Curriculum Development Centers Program

Awardee of The Office of the National Coordinator for Health Information Technology

Component 8
Installation and Maintenance of Health IT Systems

Unit 5
The Software Development Life Cycle

This material was developed by Duke University, funded by the Department of Health and Human Services, Office of the National Coordinator for Health Information Technology under Award Number IU24OC000024.

What We'll Cover

- What is the Software Development Life Cycle (SDLC)? And why do we need it?
- Phases
- Models: Waterfall, Iterative, Spiral
- Examples
- SDLC and EHR Systems

What is the SDLC?

- Software/Systems Development Life Cycle (SDLC)
  - Detailed plan for creation, development, implementation, and eventual phase-out of a software package
- Many different models exist. Two typical categories are:
  - Waterfall model
  - Iterative model

Why Do We Need the SDLC?

- Software purchases and upgrades can be costly.
- Integration of poorly designed or untested software can be devastating to a business.
- Poorly designed software increases security risks.
- Failure to plan adequately for software integration can limit efficiency and be costly in project over-runs and lost productivity.

Factors for Success

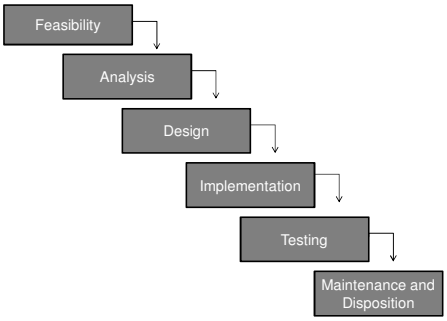
1. Management support
2. Technical and business expertise
3. Focal points of product
4. Well-defined procedure
5. Proper documentation for maintenance

Phases of a Typical SDLC

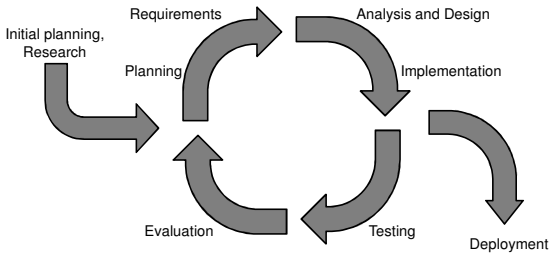
- Many different models exist for developing software systems.
- All models follow some variation of these general phases:

• **Initiation** • **Concept development** • **Planning** • **Requirements analysis**	• **Design** • **Testing** • **Implementation** • **Maintenance** • **Disposition**

Waterfall SDLC Model



Iterative and Incremental Models



Initiation Phase

- Need(s) is/are identified; e.g., Clinical needs, workflow analysis, Adminstrative input, etc.
- Project manager is assigned.
- Concept Proposal is generated.
  - Outlines the business need and justification
  - Approved by upper management

Concept Development Phase

- Needed when additional study/analysis required before beginning development
- Defines scope of development project
- Useful reports:
  - Feasibility study
  - Cost / benefit analysis
  - System boundary analysis
  - Risk management report

Planning Phase

- What must be delivered?
- What personnel will be needed?
- What external resources should you bring in, if any?
- Develop in-house or purchase software?
- What hardware constraints do you have?
- Planning document submitted for approval

Requirements Analysis Phase

- Common topics addressed
  - Operating system (OS) and interfaces
  - Input (mouse, keyboard, touchscreen)
  - Training, required user proficiency
  - Space to house hardware
- Characteristics of good requirements
  - Systematic
  - Verifiable
  - Related to business needs/opportunities
  - Details defined

Design Phase

- Blueprint of software is developed.
- Program components and workflow are established.
- Program documentation (e.g., manuals) begins to take shape.
- Flaws in original planning are often revealed, and adjustments are made.

Development Phase

- Software product is built (i.e., coded and assembled) and takes on life.
- Usually a team effort involving many software developers coordinating their efforts to realize a final product

Integration and Testing Phase

- Critical, formalized process using parameters developed during the design stage
- “Roll-Out” testing helps ensure stability in the real world environment.
- New software is tested to ensure that data can be migrated from the obsolete software into the new product easily and reliably.

Implementation Phase

- User communication and training
- Data migrated from old system and checked for integrity.
- **New system brought online**. Whenever possible, old system continues to function in case of roll out issues.
- After successful distribution, data gathered to determine successful implementation ("debriefing").

Operations and Maintenance Phase

- Day-to-day operation
- System monitored for anomalies and bugs.
- Patching and updates deployed as needed for problems or to improve functionality.
- Product lifetime can be extended.

Disposition Phase

- Closing down application once obsolete or replaced
- Many details to plan
  - Compliance with regulatory requirements
  - Safe, secure disposition of software and obsolete hardware components
  - Secure transition, with destruction or archiving of data
  - Archiving of documentation

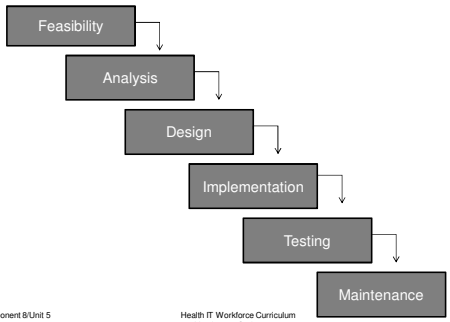
SDLC Models

- Many models, each designed to:
  - Fit a specific business need,
  - Accommodate certain resources/skills, or
  - Work with specific programming language or toolkit
- Common categories
  - Waterfall
  - Iterative

Waterfall Model

- Traditional techniques for developing software.
- Promotes strong documentation of each step.
- Uses a sequential development process.
- Formalized 1970 in critique by Winston W. Royce.
- Each phase perfected before progressing forward; derived from manufacturing, where change is very costly.
- Often criticized for use in software, where phases almost never perfected before moving forward.

Waterfall Model: Illustration of a Variation



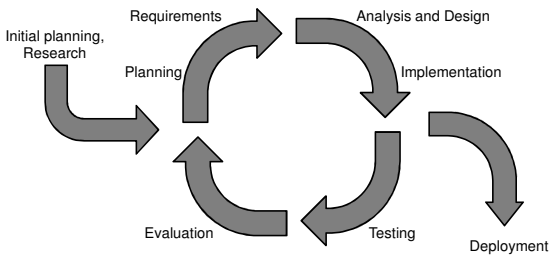
Waterfall Model: Pros & Cons

- Works best when:
  - Complexity of system is low.
  - Requirements are static.
  - Little room for mistakes.
  - No process for correcting errors after the final requirements are released.
- Limitations
  - Feedback limited.
  - In software, nearly impossible to perfect a phase before moving forward.

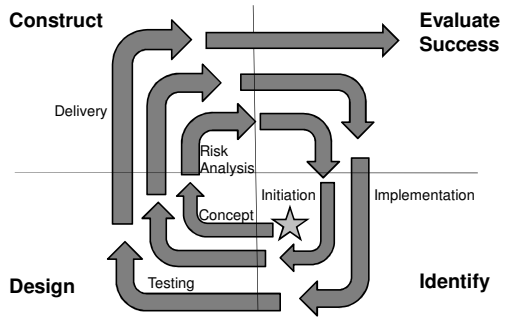
Iterative/Incremental Models

- Developed to address weaknesses in the waterfall model.
- Cyclic process which allows back-tracking, repeated cycles (iterations) for design.
- Works well when requirements subject to change or more feedback is needed.
- Variants include Spiral model.

Iterative/Incremental Models: Illustration



Spiral Model



A Not-So-Real-Life Example

- Widget Inc.'s market research identifies need for efficient square-jar canning software.
- R&D devises and tests a conceptual canning software system that should address the needs.
- Feasibility study submitted and approved.

A Not-So-Real-Life Example (cont'd)

- Design team builds blueprint, documentation of how canning software should operate.
- Implementation team begins coding modules.
  - Milestones established.
  - Documentation completed to ensure product can be adequately troubleshot and maintained.
- Three weeks in, problem identified. Project manager deems change critical, so timeline adjusted (with executive approval).

A Not-So-Real-Life Example (cont'd)

- New software tested, errors corrected, retested.
- Support infrastructure to provide customer support and upgrading as needed, using original documentation as baseline.
- Software finally brought into production!
- Quality Assurance team identifies issues for correction and passes off to the support team.

SDLC and EHR Systems

- Similar to project plan, incorporating software-specific aspects.
- Should augment (not replace) EHR project plan.
- Particularly important if planning in-house EHR design or program modifications (e.g., integration).
- Needed to ensure product satisfaction and quality assurance, mitigate risk factors, minimize downtime.

References

- Software Development Life Cycle
  - "Introduction to Software development Life Cycle" as found in Agus Soyfandi's blog http://agusofyandi.wordpress.com/2010/08/31/introduction-to-software-development-life-cycle-sdlc/
  - *"Quick Study: Systems Development Life Cycle."* By Russel Kay (also available in podcast from the website.) http://www.computerworld.com/s/article/71151/System_Development_Life_Cycle?taxonomyId=011