Analysis vs. Design: What's the Difference?

- Caution
- Requirements and Non-requirements
- Decision Process, Version 1
- Tinkertoy Tic-Tac-Toe
- Kinds of Requirements
- Decision Process, Version 2?
- Separating Requirements by Kind
- Decision Process, Version 2
- Implications of Perfect Technology
- Why Bother?
- Target Software Architecture
- Definitions for Development / Maintenance Activities
- UML for Analysis
- UML for Design

Caution

• There are no industry-wide accepted definitions of the terms “requirement”, “analysis”, or “design”

- The definitions of these terms are often the subject of almost religious debate

• Consider this to be a proposal

- Be aware that this proposal has shown benefit on a number of software projects over many years
- But be aware that there are people who may violently disagree with this proposal

Requirements and Non-requirements

• The requirements should form a contract between the developers / maintainers and the customers

• Requirements need to be
- Unambiguous
    * Interpretable in only one way
- Testable
    * Compliance (or, non-compliance) can be clearly demonstrated
- Binding
    * The customer is willing to pay for it and is unwilling to not have it

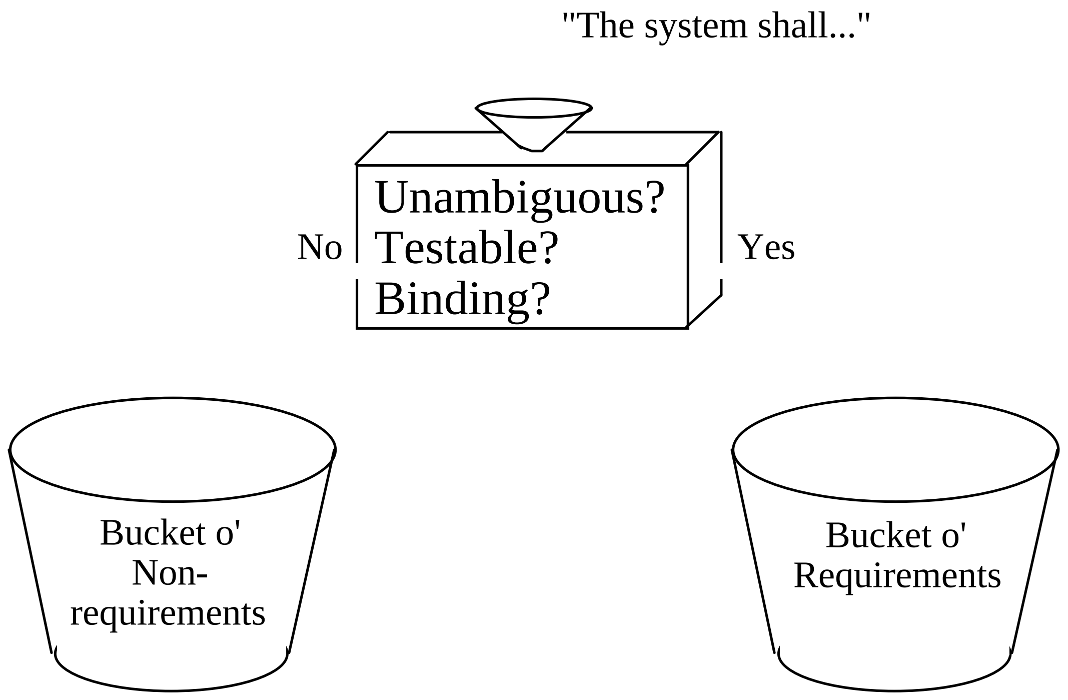
• Per the above definition, the following are not requirements

The system shall be fast
The system shall be user friendly
The system shall be maintainable
The system shall be blue
(assuming no one really cares what color it is)

- These statements are either <u>ambiguous</u>, <u>untestable</u>, or <u>non-binding</u>

Decision Process, Version 1

• Requirements should be separated from non-requirements



• What might be reasonable requirements for a system that plays tic-tac-toe?

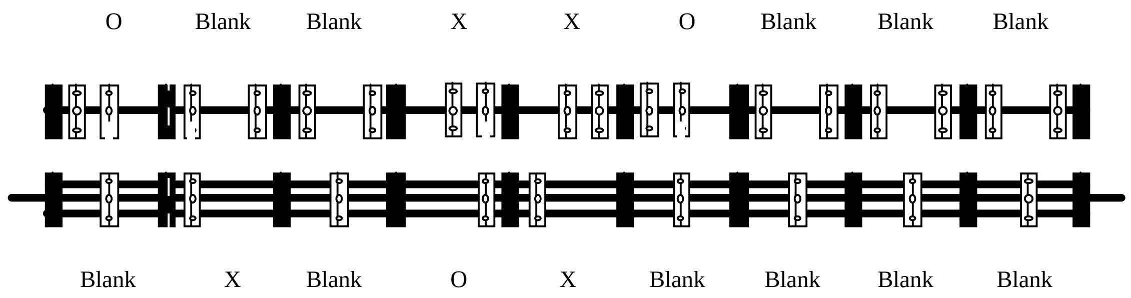
• What might be non-requirements?

• See also Donald Gause & Gerry Weinberg, *Exploring Requirements: Quality Before Design*, Dorset House, 1989 and Suzanne & James Robertson, *Mastering the Requirements Process*, Addison-Wesley, 1999

Tinkertoy Tic-Tac-Toe

• A very interesting implementation of Tic-Tac-Toe [Dewdney89]

Memory Spindle (x48)



Core Piece

Board Layout

1	2	3
4	5	6
7	8	9

Current Layout

	X	
O	X	

• What set of requirements would have led to this implementation?

[Dewdney89]

A. K. Dewdney, *A Tinkertoy computer that plays tic-tac-toe*, Scientific American, October, 1989.

Kinds of Requirements

• From the Tinkertoy Tic-Tac-Toe example, notice that there are two kinds of requirements:

- Requirements that specify what is to be built (e.g., play tic-tac-toe per referenced rules)
    - These might be called "Essential Requirements"
- Requirements that specify how it is to be built (e.g., at least 90%, by cost, of the components must be Tinkertoys)
    - These might be called "Design Requirements"

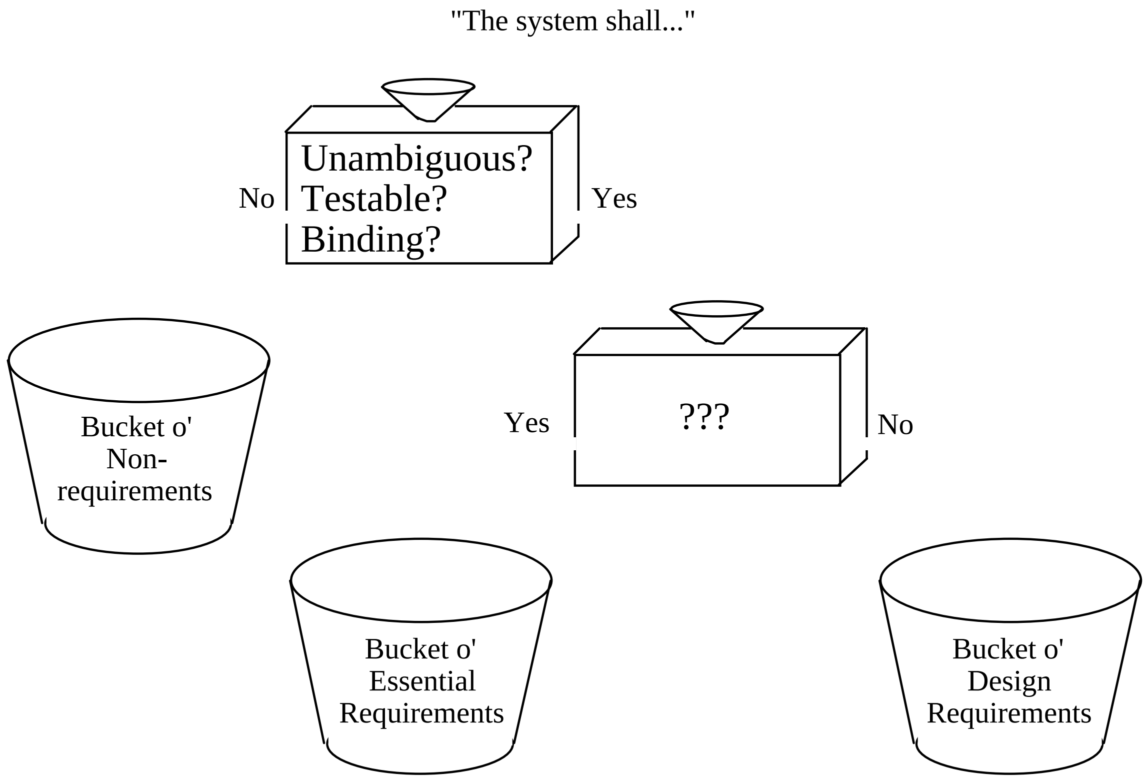
• Notice that this "what" vs. "how" distinction is neither new nor unique

- It appears in DeMarco's "Structured Analysis and System Specification" in 1979
- DoD 2167A talks about "Capabilities" and "Constraints"
- Fowler talks about "Conceptual", "Specification", and "Implementation" perspectives [Fowler97]
- ...

[Fowler97]
Martin Fowler (with Kendall Scott), *UML Distilled*, Addison-Wesley, 1997

Decision Process, Version 2?

- What might be the missing decision criteria?


"The system shall..."
Unambiguous?
Testable?
Binding?
No
Yes
???
Yes
No
Bucket o' Non-requirements
Bucket o' Essential Requirements
Bucket o' Design Requirements

Separating Requirements by Kind

• Steve McMenamin and John Palmer [McMenamin84]

"If we had perfect implementation technology (e.g., a computer with infinite speed, unlimited memory, transparent interface, no failures, and no cost), which of the requirements would still need to be stated?"

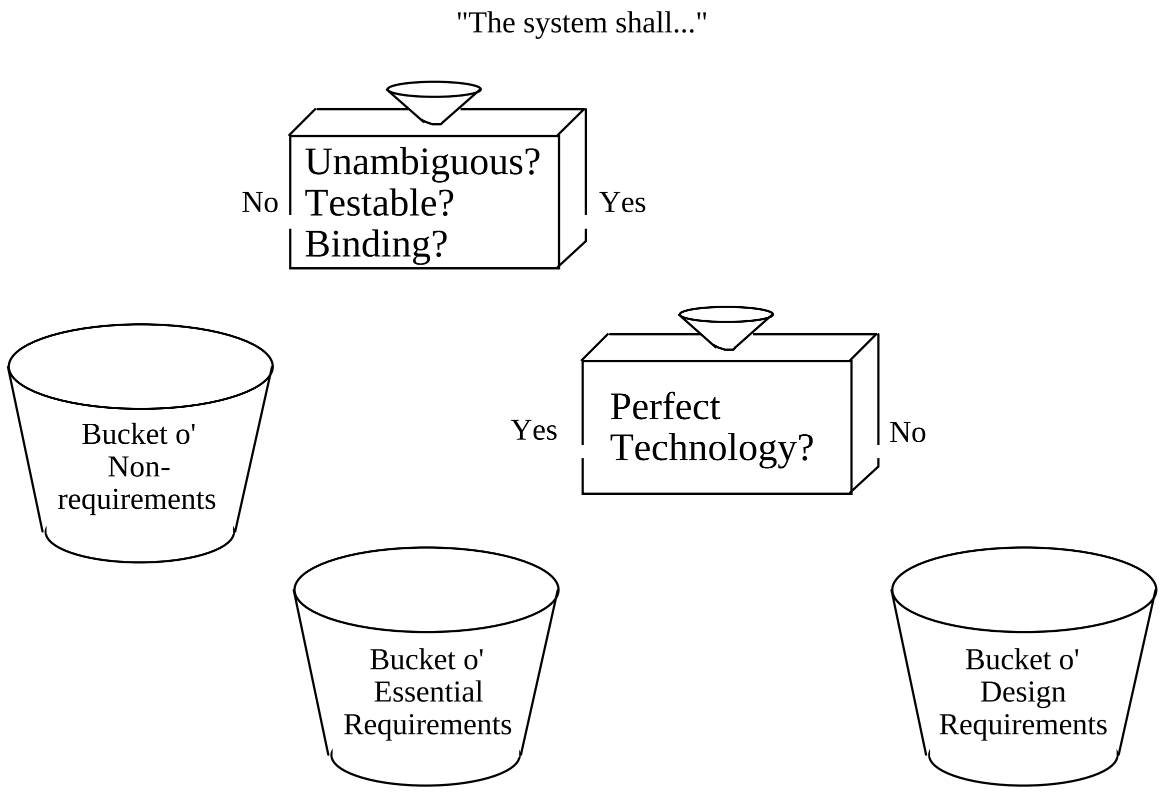
• Every requirement that is still necessary in spite of "perfect technology" is an essential requirement

• If we had this perfect computer …
- Would an ATM still need to process deposit, withdraw, and account query requests?
- Would ATM transaction records still need to be archived to mag-tape after 7 days?
- Would an ATM be usable 95% of the time?
- Would an ATM still need to record how often customers use their cards?

[McMenamin84]
Stephen M. McMenamin and John F. Palmer, *Essential Systems Analysis*, Yourdon Press, 1984, Chapters 1 through 4. See also: Ed Yourdon, *Modern Structured Analysis*, Yourdon Press, 1989, Chapter 17

Decision Process, Version 2

• The decision criteria is whether or not the requirement would still exist if we had that perfect computer


"The system shall..."
Unambiguous?
Testable?
Binding?
No
Yes
Perfect
Technology?
Yes
No
Bucket o' Non-requirements
Bucket o' Essential Requirements
Bucket o' Design Requirements

• Note: Non-requirements can also be categorized as essential or design, if you wish

Implications of Perfect Technology

• Requirements about speed, cost, and capacity go into the design bucket

• Requirements about reliability (MTBF, MTTR) go into the design bucket

• Requirements about I/O mechanisms and presentations go into the design bucket

• Requirements about computer languages go into the design bucket

• Requirements about archiving go into the design bucket

• Requirements about the customer's business policy / business process go into the essential bucket

Why Bother?

• Reduce apparent complexity: one large problem becomes two smaller ones

- Understand the customer's business policy / business process
- Figure out how to automate that business policy / process with the available technology

• Isolate areas of expertise

• Apply the principles of coupling and cohesion at the highest level of the software architecture

- More robust, less fragile systems
- Enable separate evolution of the business policy / business process and the implementation technology

Target Software Architecture

• If the separation between business policy / business process is carried through the architecture and into the code, the highest level of the software architecture will appear as follows

Input Services

Translates incoming real-world events and data into a form usable by the Essential Core:

X-Windows,
Mouse Drivers,
IEEE 488,
ARINC,
Buttons,
Analog-Digital,
etc.

Output Services

Translates outgoing data and events from the Essential Core into a form usable by the real-world :

X-Windows,
Screen Drivers,
Report Generators,
IEEE 488,
ARINC,
Lamp Drivers,
Digital-Analog,
etc.

The "Essential Core"

Implements the portion of the essential system that is automated.

Architecture Services

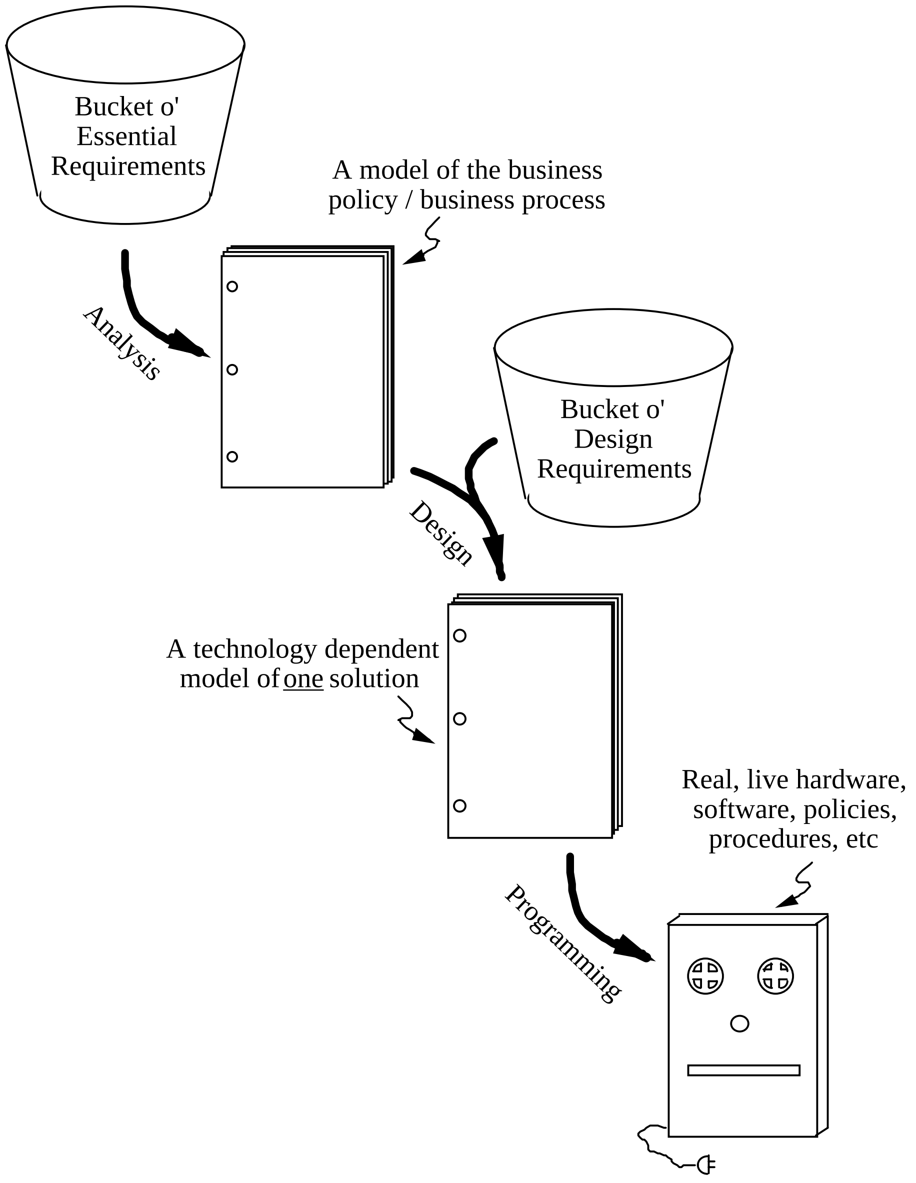
Provides internal computational support facilities to the Essential Core:

Operating System, Database, Multi-task Support, Interprocess(or) Communication, etc.

• Each of the regions in this target software architecture should be highly cohesive about itself and loosely coupled with the other regions

Definitions for Development / Maintenance Activities

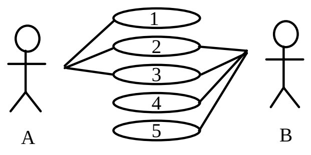
• Analysis and Design can be defined in terms of the kind of requirements being addressed

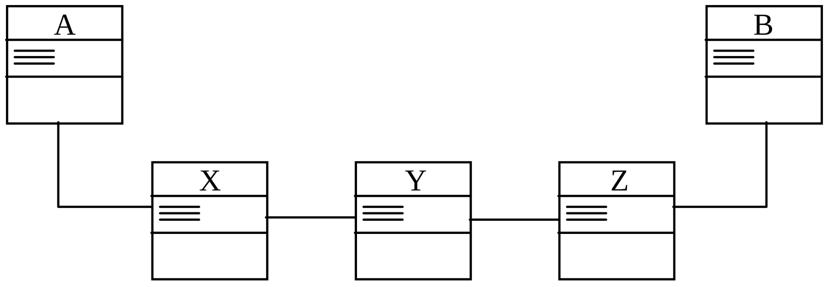

Bucket o' Essential Requirements
A model of the business policy / business process
Analysis
Bucket o' Design Requirements
Design
A technology dependent model of one solution
Real, live hardware, software, policies, procedures, etc
Programming

But this does not necessarily imply a waterfall lifecycle (note: “activity”, not “phase”)

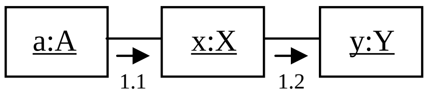
UML for Analysis

Use Case Diagram

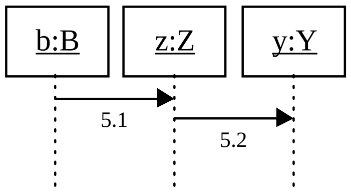




Class Diagram

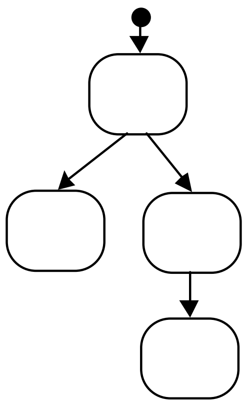


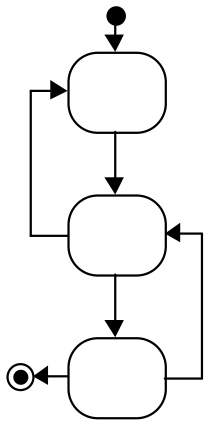
• • •

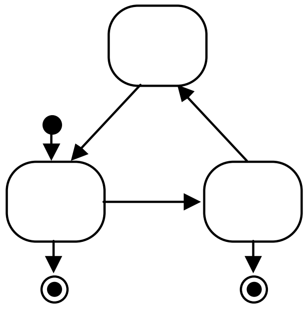


Collaboration Diagram for 1

Sequence Diagram for 5





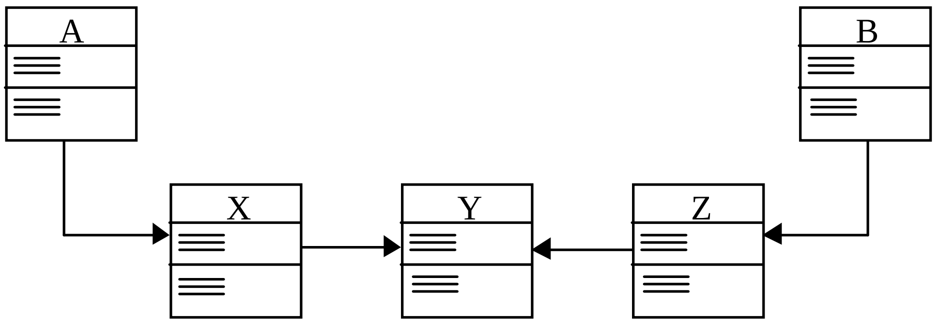


Statechart for X

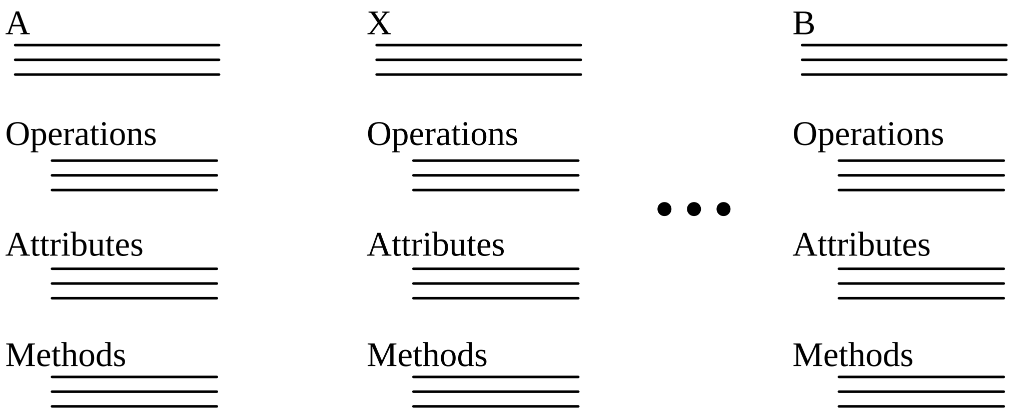
Statechart for Y

Statechart for Z

UML for Design



Class Diagram



Detailed design for A

Detailed design for X

Detailed design for B

Key Points

• Requirements should be unambiguous, testable, and binding

• There are two kinds of requirements

• There are a number of good reasons to separate requirements
- Reduce apparent complexity
- Isolate areas of expertise
- Apply the principles of coupling and cohesion at the highest level of the software architecture

• McMenamin & Palmer’s “Perfect Technology” will help you make this separation

• Analysis can be defined as modeling the customer's business policy / business process

• Design can be defined as dealing with computing technology

• Parts of UML are useful for capturing analysis models

• Parts of UML are useful for capturing design models