SOFTWARE REUSE
Architecture, Process and Organization for Business Success

Jacobson, Griss, Jonsson
Addison-Wesley, 1997

Lecture Slides
to Accompany the Text

Software Reuse (part 2)

- Object Oriented Business Engineering
  - » **Business Process Reengineering** (BPR) is "the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service and speed." (Hammer & Champy 1993, Hammer & Stanton 1995)
  - » **BPR** has risks
    - - the change process itself
    - - the technology employed
  - » The OO paradigm provides a modeling technology that spans the process and improves communication and thus "**clarity**"

Slides adapted from Software Reuse, Ivar Jacobson, Griss, Jonsson, Addison-Wesley, 1997

BPR Defining the Process.

- Create a BPR directive
  - » goals, business situation, impetus for change
- Envision the new business
  - » process & technology opportunity mining
  - » “best practice” partners & competitors
  - » goals, objectives of future business processes
- Reverse engineer the existing business
  - » unify the corporate “self-image”
  - » identify reengineering target in the “vision”
- Forward engineer the new business
  - » incrementally develop & describe new systems
- Implement the new business
  - » start with high success potential aspects
  - » build momentum, manage risks, build the “team”

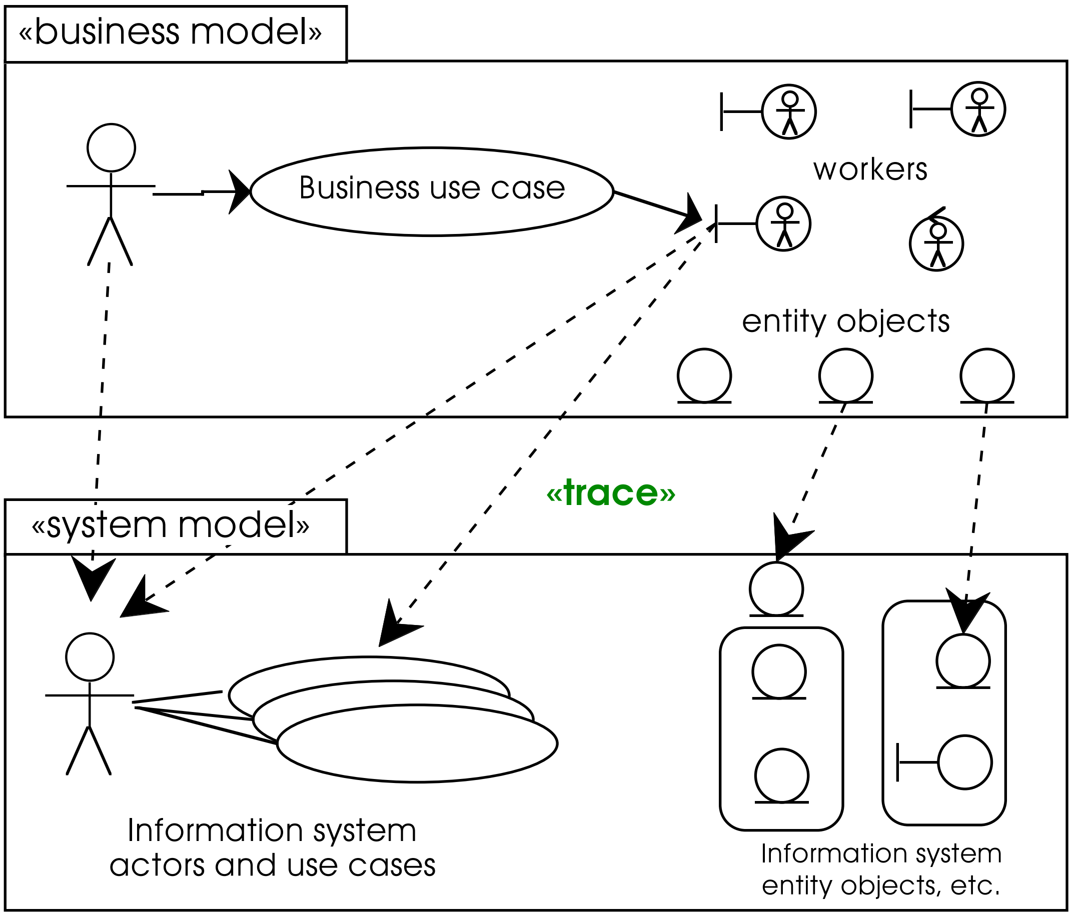
Object Modeling the Business

- Model the Business as a System
  - » a **business system** represents an organization unit that we want to understand better in order to make it more competitive
  - » a **business actor** represents a role that someone or something in the environment can play in relation to the business
  - » a **business use case** is a sequence of work steps performed in a business system that produces a result of perceived and measurable value to an individual actor of the business
- Our OO Use Cases apply directly, but at the *business model* level of abstraction

Business model parts

- Business use cases must identify not only functions but also competencies and roles that need to be played in the business process
  - » **case worker** represents people with a certain competency and interact with actors
  - » **internal workers** have a certain specialty but work within the business
  - » **entity object** represents a “thing” handled or used in the business
  - » **work step** describes a piece of work a worker performs
  - » **resource units** group persons and information systems
  - » **competence units** group workers their “tools”

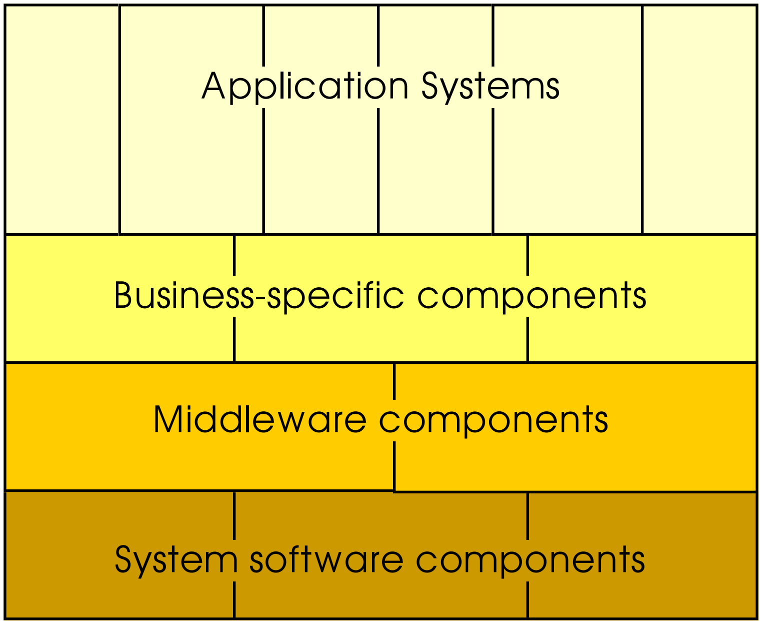
Business Models Trace to System Models


«business model»
Business use case
workers
entity objects
«trace»
«system model»
Information system actors and use cases
Information system entity objects, etc.

Applying BPR to SDLC

- Process and Structure are the key
  - » effective reuse requires significant software engineering organization and process change
  - » the organization must match the system's architecture
- Seeking architecture / business harmony
  - » processes and organization are developed **incrementally**
  - » **input from business models** drives the evolution
  - » **explicit techniques** are used to represent them
  - » we use the **same techniques** to model / represent the reuse business itself!!!

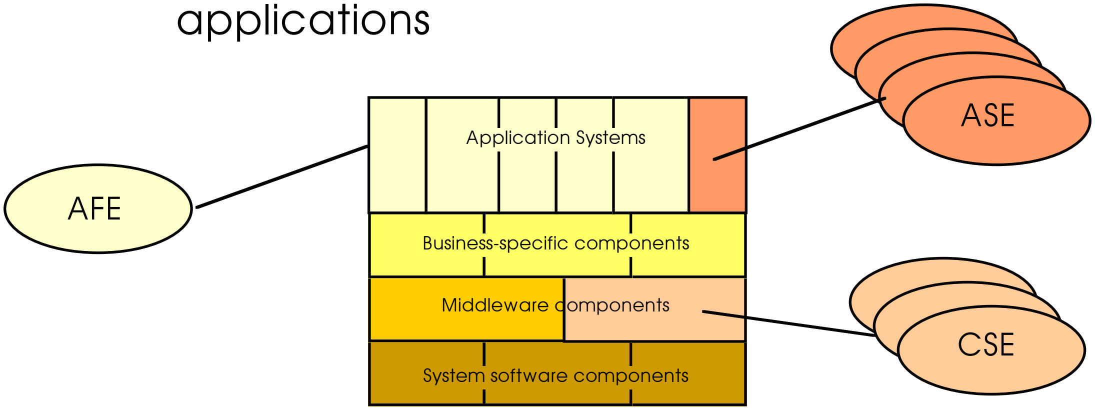
Architectural Style
(system layering)



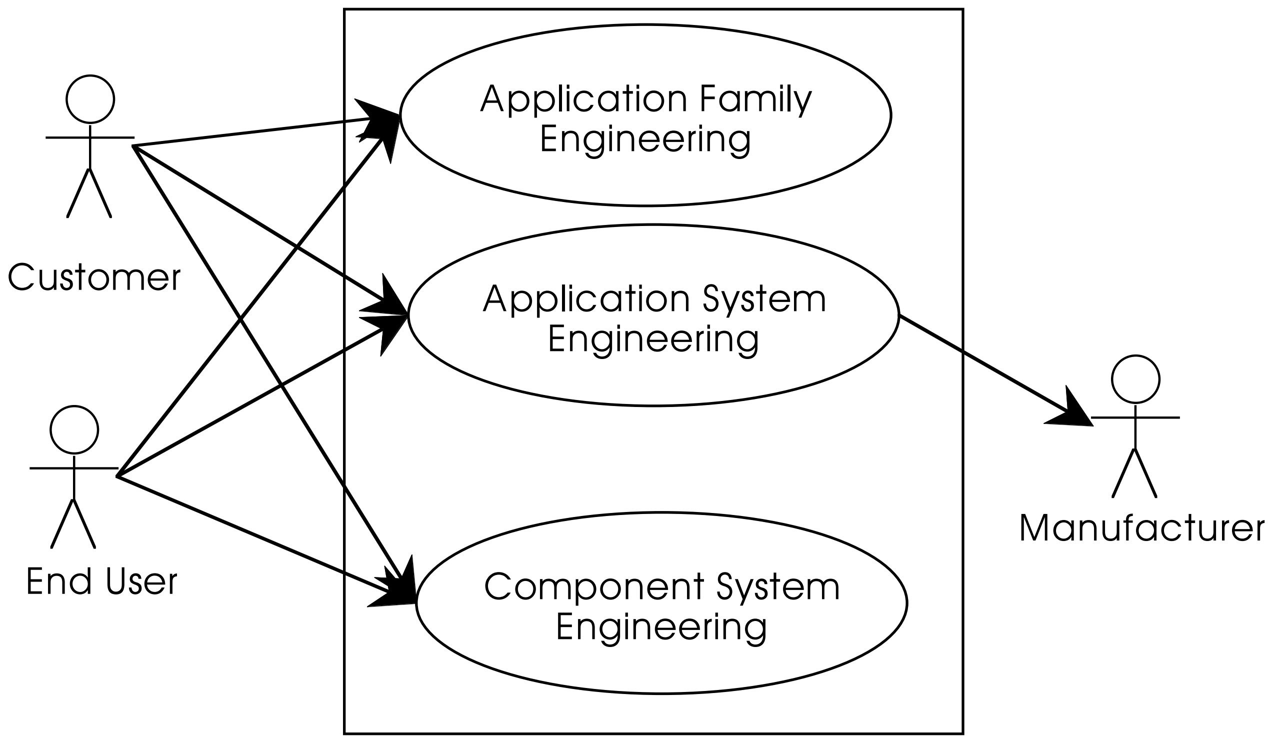
The Architectural style of a system is the denotation of the modeling languages used when modeling that system. *(Jacobson et al., 1992)*.

Software Engineering Process (SEP)

- We organize around our products:
  - » **Component System Engineering** (CSE) is organized around each component system product
  - » **Application System Engineering** (ASE) is organized around each application system product
  - » **Application Family Engineering** (AFE) is organized around each family of related applications



Reuse Business Processes

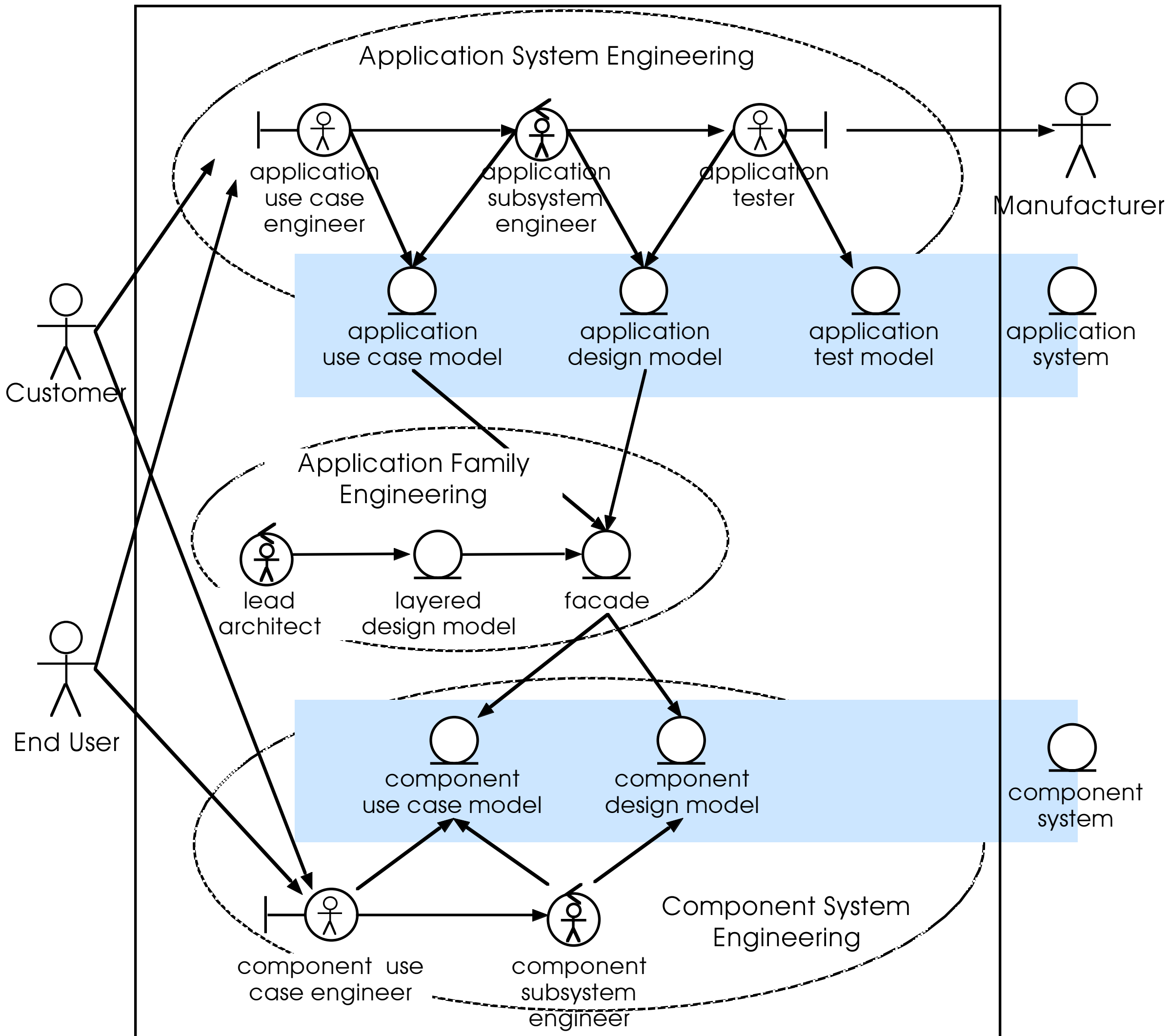

Application Family Engineering
Application System Engineering
Component System Engineering
Customer
End User
Manufacturer

Customers request application systems, place requirements on them, and usually pay for the systems.

End User is someone who will use the application system when it is installed in the target organization

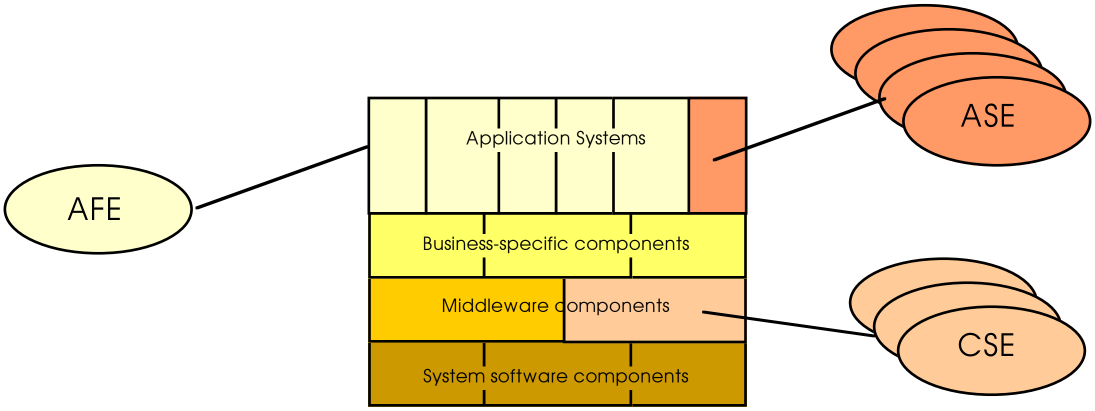
Manufacturer receives the new version of the application system when developed, customizes, configures, produces, and delivers complete applications to the customer.

Reuse Business Processes


Application System Engineering
application use case engineer
application subsystem engineer
application tester
Manufacturer
Customer
application use case model
application design model
application test model
application system
Application Family Engineering
lead architect
layered design model
facade
End User
component use case model
component design model
component system
component use case engineer
component subsystem engineer
Component System Engineering

Reuse Business Processes

- **Application Family Engineering** (AFE) process develops and maintains the overall layered architecture
- **Component System Engineering** (CSE) process develops use case, analysis, design, implementation and test models for a component system focusing on robust, extendible, and flexible components
- **Application System Engineering** (ASE) process develops use case, analysis, design, implementation and test models for a specific application system engineered from components



Application Family Engineering (AFE)

"The process of application family engineering develops and maintains the layered system, represented as a superordinate (domain) system with use case, analysis, design, deployment, and concurrency test models. Business actors and workers are used to identify actors and use cases for the layered system. The use cases are studied to identify the architecturally most relevant analysis types, which are grouped into subsystems that are molded to suit the target implementation environment. During design, the layered system is adapted to Commercial Off the Shelf products and legacy systems, and defined using explicit interfaces and facades. Finally the architecture is implemented and tested using the key use cases as drivers.

AFE proceeds in iterations, where at first only a small fraction of the use cases are selected, analyzed, designed, implemented, and tested according to the above steps. These use cases are selected so that they are the most relevant to defining the architecture. Then the second most relevant use cases are selected, and so on. Based on the resulting architecture, the application and component systems are then developed in separate processes (CSE and ASE)." (Jacobson, Griss, Jonsson)

AFE Steps:

- AFE1: capturing requirements that have impact on architecture
  - » essential to understand how the family applications will work together and support the whole user domain
  - » business, market, and customer needs and trends
  - » high level use cases represent architecturally relevant requirements
    - - static vs. dynamic domain features
    - - feature interoperability/compatibility
- AFE2: Robustness Analysis
  - » "B-C-E" boundary-control-entity patterns
  - » boundary types for actors / use cases
  - » control types / business rules
  - » high level application domain patterns of object composition and interaction

AFE Steps (cont):

- AFE3: Designing the layered system
  - » subsystems organized into layers with facades between them
    - each subsystem may ultimately become an individual application or component system
    - coupling / cohesion, middleware glue
- AFE4: Implementing a layer system architecture
  - » prototype code that implements "architecturally relevant use cases"
  - » test them
  - » test concurrency and distribution support
- AFE5: Test the layered system
  - » based on requirements from AFE1 build test suites to exercise the system interfaces

Component System Engineering (CSE)

“The process of component system engineering develops use case, analysis, design, implementation and test models similar to the ordinary OOSE software engineering process focusing on building robust, extendible, and flexible components. The component engineering uses input from business models, models of the superordinate (domain) system, and input from users, customers, and domain experts.

Business actors and workers are used to identify actors and use cases, which in turn give rise to analysis types grouped into subsystems that are adapted to the implementation environment. The superordinate use case and design models are used to define interfaces of the component system, and suggest actor and use case components.

Components are grouped into a number of facades to make them as reusable as possible -- including all documentation, examples, tools, and configuration aids available.” (Jacobson, Griss, Jonsson)

CSE Steps:

- CSE1: capturing requirements focusing on variability
  - » multiple application support
  - » long term reusable asset
  - » variability engineering means understanding the domain “articulation points”
- CSE2: Robustness Analysis to maximize flexibility
  - » finding robust and reusable object structure
  - » how are things the same and how different
  - » define “variation points” providing “hooks” into analysis types
  - » package the analysis types minimizing the dependencies (coupling)
  - » verify that if the analysis types allow for interaction of components that the components do too

CSE Steps (cont):

- CSE3: Designing the component system
  - » identify design class candidates for each analysis class
  - » use the AFE documentation to conform to interaction guidelines
  - » verify that component interfaces support the AFE documentation
  - » build classes, define methods, allocate threads
  - » implement and unit test
- CSE4: Implementing a component system
  - » finally everyone can get "code on their breath"
  - » develop and use design-to-code guidelines for the specific programming language / system
- CSE5: Test the layered system
  - » test reuse, abstraction and any configurations
- CSE6: Package Everything for Reuse
  - » documents, tutorials, guidelines, test-beds, training manuals, component encyclopedia

Application System Engineering (ASE)

“The process of application system engineering develops use case, analysis, design, implementation and test models similar to the ordinary OOSE software engineering process focusing on a specific application requirement. This process is accomplished by reusing components from CSE to improve quality and time to market.

The business models and models of the superordinate system define the interfaces that the application system must supply and define the component systems that the application system can reuse. The architecture of the reused component systems guides the application engineers as they architect and design the application system.

ASE is a sequence of iterations of controlled implementation of use cases specifically adding value to the customer and end users. It is also targeted at reducing one or more risks associated with developing the application system. ” (Jacobson, Griss, Jonsson)

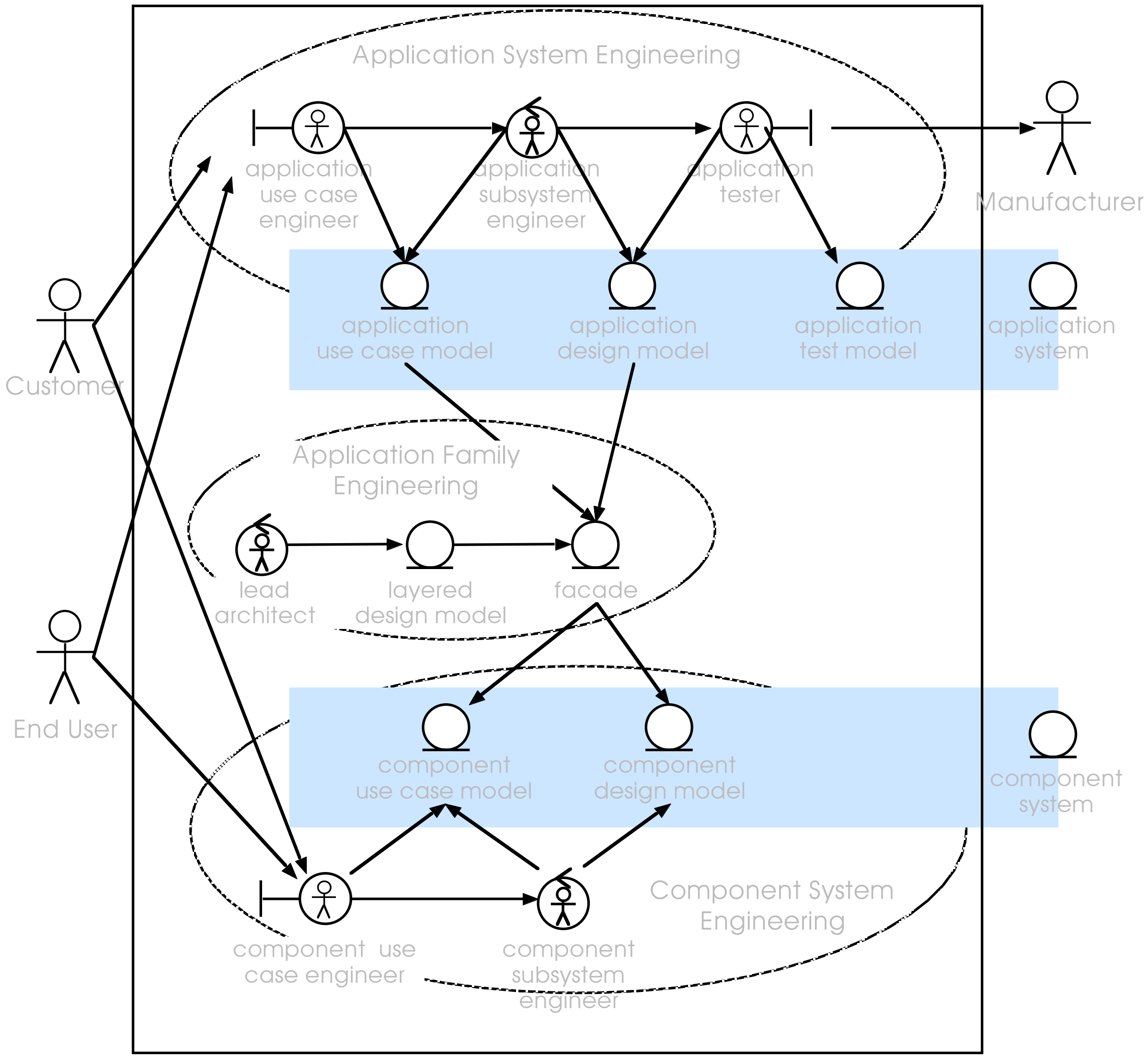
ASE Steps:

- ASE1: capturing requirements
  - » identifying customer & end user needs
  - » aligning same to AFE and CSE requirements
  - » reusing and extending AFE use cases
- ASE2: robustness analysis for flexible application systems
  - » reusing AFE and CSE use cases reusing their respective analysis types as well
  - » reuse, reuse, reuse
  - » justify extensions
  - » document and communicate the requirement to the AFE and CSE teams

ASE Steps (cont):

- ASE3: Designing the application system
  - » reuse design classes and design models from AFE and CSE
  - » exploit "wrappers," "decorators," and "mediators" (Gamma, pattern languages)
- ASE4: Implementing an application system
  - » scripting tools, VBScript, JAVAScript, etc.
- ASE5: Test the application system
  - » use the AFE and CSE test models
  - » use the application requirement test model
- ASE6: Package Everything for Support
  - » description of purpose, function, configuration
  - » guides for installation, tuning, configuration
  - » environmental requirements, platform, resources
  - » end user support, application history
  - » FAQ's, defects, limitations
  - » everything that any well OOSE'd system would have!!

Reuse Business Processes


Application System Engineering
application use case engineer
application subsystem engineer
application tester
Manufacturer
Customer
application use case model
application design model
application test model
application system
Application Family Engineering
lead architect
layered design model
facade
End User
component use case model
component design model
component system
component use case engineer
component subsystem engineer
Component System Engineering