SOFTWARE REUSE
Architecture, Process and Organization for Business Success

Jacobson, Griss, Jonsson
Addison-Wesley, 1997

Lecture Slides
to Accompany the Text

Software Reuse (part 3)

- Transitioning to a reuse business
  - » systematic, incremental transition controls risk:
    - + assessing reuse readiness
      - - business
      - - process
      - - domain
      - - organization
    - - designing a multi-step, pilot-driven plan
    - - customizing the generic RSEB org / design
    - - training, tool development, deployment
  - » employ BPR which is process centered
  - » manage people issues
    - - stakeholders, fear mgmt, change agents & champions, success mgmt, leadership

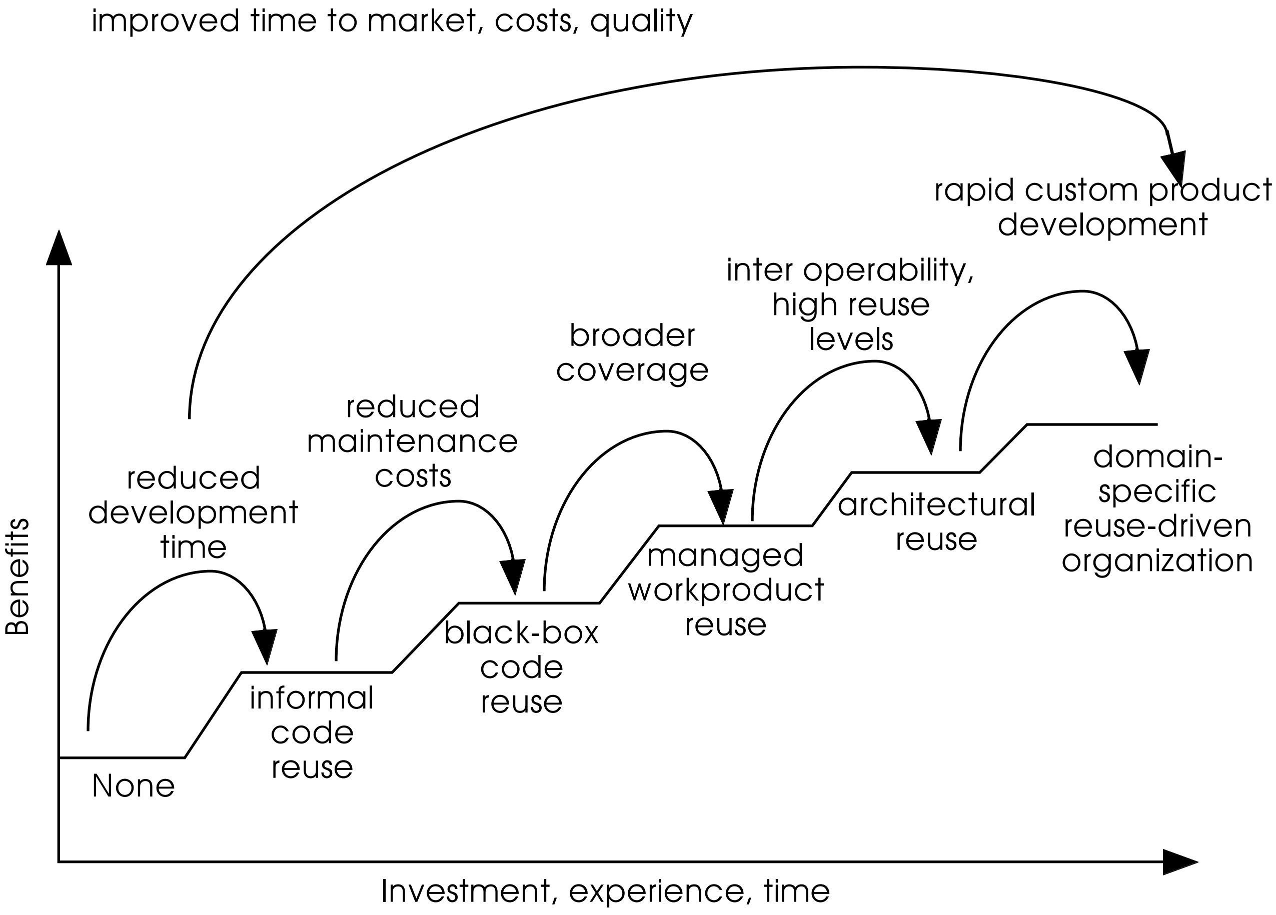
Slides adapted from Software Reuse, Ivar Jacobson, Griss, Jonsson,
Addison-Wesley, 1997

Transition to reuse business

- Create a reengineering directive
  - » public commitment to the reuse thrust
- Envision the new reuse business
  - » first cut of new architecture, bus. processes and organization, stakeholders, & champions
- Reverse engineer the existing development organization
  - » study current architecture, assets, processes
- Forward engineer the new reuse business
- Implement the new reuse business
  - » training, incrementally replace old systems
- Continuous process improvement
  - » systematically collect & analyze reuse metrics

Adopting Reuse Incrementally

HP’s systematic reuse adoption


improved time to market, costs, quality
rapid custom product development
inter operability, high reuse levels
broader coverage
reduced maintenance costs
reduced development time
Benefits
None
informal code reuse
black-box code reuse
managed workproduct reuse
architectural reuse
domain-specific reuse-driven organization
Investment, experience, time

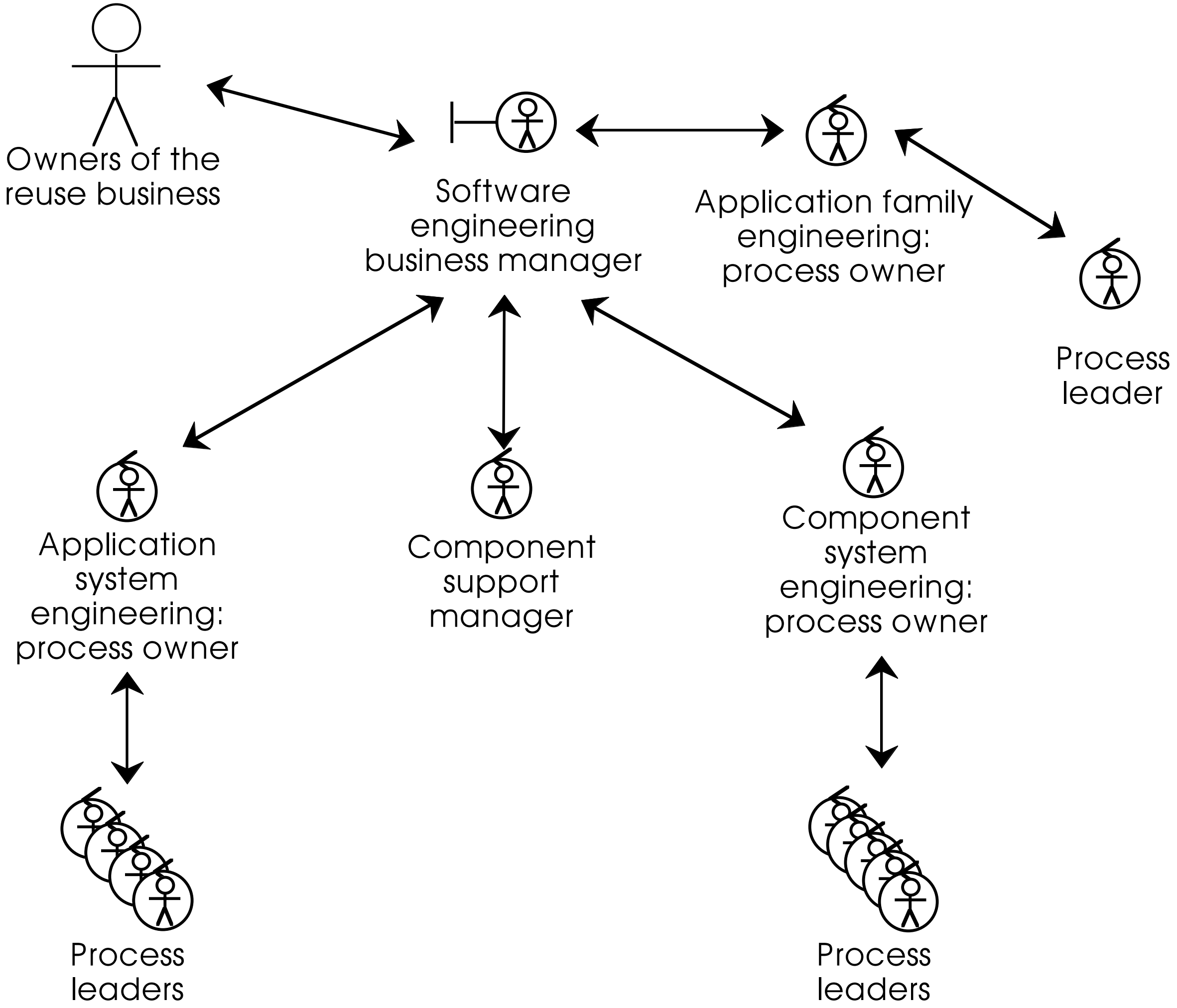
Building Skills/Trust Incrementally

- black box code reuse
  - » trust off-the-shelf code reuse (not invented here)
- library and workproduct management
  - » trust formal use cases and analysis products
- architected components and systems
  - » defining architectures to support consistency
- application and component eng. skills
  - » choosing parts rather than writing code
- reuse-oriented process and org. mgmt.
  - » trusting global vs. local productivity measures
- new tools and technology
  - » acquiring tools to focus on sustained reuse

Transition Iterations

	1st Iteration	2nd Iteration	3rd Iteration	4th Iteration
Business need and opportunities	Rengineering directive	Product plan	Customer Orders	End-user feedback
Application family and architecture	Architecture outline	Architecture baseline	Component systems	Application systems
Teams enabled	Architects	Component engineers	Application engineers	Component support
Processes defined	AFE	CSE	ASE	Custom ASE & CSE processes

Reuse Team Management



Owners of the reuse business

Software engineering business manager

Application family engineering: process owner

Process leader

Application system engineering: process owner

Component support manager

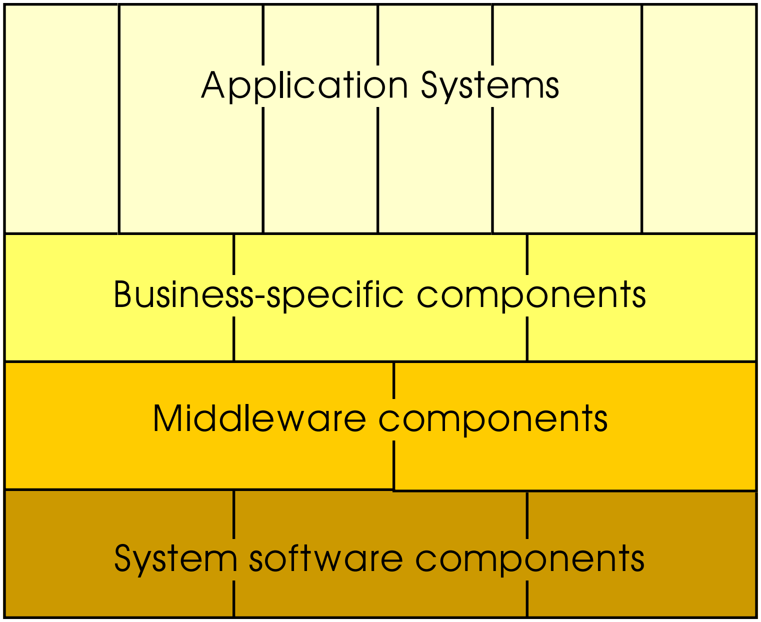
Component system engineering: process owner

Process leaders

Process leaders

Reuse Driven Architecture

(system layering)



Application Systems

Business-specific components

Middleware components

System software components

Managing reuse business

- Business goal focus
  - » customer demand consciousness
  - » multi-project management
- Measurement is the key
  - » you only see change in what you watch
- “It’s the economy, stupid”
  - » you don’t get something for nothing
- Continuous process improvement
  - » testing our theories for “where we are”
- Managing people and organization
  - » so many coexisting and interacting projects (CSE, AFE, ASE) competing for the business resources

Management: business goal focused

- ensuring continued progress to meet the business goals that first motivated the transition
- leading and supporting the transition, clearing roadblocks
- adjusting to respond to changing business conditions and discovered process weaknesses
- keeping projects and goals aligned

Project management++

- traditional management techniques are relevant and effective *(to a point)*
- each project is project manageable, but they are all interconnected
- standard metrics won't work for "dependent" projects
- the combination of projects puts a reuse business on a larger scale than most metrics programs
- most of the "players" do not have an "academic" background in reuse, reuse metrics, or reuse business management

Measurement is the key

- measurement answers questions:
  - » How is our project doing?
  - » How much are we doing?
  - » Are we doing it efficiently?
  - » Are we investing the right amount?
  - » Will we get the expected long-term benefits?
- measuring key driver indicators
  - » customer demand drives the size and shape of the application families --> staffing, training
  - » reuse levels, application engineering steps, component delivery delays --> time to market
  - » component reuse frequency (or infrequency)
    - --> cost / productivity
  - » component specialization cost / flexibility --> target application variety

Measurement goals

- measuring the project "matrix"
    - » AFE, ASE, CSE, Component system support CSS
- fund-allocating levels of management want to know about
    - » return on investment - ROI
    - » cost center profit/loss
    - » investment amount per time period
    - » estimated break-even crossover point
- project managers want to know about
    - » estimated effort
    - » scheduled time
    - » defect rate
- the software engineering business manager wants to support ***CPI***

Reuse business measures

- **SIZE**: some measure of the amount of text or function within a workproduct
    - » source lines of code (SLC), function points, SLC equivalence for non-code workproducts
- **REUSE LEVEL (R)**: the ratio of "size of workproduct derived from reusable components" to "total application size"
- Quality: a measure of the number of defects in a workproduct related to size
- Complexity: combination of size and structure complexity to estimate maintenance
- Cohesion & Coupling: integration

Reuse economics

- Measurement
  - » define and collect raw data, size, reuse level, and time spent
- Cost/benefit *Estimation*
  - » interrelate the measures approaching effort, cost, or time. *(e.g. cost savings to reuse level)*
- Reuse *investment analysis*
  - » efficiency / effectiveness of reuse in the business operations
  - » locating "room for improvement"
  - » connection those to process steps

Estimating reuse costs

- **$C_{no\text{-}reuse}$**
  - » cost of developing without reuse
  - » what the application would have costed to build without the reuse business goals
- **Reuse Level**
  - » **R** = (size of reused components) / (size of application system)
  - » "how much of the application was reused"
- **F_{use}**
  - » relative cost to reuse a component
  - » overhead to locate, configure and apply a reusable component
  - » *(0.10 - 0.25, 0.2 default)*

Estimating reuse costs

- Application development includes reuse and non-reuse
  - » $\mathbf{C}_{\textbf{part-with-reuse}} = \mathbf{C}_{\textbf{no-reuse}} * (\mathbf{R} * \mathbf{F}_{\textbf{use}})$
  - » $\mathbf{C}_{\textbf{part-with-no-reuse}} = \mathbf{C}_{\textbf{no-reuse}} * (1 - \mathbf{R})$
- Total cost of reuse
  - » $\mathbf{C}_{\textbf{with-reuse}} = \mathbf{C}_{\textbf{part-with-reuse}} + \mathbf{C}_{\textbf{part-with-no-reuse}}$
  - » $\mathbf{C}_{\textbf{with-reuse}} = \mathbf{C}_{\textbf{no-reuse}} * (\mathbf{R} * \mathbf{F}_{\textbf{use}} + (1 - \mathbf{R}))$
    - *E.G. if* R = 50% and $\mathbf{F}_{\textbf{use}} = 0.2$ *then*

      $\mathbf{C}_{\textbf{with-reuse}} = 0.6 * \mathbf{C}_{\textbf{no-reuse}}$
  - » $\mathbf{C}_{\textbf{saved}} = \mathbf{C}_{\textbf{no-reuse}} - \mathbf{C}_{\textbf{with-reuse}}$
  - » $\mathbf{C}_{\textbf{saved}} = \mathbf{C}_{\textbf{no-reuse}} * \mathbf{R} * (1 - \mathbf{F}_{\textbf{use}})$
  - » $\mathbf{ROI}_{\textbf{saved}} = \mathbf{C}_{\textbf{saved}} \,/\, \mathbf{C}_{\textbf{no-reuse}} = \mathbf{R} * (1 - \mathbf{F}_{\textbf{use}})$
    - *E.G. if* R = 50% and $\mathbf{F}_{\textbf{use}} = 0.2$ *then* $\mathbf{ROI}_{\textbf{saved}} = 40\%$

Component system costs

- $\mathbf{F_{create}}$
  - » cost of developing and maintaining a reusable component system (usually $\mathbf{F_{create}} >> \mathbf{F_{use}}$)
- $\mathbf{C_{component\text{-}system}}$
  - » cost to develop a library of component systems for **R** percent
    - $\mathbf{R} * \mathbf{F_{create}} * \mathbf{C_{no\text{-}reuse}}$
    - each reusable component must be reused several times to be cost effective
    - $\mathbf{C_{family\text{-}saved}} = n * \mathbf{C_{saved}} - \mathbf{C_{component\text{-}system}}$
    - $\mathbf{C_{family\text{-}saved}} =$

      $$\mathbf{C_{no\text{-}reuse}} * (n * \mathbf{R} * (1 - \mathbf{F_{use}}) - \mathbf{R} * \mathbf{F_{create}})$$
  - » **ROI** = $\mathbf{C_{family\text{-}saved}} / \mathbf{C_{component\text{-}system}}$
    - E.G. $\mathbf{F_{use}} = 0.2$, $\mathbf{F_{create}} = 1.5$:: break even is n=2

Reuse vs. Classic projects and metrics

- Classic metrics are single project based
- Reuse business manages “multi-projects”
  - Component System Engineering
    - single projects
    - long term investment
    - generally more expensive than one-time
  - Application System Engineering
    - single projects
    - short term investment
    - generally less expensive than one-time
  - Application Family Engineering
    - continuous project, long term
  - Component System Support
    - continuous project, long term

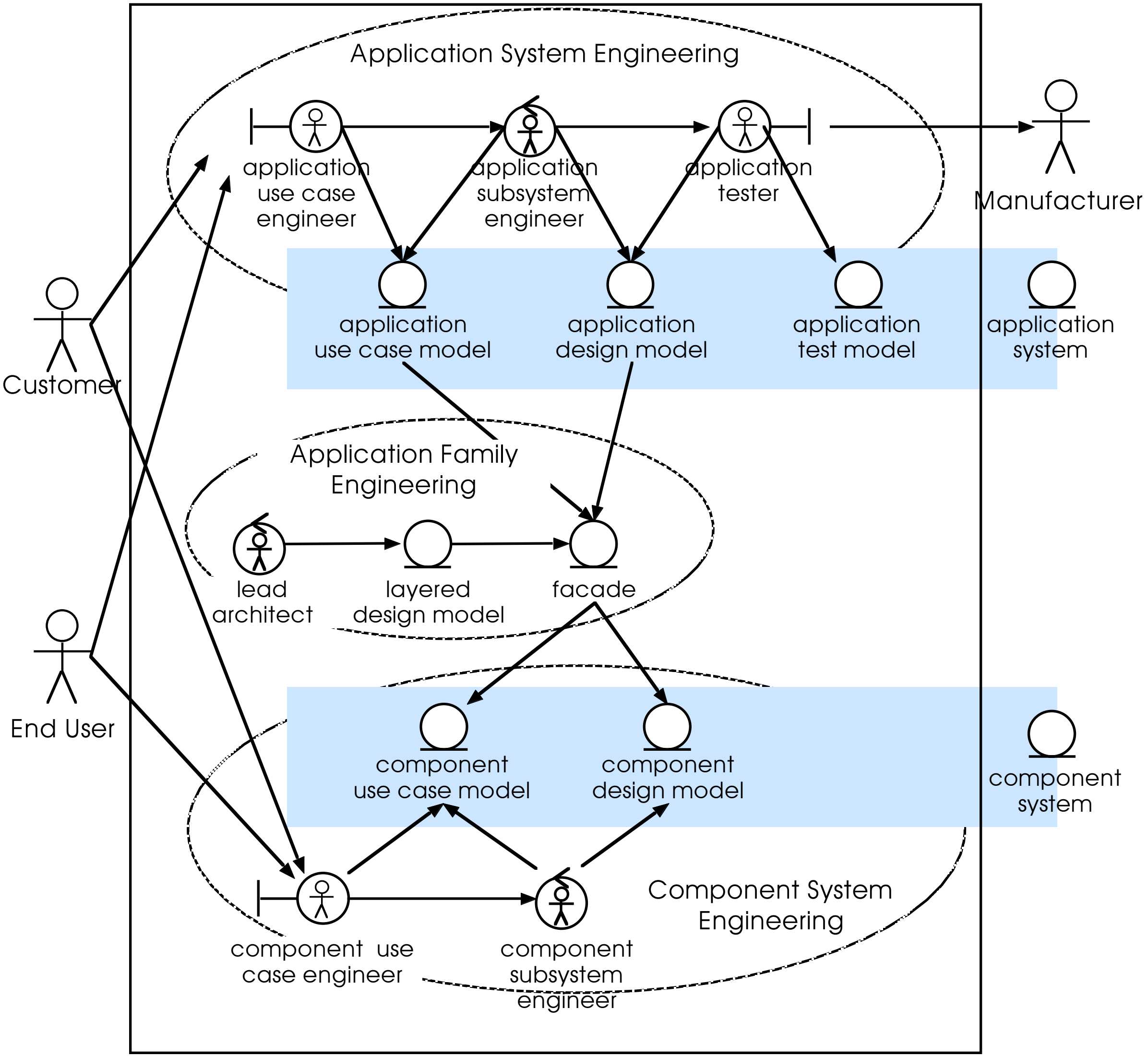
Continuous process improvement

- “Change is the name of the game”
  - » business goals
  - » priorities
  - » organization
  - » metrics
- Maturing processes change the ROI factors and opportunities
- Adapt the processes based on improvement opportunity
- Measure, monitor, evaluate and adapt
- Cycle . . . .

“People are our most important assets”

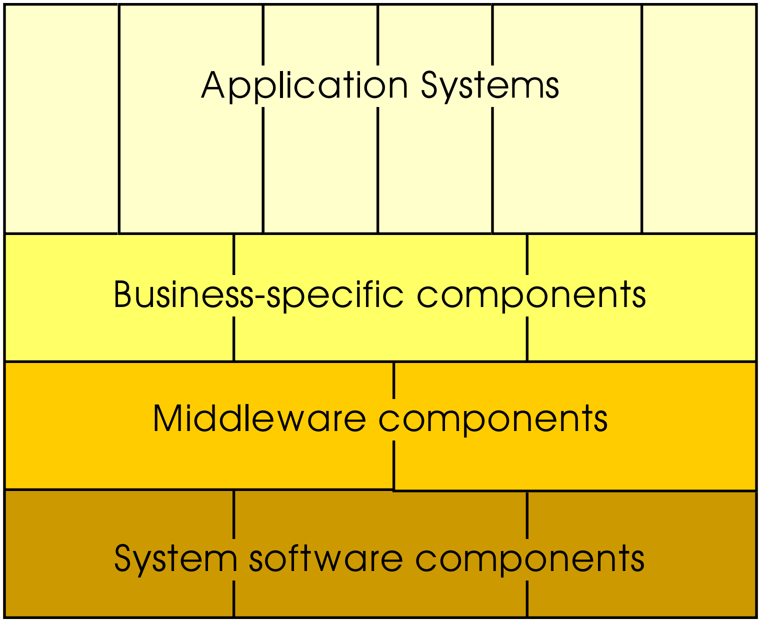
- Education, training and integration
  - » domain knowledge
  - » technology knowledge and skills
    - Application Family Engineering is more of a “family” business than you might think
- Applications and Components are “driven” differently
  - » AFE, CSE, ASE communication and delivery coordination are critical to efficiency
  - » In-house component quality perception is at least as important as out-of-house perception
- Reward professionalism and teamwork

Reuse Business


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 Driven Architecture

(system layering)


Application Systems
Business-specific components
Middleware components
System software components