REQUIREMENTS - WHY WHAT AND HOW?

Based on presentations by Steve Chenoweth & Sriram Mohan

WHY REQUIREMENTS

"Why not just ask them what they want, they tell you, and you build it?"

Some facts

- 250 billion dollar industry
- 31% projects cancelled(money lost 81 billion dollars)
- 52.7% cost 189% of their original estimates

Biggest Reasons for Project Success and Failure

Project Failure

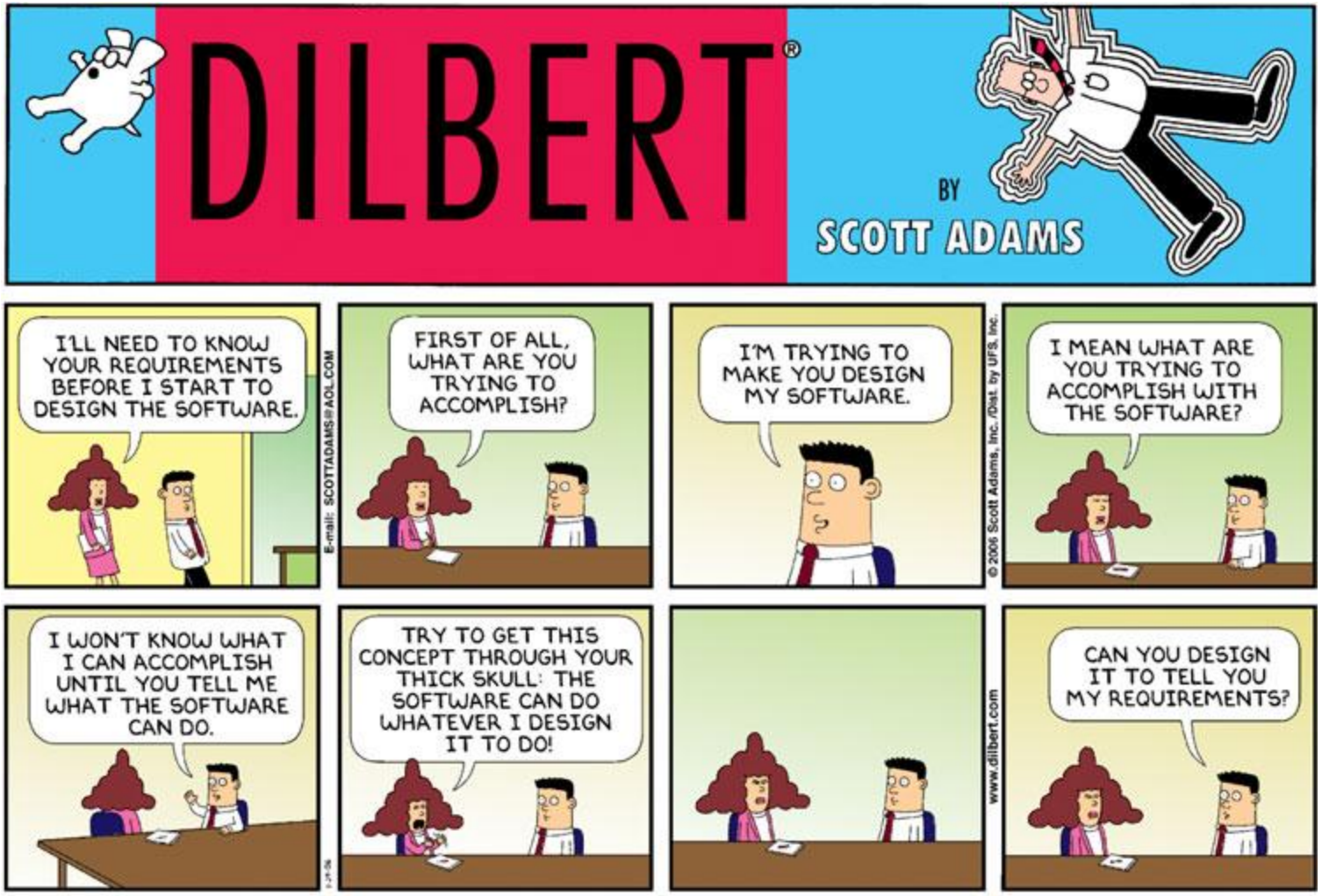
- Lack of user input (13%)
- Incomplete requirements (12%)
- Changing requirements (12%)

Note the heavy mention of requirements! !, there are other reasons such as poor planning, inadequate resources, but they all pale under the influence of requirements.

Project Success

- User involvement (16%)
- Executive management support (14%)
- Clear statement of requirements (12%)

Note the heavy mention of requirements, other surveys have shown a similar problem – effective requirements specification and managing changing requirements stand between success and failure

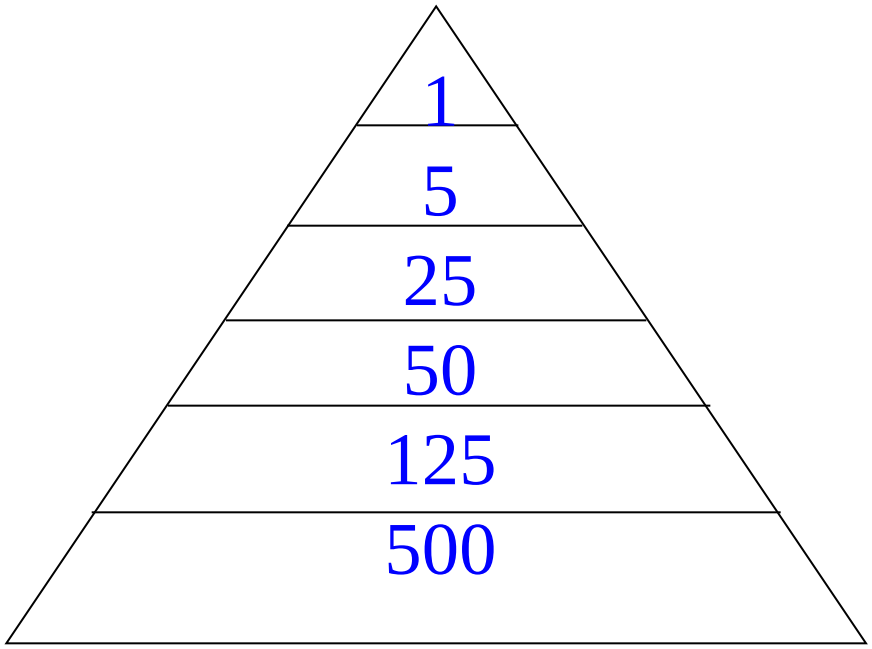


© Scott Adams, Inc./Dist. by UFS, Inc.

Defects by Development Phase

- Requirements – 31%
- Design – 25%
- Coding – 12%
- Documentation – 16%
- Bad fixes – 16%

Cost to Fix Errors



Requirements
Design
Coding
Unit Test
Acceptance Test
Maintenance

If requirement errors can be fixed quickly and economically, we may still be safe. But it is not the case, we will have to worry about leakage.

Why ?

- Requirements errors are likely to be the most common class of error
- Requirements errors are likely to be the most expensive to fix

WHAT IS A SOFTWARE REQUIREMENT?

"I'd like to see the web pages in green…"

What is a Software Requirement?

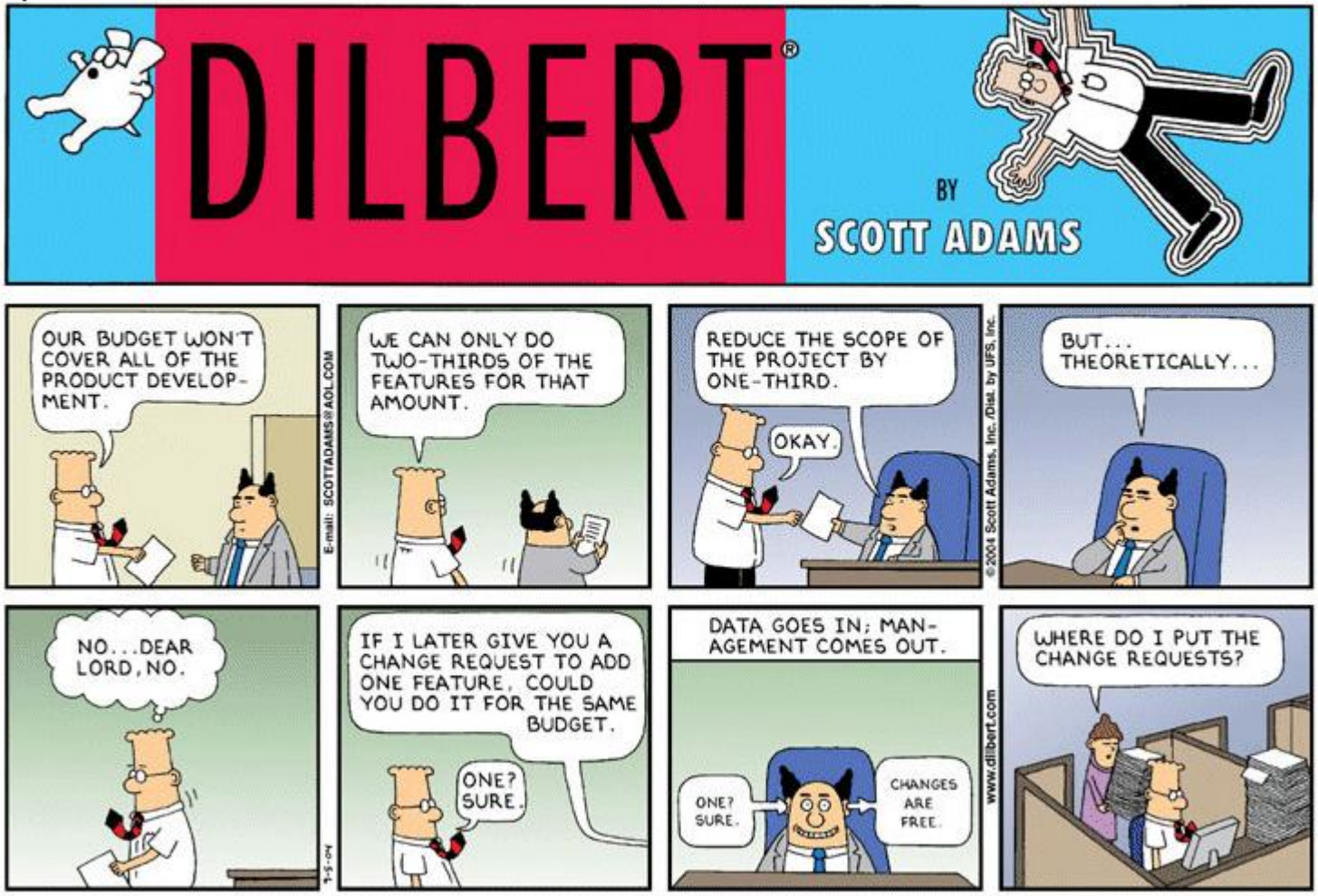
1. A software capability needed by the user to solve a problem to achieve an objective
2. A software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documentation

What is Software Requirements Management?

- The process of systematically, eliciting, organizing and documenting requirements for the software system, and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system.

Dilbert

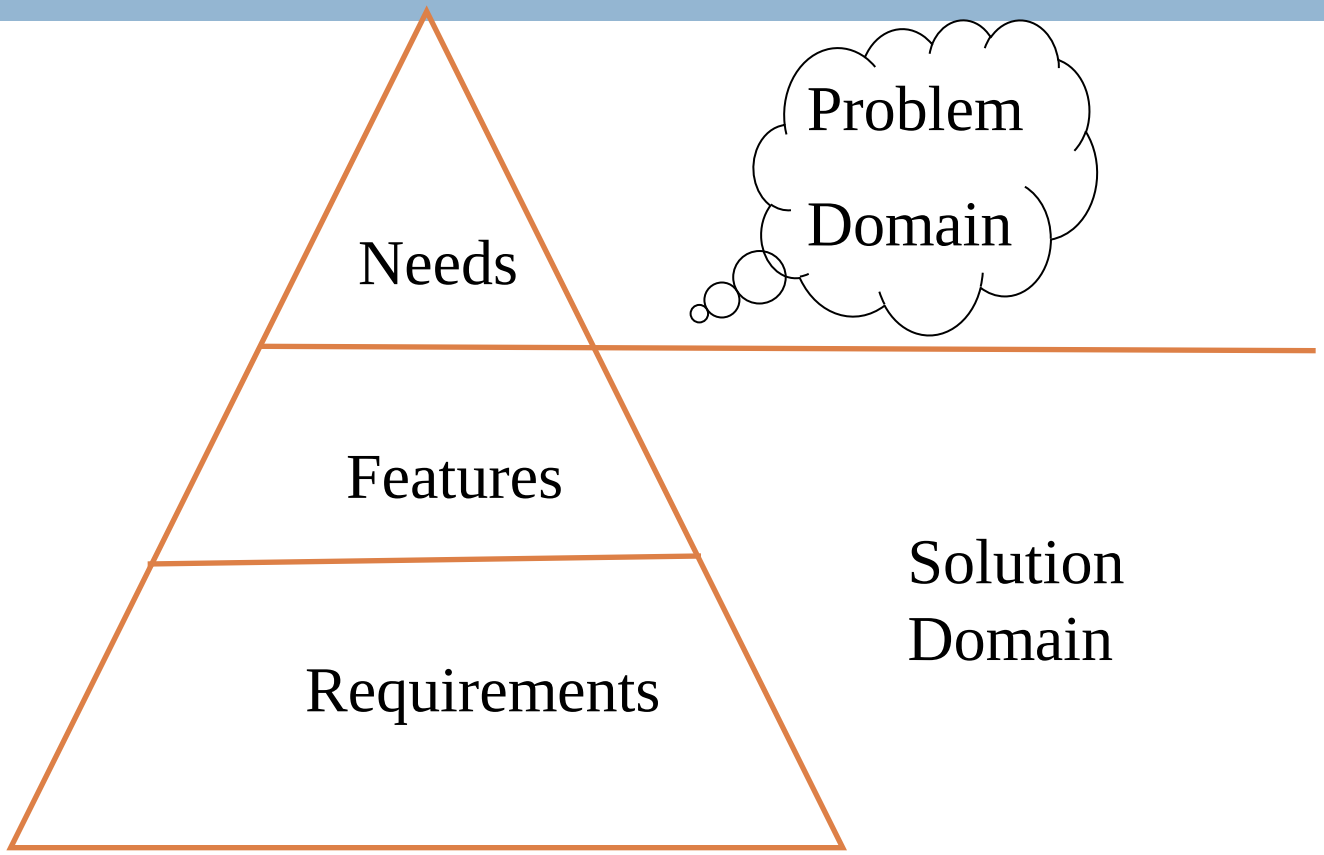
by Scott Adams



© UFS, Inc.

What does it involve?

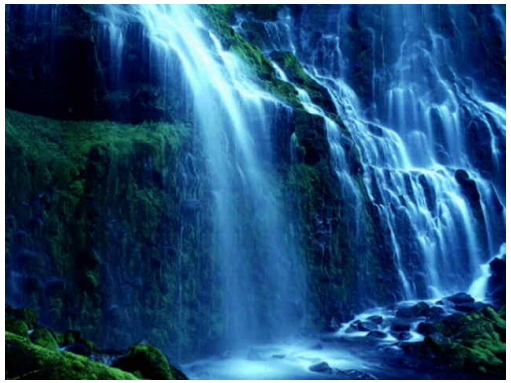
- Understand the need - the problem to be solved
- Propose features - how the need will be resolved
- Requirements are the specific conditions that are imposed on the solution



You can think of Features as a broad solution to a problem and requirements gets to the specifics of a solution.

SOFTWARE LIFECYCLE PROCESS MODELS

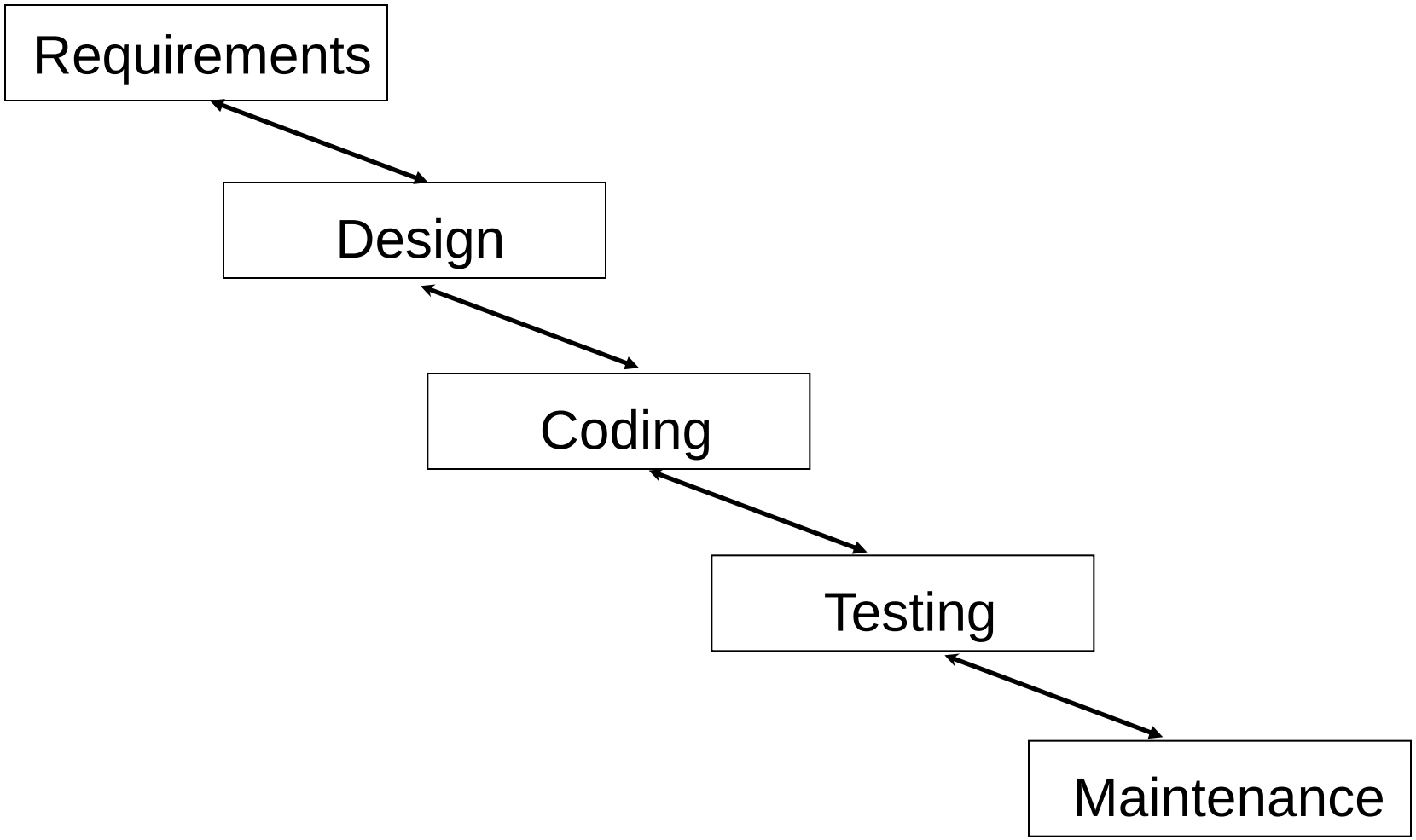
Waterfall Model



- Manufacturing life cycle model
- Assumes
  - multi-stage development cycle
  - completely separate stages
  - output of one stage is input for next
  - each stage complete before next is begun

Manufacturing uses this because they use physical material.

Waterfall Stages

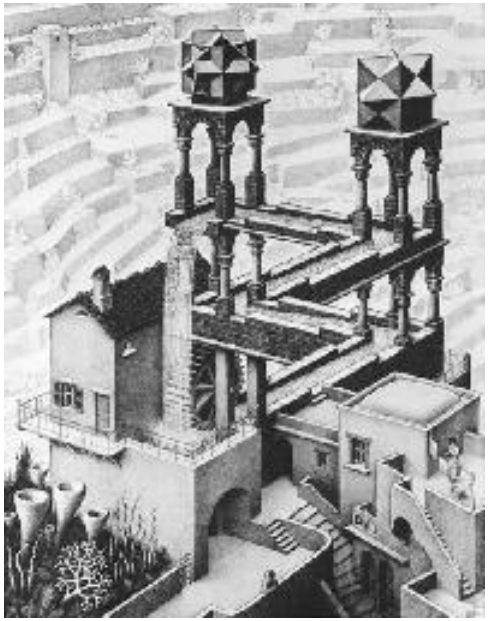


Advantages

- Much better than chaos!
- Clearly defined stages help with
  - planning, scheduling
  - management, organization
  - accountability, control

It has been real useful in reinforcing the role of requirements in the software development process. Requirements before design coding is a good mantra for this phase. Suffers from scope management issues, what happens if we start off with a big scope, what happens if we start off with a small scope.

Disadvantages



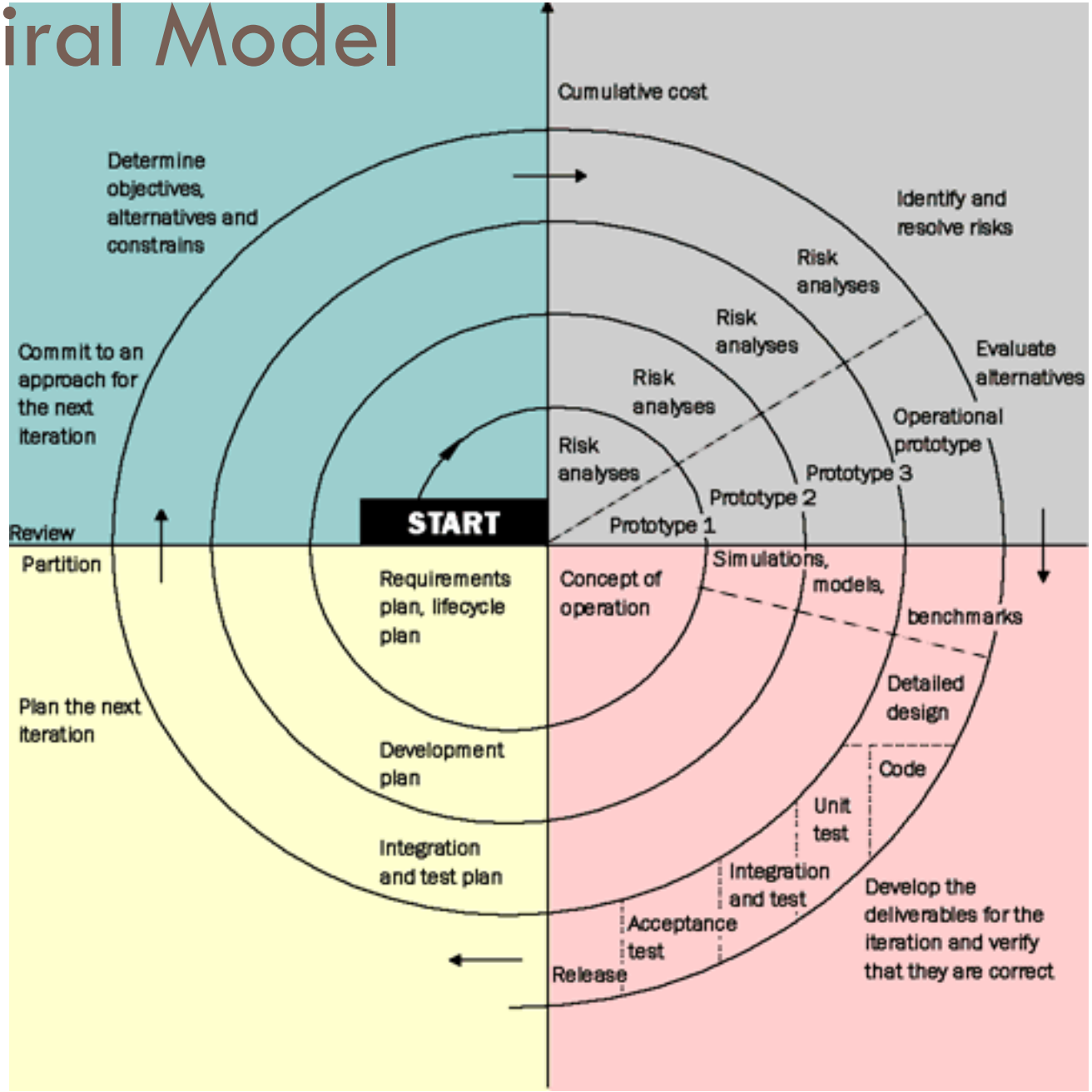
- Not very practical (do not know requirements in the beginning)
- Customer sees nothing until last step
- Change is Anathema

These disadvantages may be overcome with software because it is non-physical. Requirements are typically frozen early on and it is a problem as it may be difficult to define the requirements early on in the life cycle. Scope management is a big issue, what happens if you start with too large a scope, what happens if you start off with too small a scope.

Spiral Model

- Incremental
  - more feedback from customer
- Risk analysis
  - more rigorous analysis of risk at each stage

Spiral Model



Spiral Model

Each stage has the generation of a risk driven prototype; followed by a structural waterflow model. Projects can follow a cut and try approach, incremental coding might result in bad design and bad code.

Iterative (or Incremental) Approach

- Four phases
  - Inception
  - Elaboration
  - Construction
  - Transition

Iterative (or Incremental) Approach

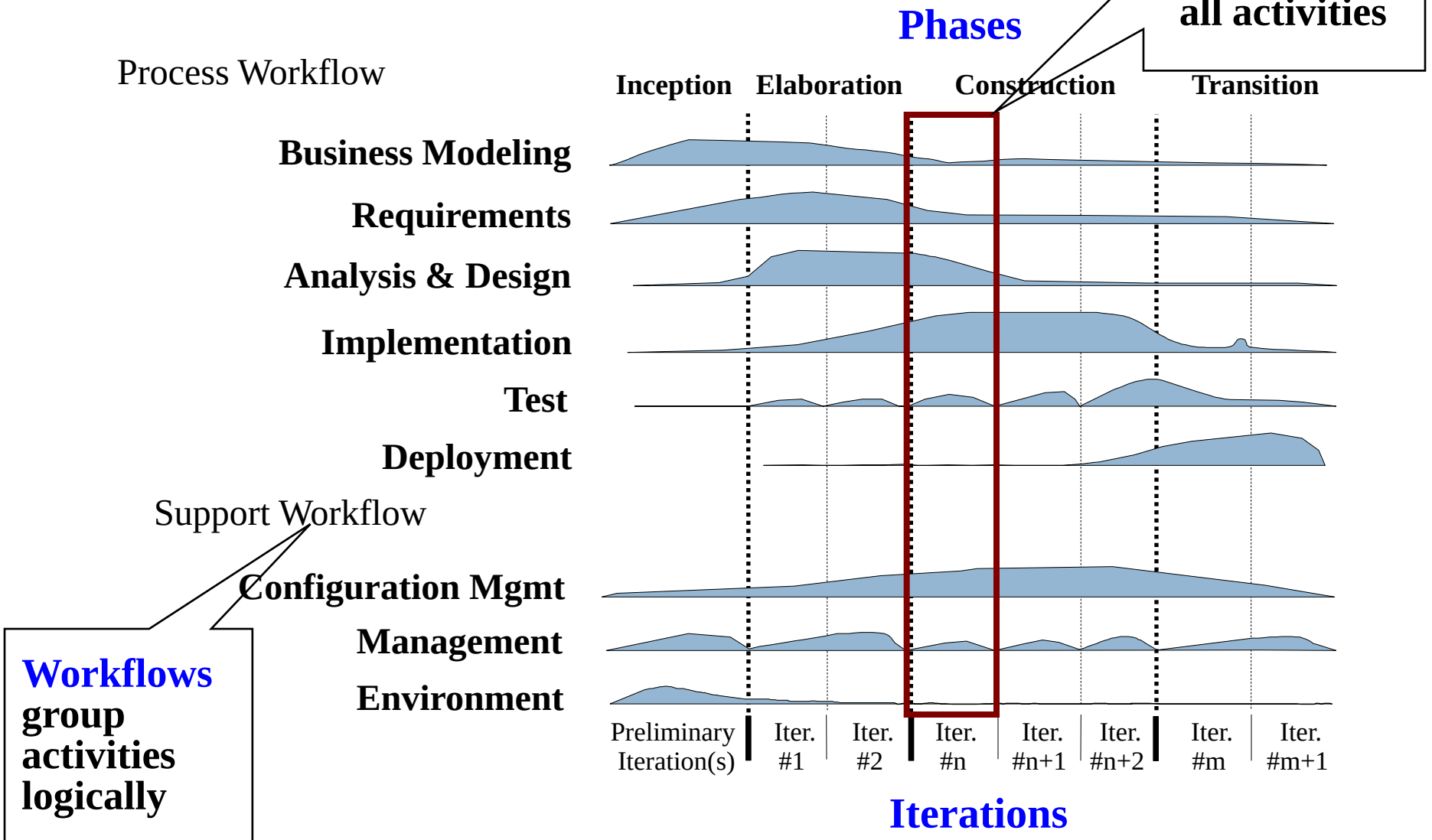
- Note that each iteration delivers a production system, although not with the complete functionality (except for the last iteration). This is unlike the spiral model, which uses prototypes. Life cycle is decoupled from the actual stages , thus each phase can visit the various things involved in software development such as requirements, design and coding.
- Inception phase – understand the project, what is the scope, the vision – involves mainly problem analysis and preliminary resource estiamtion
- Elaboration phase – refine the requirements, a potential architecture and an early feasibility prototype

Iterative(or Incremental) Approach

- An *iteration* is a sequence of activities with an established plan and evaluation criteria, resulting in an executable of some type
- Each phase goes under a # of iterations

Rational Unified Process Model

In an iteration, you walk through all activities



Workflows group activities logically

Rational Unified Process Model

- This graphic illustrates how phases and iterations, or the time dimension, relates to the development activities performed, or the workflow dimension. The relative size of the color area indicates how much of the activity is performed in each phase/iteration.
- Each iteration involves activities from all workflows. The relative amount of work related to the workflows changes between iterations. For instance, during late Construction, the main work is related to Implementation and Test and very little work on Requirements is done.
- Note that requirements are not necessarily complete by the end of Elaboration. It is acceptable to delay the analysis and design of well-understood portions of the system until Construction because they are low in risk.
- Works well with changing requirements and you can scope better too.

Software Development

- Is this a team activity? Why?
  - Computer programming is a human activity and it is a lot like football. A diverse skill set with effective collaboration and communication is necessary to succeed.

Requirements Team Skills

1. Analyzing the Problem
2. Understanding User and Stakeholder Needs
3. Defining the System
4. Managing Scope
5. Refining the System Definition
6. Building the Right System