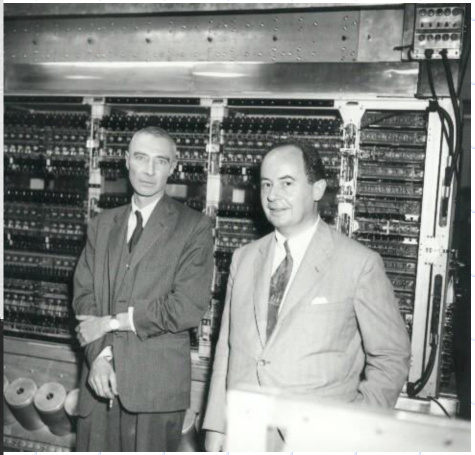
KG REDDY College of Engineering & Technology

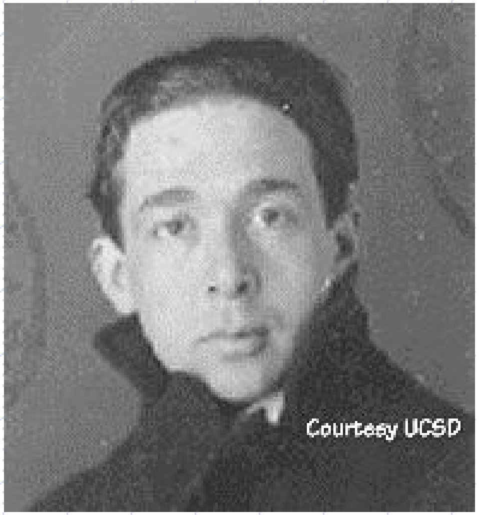
Software Engineering: A Perspective for 2003

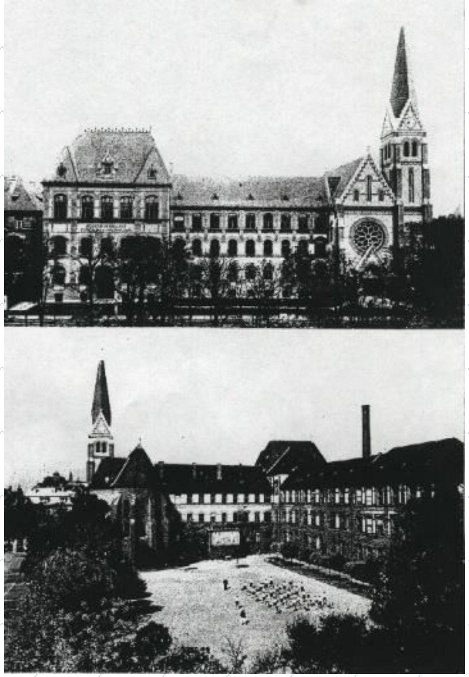
Ms Kavya Mandavalli

Seminar Dedication

Enrico Fermi referred to these brilliant Hungarian scientists as "the Martians," based on speculation that a spaceship from Mars dropped them all off in Budapest in the early 1900's.




Courtesy UCSD



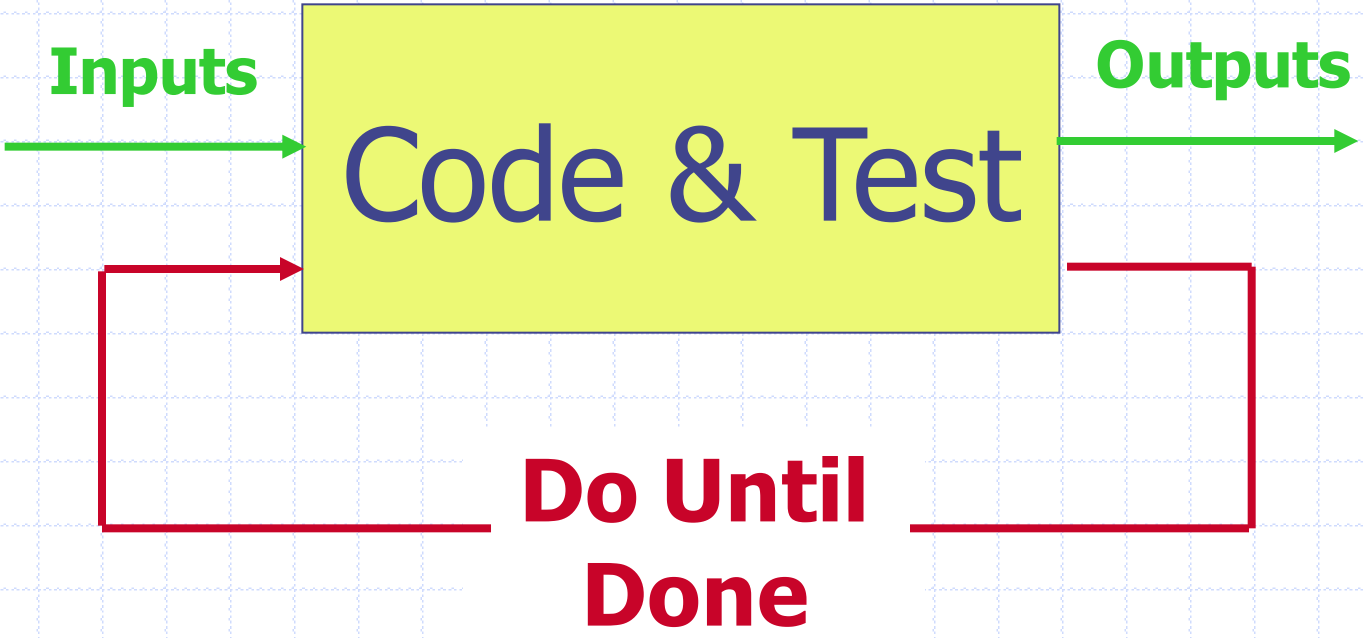
Software Engineering Seminar

Many professionals feel that the Waterfall Model is old fashioned or simplistic, having long ago outlived its usefulness – the very name seems wrong, since water cannot "fall" uphill to accommodate the backward arrows. All sorts of new models have been depicted to better show how the "real world" works, or how software can be developed faster, or how customers can become more engaged in the process to improve functionality. The Spiral Model, the Evolutionary Rapid Prototyping Model, the "V"-Shaped Model and others have emerged to solve one issue or another. Today, most practitioners might agree that there are so many different types of projects, a one size SLC cannot possible fit all. The modern viewpoint is that unique projects require unique models, or combinations of models to succeed. We will discuss the choice of appropriate SLC models, or modified versions of SLC models, the real baseline for beginning software engineering. We will describe several of the more modern SLC's (e.g. eXtreme, RUP), and how a project manager can decide which one to use. We will also explain what the various bodies of knowledge (e.g. PMBOK, SWEBOK) map to our life cycles.

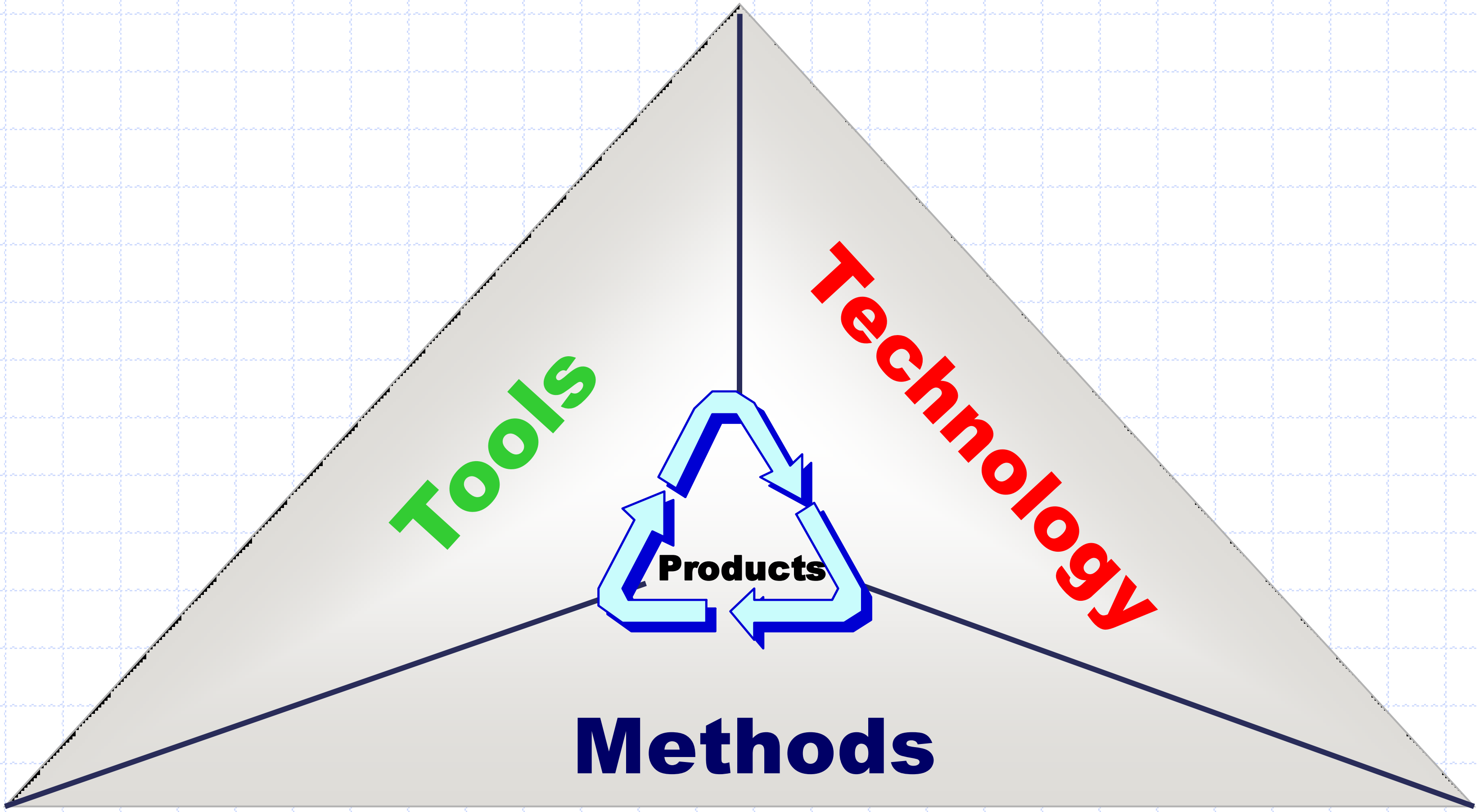
Presentation Description

- The key to managing a software development project is having a high level road map to identify where you are on the project. The life cycle model you adopt for your development project is this roadmap. Using IEEE 1074, we will walk through a "standard" development life cycle and all the supporting processes required; e.g. configuration management, documentation, project management, software quality assurance. Using this as the baseline we'll construct a first pass WBS for the life cycle.
- The next steps will be to customize the baseline life cycle for two different types of development: evolutionary rapid prototyping and commercial-of-the-shelf package selection.
- To wrap up, some metrics on life cycles for web-based application delivery.

NOT the Model you want!



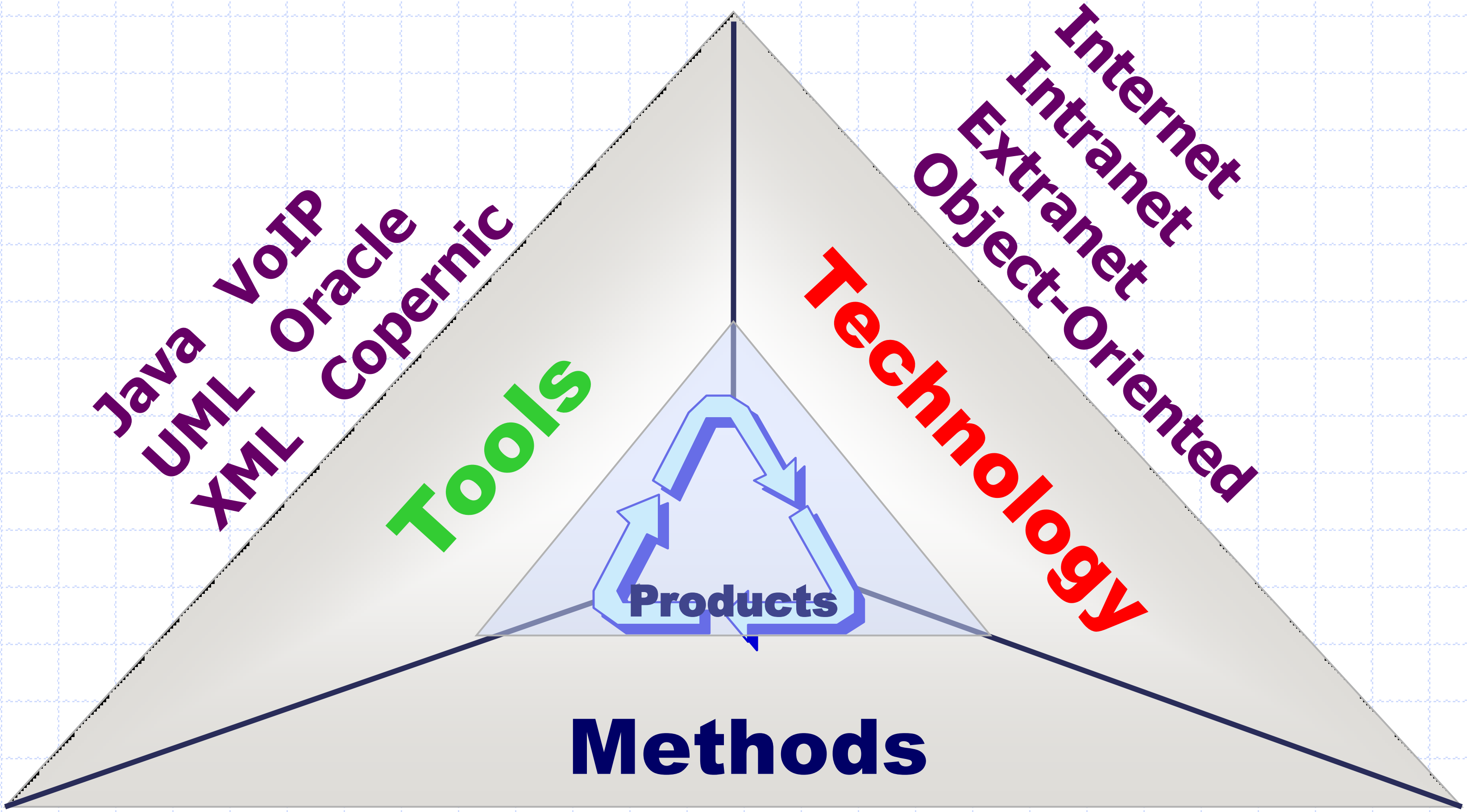
Product Development



A Quick Level Set

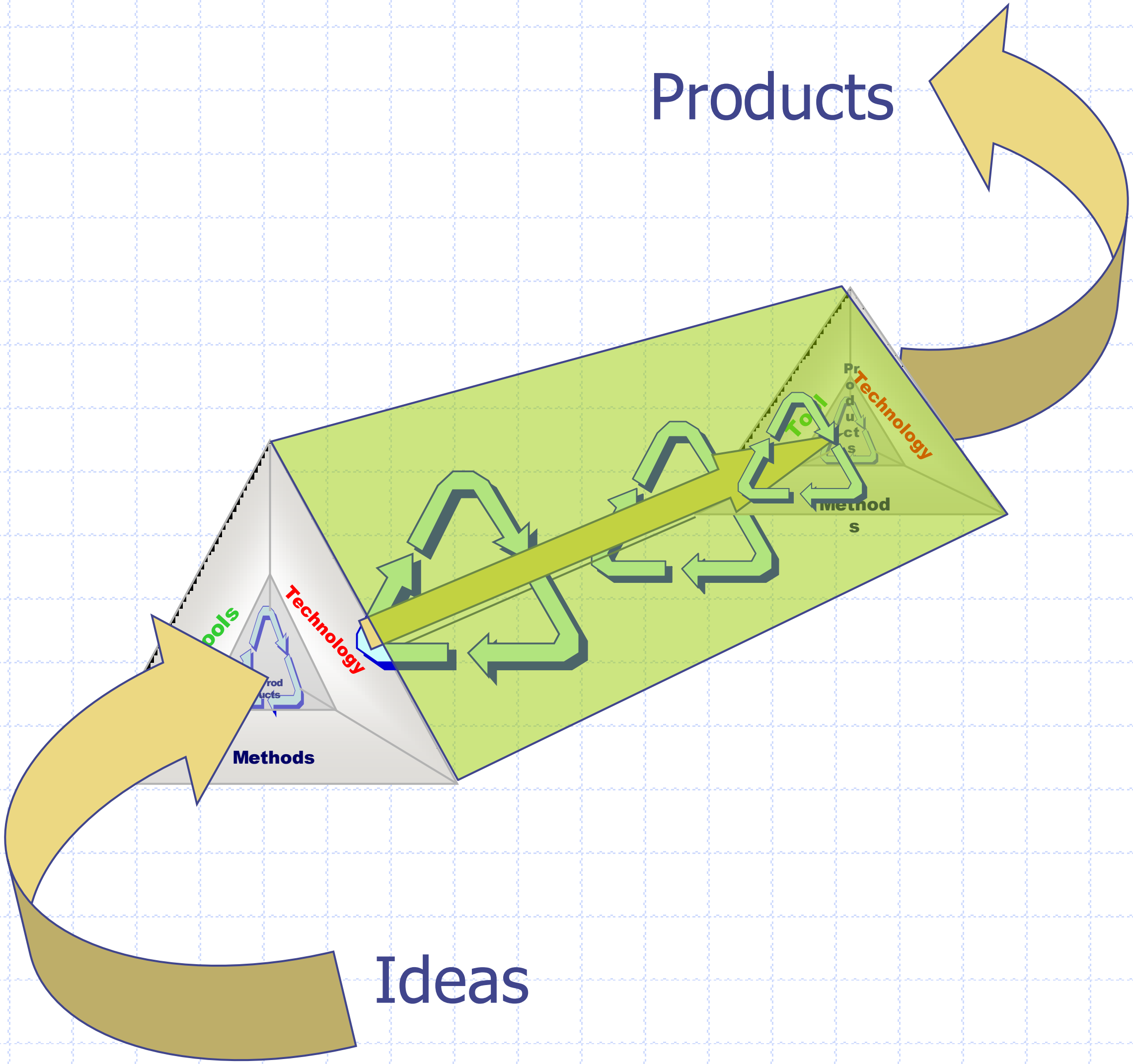
- Technology
  - Application of scientific knowledge in industry or business
- Tool
  - An implement or machine used to do work or perform a task.
- Method
  - A manner, means or process for accomplishing something.

What’s in each segment?

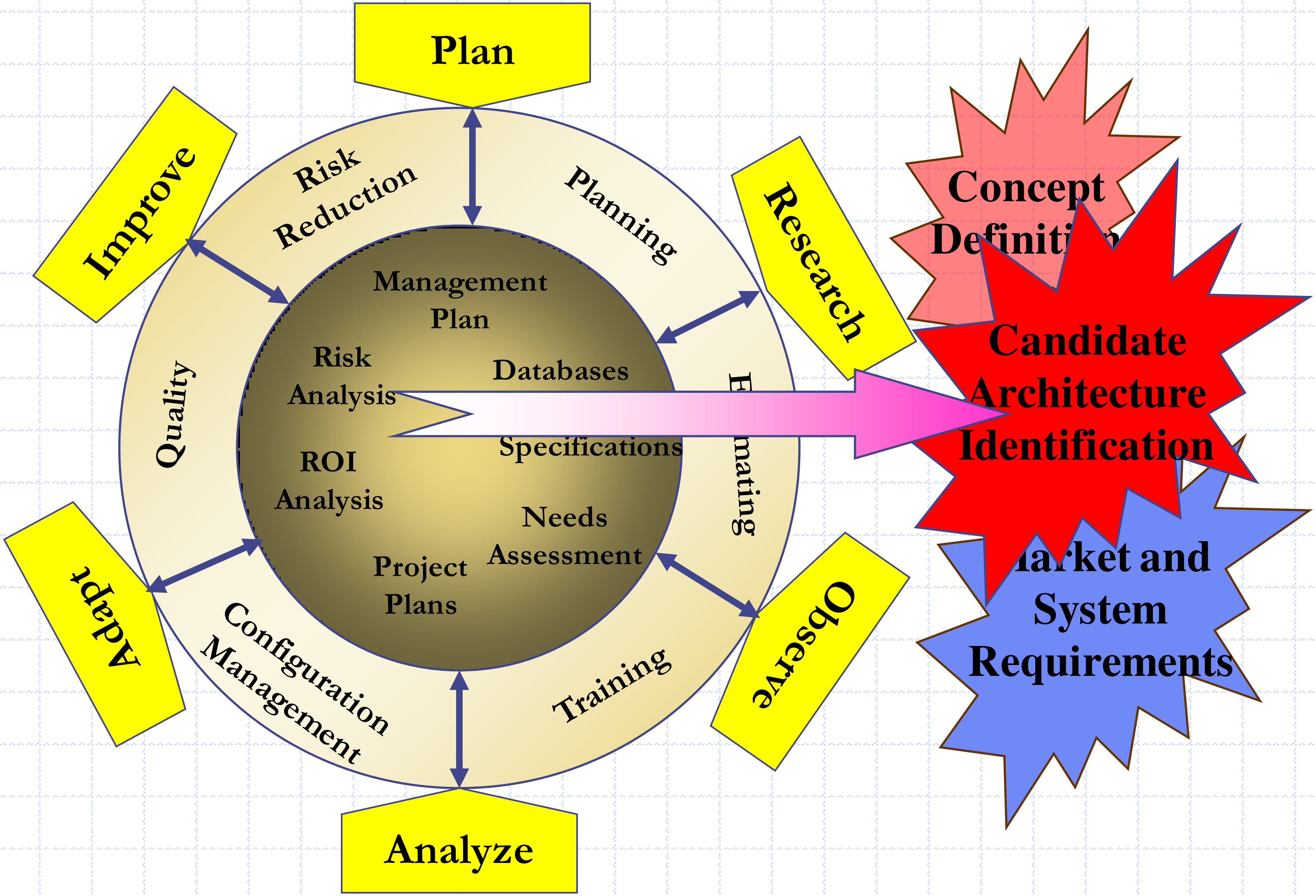


Software Engineering Project Management

How do products happen?



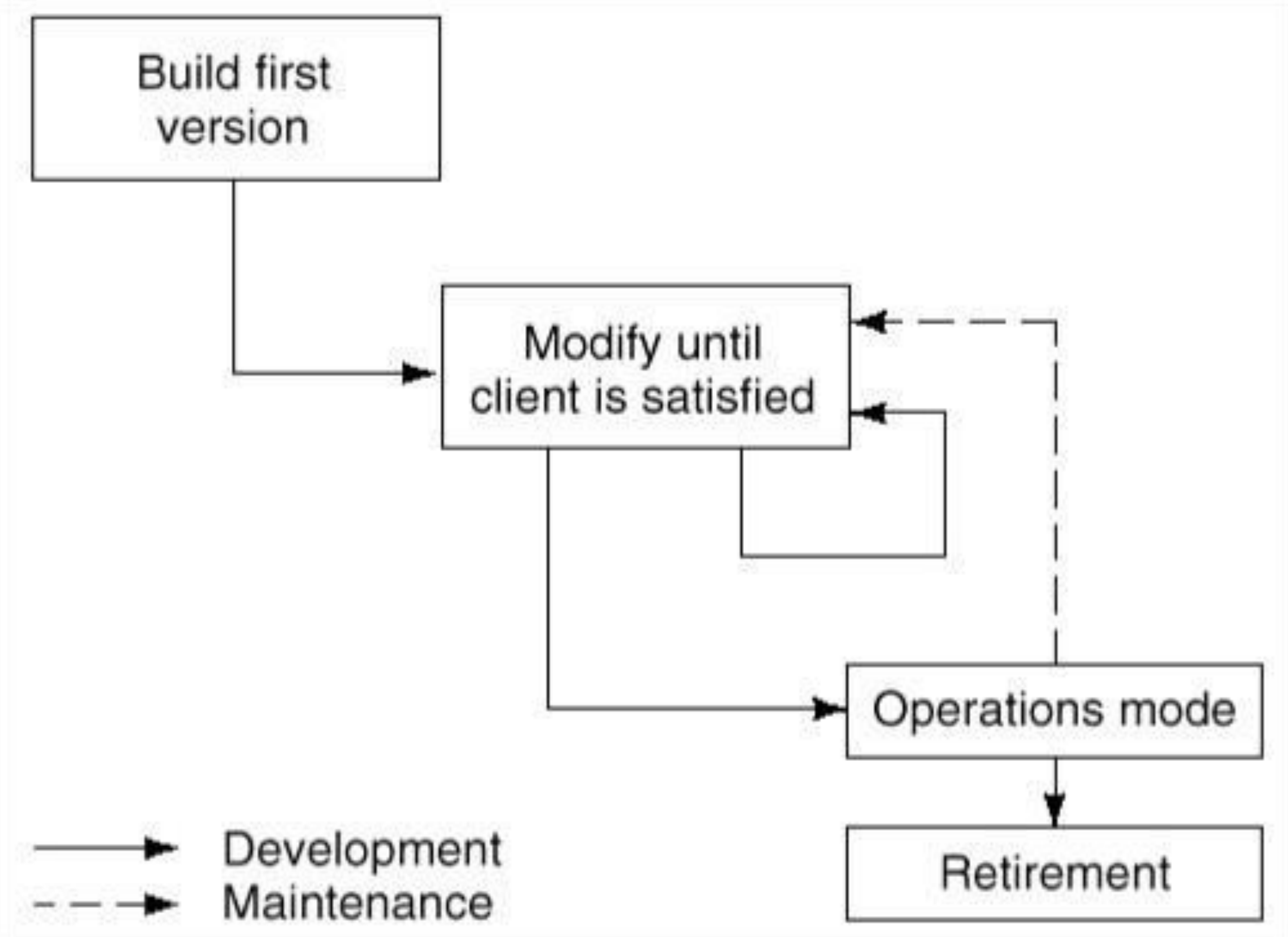
Project Management Mitigates the Front End Risks



Defining Your Life Cycle Model

1) Become familiar with the various models
2) Review, analyze the type of work: development, enhancement, maintenance, etc.
3) Review project criteria
4) Identify a minimum set of phases
5) Identify phase activities
6) Establish a minimum set of deliverables
7) Define templates and content guides for deliverables
8) Evaluate progress and effectiveness of the life cycle framework
9) Implement improvements

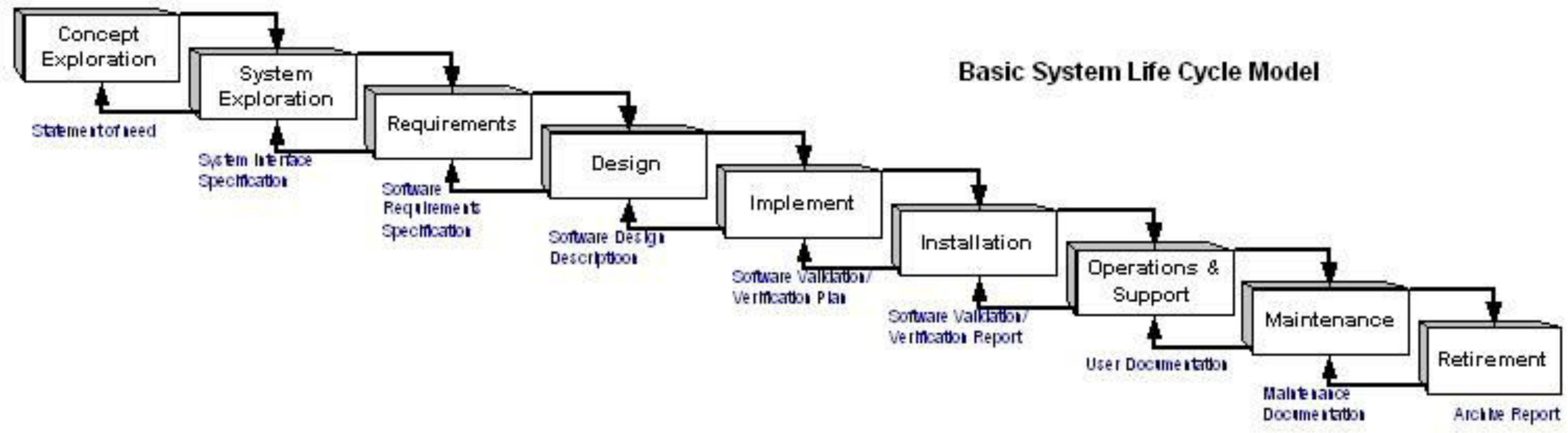
Build and Fix



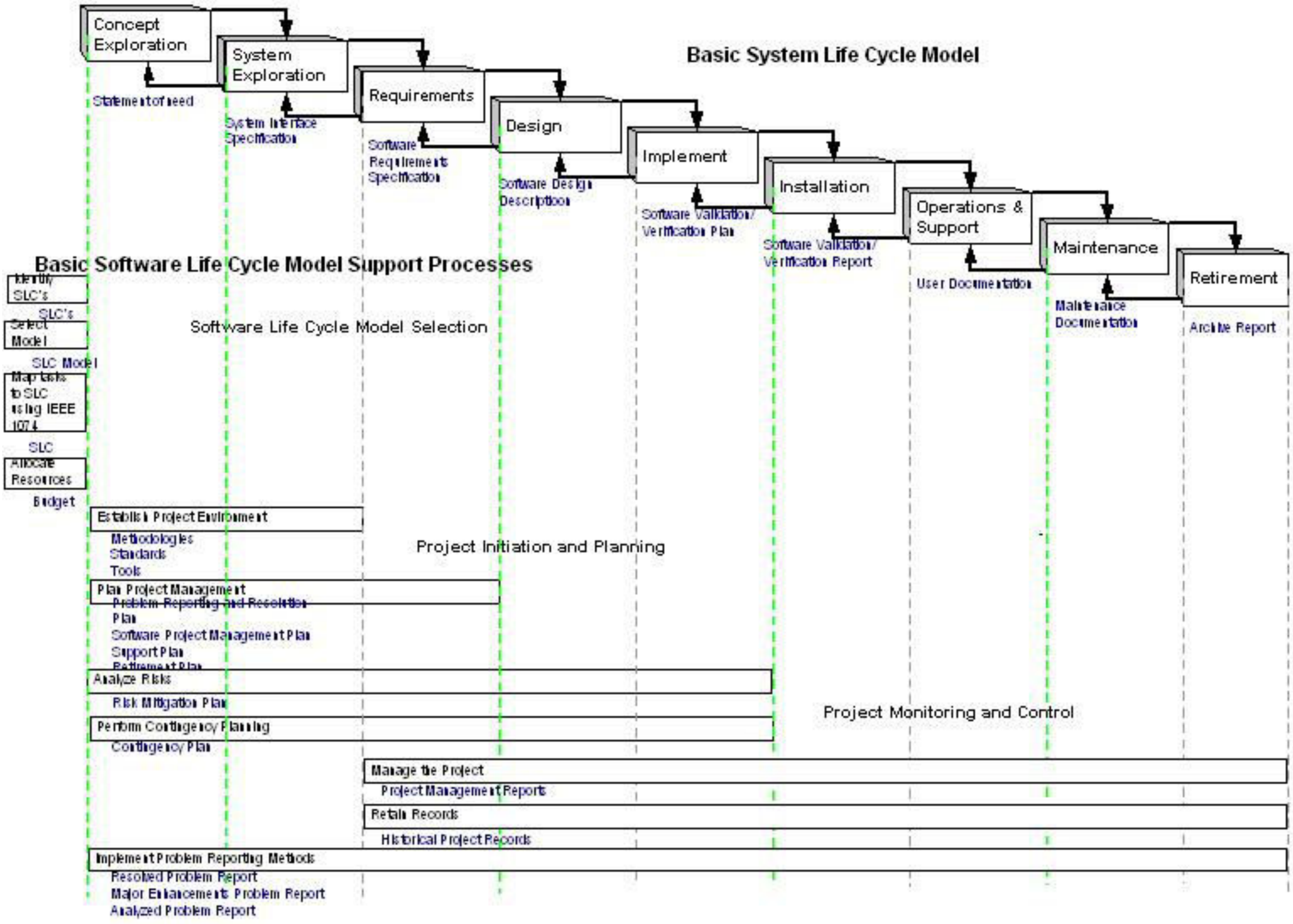
Build and Fix – Good and Bad

Pros:	Cons:
Works for projects generating less than 200 LOC	One step beyond code and test
	Does not scale with large projects
	No specifications
	Not a life cycle model

Basic 1074 Life Cycle


Basic System Life Cycle Model
Concept Exploration
System Exploration
Requirements
Design
Implement
Installation
Operations & Support
Maintenance
Retirement
Statement of need
System Interface Specification
Software Requirements Specification
Software Design Description
Software Validation/ Verification Plan
Software Validation/ Verification Report
User Documentation
Maintenance Documentation
Archive Report

Full 1074 Life Cycle (1)



Basic System Life Cycle Model

Concept Exploration

Statement of need

System Exploration

System Interface Specification

Requirements

Software Requirements Specification

Design

Software Design Description

Implement

Software Validation/ Verification Plan

Installation

Software Validation/ Verification Report

Operations & Support

User Documentation

Maintenance

Maintenance Documentation

Retirement

Archive Report

Basic Software Life Cycle Model Support Processes

Identify SLC's

SLC's

Select Model

SLC Model

Map tasks to SLC using IEEE 1074

SLC

Allocate Resources

Budget

Software Life Cycle Model Selection

Establish Project Environment

Methodologies
Standards
Tools

Project Initiation and Planning

Plan Project Management

Problem Reporting and Resolution Plan
Software Project Management Plan
Support Plan
Retirement Plan

Analyze Risks

Risk Mitigation Plan

Perform Contingency Planning

Contingency Plan

Project Monitoring and Control

Manage the Project

Project Management Reports

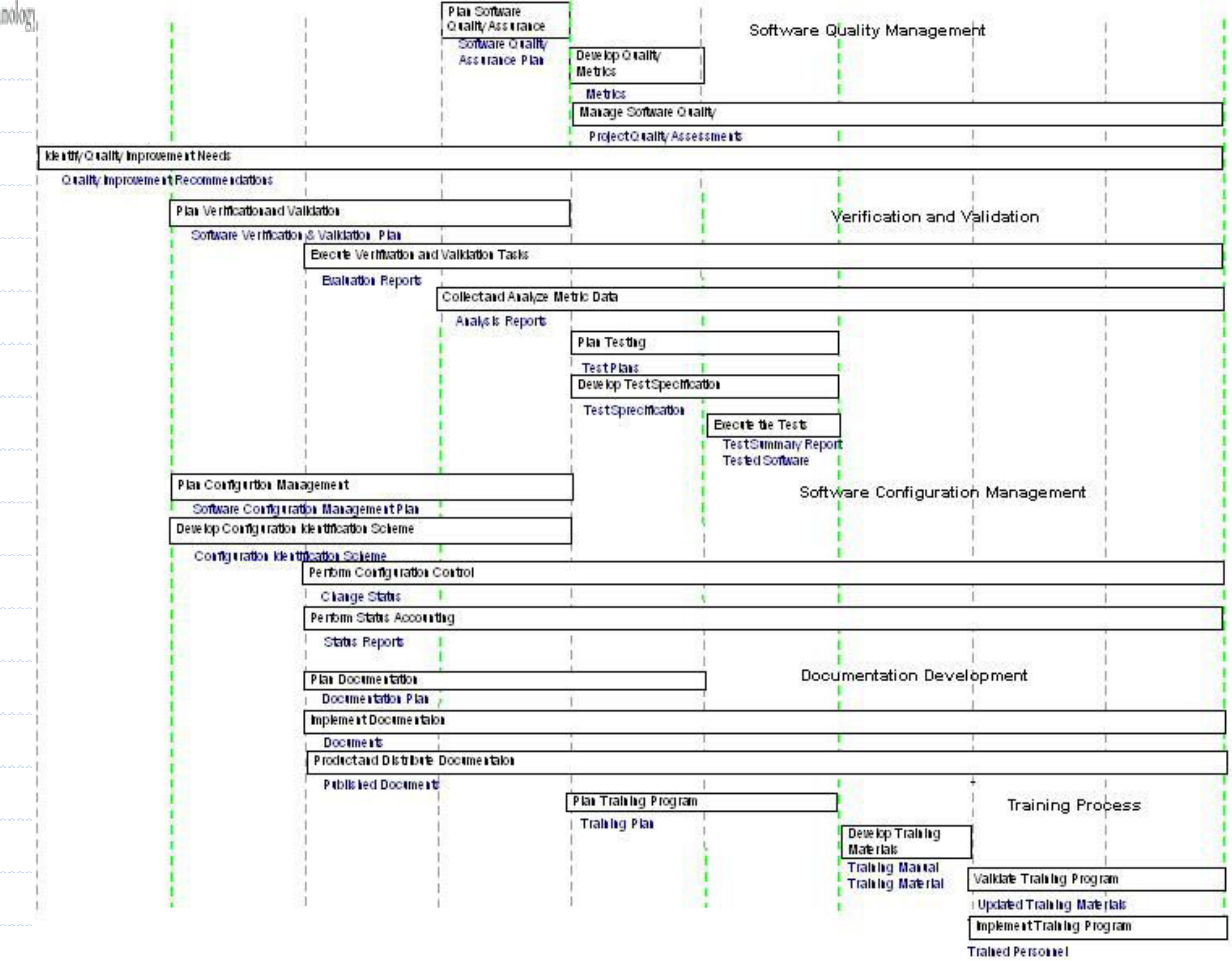
Retain Records

Historical Project Records

Implement Problem Reporting Methods

Resolved Problem Report
Major Enhancements Problem Report
Analyzed Problem Report

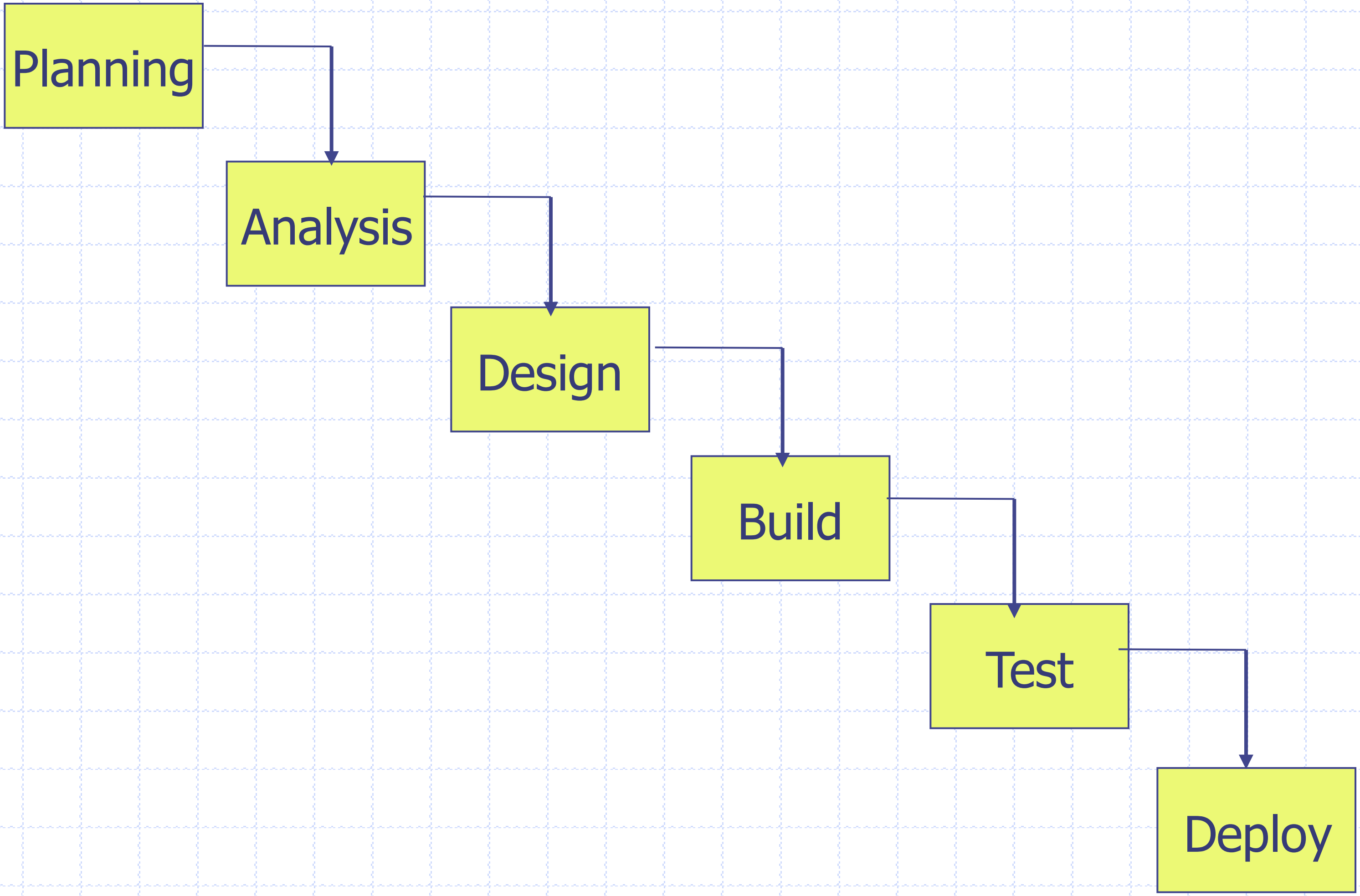
Full 1074 Life Cycle (2)


Plan Software Quality Assurance
Software Quality Assurance Plan
Software Quality Management
Develop Quality Metrics
Metrics
Manage Software Quality
Project Quality Assessments
Identify Quality Improvement Needs
Quality Improvement Recommendations
Plan Verification and Validation
Verification and Validation
Software Verification & Validation Plan
Execute Verification and Validation Tasks
Evaluation Reports
Collect and Analyze Metric Data
Analysis Reports
Plan Testing
Test Plans
Develop Test Specification
Test Sprecification
Execute the Tests
Test Summary Report
Tested Software
Plan Configuration Management
Software Configuration Management
Software Configuration Management Plan
Develop Configuration Identification Scheme
Configuration Identification Scheme
Perform Configuration Control
Change Status
Perform Status Accounting
Status Reports
Plan Documentation
Documentation Development
Documentation Plan
Implement Documentation
Documents
Product and Distribute Documentation
Published Documents
Plan Training Program
Training Process
Training Plan
Develop Training Materials
Training Manual
Training Material
Validate Training Program
Updated Training Materials
Implement Training Program
Trained Personnel

Full 1074 Life Cycle – Good and Bad

Pros:	Cons:
THE starting point for defining you life cycle	Too much process
Contains all the life cycle supports you would need	Contains more than you may reasonably use
Is a process for defining your life cycle	Is not in and of itself a life cycle to implement

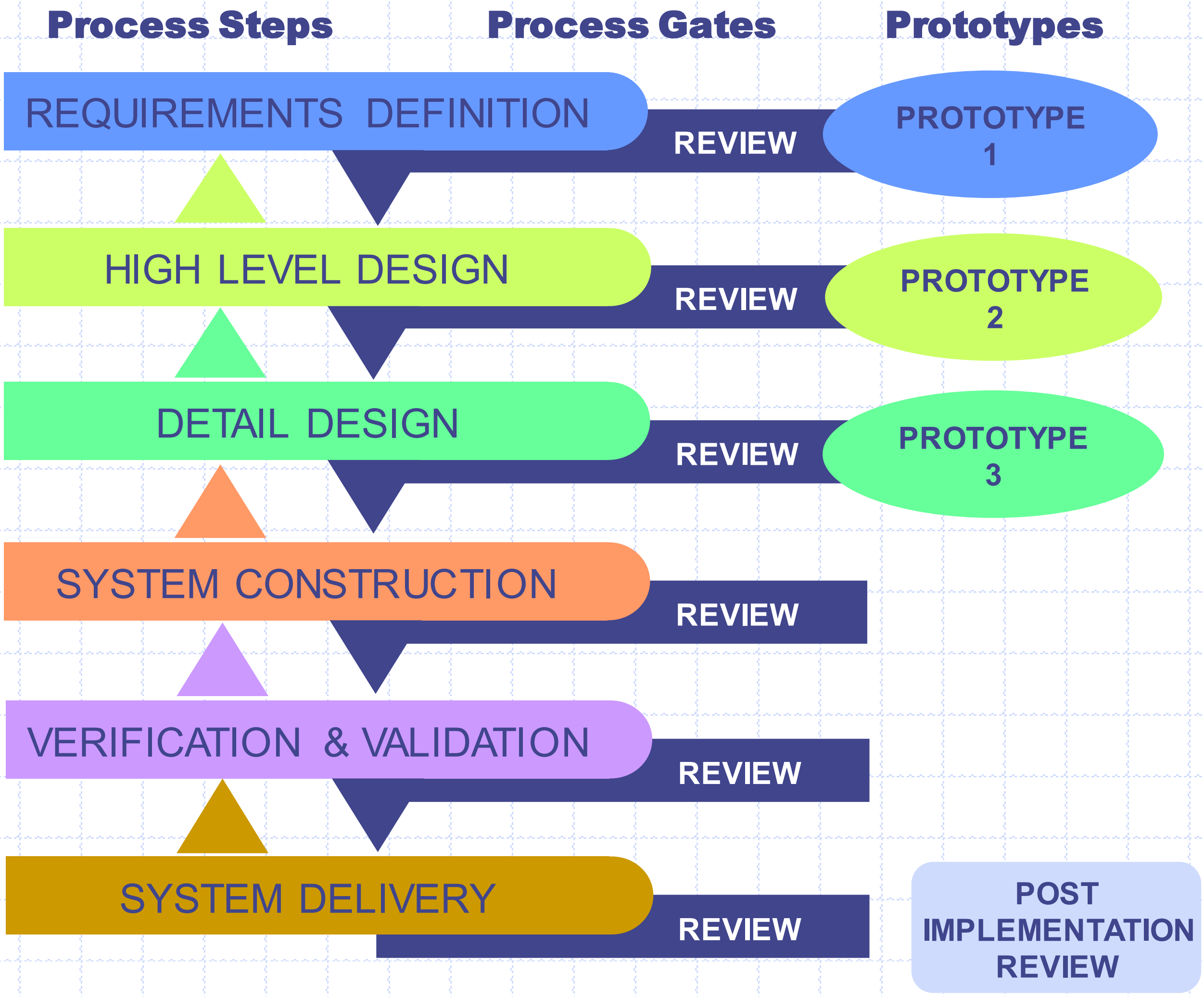
Waterfall Model



Waterfall Model – Good and Bad

Pros:	Cons:
Easiest to understand	Does not model the real world
Easiest to instrument	Too much documentation
Enforced discipline	
Document and deliverable driven	

Waterfall with Prototyping



Project Management Support Processes

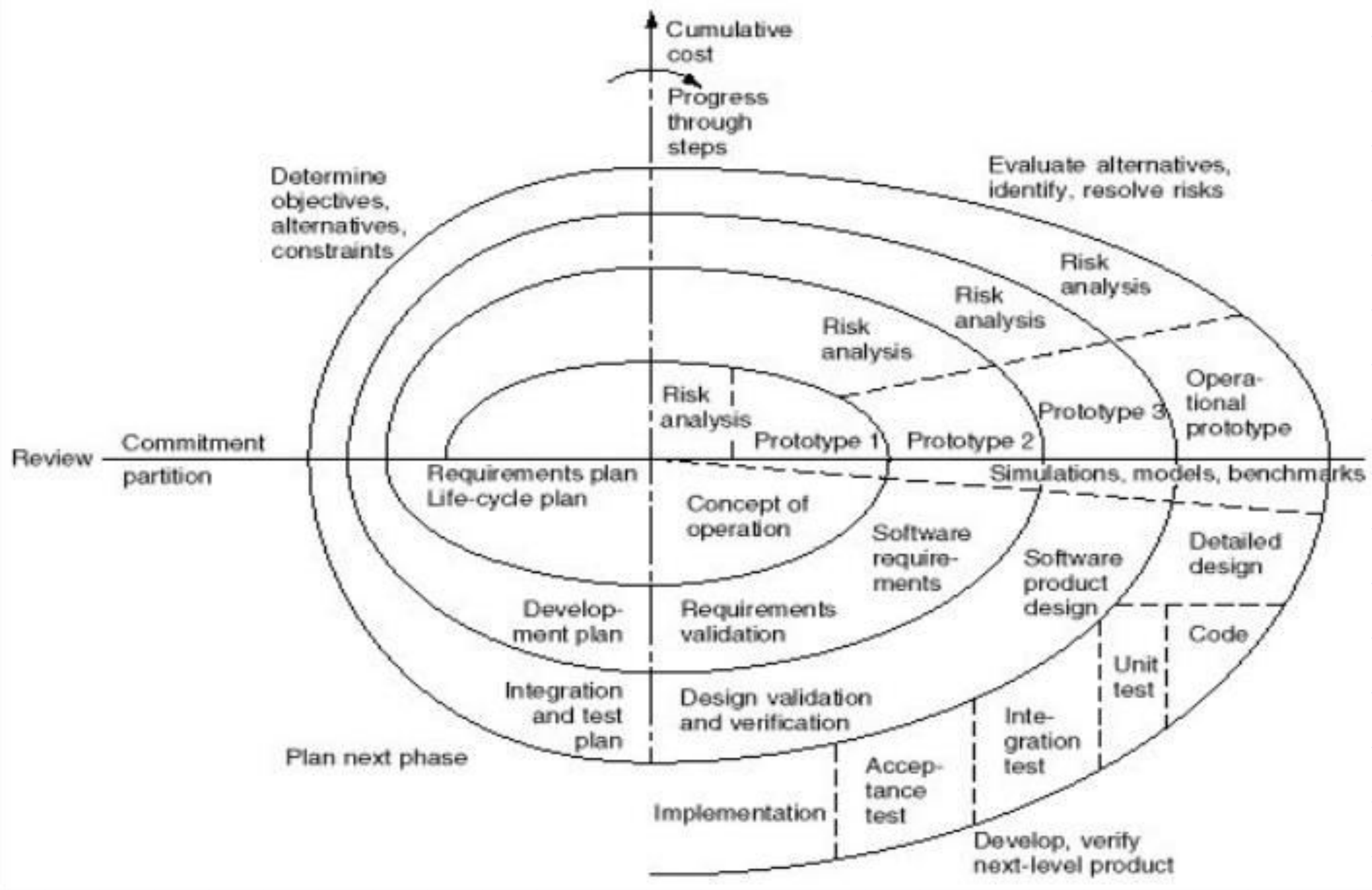
Risk Reduction Training Planning Configuration Management Estimating Metrics Quality Assurance

Prototyping Model - Pros and Cons

Pros:	Cons:
Easiest to understand	Not stopping the prototyping
Easiest to instrument	Prototyping becomes early code hacking
Real world modeling	
Recursion among process steps	
Document and deliverable driven	

KG REDDY College of Engineering & Technology

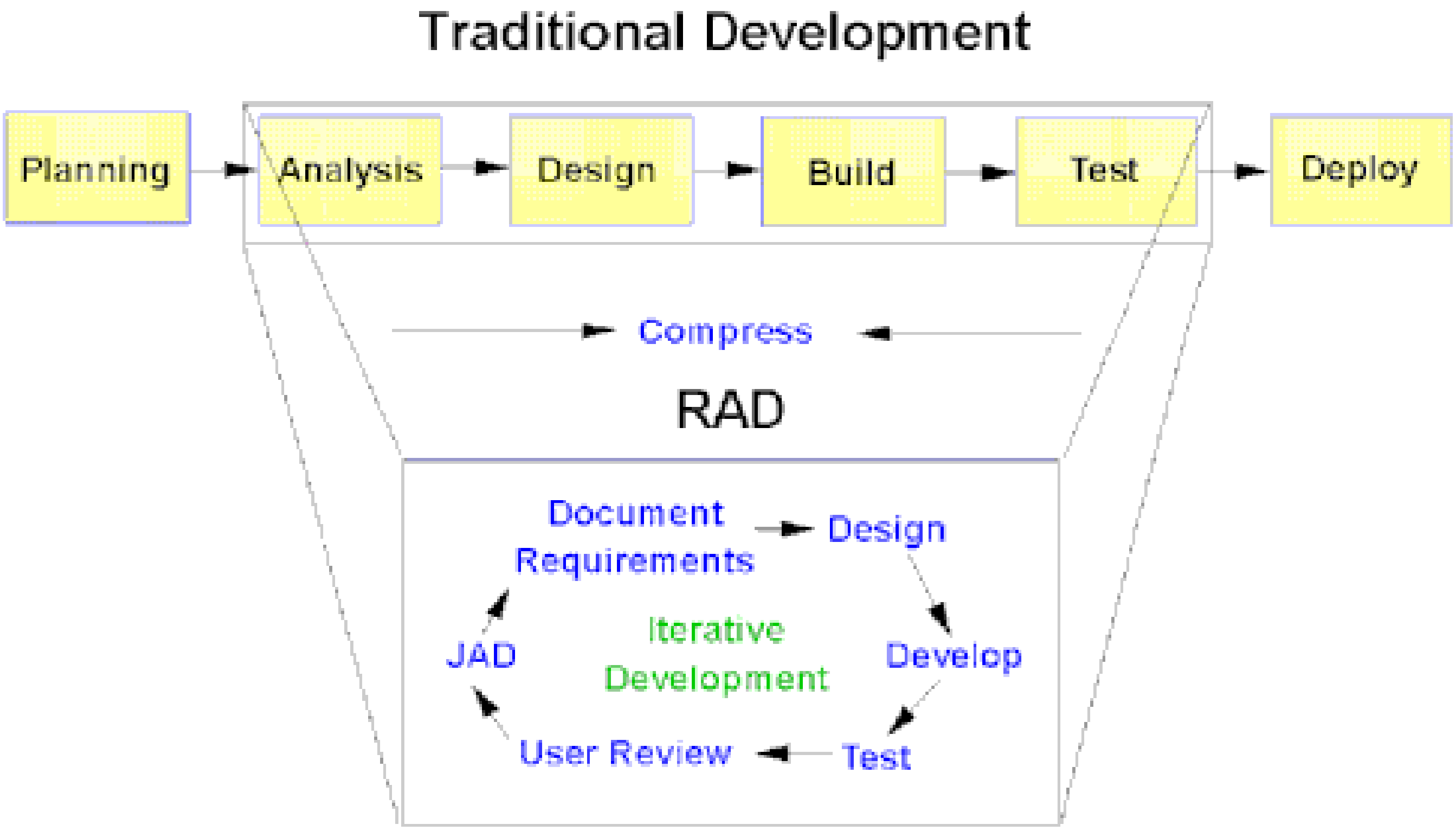
Spiral Model



Spiral Good and Bad

Pros:	Cons:
Emphasizes risk reduction	Internal development of large systems
Supports reuse	High overhead costs
Maintenance and development mesh	Requires a mature organization
Easy look at product with prototypes	Risk and prototyping tools a must
Risk focused testing	

Rapid Application Development



RAD – Good/Bad

Pros:	Cons:
Lots of user interaction	Users intimately involved
Early proof of concept	Needs maturity of tools and process
Incremental building	Increased overhead if too many prototypes
	Tight delivery control
	Poorly set expectations

Selecting a Life Cycle Model - Project Characteristic Category Matrix Requirements

Requirements	Waterfall	Prototype	Spiral	RAD
Are the requirements easily defined and/or well known?	Yes	No	No	Yes
Can the requirements be defined early in the cycle?	Yes	No	No	Yes
Will the requirements change often in the cycle?	No	Yes	Yes	No
Is there a need to demonstrate the requirements to achieve definition?	No	Yes	Yes	Yes
Is a proof of concept required to demonstrate capability?	No	Yes	Yes	Yes

Selecting a Life Cycle Model - Project Characteristic Category Matrix Project Team

Project Team	Waterfall	Prototype	Spiral	RAD
Are the majority of team members new to the problem domain for the project?	No	Yes	Yes	No
Are the majority of team members new to the technology domain for the project?	Yes	No	Yes	No
Are the majority of team members new to the tools used on the project?	Yes	No	Yes	No
Are the team members subject to reassignment during the life cycle?	No	Yes	Yes	No
Is there training available for the project team if required?	No	No	No	Yes

Selecting a Life Cycle Model - Project Characteristic Category Matrix User Community

User Community	Waterfall	Prototype	Spiral	RAD
Will the availability of the user representatives be restricted, or limited during the life cycle?	Yes	No	Yes	No
Are the user representatives new to system definition?	No	Yes	Yes	No
Are the user representatives experts in the problem domain?	No	Yes	No	Yes
Do the users want to be involved in all phases of the life cycle?	No	Yes	No	Yes

Selecting a Life Cycle Model - Project Characteristic Category Matrix Project Type and Risk

Project Type & Risk	Waterfall	Prototype	Spiral	RAD
Does the project identify a new product direction for the organization?	No	Yes	Yes	No
Is the project a system integration project?	No	Yes	Yes	Yes
Is the project an enhancement to an existing system?	No	No	No	Yes
Is the funding for the project expected to be stable throughout the life cycle?	Yes	Yes	No	Yes
Is the product expected to have a long life in the organization?	Yes	No	Yes	No

Two Derived Development Methods

- COTs
- eXtreme Programming

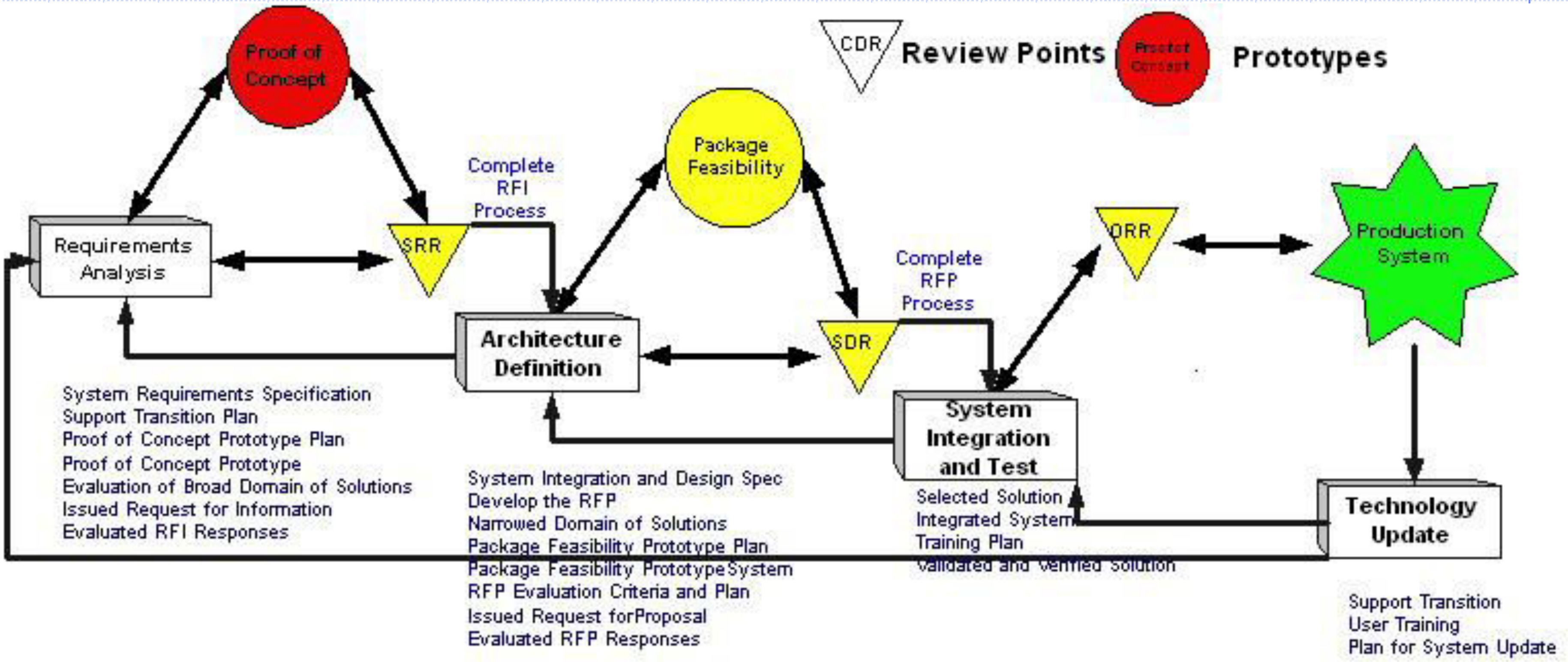
Before Customizing Remember the Framework Activities …

- An effective process model should define a small set of framework activities that are always applicable, regardless of project type. The APM defines the following set of framework activities:
  - I. project definition - tasks required to establish effective communication between developer and customer(s) and to define requirements for the work to be performed
  - II. planning - tasks required to define resources, timelines and other project related information and assess both technical and management risks
  - III. engineering and construction - tasks required to create one or more representations of the software (can include the development of executable models, i.e., prototypes or simulations) and to generate code and conduct thorough testing
  - IV. release - tasks required to install the software in its target environment, and provide customer support (e.g., documentation and training)
  - V. customer use - tasks required to obtain customer feedback based on use and evaluation of the deliverables produced during the release activity
- Each of the above framework activities will occur for every project. However, the set of tasks (we call this a task set) that is defined for each framework activity will vary depending upon the project type (e.g., Concept Development Projects will have a different task set than Application Enhancement Projects) and the degree of rigor selected for the project.

… and the Umbrella Activities

- In addition to the framework activities, a set of umbrella activities persist across the entire software process. These umbrella activities include:
- software project management
- formal technical reviews
- software quality assurance
- software configuration management
- reusability management
- measurement
- document preparation and production
- risk management
- Each of these umbrella activities is defined by a set of tasks that are adapted to the project type and degree of rigor with which software engineering is to be applied.

COTS Application Selection (1)



CDR Review Points

Proof of Concept Prototypes

Proof of Concept

Requirements Analysis

SRR

Complete RFI Process

Architecture Definition

Package Feasibility

SDR

Complete RFP Process

System Integration and Test

ORR

Production System

Technology Update

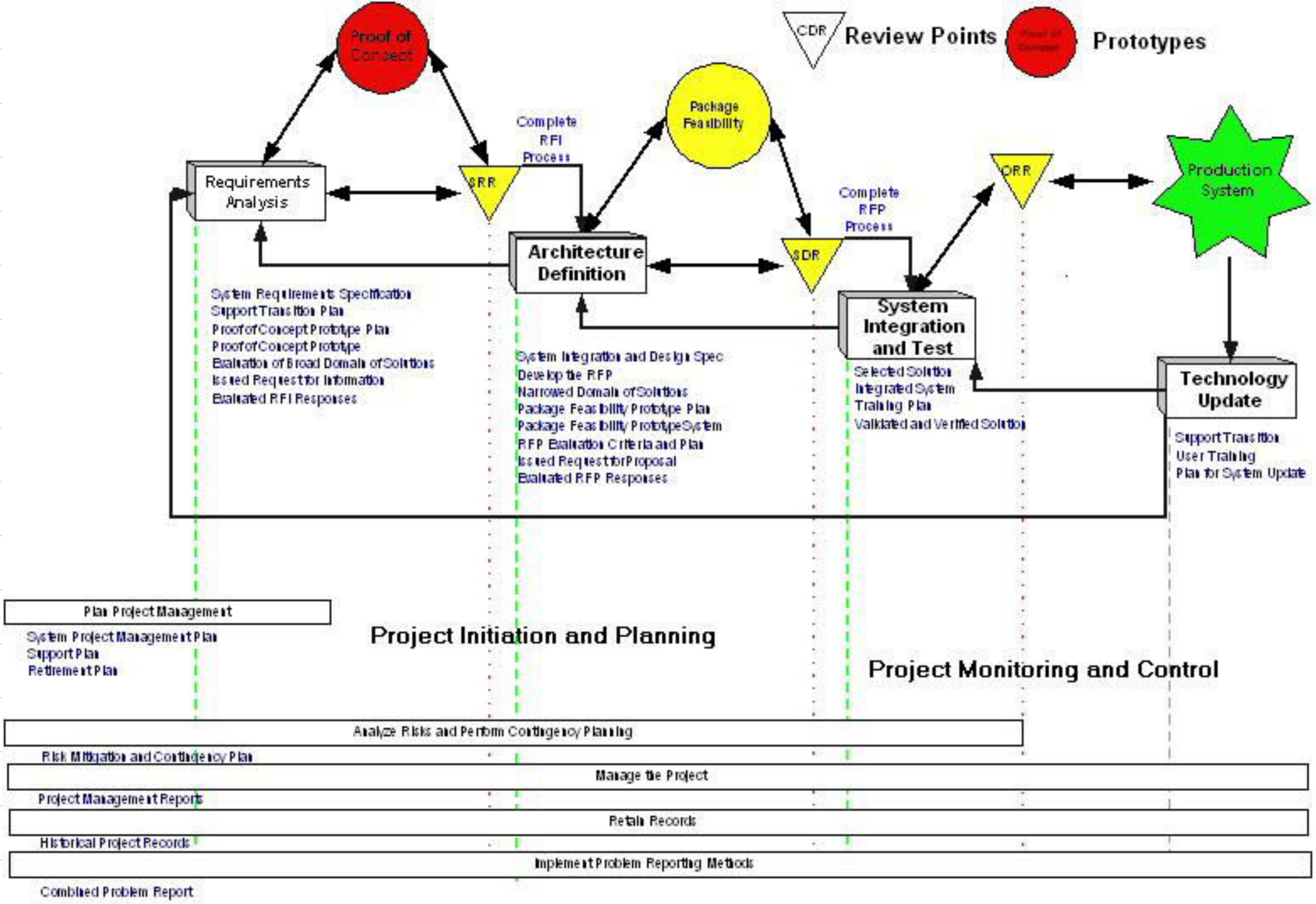
System Requirements Specification
Support Transition Plan
Proof of Concept Prototype Plan
Proof of Concept Prototype
Evaluation of Broad Domain of Solutions
Issued Request for Information
Evaluated RFI Responses

System Integration and Design Spec
Develop the RFP
Narrowed Domain of Solutions
Package Feasibility Prototype Plan
Package Feasibility Prototype System
RFP Evaluation Criteria and Plan
Issued Request forProposal
Evaluated RFP Responses

Selected Solution
Integrated System
Training Plan
Validated and Verified Solution

Support Transition
User Training
Plan for System Update

COTS Life Cycle (2)



CDR Review Points

Prototypes

Proof of Concept

Complete RFI Process

SRR

Package Feasibility

Requirements Analysis

Complete RFP Process

ORR

Production System

Architecture Definition

SDR

System Integration and Test

Technology Update

System Requirements Specification
Support Transition Plan
Proof of Concept Prototype Plan
Proof of Concept Prototype
Evaluation of Broad Domain of Solutions
Issued Request for Information
Evaluated RFI Responses

System Integration and Design Spec
Develop the RFP
Narrowed Domain of Solutions
Package Feasibility Prototype Plan
Package Feasibility Prototype System
RFP Evaluation Criteria and Plan
Issued Request for Proposal
Evaluated RFP Responses

Selected Solution
Integrated System
Training Plan
Validated and Verified Solution

Support Transition
User Training
Plan for System Update

Plan Project Management

System Project Management Plan
Support Plan
Retirement Plan

Project Initiation and Planning

Project Monitoring and Control

Analyze Risks and Perform Contingency Planning

Risk Mitigation and Contingency Plan

Manage the Project

Project Management Reports

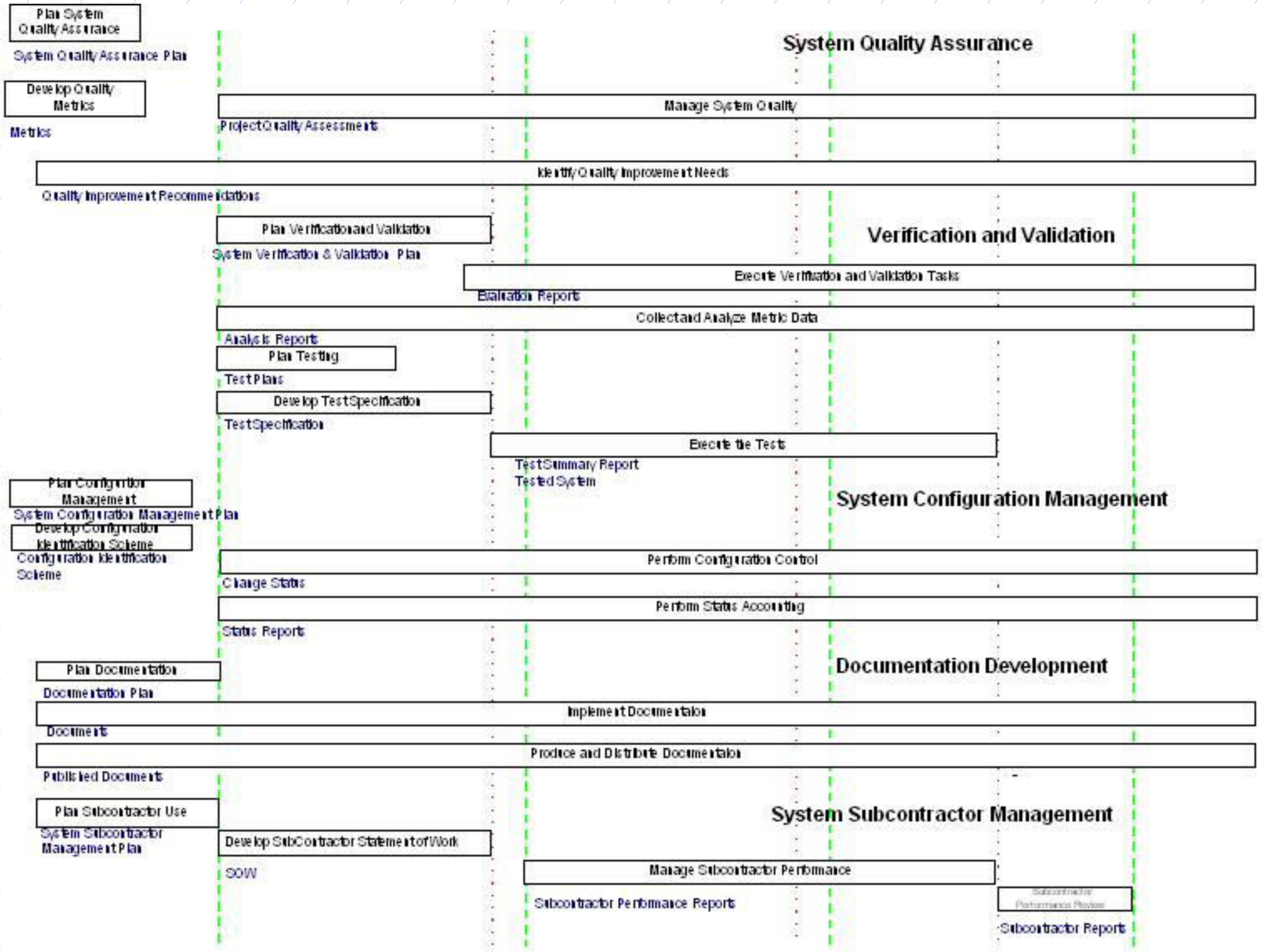
Retain Records

Historical Project Records

Implement Problem Reporting Methods

Combined Problem Report

COTS Life Cycle (3)



System Quality Assurance

Plan System Quality Assurance

System Quality Assurance Plan

Develop Quality Metrics

Metrics

Manage System Quality

Project Quality Assessments

Identify Quality Improvement Needs

Quality Improvement Recommendations

Verification and Validation

Plan Verification and Validation

System Verification & Validation Plan

Execute Verification and Validation Tasks

Evaluation Reports

Collect and Analyze Metric Data

Analysis Reports

Plan Testing

Test Plans

Develop Test Specification

Test Specification

Execute the Tests

Test Summary Report

Tested System

System Configuration Management

Plan Configuration Management

System Configuration Management Plan

Develop Configuration Identification Scheme

Configuration Identification Scheme

Perform Configuration Control

Change Status

Perform Status Accounting

Status Reports

Documentation Development

Plan Documentation

Documentation Plan

Implement Documentaion

Documents

Produce and Distribute Documentaion

Published Documents

System Subcontractor Management

Plan Subcontractor Use

System Subcontractor Management Plan

Develop SubContractor Statement of Work

SOW

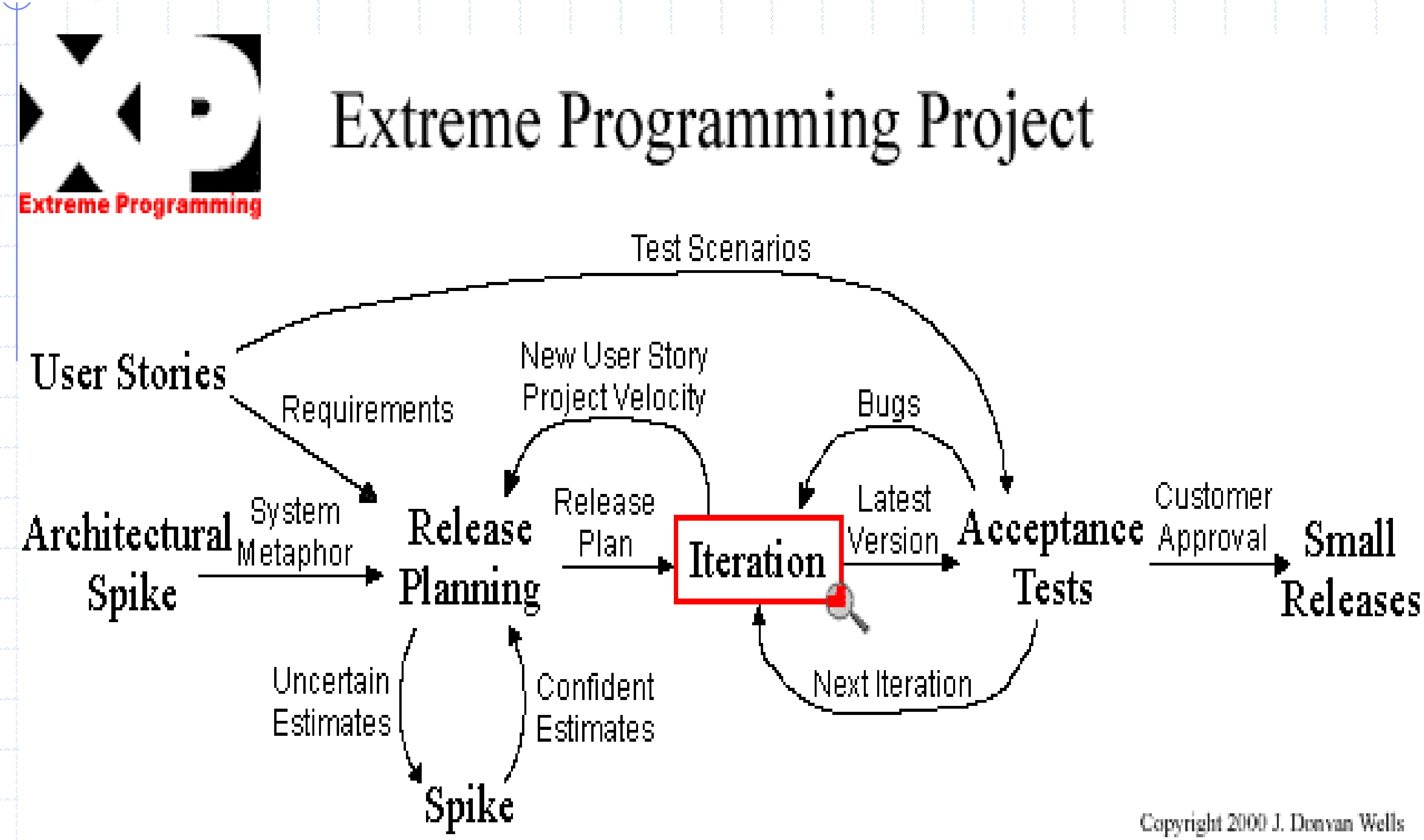
Manage Subcontractor Performance

Subcontractor Performance Reports

Subcontractor Performance Review

Subcontractor Reports

eXtreme Programming



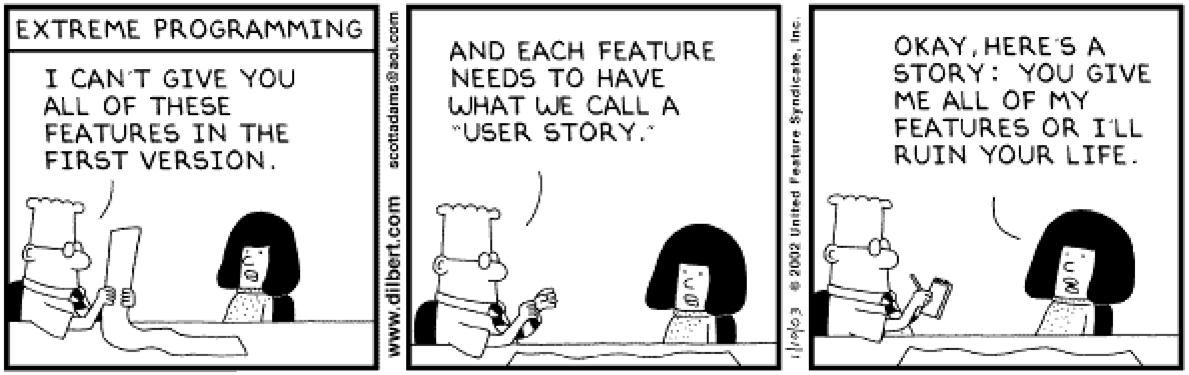
eXtreme Programming - the propaganda

- Light methods are adaptive rather than predictive. Heavy methods tend to try to plan out a large part of the software process in great detail for a long span of time, this works well until things change. So their nature is to resist change. The light methods, however, welcome change. They try to be processes that adapt and thrive on change, even to the point of changing themselves.
- Light methods are people-oriented rather than process-oriented. They explicitly make a point of trying to work with peoples' nature rather than against them and to emphasize that software development should be an enjoyable activity.

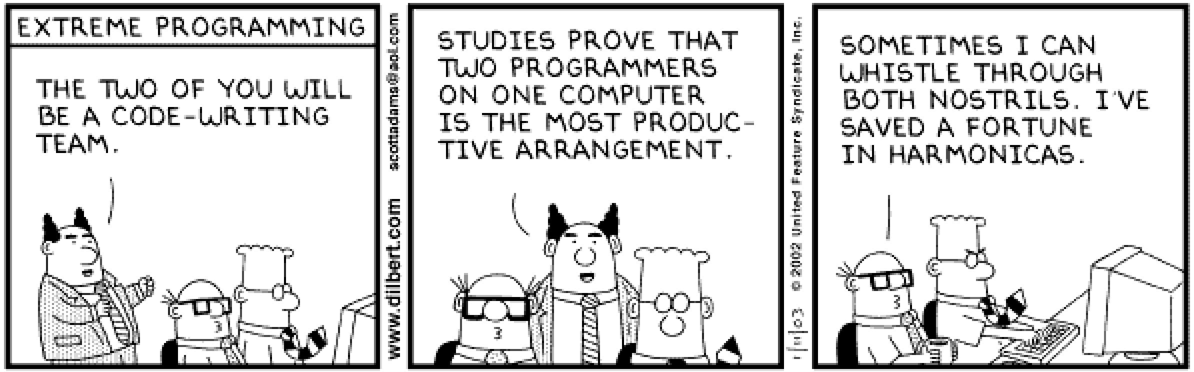
eXtreme Programming - the truth :-)

WE'RE GOING TO TRY SOMETHING CALLED EXTREME PROGRAMMING.
FIRST, PICK A PARTNER. THE TWO OF YOU WILL WORK AT ONE COMPUTER FOR FORTY HOURS A WEEK.
THE NEW SYSTEM IS A MINUTE OLD AND I ALREADY HATE EVERYONE.
www.dilbert.com scottadams@aol.com
© 2002 United Feature Syndicate, Inc.
1/9/03

Copyright © 2003 Ur

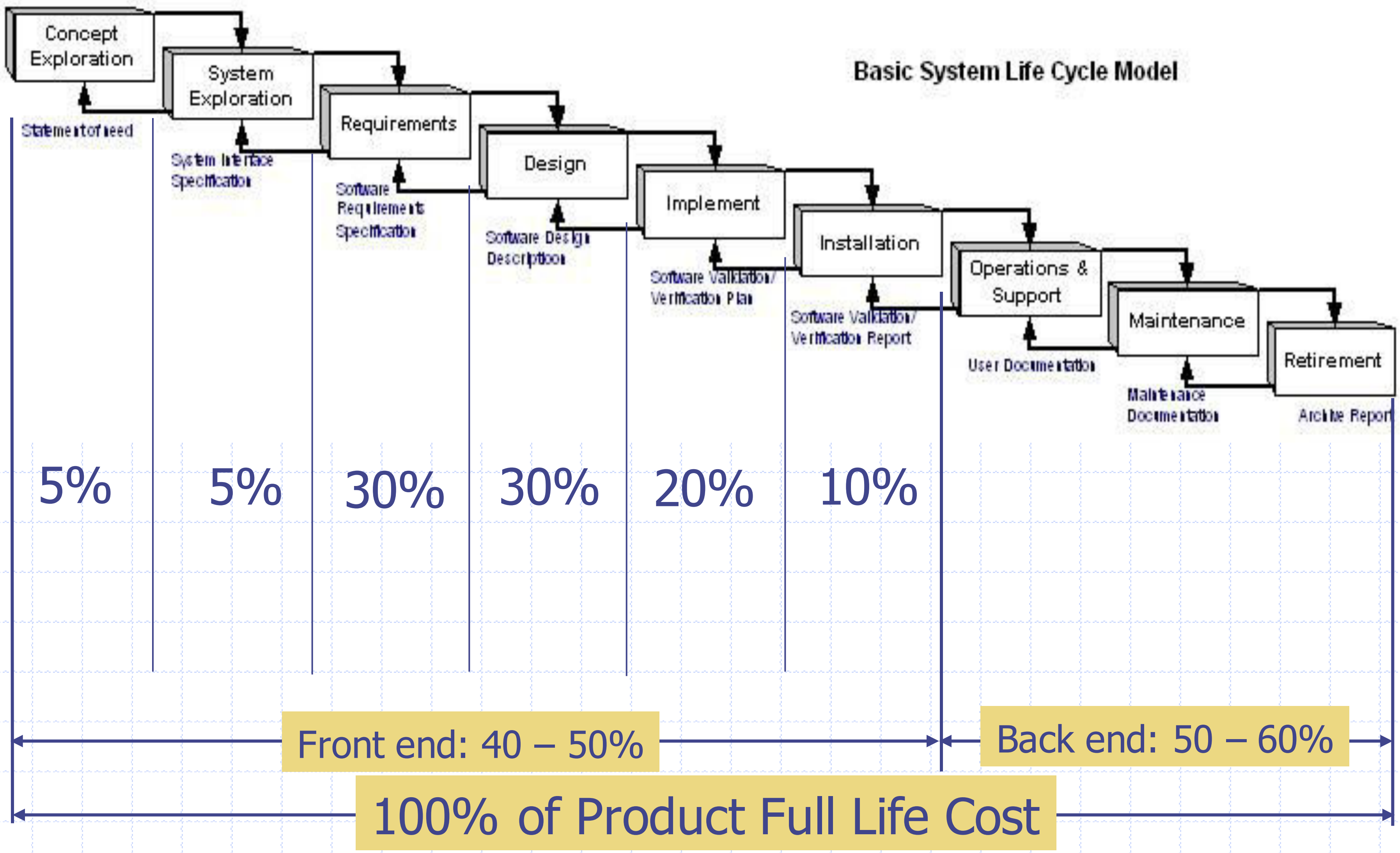

EXTREME PROGRAMMING
I CAN'T GIVE YOU ALL OF THESE FEATURES IN THE FIRST VERSION.
AND EACH FEATURE NEEDS TO HAVE WHAT WE CALL A "USER STORY."
OKAY, HERE'S A STORY: YOU GIVE ME ALL OF MY FEATURES OR I'LL RUIN YOUR LIFE.
www.dilbert.com scottadams@aol.com
© 2002 United Feature Syndicate, Inc.
1/10/03

Copyright © 2003 Un


EXTREME PROGRAMMING
THE TWO OF YOU WILL BE A CODE-WRITING TEAM.
STUDIES PROVE THAT TWO PROGRAMMERS ON ONE COMPUTER IS THE MOST PRODUCTIVE ARRANGEMENT.
SOMETIMES I CAN WHISTLE THROUGH BOTH NOSTRILS. I'VE SAVED A FORTUNE IN HARMONICAS.
www.dilbert.com scottadams@aol.com
© 2002 United Feature Syndicate, Inc.
1/11/03

Copyright © 2003 United Feature Syndicate, Inc.

Classical "Best" Effort per Phase



Basic System Life Cycle Model

Concept Exploration

System Exploration

Requirements

Design

Implement

Installation

Operations & Support

Maintenance

Retirement

Statement of Need

System Interface Specification

Software Requirements Specification

Software Design Description

Software Validation/ Verification Plan

Software Validation/ Verification Report

User Documentation

Maintenance Documentation

Archive Report

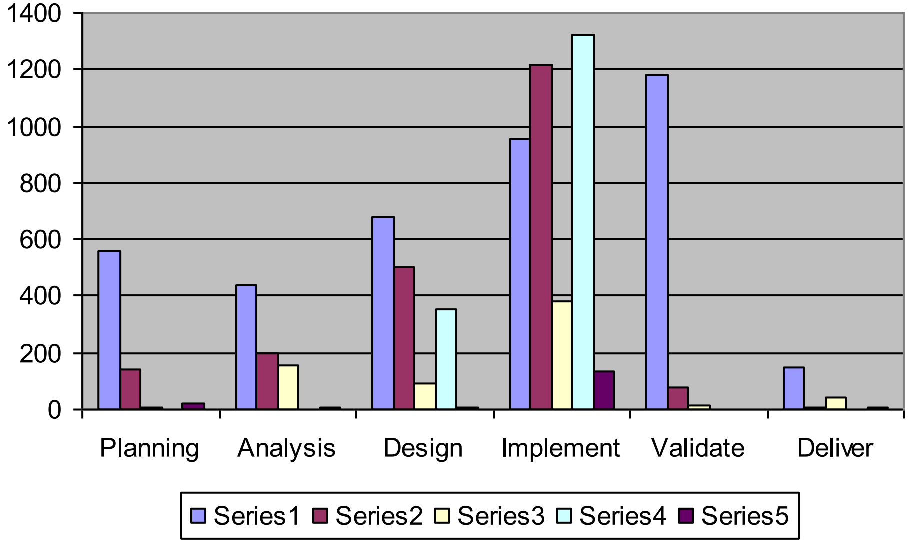
5% 5% 30% 30% 20% 10%

Front end: 40 – 50%

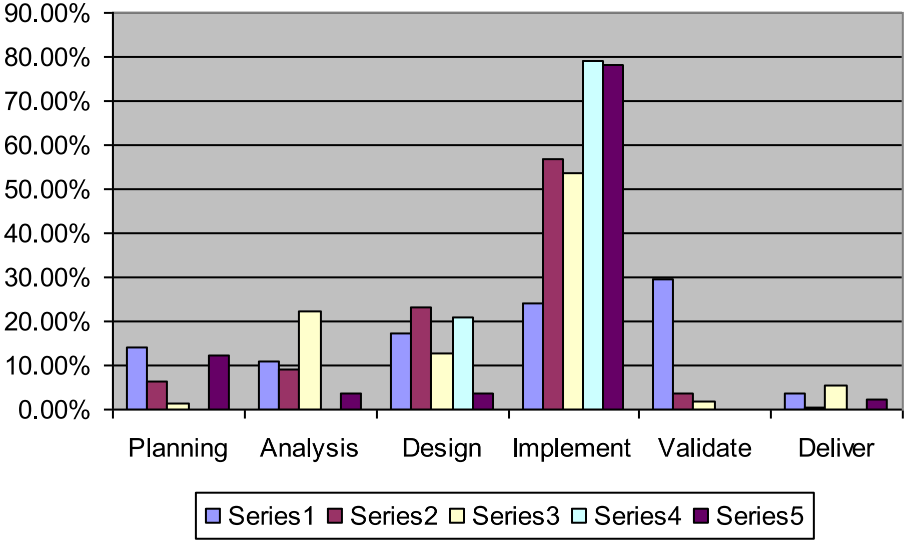
Back end: 50 – 60%

100% of Product Full Life Cost

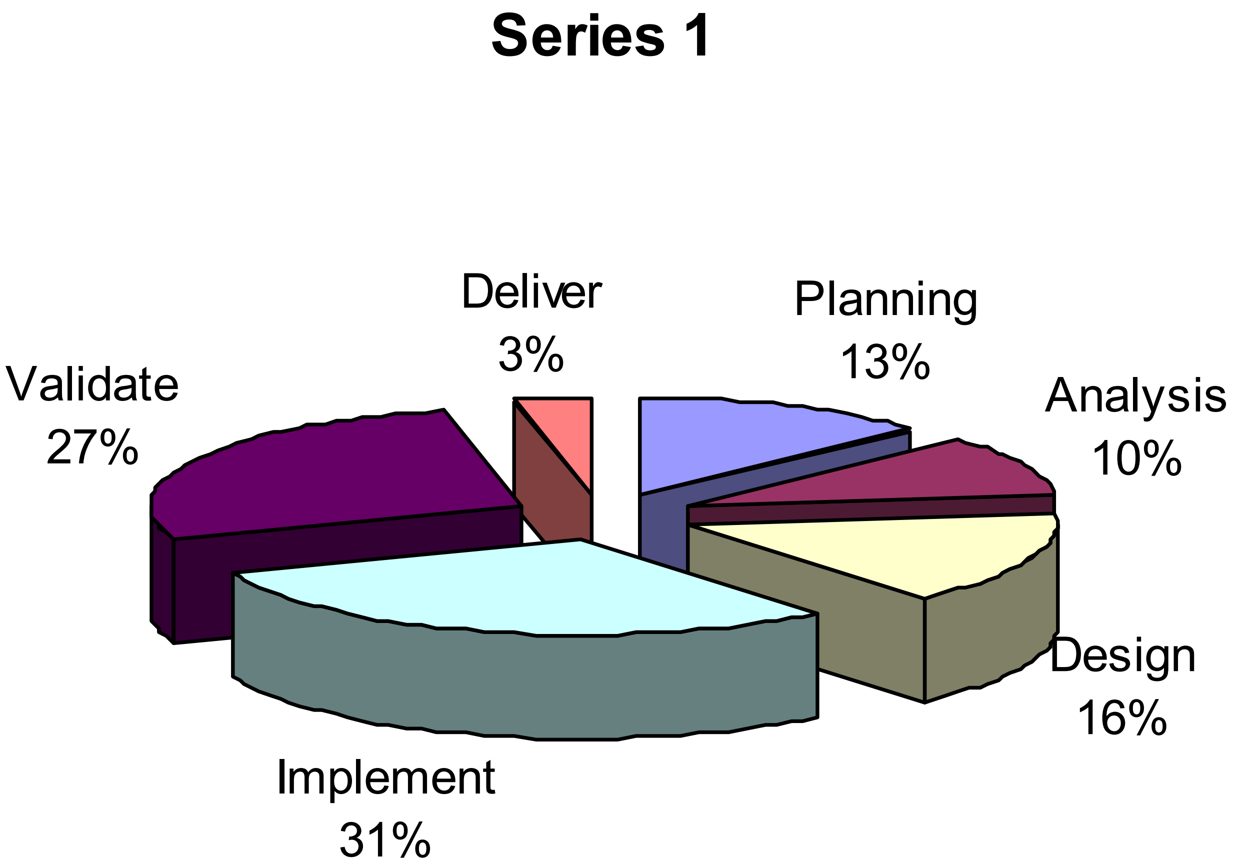
Real Web Project Metrics(1)



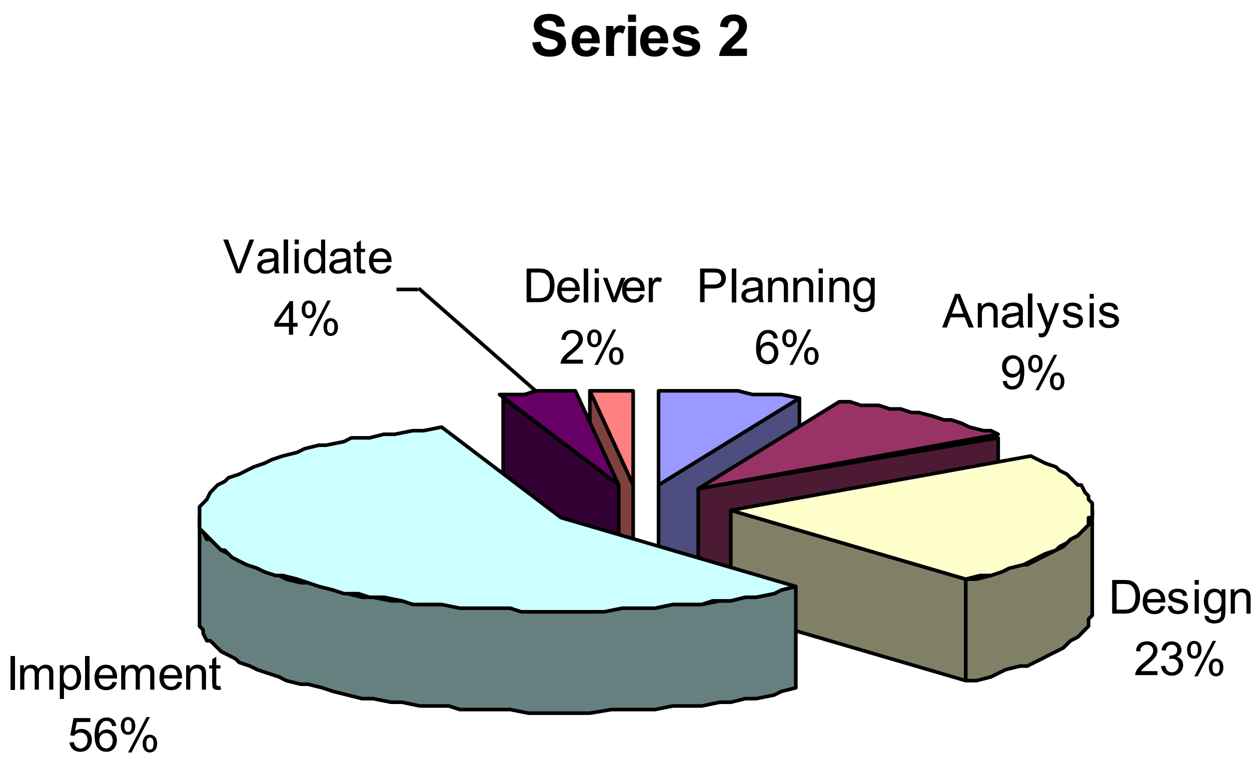
What is the message here?



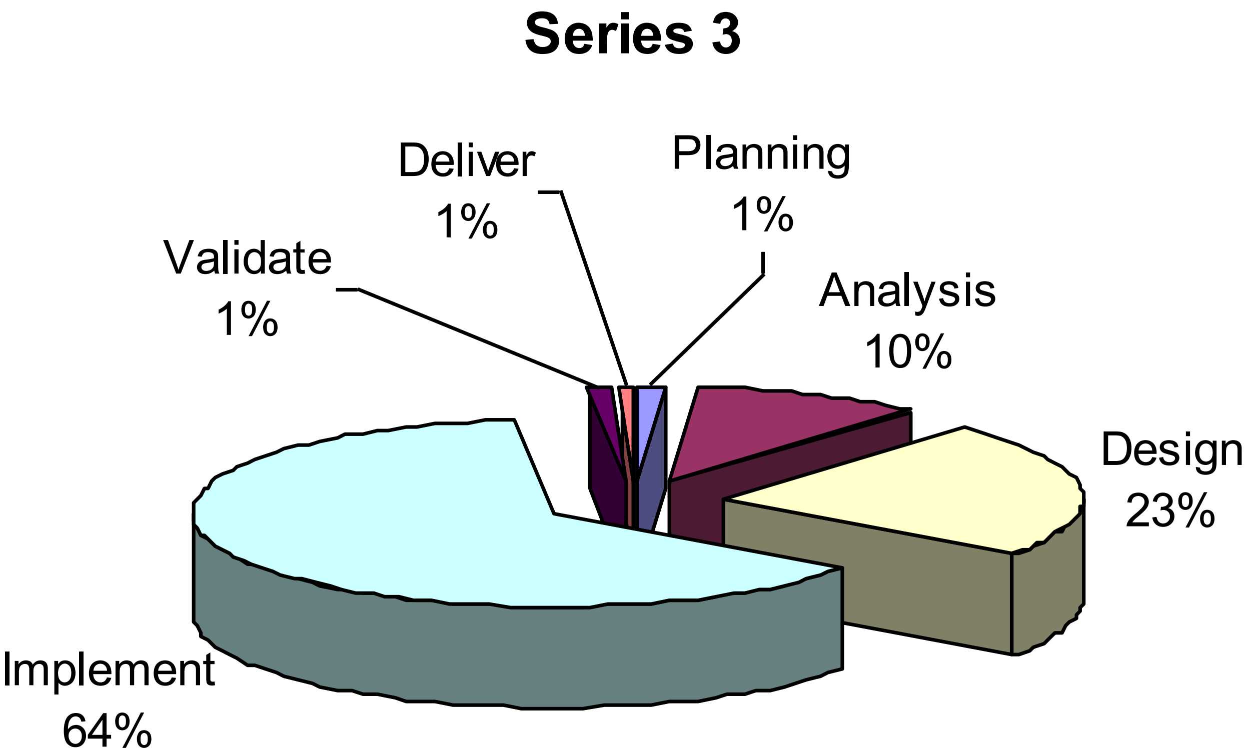
Real Web Project Metrics(2)



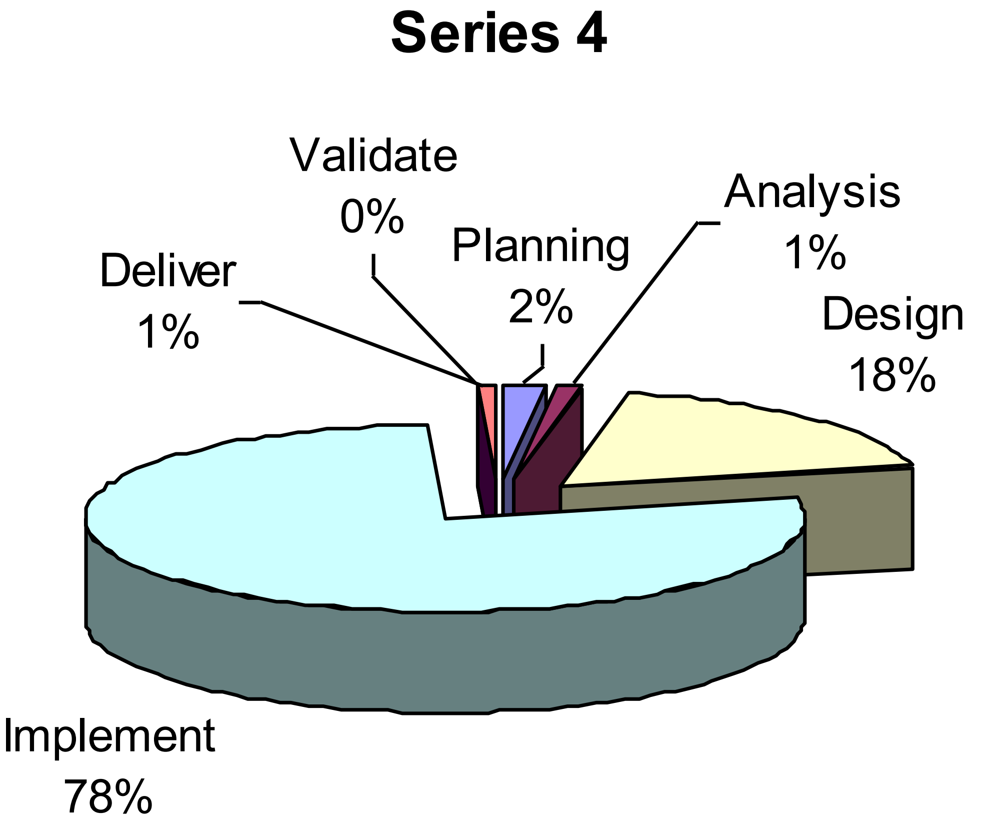
Real Web Project Metrics(3)



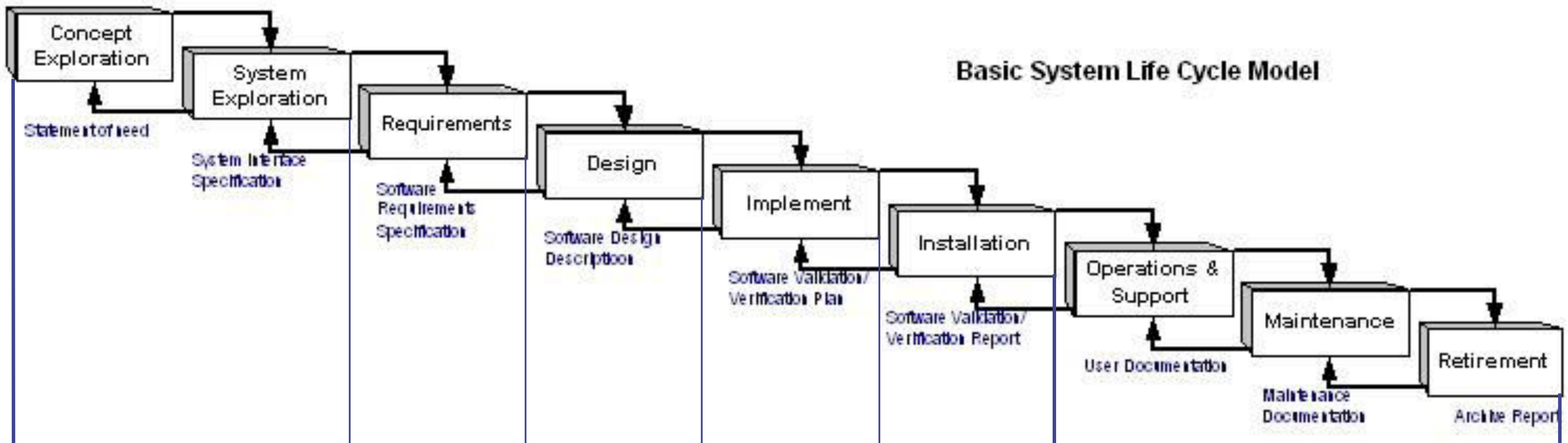
Real Web Project Metrics(4)



Real Web Project Metrics(5)



Web Effort per Phase (Preliminary Research)


Basic System Life Cycle Model
Concept Exploration
System Exploration
Requirements
Design
Implement
Installation
Operations & Support
Maintenance
Retirement
Statement of need
System Interface Specification
Software Requirements Specification
Software Design Description
Software Validation/ Verification Plan
Software Validation/ Verification Report
User Documentation
Maintenance Documentation
Archive Report

7%
△-3%

10%
△-20%

16%
△-14%

65%
△+45%

2%
△-8%

Front end: 40 – 50%

Back end: 50 – 60%

100% of Product Full Life Cost

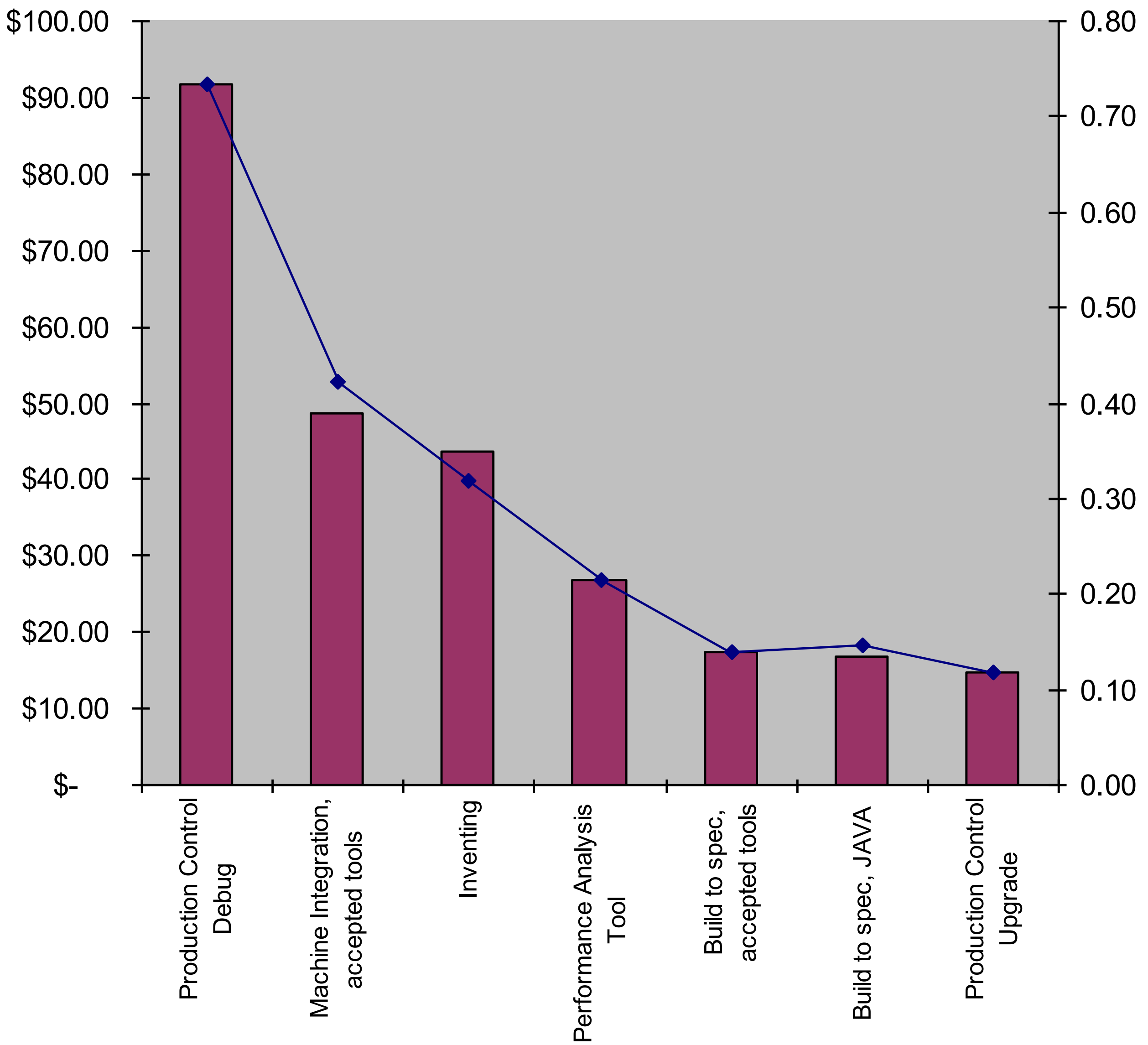
Best Practices that Work

1. Define your life cycle
2. Set up a metrics system
3. Formalize project management
4. Develop a prototyping process
5. Institute reviews and inspections
6. Implement non-invasive configuration management
7. JAD with your customers

Defining Your Life Cycle Model

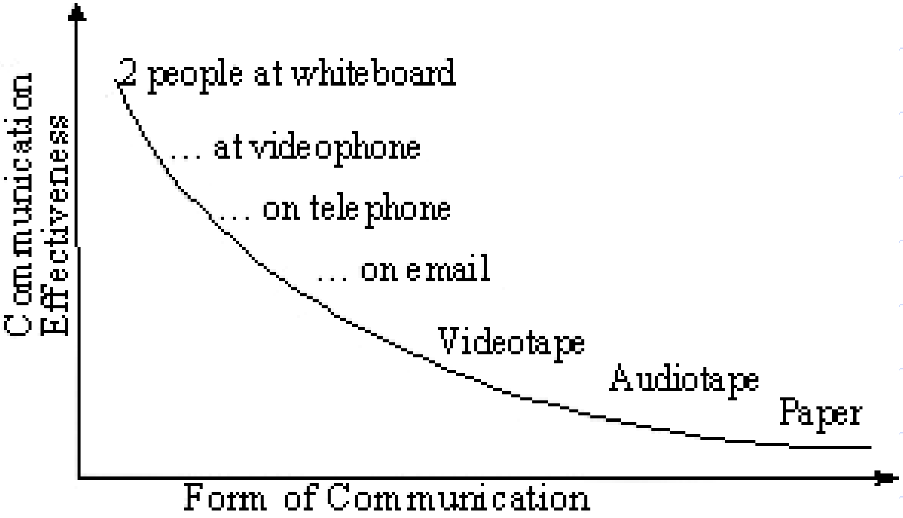
1) Become familiar with the various models
2) Review, analyze the type of work: development, enhancement, maintenance, etc.
3) Review project criteria
4) Identify a minimum set of phases
5) Identify phase activities
6) Establish a minimum set of deliverables
7) Define templates and content guides for deliverables
8) Evaluate progress and effectiveness of the life cycle framework
9) Implement improvements

Why a metrics system?



Best Practices that Work

8. Evolve to an object-oriented model
9. Embrace modeling with UML
10. Build early and often
11. Build anywhere
12. Communicate, communicate, communicate



Key Life Cycle Message

Whatever life cycle you start with will not be the one that will really work for you. You have to take charge of your life cycle, monitor it and adapt it to your circumstances. In the end it must become yours!

Are you secure with your process???

Linda Shafer Bio:

Linda Shafer has been working with the software industry since 1965, beginning with NASA in the early days of the space program. Her experience includes roles of programmer, designer, analyst, project leader, manager, and SQA/SQE. She has worked for large and small companies, including IBM, Control Data Corporation, Los Alamos National Laboratory, Computer Task Group, Sterling Information Group, and Motorola. She has also taught for and/or been in IT shops at The University of Houston, The University of Texas at Austin, The College of William and Mary, The Office of the Attorney General (Texas) and Motorola University. Ms. Shafer's publications include 25 refereed articles, and three books. She currently works for the Software Quality Institute and co-authored a SQI Software Engineering Series book published by PrenHall in 2002: Quality Software Project Management. She is on the International Press Committee of the IEEE and an author in the Software Engineering Series books for IEEE. Her MBA is from the University of New Mexico.

Don Shafer Bio:

Don Shafer is a co-founder, corporate director and Chief Technology Officer of Athens Group, Inc. Incorporated in June 1998, Athens Group is an employee-owned consulting firm, integrating technology strategy and software solutions. Prior to Athens Group, Shafer led groups developing and marketing hardware and software products for Motorola, AMD and Crystal Semiconductor. He was responsible for managing a $129 million-a-year PC product group that produced the award-winning audio components. From the development of low-level software drivers in yet-to-be-released Microsoft operating systems to the selection and monitoring of Taiwan semiconductor fabrication facilities, Shafer has led key product and process efforts. In the past three years he has led Athens engineers in developing industry standard semiconductor fab equipment software interfaces, definition of 300mm equipment integration tools, advanced process control state machine data collectors and embedded system control software agents. His latest patents are on joint work done with Agilent Technologies in state-based machine control. He earned a BS degree from the USAF Academy and an MBA from the University of Denver. Shafer's work experience includes positions held at Boeing and Los Alamos National Laboratories. He is currently an adjunct professor in graduate software engineering at Southwest Texas His faculty web site is http://www.cs.swt.edu/~donshafer/. With two other colleagues in 2002, he wrote Quality Software Project Management for Prentice-Hall now used in both industry and academia. Currently he is working on an SCM book for the IEEE Software Engineering Series.