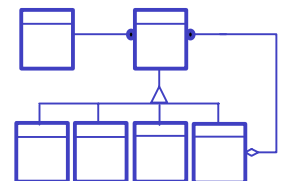# Vertical Industry Frameworks & Components

Connecticut Object Oriented Users Group
December 9, 1997

Scott Koehler

Koehler Consulting, Inc.
Holliston, MA
(508) 429-1589 Tel.
email: info@koehlerconsult.com

Our Background

- Vertical industry business object applications since mid-80s
  - insurance and finance
  - C++ and Smalltalk
- Designed and assisted marketing an insurance framework (selling system) for a major insurance company
- Participated in object harvesting project sponsored by an industry consortium
- Built and market a life insurance tax compliance business object framework
  - (www "*KCI Tax Server*")

Definitions

- Business object
  - an object that corresponds to an entity familiar in the business domain
  - e.g. Stock, Bond, InsurancePolicy
- Class library
- a collection of classes that represent entities of the business used for creating business objects

Definitions (cont.)

- Component
  - encapsulated software services that can be used to build systems, which are packaged for reusability
- Framework
  - a group of business objects or components that work together to provide some useful business application
  - a subsystem
  - e.g. claims processing, order entry

Characteristics of this Market

- Software application comprised of cooperating business objects that deliver a particular business benefit
- Generally strong interrelationships among the business objects
- Multiple business objects combine to provide higher grain components
- Business objects are rarely "generic"
  - company specific rules and methods

Characteristics (cont.)

- Not "black box" components
- Customization is required for a particular company's business rules
- Can become a "black box" component for an individual company

Buying Issues

- Is there value added? How close is the fit?
- 'Build versus buy'
  - conventional thinking
  - decision criteria:
    - quality of the offering, how OO is it?
    - time-to-market
    - functionality
    - success rate of new development projects
    - ease of integration
    - maintainability

Buying Issues (cont.)

- Skill set of staff
- Commitment to newer technologies
- Project characteristics
  - enterprise solution or project-based
- Corporate culture

Vendor Issues

- Customer sincerity:
  - "If there was a model of my industry, I'd buy it in a minute."
  - Not Invented Here friction
  - Professional curiousity
- Intellectual property
 - "please send your design guide"
- Pay as you go...

Vendor Issues (cont.)

- Generic / specific tradeoffs
- Product vs. engagement aid
- Pricing
  - few frame of references
  - shrinked-wrapped based expectations
- Knowledge transfer
- References

Where's the Value

- Intellectual property
- A foundation to extend
  - gets past the clean sheet of paper
  - infrastructure
- A defined set of requirements
- Existing design and implementation
- Tested code
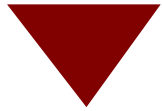
What's Important

- Architecture
  - application architecture
  - application components
  - design principles
  - design patterns

Application Architecture
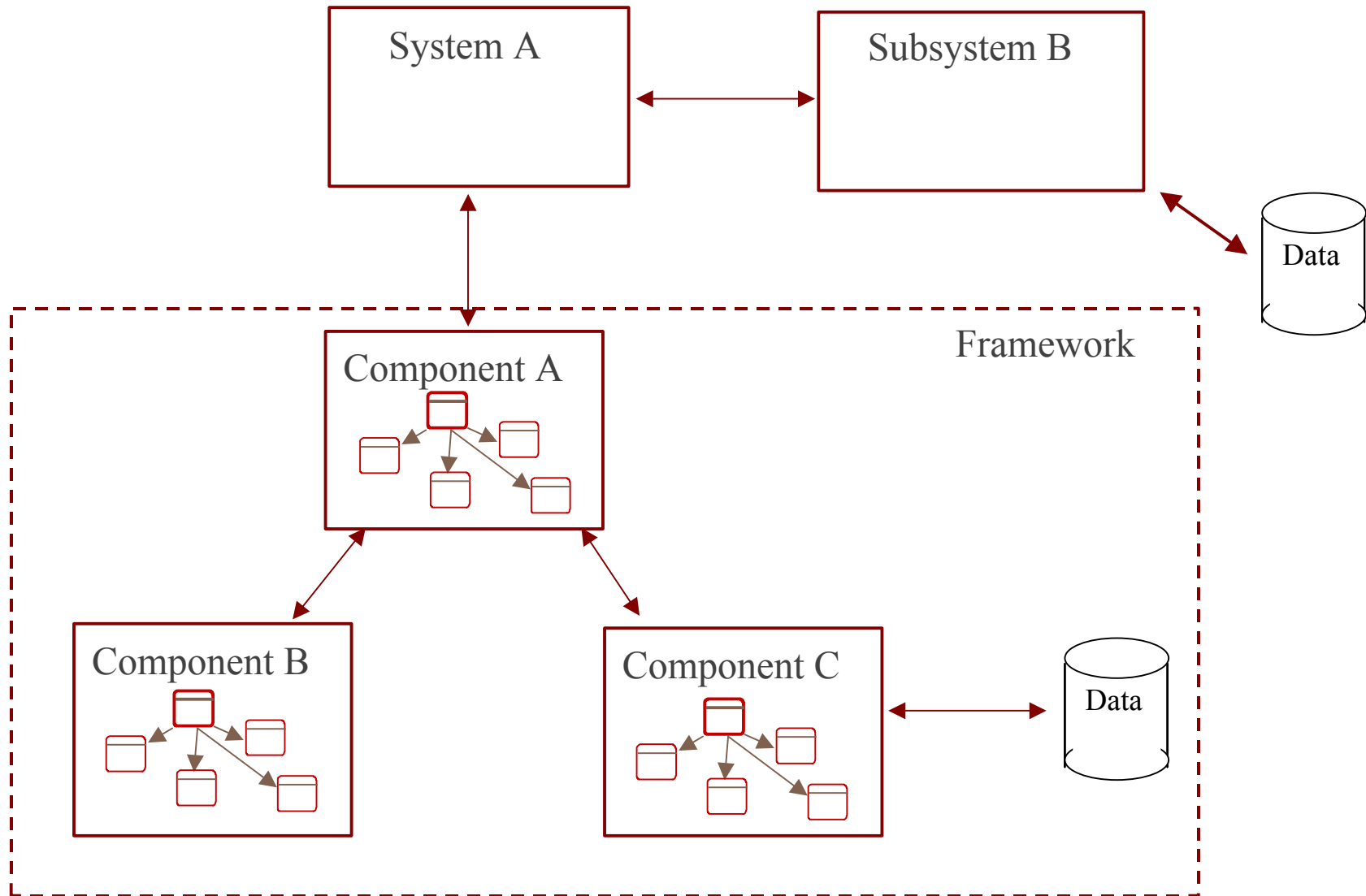
Design Principles

- A written set of stated design objectives to use as a reference during the development phase
- Addresses current and future uses of the framework
- Affects the representation and placement of knowledge
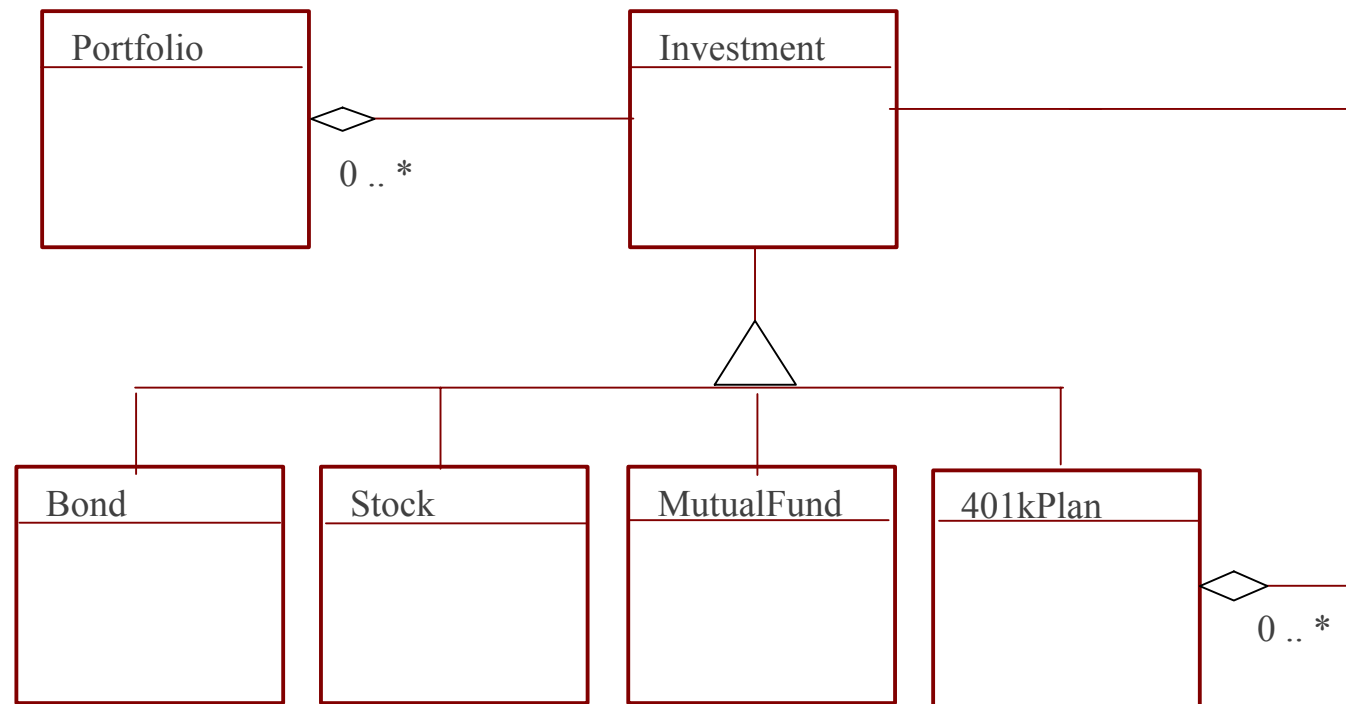- Influences design decisions

Design Principle

- *"The tax subsystem will contain the knowledge of the tax law with other interfacing subsystems exhibiting little or no knowledge of the tax law."*

reference KCI, *KCI Tax Server Architecture*
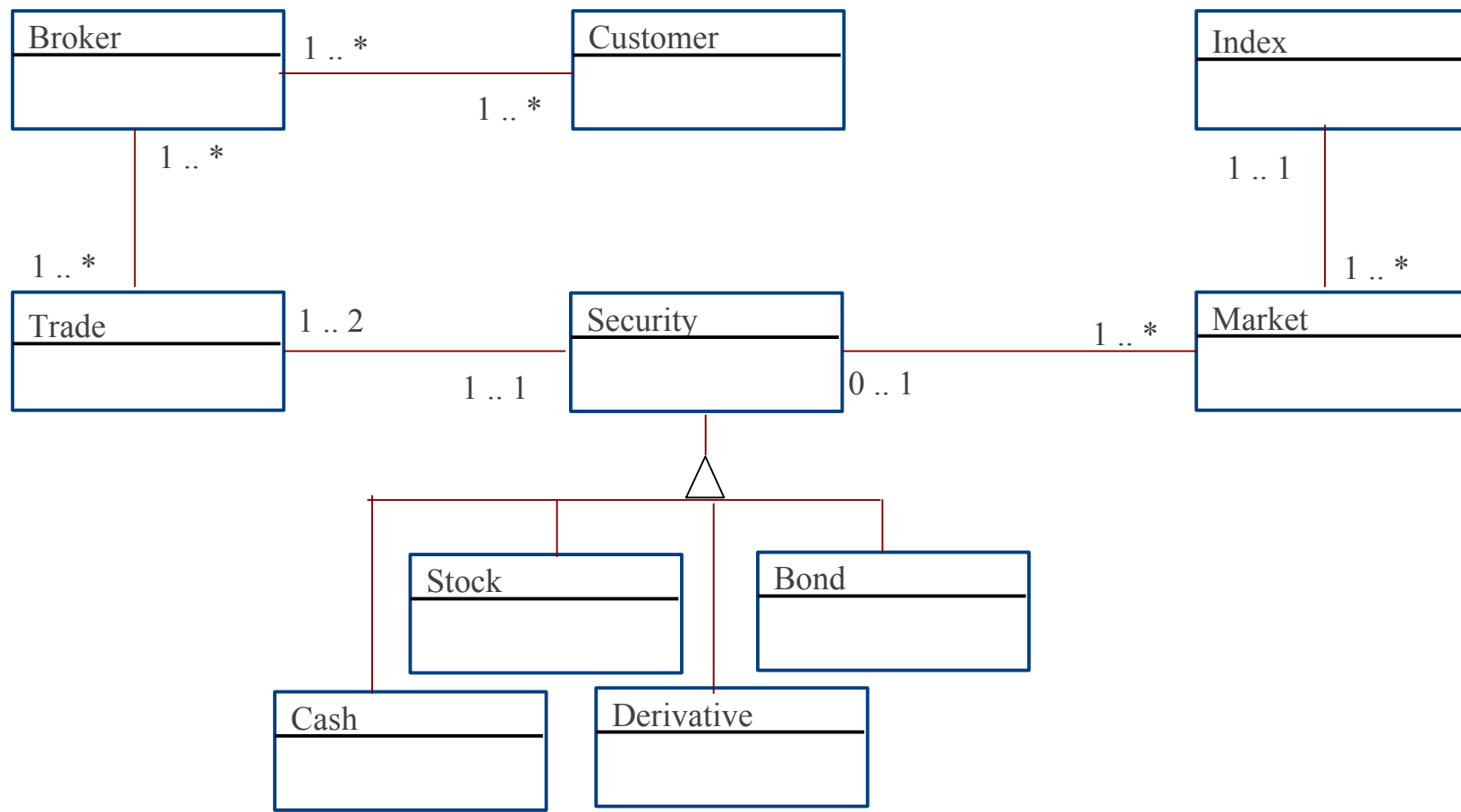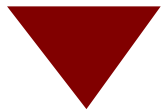
Design Patterns - Composite
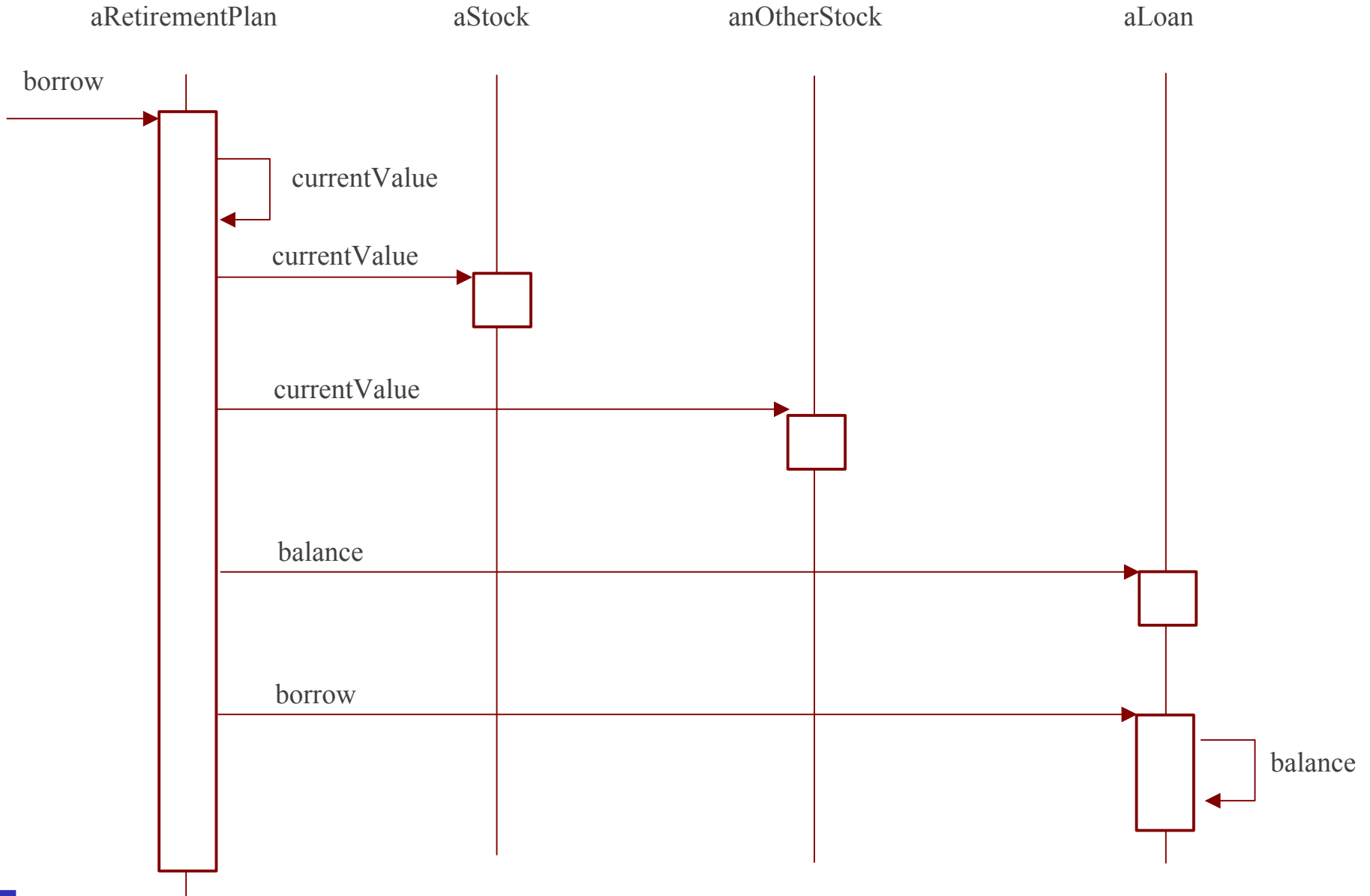
What's Important (cont.)

- Functional specifications
  - use cases
- Design documentation
  - object models
  - interaction diagrams, etc.

Finance Industry Object Model

Broker		1 .. *		Customer			Index	

Broker — 1 .. * — Customer

1 .. *

1 .. *

1 .. 1

1 .. *

1 .. *

Trade	1 .. 2	Security	1 .. *	Market

1 .. 1

0 .. 1

Security
├── Stock
├── Bond
├── Cash
└── Derivative

Business Object Interaction Diagram

aRetirementPlan aStock anOtherStock aLoan

borrow

currentValue

currentValue

currentValue

balance

borrow

balance

What's Important (cont.)

- Implementation language (??)
- Source code
  - right to modify, enhance
  - coding guidelines, commenting
- Test suites

Evaluation Issues

- How well does the model map to your enterprise model?
- How consisent is the business terminology?
- How flexible is the model for adapting your specific business rules and methods?
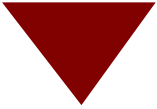- Is it a good fit for your needs?

Evaluation Issues (cont.)

- What is the OO experience of the vendor?
- Pricing
  - How does the price compare to your cost to develop?
- Delivery time
- Implementation language
- Platform support
- Standards compliance
- Customer references
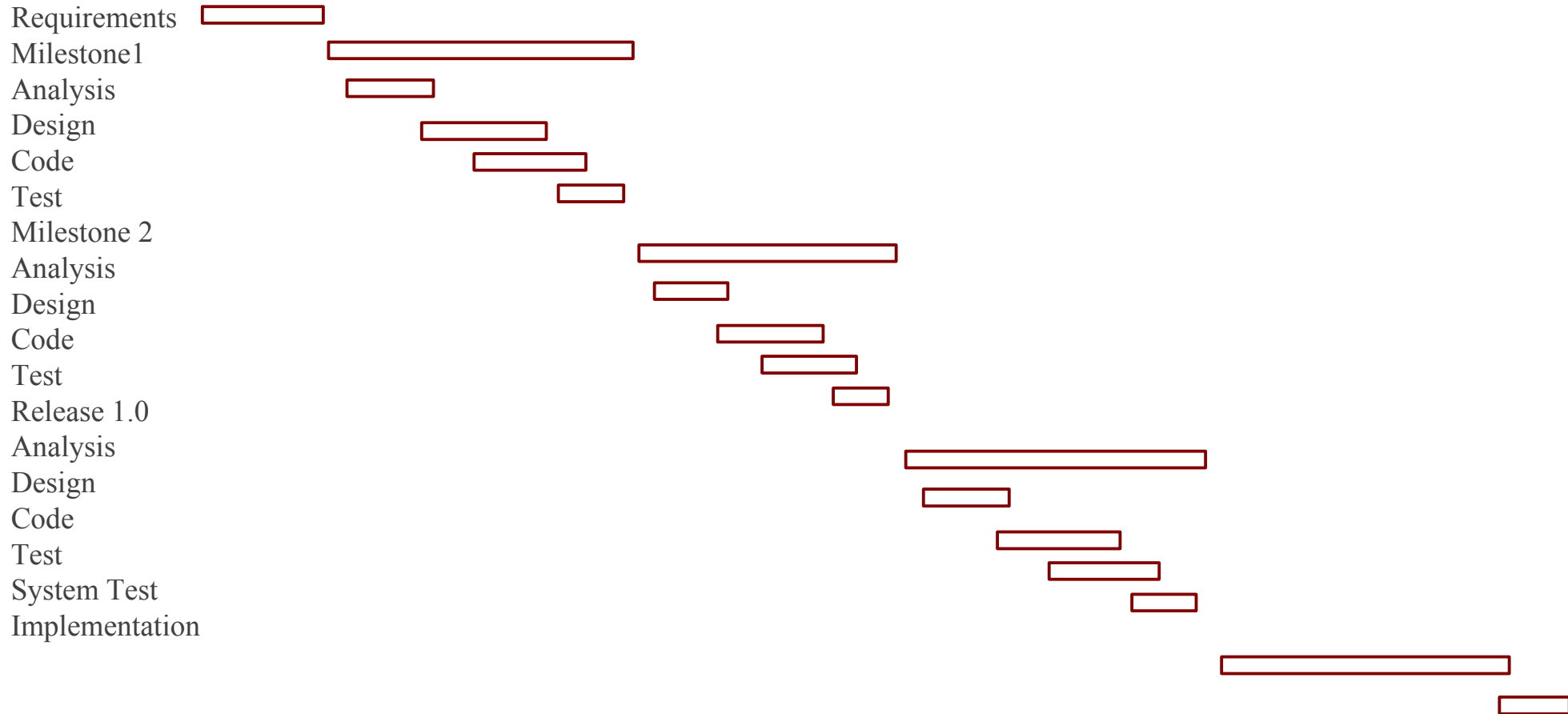- Consider conducting a sample test

Implementation Issues

- Knowledge transfer
- Support
- Extension:
  - Reviews
    - architecture
    - design
    - code

A Project

Requirements
Milestone1
Analysis
Design
Code
Test
Milestone 2
Analysis
Design
Code
Test
Release 1.0
Analysis
Design
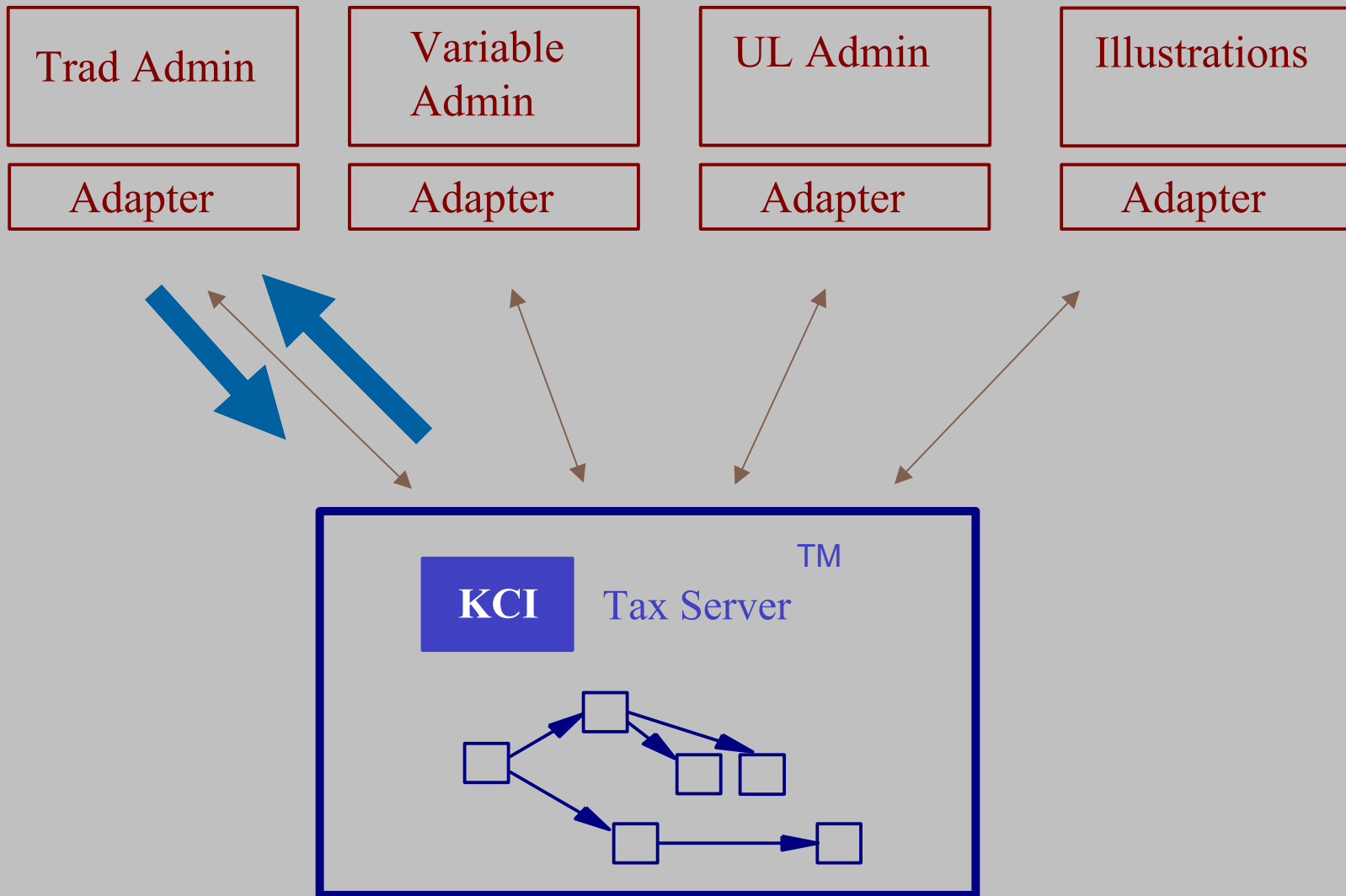Code
Test
System Test
Implementation

A Framework / Component Project

- Project with large life insurance company
- Tax subsystem
- 25 year old COBOL legacy administrative system
- Component-based legacy transition
- Daily batch transactions
  - premium payments, loans, etc.

Tax Subsystem Architecture

Trad Admin	Variable Admin	UL Admin	Illustrations
Adapter	Adapter	Adapter	Adapter

KCI Tax Server ™

Project (cont.)

- 10 person development team
  - 3 external resources (KCI)
  - client personnel with conventional skill set
- Architectural components:
  - 25 year old COBOL legacy system -- variable life insurance
  - IBM MQSeries messaging
  - DB2
  - Windows NT Server based KCI Tax Server
    - Smalltalk business object component

Project (cont.)

- Started with initial framework & architecture
- ~14 month duration
- The result:
  - 1st production run on 10/27/97
  - It works!
- Additional connections planned

Summary

- Business object frameworks / components are an important part of software development
- It is a strategic investment
- Make an honest assessment of build v. buy
- If you buy:
  - gain an understanding of what you're are getting
  - get the source code
  - achieve a proper knowledge transfer

Summary (cont.)

- If you build one:
  - Consider a development partner or get help if this is your first time
  - End users should resist entering the software business
  - Stay focused on your distinctive competence
  - Stay focused on meeting the needs of the business