

# Validation: Never an Endpoint

---

## Implementing a Systems Development Life Cycle

Bucky Walsh

Vice President, Systems Management

Greg Johnson

Product Director

## Weinberg's Second Law

---

- “If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization.”

# Short History of Programming

---

**Broader  
Distribution  
of the Tools  
is not Having  
a “Magic”  
Effect on the  
Quality of the  
Programs we  
Produce**

- 50 years ago programming was an unknown craft
- 40 years ago it was an esoteric art
- 30 years ago it was the domain of a select few initiates
- 20 years ago it was in its widespread infancy
- 10 years ago tools became widely available to the general public
- Today my children learn programming in elementary school

# Programming: Art vs. Engineering

---

Systems

Development

is at the

Beginning of

its Life as a

Scientific

Discipline

- Programming is still an art, not a science
- In manufacturing terms, we are just now on the brink of the transition to interchangeable parts
- Programming needs to transition to Software Engineering
- All software systems have bugs in them
- Pharmaceuticals industry has a clear requirement for the use of “validated” systems
- This started years ago with the GMP requirements, however, modern requirements for systems under GCP are very different from GMP requirements

# The Validation Question

---

Does  
Everyone  
Mean the  
Same Thing  
When They  
Say  
“Validation”?

- How does one “DO” validation?
- This question had surfaced off and on for years at PRA International, as we moved through various approaches to meet the industry requirements.
- Having a large quantity of printed material seems like a mandatory requirement
- Retrospective, following a plan, following an SOP, how much testing is enough, etc.

# Validation is Not an Event

---

When is  
Validation  
Done?

- Validation can only refer to a particular configuration of a system (hardware and software)
- The “System” is always changing, though:
  - Hardware upgrades
  - Operating System updates
  - DBMS upgrades
  - Software system updates
- Avoiding these upgrades is not always possible or desirable
- Even if Validation was planned as a single event, it is something that must be done over and over as the system changes

# Considering the Life of a System

---

A Systems  
Development  
Life Cycle  
Addresses  
the Need to  
Treat  
Validation as  
a Process,  
Not an Event

- Our experience required us to acknowledge the need to revisit validation during the life of a system
- As we were always doing and re-doing validation, the question came up:
  - Were we ever done with Validation?
- This led to considering a systems development life cycle model, including validation as a part of that life cycle

## Implementation

---

- What is a Systems Development Life Cycle?
- How does this fit my needs?
- Where is the validation step?
- Why is it a good approach?
- Who knows how to do this?
- When can you start?
- What have we learned?

# What is a Systems Development Life Cycle?

---

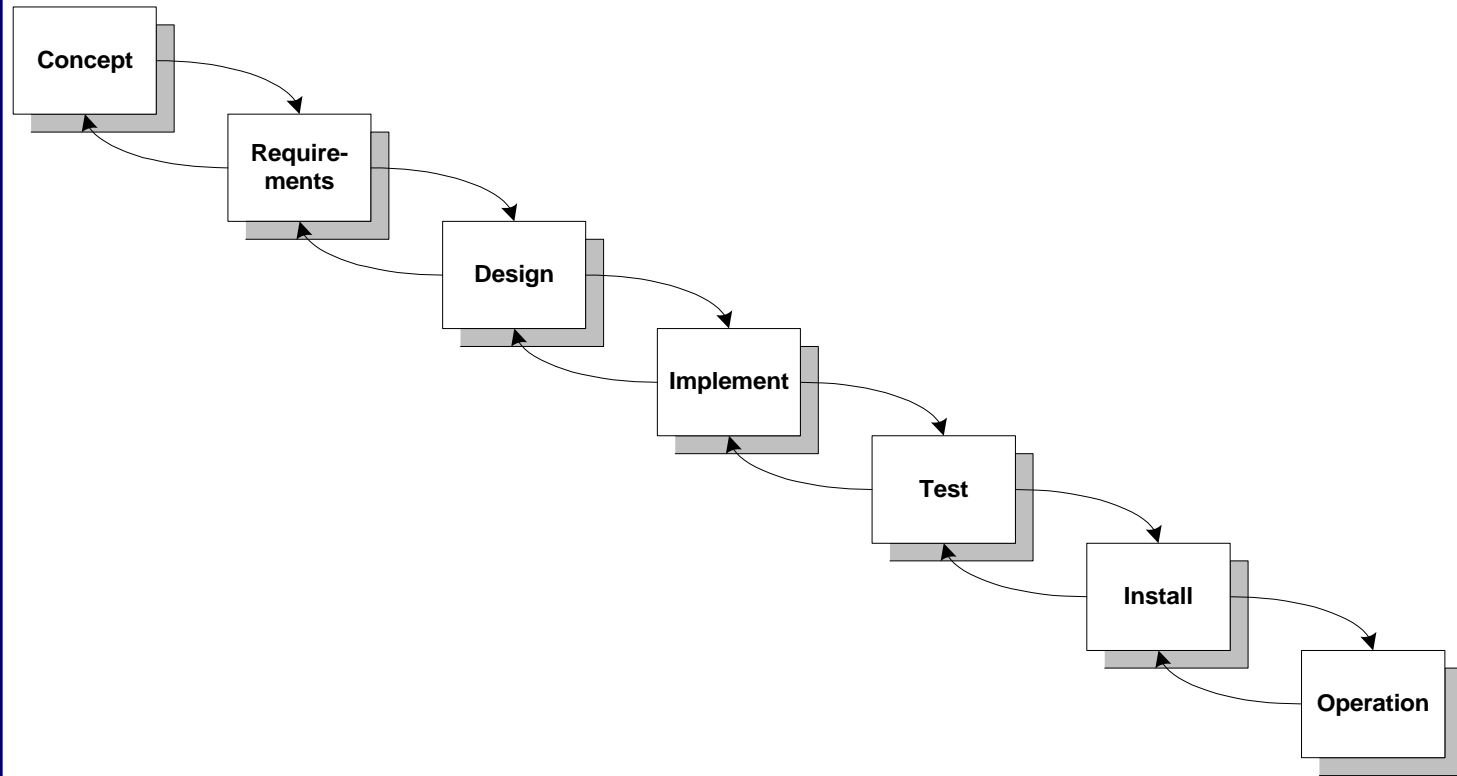
## Life Cycle

Phases can  
be Repeated,  
and Will be  
Repeated for  
most  
Systems

- An approach that recognizes that systems go through a life cycle, a series of phases in their existence, that may be repeated several times during the life of a system before it is retired.
- Key point: Phases that may be repeated
- Systems change, the environments in which they run change, there are natural cycles.

# Example Life Cycle

---



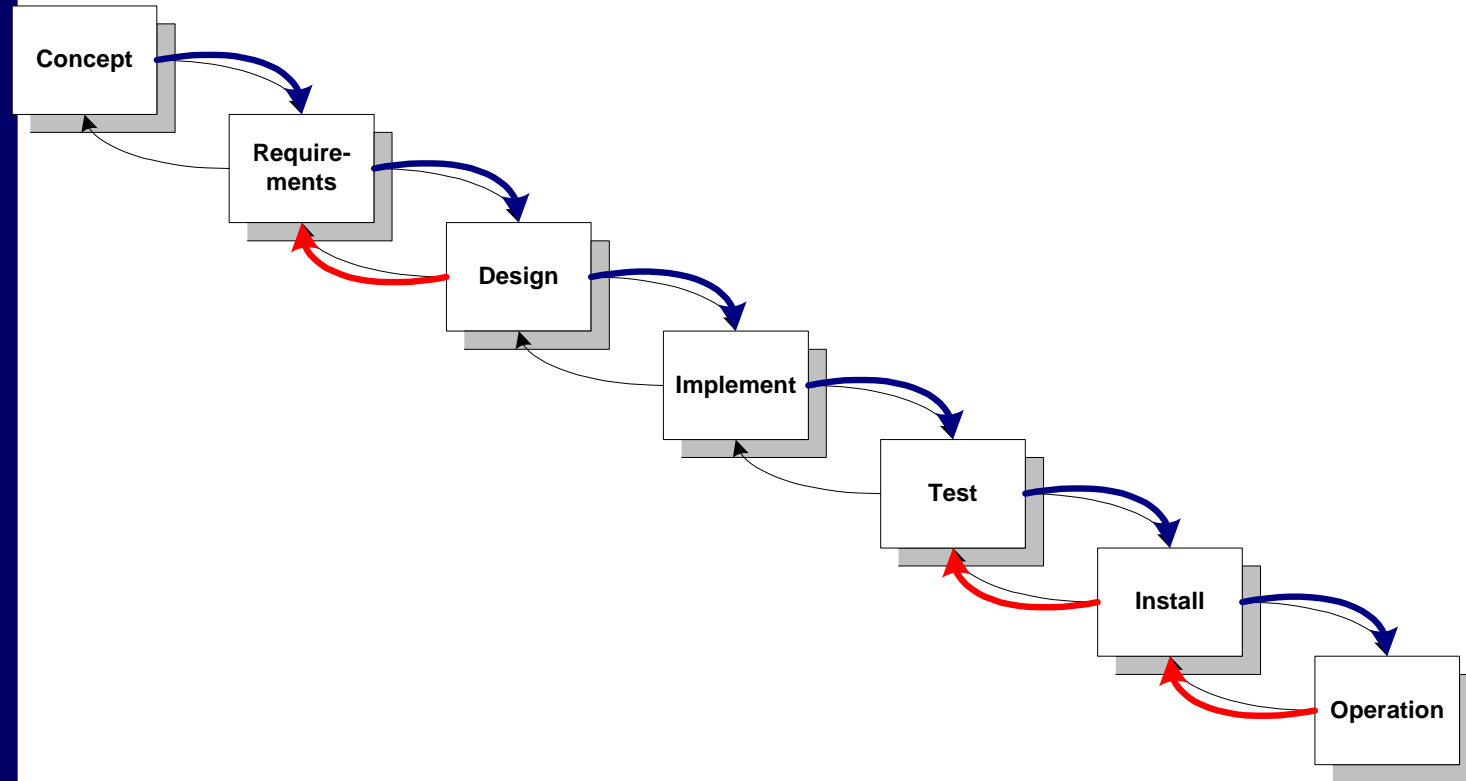
# Cycle

---

This Concept  
of a Cycle is  
Critical to the  
Systems  
Development  
Life Cycle  
Approach

- Cycle: A series of events or operations that **repeat** regularly in the same **order** – Oxford American Dictionary

# Cycling Through an Example



# Selecting a Life Cycle

---

Creating a  
Systems  
Development  
Life Cycle  
Model  
Involves  
Making  
Choices from  
Established  
Standards

- What are the steps in a Systems Development Life Cycle?
- That depends on who you ask
- PRA selected the IEEE Standards on Software Engineering as a guide
- IEEE is the Institute of Electrical and Electronics Engineers
- These standards are presently found in a four volume set, which is recommended as reading for the strong willed.

# IEEE Software Engineering Standards

Distilling out  
the Best  
Starting  
Point May  
Require  
Some  
Familiarity  
with the  
Standards

- IEEE Std 610.12-1990, IEEE Standard Glossary for Software Engineering Terminology
- IEEE Std 730-1998, IEEE Standard for Software Quality Assurance Plans
- IEEE Std 730.1-1995, IEEE Guide for Software Quality Assurance Planning
- IEEE Std 828-1998, IEEE Standard for Software Configuration Management Plans
- IEEE Std 829-1998, IEEE Standard for Software Test Documentation
- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- IEEE Std 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software
- IEEE 982.2-1988, IEEE Guide for the Use of Standard Dictionary of Measures to Produce Reliable Software
- IEEE Std 1008-1987 (Reaff 1993), IEEE Standard for Software Unit Testing
- IEEE Std 1012-1998, IEEE Standard for Software Verification and Validation
- IEEE Std 1012a-1998, Supplement to IEEE Standard for Software Verification and Validation: Content Map to IEEE/EIA 12207.1-1997
- IEEE Std 1016-1998, IEEE Recommended Practice for Software Design Descriptions
- IEEE Std 1028-1997, IEEE Standard for Software Reviews
- IEEE Std 1042-1987 (Reaff 1993), IEEE Guide to Software Configuration Management
- IEEE Std 1044-1993, IEEE Standard Classification for Software Anomalies
- IEEE Std 1044.1-1995, IEEE Guide to Classification of Software Anomalies
- IEEE Std 1045-1992, IEEE Standard for Software Productivity Metrics
- IEEE Std 1058-1998, IEEE Standard for Software Project Management Plans
- IEEE Std 1059-1993, IEEE Guide for Software Verification and Validation Plans
- IEEE Std 1061-1998, IEEE Standard for a Quality Metrics Methodology
- IEEE Std 1062, 1998 Edition, IEEE Recommended Practice for Software Acquisition
- IEEE Std 1063-1987 (Reaff 1993), IEEE Standard for Software User Documentation
- IEEE Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes
- IEEE Std 1219-1998, IEEE Standard for Software Maintenance
- IEEE Std 1220-1998, IEEE Standard for the Application and Management of the Systems Engineering Process
- IEEE Std 1228-1994, IEEE Standard for Software Safety Plans
- IEEE Std 1233, 1998 Edition, IEEE Guide for Developing System Requirements Specifications
- IEEE Std 1320.1-1998, IEEE Standard for Functional Modeling Language – Syntax and Semantics for IDEF0
- IEEE Std 1320.2-1998, IEEE Standard for Software Conceptual Modeling Language – Syntax and Semantics for IDEFIX<sub>97</sub>
- IEEE Std 1348-1995, IEEE Recommended Practice for the Adoption of Computer-Aided Software Engineering (CASE) Tools
- IEEE Std 1362-1998, IEEE Guide for Information Technology – System Definition – Concept of Operations Document
- IEEE Std 1420.1-1995, IEEE Standard for Information Technology – Software Reuse – Data Model for Reuse Library Interoperability: Basic Interoperability Data Model (BIDM)
- IEEE Std 1420.1a-1995, Supplement to IEEE Standard for Information Technology – Software Reuse – Data Model for Reuse Library Interoperability: Asset Certification Framework
- IEEE Std 1430-1996, IEEE Guide for Information Technology – Software Reuse – Concept of Operations for Interoperating Reuse Libraries
- IEEE Std 1462-1998, IEEE Standard Adoption of ISO/IEC 14102: 1995 – Information Technology – Guidelines for the Evaluation and Selection of CASE Tools
- IEEE Std 1465-1998, IEEE Standard Adoption of ISO/IEC 12119:1994 (E) – Information Technology – Software Packages – Quality requirements and testing
- IEEE Std 1490-1998, IEEE Guide – Adoption of PMI Standard – A Guide to the Project Management Body of Knowledge
- IEEE/EIA Std 12207.0-1996, Software life cycle processes
- IEEE/EIA Std 12207.1-1997, Software life cycle processes – Life cycle data
- IEEE/EIA Std 12207.2-1997, Software life cycle processes – Implementation considerations

# Creating a Life Cycle That Fits

---

Each Step in  
the Systems  
Development  
Life Cycle  
Should Add  
Value to the  
System as a  
Whole

- IEEE Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes
- We pared the steps and processes from the IEEE down to those elements that made sense for our industry and our needs
- Contrasting Examples of Development Requirements:
  - Space Shuttle Atmospheric Flight System
    - Classic example of a life or death system, every 40ms deciding the astronaut's fate
  - Word Processor
    - Output usually reviewed before contents are finalized

# Too Much or Too Little

---

Validation

Activities

Can Cost

More than a

Particular

System is

Worth,

Selecting the

Best Value is

a Critical

Success

Factor

- Contrast GCP requirements/needs with GMP and GLP
  - A single tablet manufactured incorrectly could result in a death
  - A clinical database for a trial has source data from which the database can be reconstructed
- Validation workshops should be carefully selected, GMP is not GCP
- It is possible to set the quality bar too high, and this needlessly drives up the cost of our products
- It is possible to set the quality bar too low, and then your “validation” proves little

Validation in  
this  
Environment  
is About  
Having a  
Quality Plan  
and Showing  
That You  
Have  
Followed It

## How can this fit our needs?

---

- Validation, within the systems development life cycle that we have selected, is a process, not an event.
- Every Systems Development Life Cycle step has ramifications from a validation standpoint. Some steps produce documents to be checked and used in later steps, some steps consume documents. Each of these events offers an opportunity to window an event in the life of the system, and to determine if it occurred as planned.

# IEEE Software Engineering Standards - Title Review

Red Titles

Deal with

Software

Development

Issues,

Yellow Titles

are Other

Issues,

Leaving

Much Less

Material to

Review

- IEEE Std 610.12-1990, IEEE Standard Glossary for Software Engineering Terminology
- IEEE Std 730-1998, IEEE Standard for Software Quality Assurance Plans
- IEEE Std 730.1-1995, IEEE Guide for Software Quality Assurance Planning
- IEEE Std 828-1998, IEEE Standard for Software Configuration Management Plans
- IEEE Std 829-1998, IEEE Standard for Software Test Documentation
- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- IEEE Std 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software
- IEEE 982.2-1988, IEEE Guide for the Use of Standard Dictionary of Measures to Produce Reliable Software
- IEEE Std 1008-1987 (Reaff 1993), IEEE Standard for Software Unit Testing
- IEEE Std 1012-1998, IEEE Standard for Software Verification and Validation
- IEEE Std 1012a-1998, Supplement to IEEE Standard for Software Verification and Validation: Content Map to IEEE/EIA 12207.1-1997
- IEEE Std 1016-1998, IEEE Recommended Practice for Software Design Descriptions
- IEEE Std 1028-1997, IEEE Standard for Software Reviews
- IEEE Std 1042-1987 (Reaff 1993), IEEE Guide to Software Configuration Management
- IEEE Std 1044-1993, IEEE Standard Classification for Software Anomalies
- IEEE Std 1044.1-1995, IEEE Guide to Classification of Software Anomalies
- IEEE Std 1045-1992, IEEE Standard for Software Productivity Metrics
- IEEE Std 1058-1998, IEEE Standard for Software Project Management Plans
- IEEE Std 1059-1993, IEEE Guide for Software Verification and Validation Plans
- IEEE Std 1061-1998, IEEE Standard for a Quality Metrics Methodology
- IEEE Std 1062, 1998 Edition, IEEE Recommended Practice for Software Acquisition
- IEEE Std 1063-1987 (Reaff 1993), IEEE Standard for Software User Documentation
- IEEE Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes
- IEEE Std 1219-1998, IEEE Standard for Software Maintenance
- IEEE Std 1220-1998, IEEE Standard for the Application and Management of the Systems Engineering Process
- IEEE Std 1228-1994, IEEE Standard for Software Safety Plans
- IEEE Std 1233, 1998 Edition, IEEE Guide for Developing System Requirements Specifications
- IEEE Std 1320.1-1998, IEEE Standard for Functional Modeling Language – Syntax and Semantics for IDEF0
- IEEE Std 1320.2-1998, IEEE Standard for Software Conceptual Modeling Language – Syntax and Semantics for IDEFIX<sub>97</sub>
- IEEE Std 1348-1995, IEEE Recommended Practice for the Adoption of Computer-Aided Software Engineering (CASE) Tools
- IEEE Std 1362-1998, IEEE Guide for Information Technology – System Definition – Concept of Operations Document
- IEEE Std 1420.1-1995, IEEE Standard for Information Technology – Software Reuse – Data Model for Reuse Library Interoperability: Basic Interoperability Data Model (BIDM)
- IEEE Std 1420.1a-1995, Supplement to IEEE Standard for Information Technology – Software Reuse – Data Model for Reuse Library Interoperability: Asset Certification Framework
- IEEE Std 1430-1996, IEEE Guide for Information Technology – Software Reuse – Concept of Operations for Interoperating Reuse Libraries
- IEEE Std 1462-1998, IEEE Standard Adoption of ISO/IEC 14102: 1995 – Information Technology – Guidelines for the Evaluation and Selection of CASE Tools
- IEEE Std 1465-1998, IEEE Standard Adoption of ISO/IEC 12119:1994 (E) – Information Technology – Software Packages – Quality requirements and testing
- IEEE Std 1490-1998, IEEE Guide – Adoption of PMI Standard – A Guide to the Project Management Body of Knowledge
- IEEE/EIA Std 12207.0-1996, Software life cycle processes
- IEEE/EIA Std 12207.1-1997, Software life cycle processes – Life cycle data
- IEEE/EIA Std 12207.2-1997, Software life cycle processes – Implementation considerations

# Systems Development Life Cycle

---

More Phases  
or Fewer  
Phases are  
Possible and  
may be  
Desirable in  
a Different  
Environment

- Concept Phase
- Requirements Phase
- Design Phase
- Implementation Phase
- Acceptance Testing Phase
- Installation and Checkout Phase
- Operation Phase

# Inputs and Outputs

---

Each Output  
Has Some  
Flexibility in  
Its Content,  
Allowing  
That  
Document to  
Reflect the  
Complexity  
of the  
System That  
It Supports

- Concept Paper
- Systems Requirements Specification
- System Design Description
- Implementation Testing Plan
- Traceability Analyses
- Acceptance Test Plan
- Validation Report
- System Configuration Management Plan
- Training Plan
- Test and Operations Outputs

# Where is the Validation Step?

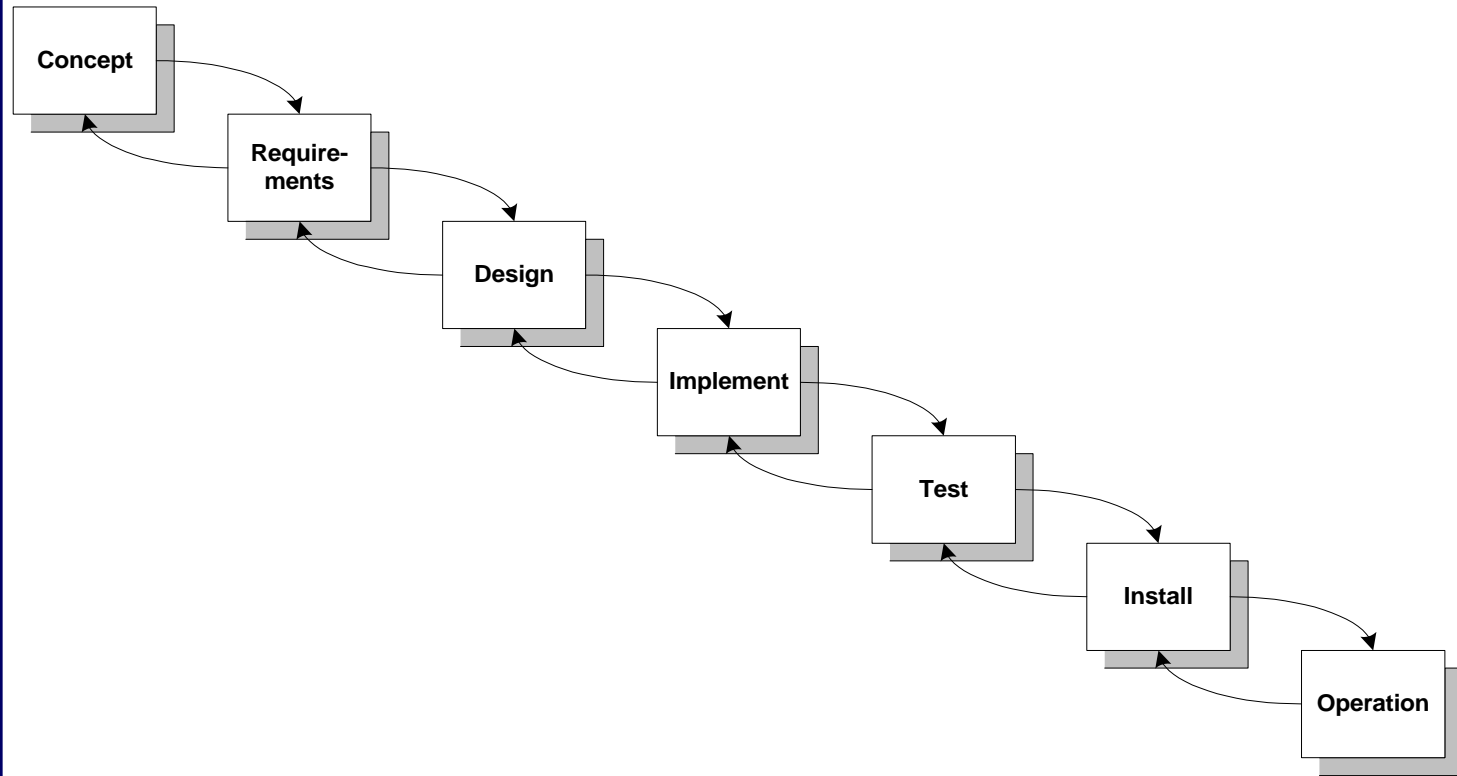
---

Validation  
Steps Are a  
Part of Every  
Phase of the  
Systems  
Development  
Life Cycle

- There is none

# Validation is Part of Each Step

---



# Why is This a Good Approach?

---

The Solution  
Meets Our  
Require-  
ments

- It fits our need
- It is flexible

**A Systems  
Development  
Life Cycle  
Involves  
Everyone  
who Touches  
that System  
During its  
Life**

# Who Knows How to Do This?

---

- In other words, how do you get started
  - Consultants, reading
  
- Who does this within the organization
  - IT staff, users, and added staff
  - Added staff are Technical Writers
  
- Everyone in the organization has some knowledge of this
  - Potential ramification of outside changes

# When can you start?

---

Implement-  
ing a  
Systems  
Development  
Life Cycle  
Has its Own  
Life Cycle

- We started three years ago
- It takes time to migrate systems out of one process an into another
- It really takes time to change a culture
- This is done

# What have we learned?

---

**Planning is a  
Critical  
Success  
Factor for  
any Project**

- Requirements are key – true not only for SDLC approaches
- Documentation can be copious
  - To stand alone, it may need an overall guide (System Quality Assurance Plan or System Verification and Validation Plan)
- Flexibility and judgement
  - Different systems have different needs
- Everyone has to be on board
  - Systems owners are not in IT, except for infrastructure systems
- Documentation needs – various offices, same systems

The Systems  
Development  
Life Cycle  
Can Grow to  
Meet  
Changing  
Needs

# Take Aways

---

- This is a good approach
- Can not foresee replacing it
- Will definitely grow it as our needs change