Introduction to Software Project Management

CITS3220
Software Requirements & Project Management

- "A project gets a year late one day at a time."
- "Anything that can be changed will be changed until there is no time left to change anything."
- "Furious activity does not necessarily equate to progress and is no substitute for understanding."
- "People under pressure do not think faster."
- "The nice thing about not planning is that failure comes as a complete surprise rather than being preceded by a period of worry and depression."
- "Any project can be estimated accurately (once it's completed)."

http://en.wikiquote.org/wiki/Project_management

Lecture Overview

- Issues
  - how should we manage the *process* which produces a software *product*?
  - does improving the process improve the product?
- Definitions
  - process, phases and tasks, activities
- Techniques
  - a selection of *software life cycle models*
- Recommended Reading
  http://en.wikipedia.org/wiki/Software_development_process

Project Management

- is the discipline of defining and achieving targets while optimizing the use of resources (time, money, people, materials, energy, space, etc) over the course of a project (a set of activities of finite duration).

Project Management Activities

1. Planning the work
2. Estimating resources
3. Organizing the work
4. Acquiring human and material resources
5. Assigning tasks
6. Directing activities
7. Controlling project execution
8. Reporting progress
9. Analysing the results based on the facts achieved

The 5 Variables of Project Control

1. Time
2. Cost
3. Quality
4. Scope
5. Risk

Project control variables (1)

- **time** - amount of time required to complete the project.
- Typically broken down for analytical purposes into the time required to complete the components of the project, which is then further broken down into the time required to complete each task contributing to the completion of each component.

- **cost** - calculated from the time variable.
- Cost to develop an internal project is time multiplied by the cost of the team members involved. When hiring an independent consultant for a project, cost will typically be determined by the consultant or firm's hourly rate multiplied by an estimated time to complete.

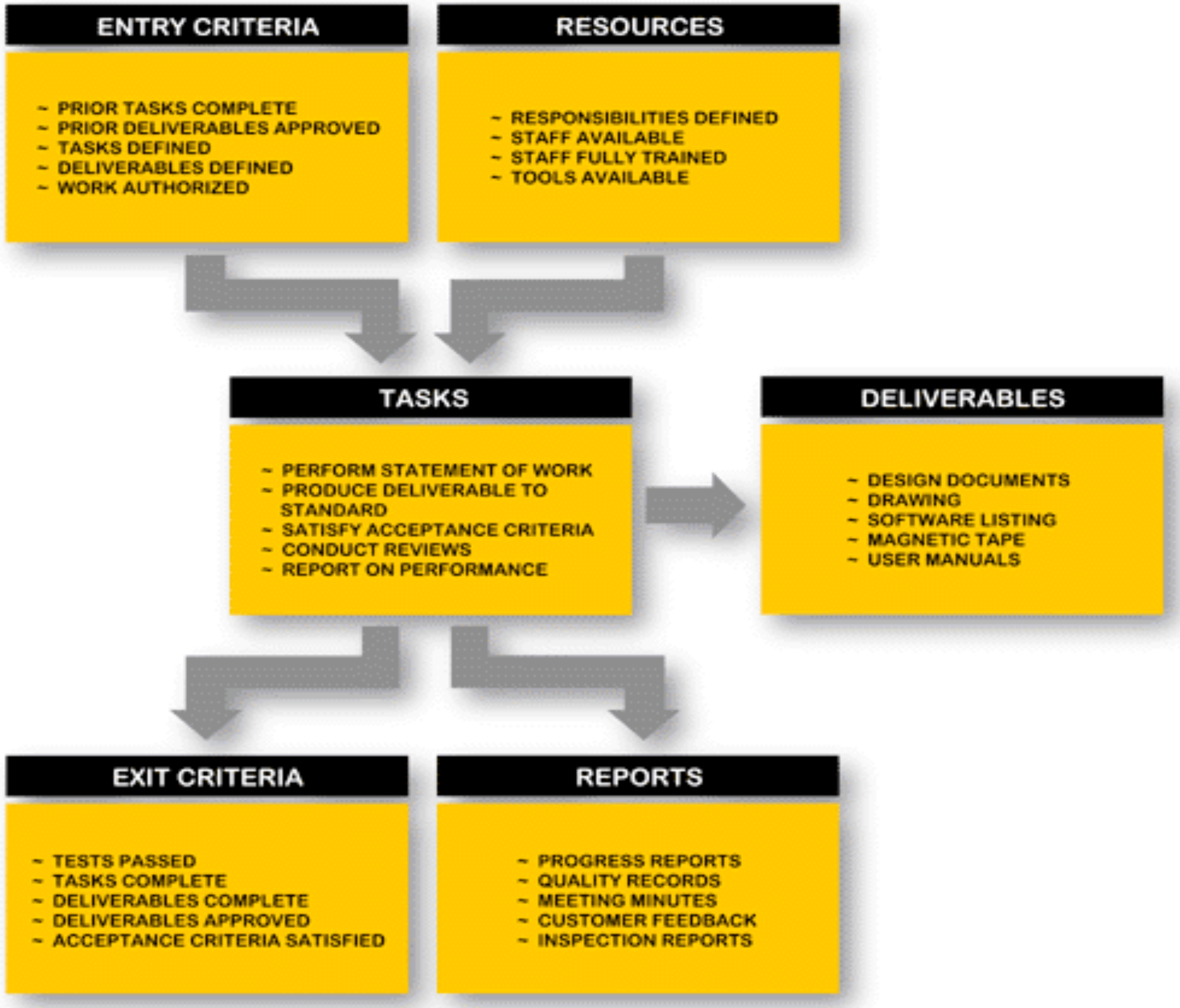
Project control variables (2)

- **quality** - The amount of time put into individual tasks determines the overall quality of the project. Some tasks may require a given amount of time to complete adequately, but given more time could be completed exceptionally. Over the course of a large project, quality can have a significant impact on time and cost (or vice versa).
- **scope** - Requirements specified for the end result. The overall definition of what the project is supposed to accomplish, and a specific description of what the end result should be or accomplish.
- **risk** - Potential points of failure. Most risks or potential failures can be overcome or resolved, given enough time and resources.

Phases or Tasks

- A project can be broken down into *phases*, or *tasks* to be done.
- Each *phase* is defined by its
  - entry criteria,
  - exit criteria,
  - resources,
  - deliverables and
  - reports.

Phase Definition



8 Phases of the SW Lifecycle

[Boehm Software Engineering Economics]

- Plans & Requirements
- Product Design
- Detailed Design
- Code and Unit Test
- Integration and Test
- Implementation
- Operations and Maintenance
- PhaseOut

Activities

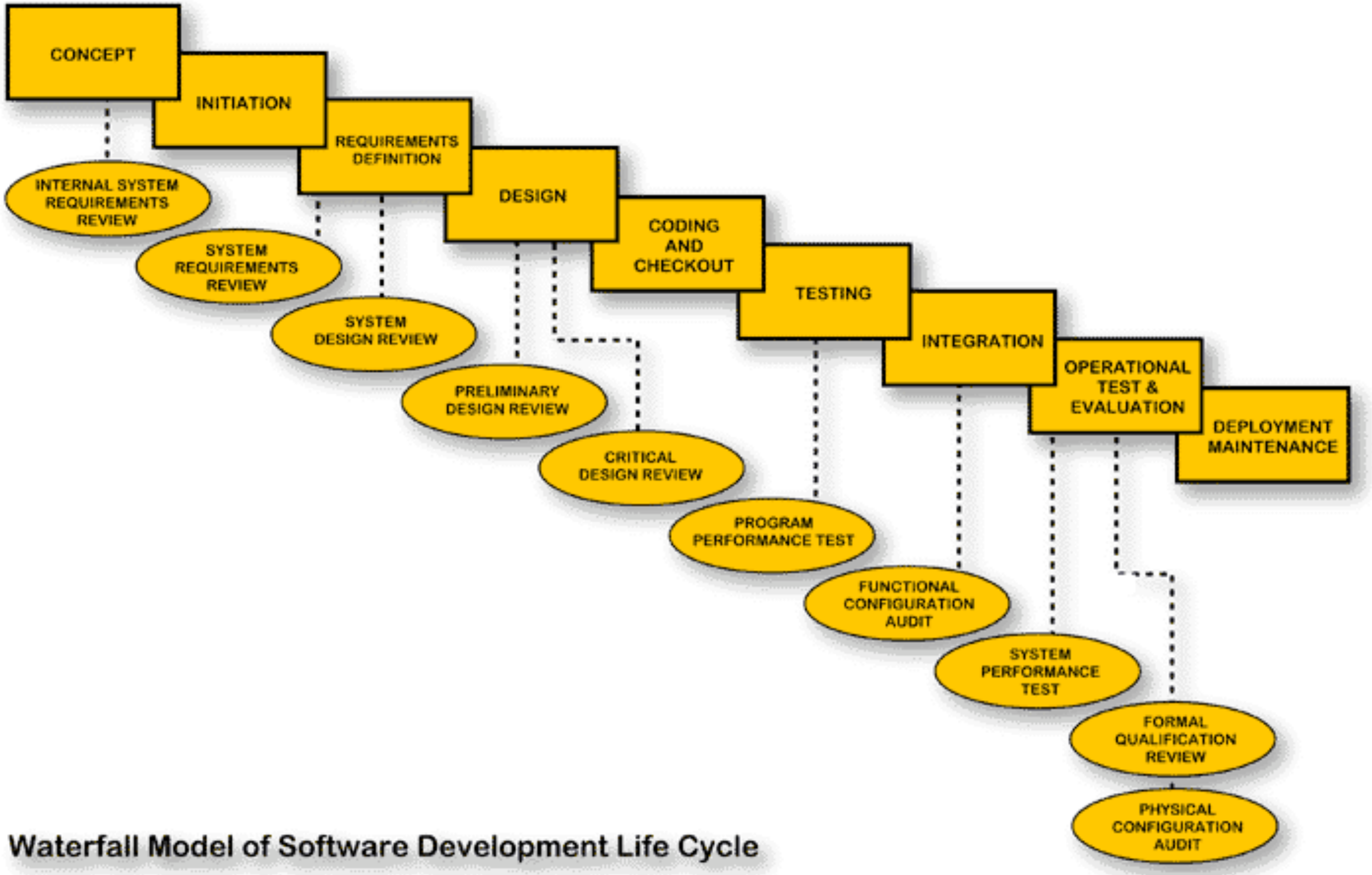
- Requirements Analysis
- Product Design
- Programming
- Test Planning
- Verification & Validation
- Project Office functions
- Configuration management and quality assurance
- Manuals

What’s the difference between activities and phases ?

- ***Activity:*** a set of tasks performed towards a specific purpose [B&D,p707]
- ***Phase:*** A set of tasks performed over time; defined by its start and end points
- The same *activity* may occur in different *phases* of a software life cycle

SW Lifecycle Models
Some Examples

- Waterfall model
- Boehm’s Risk Spiral model
- Prototyping Paradigm
- Evolutionary (Incremental) Development
- Extreme Programming
- Agile Programming
- Synchronize-and-stabilize

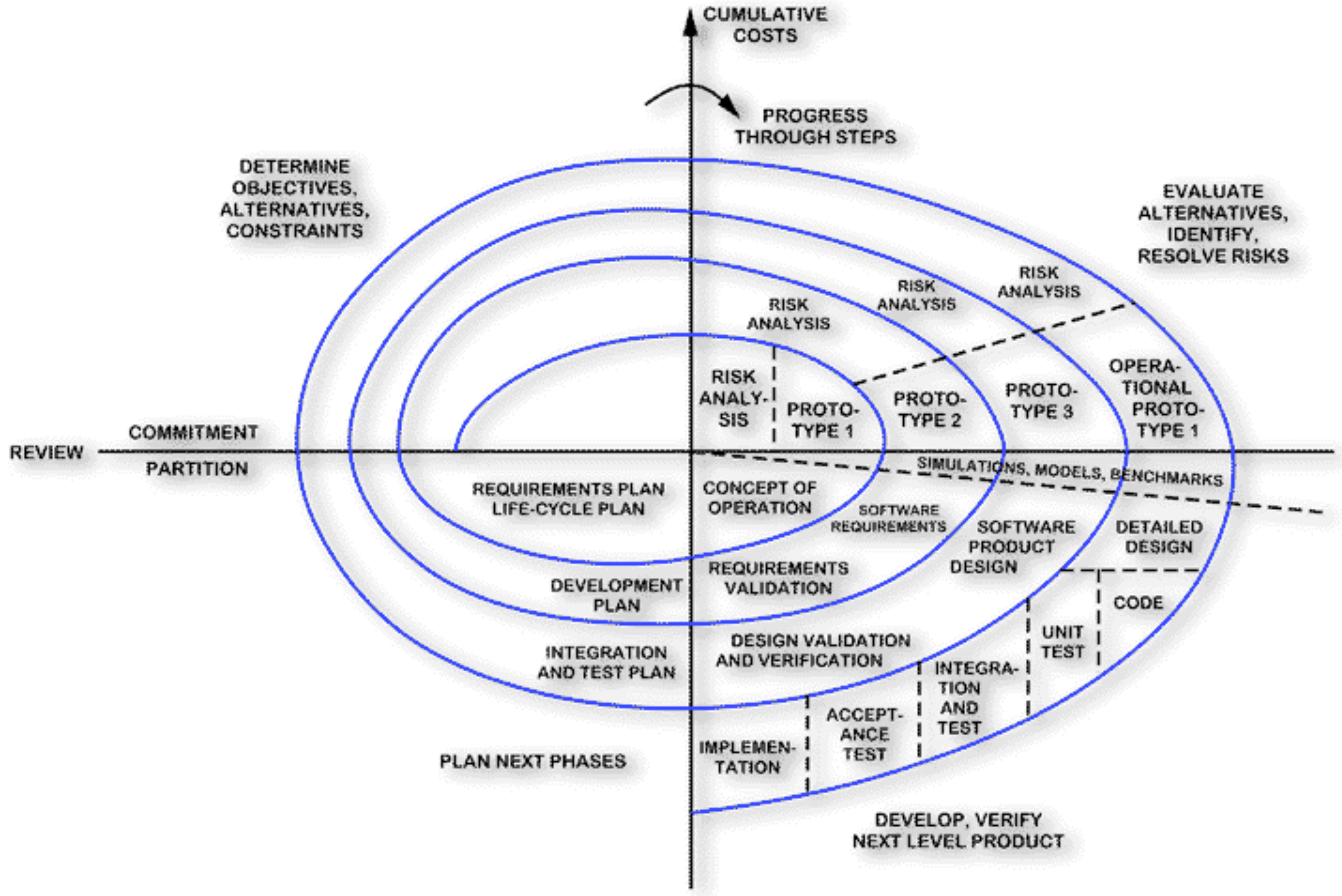

CONCEPT
INITIATION
REQUIREMENTS DEFINITION
DESIGN
CODING AND CHECKOUT
TESTING
INTEGRATION
OPERATIONAL TEST & EVALUATION
DEPLOYMENT MAINTENANCE
INTERNAL SYSTEM REQUIREMENTS REVIEW
SYSTEM REQUIREMENTS REVIEW
SYSTEM DESIGN REVIEW
PRELIMINARY DESIGN REVIEW
CRITICAL DESIGN REVIEW
PROGRAM PERFORMANCE TEST
FUNCTIONAL CONFIGURATION AUDIT
SYSTEM PERFORMANCE TEST
FORMAL QUALIFICATION REVIEW
PHYSICAL CONFIGURATION AUDIT

Waterfall Model of Software Development Life Cycle

Source: US Department of Defence

Waterfall Model Issues

- To be effective there must be a *review* step at each phase (shown as ovals) to check for completion.
- In theory we should not proceed down the waterfall until the phase is "complete".
- Problem: real projects don’t work this way – “change is the only constant”



Spiral model of the software process

Source: Barry Boehm, IEEE paper

Boehm’s Spiral Model

- Essence: process contains several cycles, starting from the centre; at each iteration the tasks involve more detailed knowledge and design tasks.
- Focus on addressing risks incrementally, in order of priority
- Distance from origin represents **cost** accumulated by the project
- Angle from the horizontal represents the **type of activity** (e.g. risk, risk analysis, testing)

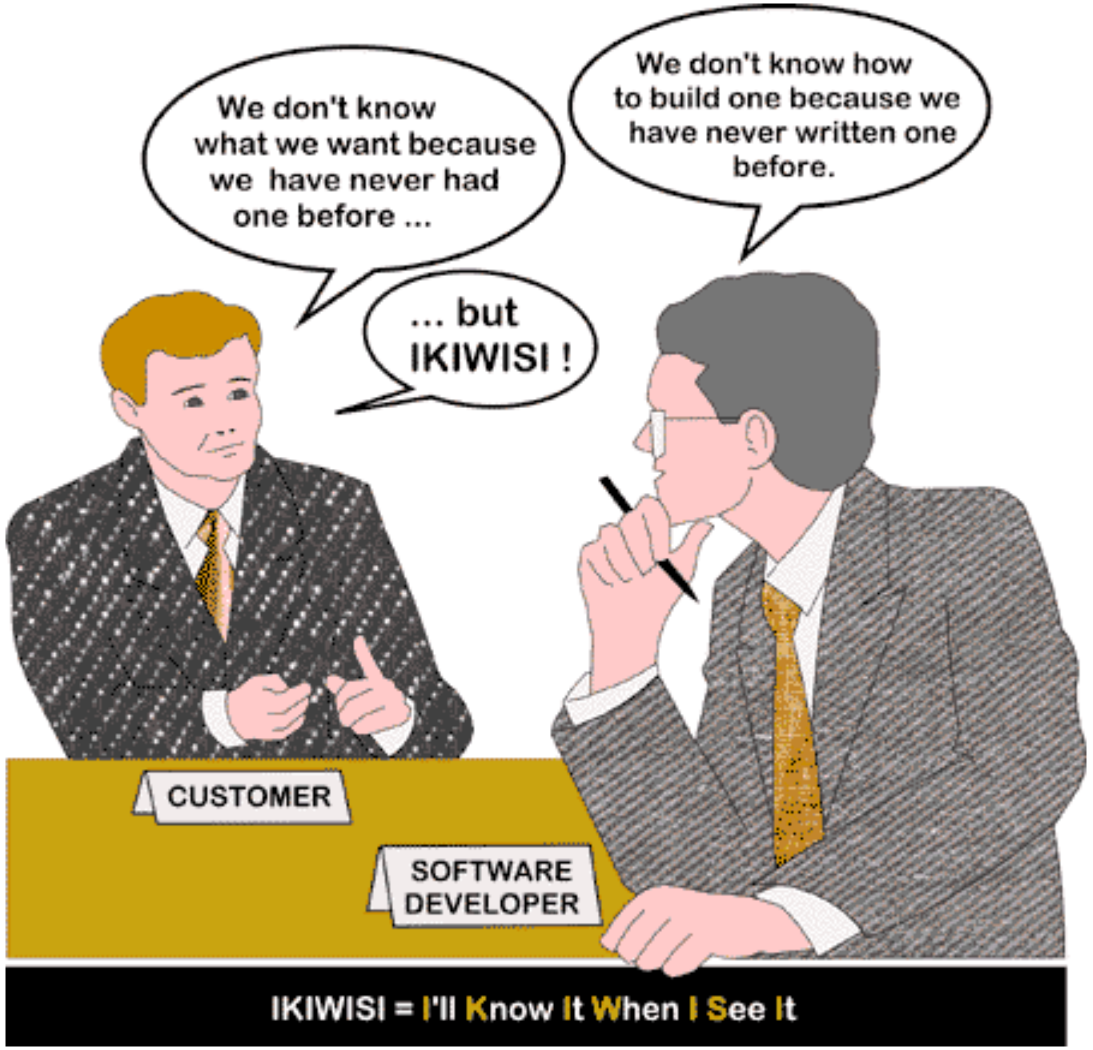
Spiral Model: Key Assumptions

- Assume sufficient *flexibility* to adapt and change after each analysis spiral
- Sensitive to having a thorough *risk assessment* at each cycle.
- Requires *confidence* in the project managers to carry it off, and to maintain the confidence of both the development team and the client!

A Requirements Capture Dilemma

- Where does these leave both the customer and the developer? They both think they know what is required but either do not know how to do it or cannot be sure of what is really required.

- **This might be described as the software engineer's dilemma!**


We don't know what we want because we have never had one before ...
... but IKIWISI !
We don't know how to build one because we have never written one before.
CUSTOMER
SOFTWARE DEVELOPER
IKIWISI = I'll Know It When I See It

Prototyping (a solution to IKIWISI)

- Developing some high level strategies, and some simple frameworks for testing out the concepts that the customer wants and the developer can implement.
- From here both are able to respond with suggestions for improvements and changes. The cycle may then be able to be repeated, each time progressing closer to the required solution.
- **This iterative process is called prototyping.**

Problems with prototyping

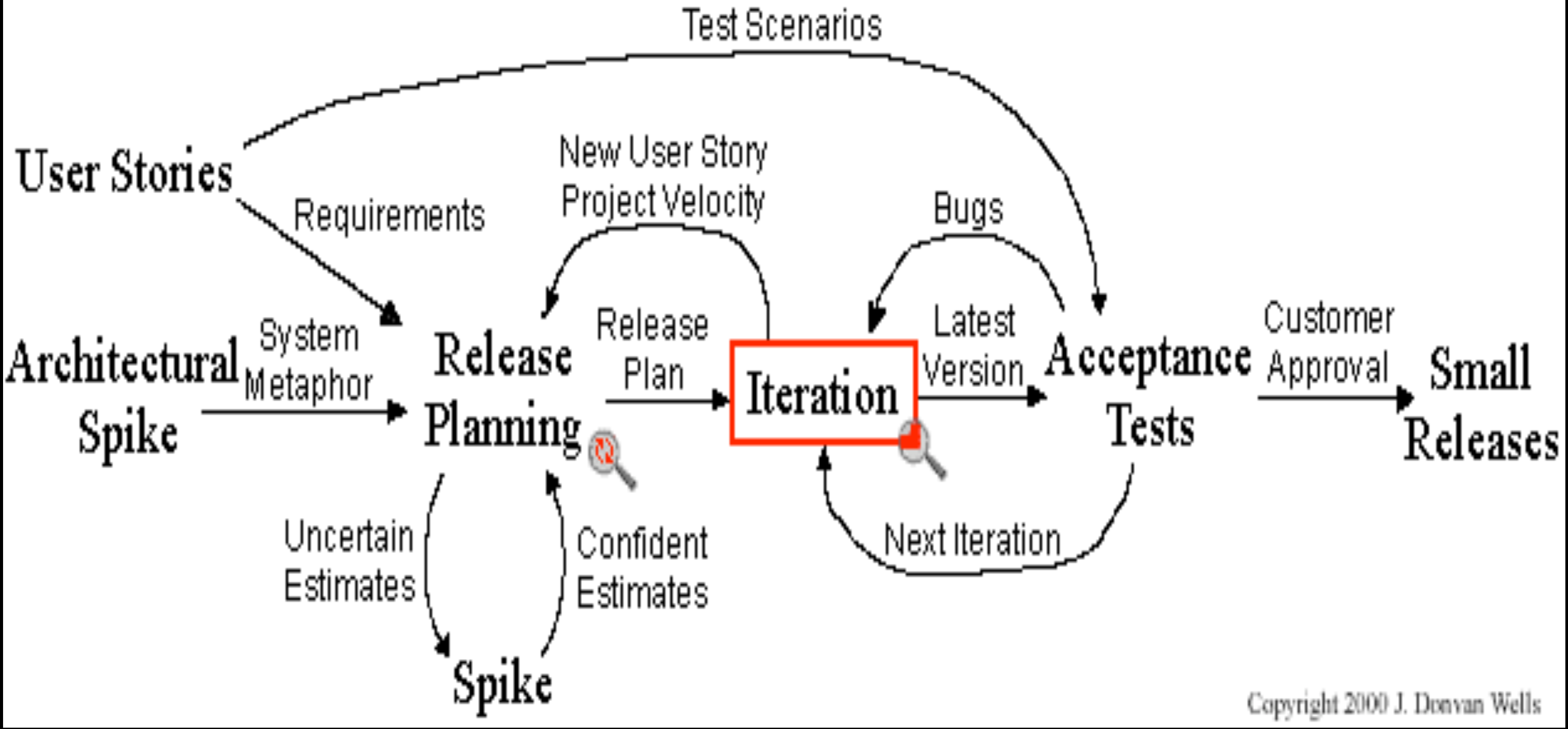
- **GIVES A 'MIND-SET'** All following versions tend to carry on the assumptions and faults of the original.
- **PROTOTYPES ARE RARELY THROWN AWAY** Quick and dirty jobs become the final long-term system.
- **USER DOMINATION vs INVOLVEMENT** 'Suck it and see' is the opposite of good change control.
- **QUICK, QUICK, SLOW CRITICISM** If the prototype can be developed in days, then why does the real system take many months?

Evolutionary Paradigm (Gilb)

"Grow, don't build software." - Fred Brooks

- Step 1: Prototype
  - produce a useable but small part of the system
  - solidifies user requirements
  - sketch of system design
- Step 2: Expansion
  - add functionality
  - determine "hot-spots"
- Step 3: Consolidation
  - correct design defects
  - introduce new abstractions

Extreme Programming Project



Extreme Programming

- Extreme Programming (XP) stresses customer satisfaction and emphasizes team work.
- XP principles: communication, simplicity, feedback, and courage.
- Feedback = testing SW from day one; deliver to customers as early as possible; implement changes as suggested.
- With this foundation XP programmers are able to respond courageously to changing requirements and technology.

Pair Programming (used in XP)

- All code to be included in a production release is created by two people working together at a single computer.
- Pair programming increases software quality without impacting time to deliver. It is counter intuitive, but 2 people working at a single computer will add as much functionality as two working separately except that it will be much higher in quality.
- With increased quality comes big savings later in the project.

How to Pair Program

- The best way to pair program is to just sit side by side in front of the monitor. Slide the key board and mouse back and forth. One person types and thinks tactically about the method being created, while the other thinks strategically about how that method fits into the class. It takes time to get used to pair programming so don't worry if it feels awkward at first.

Agile Programming [Cockburn 2002]

“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.”

Source: http://www.agilealliance.org/home

Underlying Assumptions for Agile SW Development

- “Different projects need different processes or methodologies”
- “Focussing on skills, communication and community allows the project to be more effective and more agile than focussing on process”
  - Reference: *Agile Software Development*, by Alistair Cockburn, Addison Wesley, 2002

Synchronize-and-stabilize

- teams work in parallel on individual application modules
- teams frequently synchronize their code with that of other teams, and
- debug (stabilize) code regularly throughout the development process
- flexible since allows for changes at any point
- successfully used by Microsoft and Netscape

Process Maturity

- A software development process is **mature** if the development activities are well defined and if management has some control over the management of the project
- *Assumption:* With increasing maturity the risk of project failure decreases

- CMMI Capability Maturity Model Integration
- http://www.sei.cmu.edu/cmmi/cmmi.html

Capability maturity levels

1. Initial Level (also called ad hoc or chaotic)
2. Repeatable Level (Process depends on individual "champions")
3. Defined Level
   - Process is institutionalized (sanctioned by management)
4. Managed Level
   - Activities are measured and provide feedback for resource allocation (process itself does not change)
5. Optimizing Level
   - Process allows feedback of info. to change process itself

SPICE

- International Standard for Software Process Assessment (developing)
- http://www.sqi.gu.edu.au/spice/
- **Process assessment** examines the processes used by an organization with the purpose of determining whether they are effective in achieving their goals
- Used to assist with “planning, managing, monitoring, controlling and improving the acquisition, supply, development, operation, evolution and support of software.”

Issues in Project Management (to be covered in future lectures)

- Before starting the project:
  - planning, estimation, risk analysis;
- While doing the project:
  - monitoring, evaluation, metrics;
- After the project is done:
  - assessment and improvement