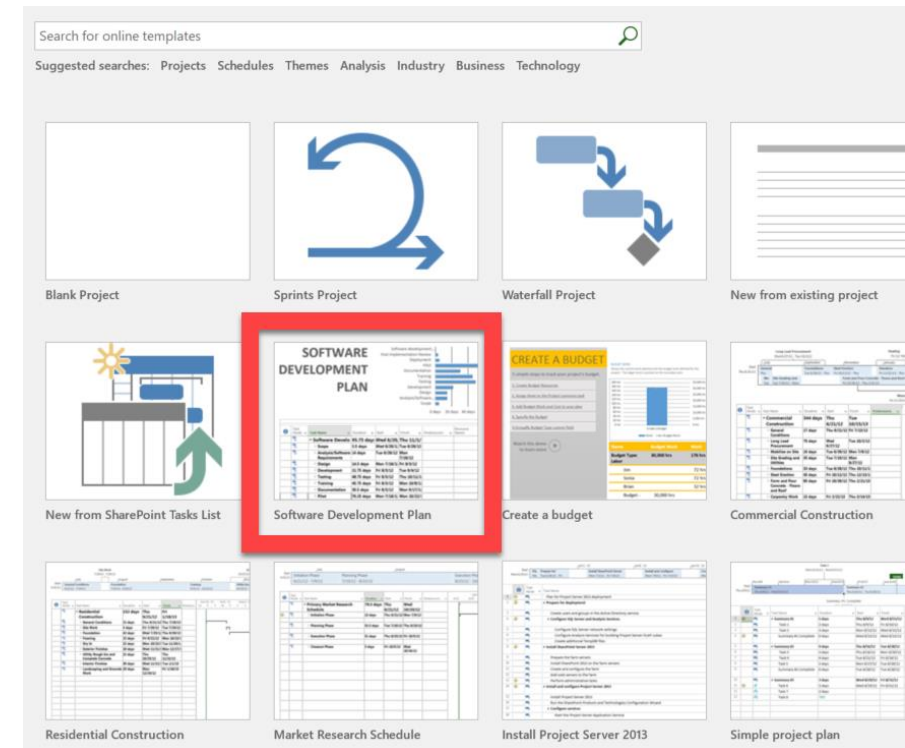
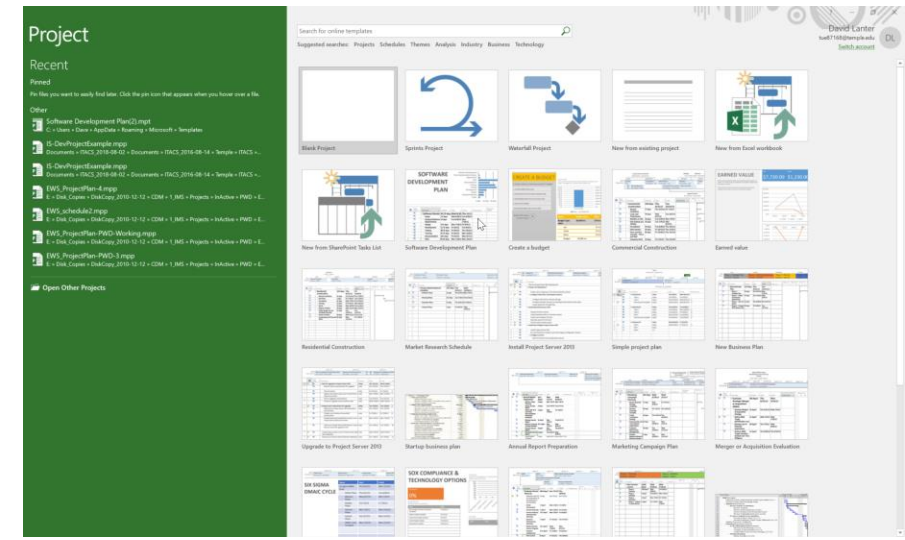
1. Overall assessment of the controlled progress of the project
2. Areas requiring improvement to complete the project, as specified, within budget and at an appropriate level of quality

Requires an in-depth understanding of both:

1. The overall information systems development processes adopted
2. The business processes being computerized

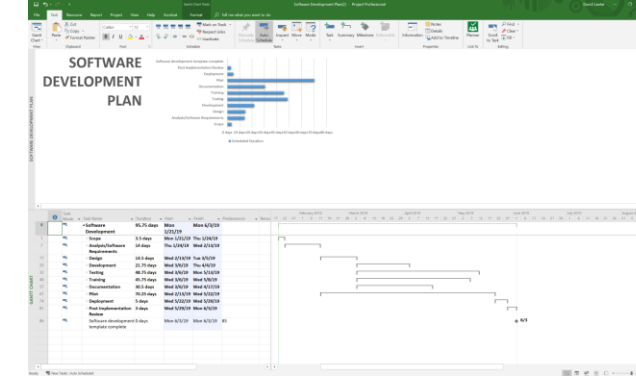
In-class exercise

1. Start up Microsoft Project 2019
2. Select Software Development Plan Template
3. ***What do you see within MS Project?***



In-class exercise

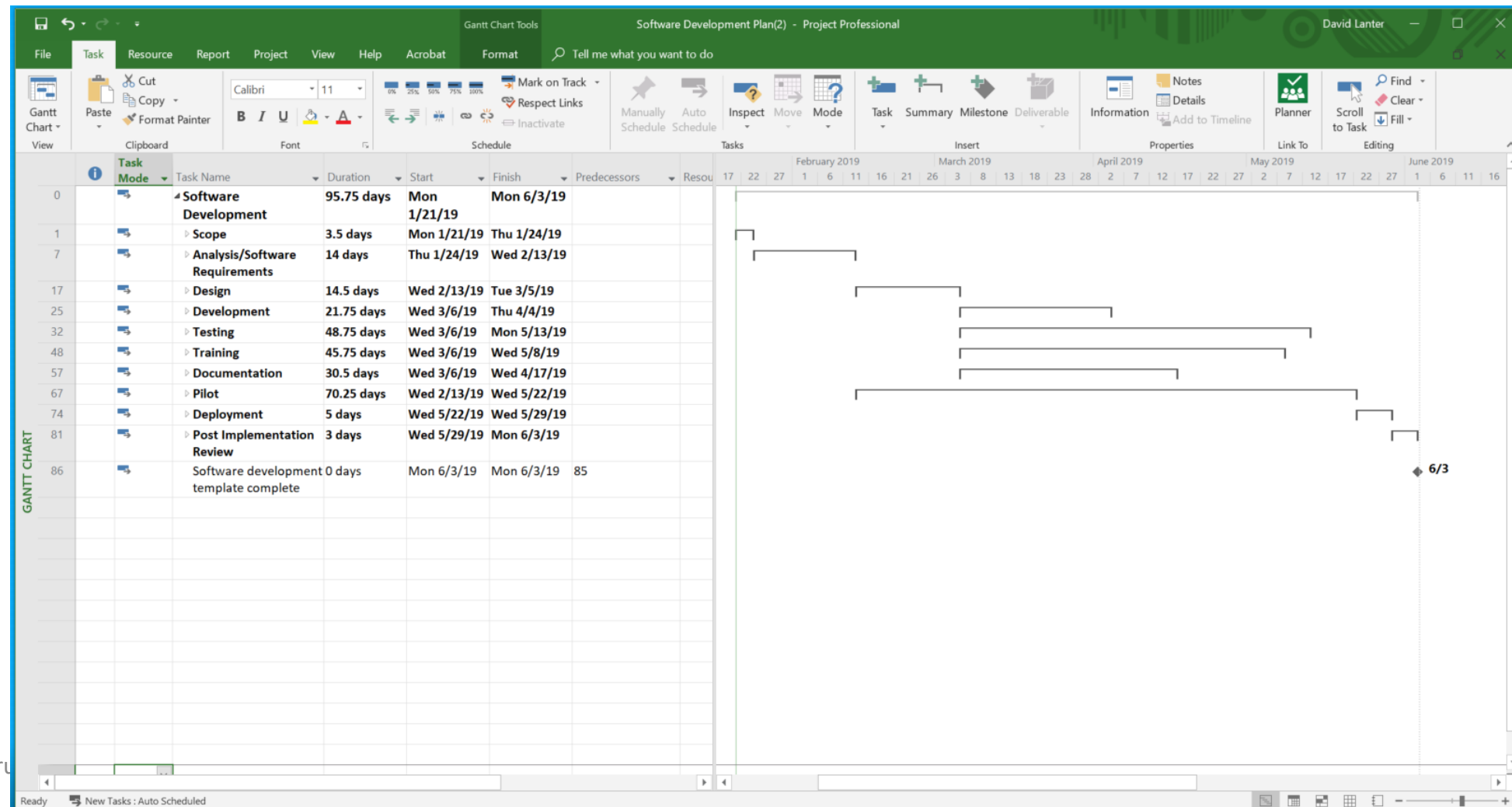
4. Pull the GANTT chart window up...



A Gantt chart is a type of bar chart that illustrates a project schedule

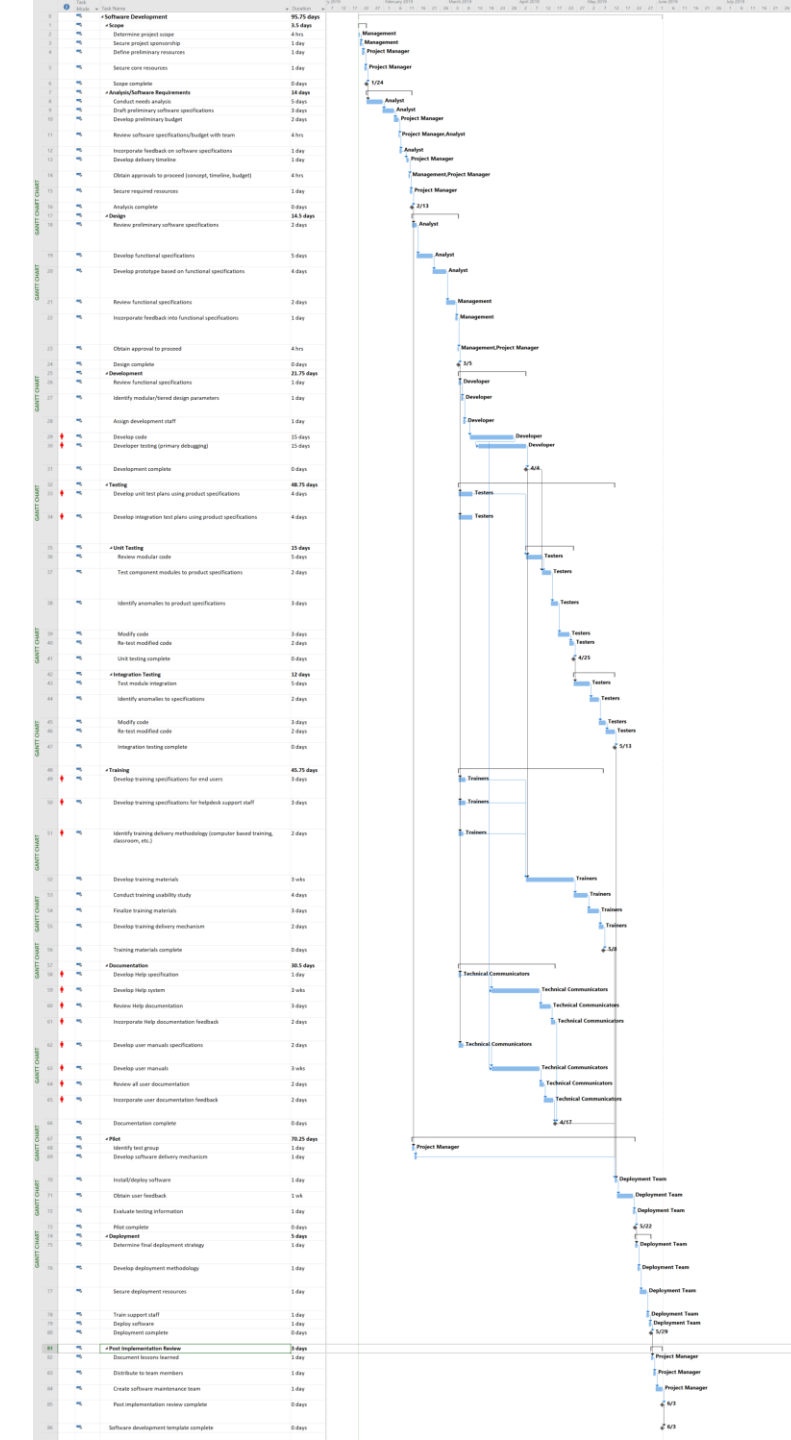
Named after its inventor, Henry Gantt (1861–1919), who designed such a chart around the years 1910–1915

Modern Gantt charts also show the dependency relationships between activities (“tasks”) and current schedule status



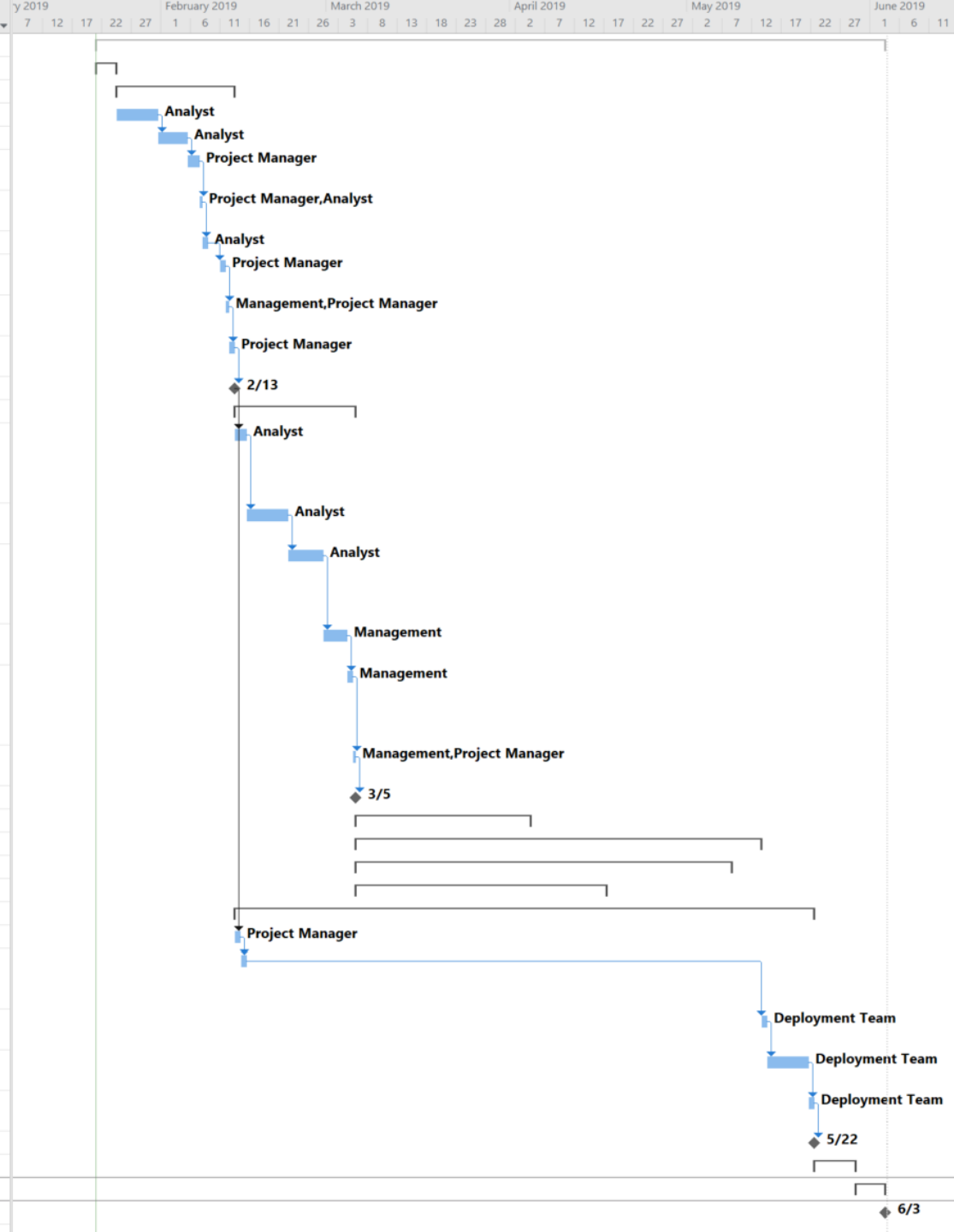
Analyze the software development project plan

What dependencies can you see among activities?

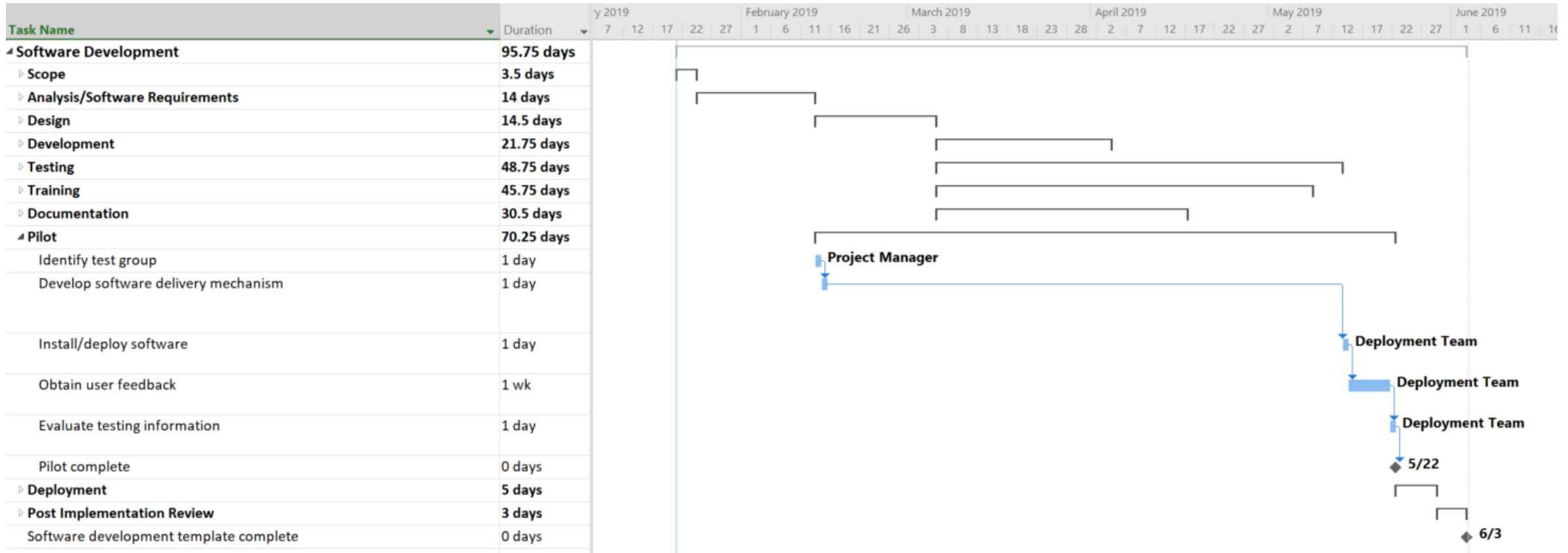


Which SDLC methodology is evident in this project plan?

Task Name	Duration
Software Development	95.75 days
Scope	3.5 days
Analysis/Software Requirements	14 days
Conduct needs analysis	5 days
Draft preliminary software specifications	3 days
Develop preliminary budget	2 days
Review software specifications/budget with team	4 hrs
Incorporate feedback on software specifications	1 day
Develop delivery timeline	1 day
Obtain approvals to proceed (concept, timeline, budget)	4 hrs
Secure required resources	1 day
Analysis complete	0 days
Design	14.5 days
Review preliminary software specifications	2 days
Develop functional specifications	5 days
Develop prototype based on functional specifications	4 days
Review functional specifications	2 days
Incorporate feedback into functional specifications	1 day
Obtain approval to proceed	4 hrs
Design complete	0 days
Development	21.75 days
Testing	48.75 days
Training	45.75 days
Documentation	30.5 days
Pilot	70.25 days
Identify test group	1 day
Develop software delivery mechanism	1 day
Install/deploy software	1 day
Obtain user feedback	1 wk
Evaluate testing information	1 day
Pilot complete	0 days
Deployment	5 days
Post Implementation Review	3 days
Software development template complete	0 days



What risks threaten projects based on the kind of SDLC project methodology implied by this plan?



In-class exercise

1. Pair up with a classmate and work together
2. Save a copy of the Software Development project Plan as “MyRAD_Project”
3. Edit the project plan and transform it into a RAD (Rapid Application Development) project plan
4. Present your results to the class

Quiz

1. Which one of the following statements about the SDLC is correct?
 - a. The SDLC requires the use of an iterative approach to software development
 - b. The SDLC requires the use of a sequential approach to software development
 - c. The SDLC does not include training for end users and support staff
 - d. The waterfall methodology is compatible with the SDLC

2. When using the SDLC, which one of these steps should be taken before the others?
 - a. Functional requirements determination
 - b. Control specifications development
 - c. Code review
 - d. Design review

3. When should design take place in following an SDLC approach to software development?
 - a. After the code review
 - b. After user acceptance testing
 - c. After the development of functional requirements
 - d. After the completion of unit testing

4. What is the main reason for reviews at the end of each phase in the SDLC?
 - a. Funding approval to continue development
 - b. Approval by management to proceed to the next phase or possibly kill the project (“stage gate”)
 - c. Design and code familiarity
 - d. Internal Auditor compliance

Quiz

1. Which one of the following statements about the SDLC is correct?
 - a. The SDLC requires the use of an iterative approach to software development
 - b. The SDLC requires the use of a sequential approach to software development
 - c. The SDLC does not include training for end users and support staff
 - d. **The waterfall methodology is compatible with the SDLC**

2. When using the SDLC, which one of these steps should be taken before the others?
 - a. **Functional requirements determination**
 - b. Control specifications development
 - c. Code review
 - d. Design review

3. When should design take place in following an SDLC approach to software development?
 - a. After the code review
 - b. After user acceptance testing
 - c. **After the development of functional requirements**
 - d. After the completion of unit testing

4. What is the main reason for reviews at the end of each phase in the SDLC?
 - a. Funding approval to continue development
 - b. **Approval by management to proceed to the next phase or possibly kill the project (“stage gate”)**
 - c. Design and code familiarity
 - d. Internal Auditor compliance

Next week...

1. Practice using the [CalculateProjectNPV B.xlsx](#) spreadsheet to calculate costs and benefits for the Customer Tracking System (Pine Valley Furniture) as you read Chapter 5 in Modern Systems Analysis and Design (MSAD). Apply the spreadsheet to illustrate:
 - A. Cash flow analysis
 - B. Use of Net Present Value (NPV) in calculating Return on Investment (ROI)
 - C. Break even analysis
2. Apply the [CalculateProjectNPV B.xlsx](#) spreadsheet to Case Study 1: Teradata Data Mart Consolidation Return on Investment at GST to illustrate:
 - A. Cash flow analysis
 - B. Use of NVP in calculating ROI for the Data Mart Pilot Project
 - C. Break even analysis
3. How does the project structure illustrated in the [Microsoft Project Software Development Plan template](#) compare with the project plan presented in Exhibit 10 of the Teradata Data Mart Consolidation ROI case study?

<http://community.mis.temple.edu/mis5203sec001sp2019/category/unit-03-project-initiation-and-selection/>

Agenda

- ✓ Information System Development Lifecycle (SDLC) Models and Methods
- ✓ IT Auditor's Role
- ✓ In-Class Exercise
- ✓ Quiz
- ✓ Next week...