

Component-Based Software Engineering

Software Engineering Lab

Johan Fredriksson
Mikael Åkerholm



What is CBSE

- Something entirely new and exciting?
- Component Based Development (CBD)
 - Reusable code
 - Strong interfaces
 - Well defined *interfaces* (contracts)
 - Modular systems
- It's what "good-style" programming, is all about!



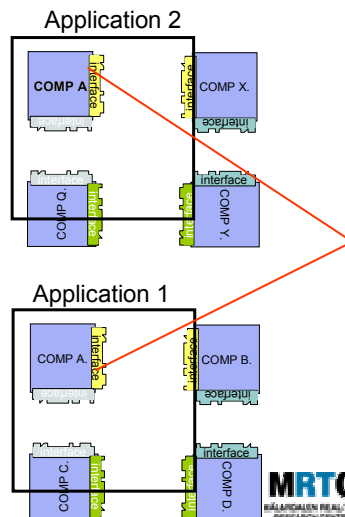
Component-based Software Engineering (CBSE)

- Basic motivation
 - Structure
 - Giving structure to system development
 - Reuse
 - Reuse of development effort
 - Maintenance
 - Making verification and maintenance more tractable.



Why is CBD better?

- Traditional programming:
 - Re-invented
 - Re-coded
 - Re-tested
- Component Based Development:
 - Focuses on reuse of
 - Subsystems
 - Infrastructure



Component definition

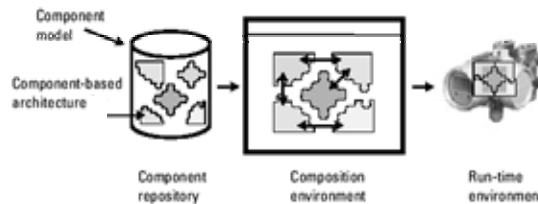
- Szyperski C.
 - "A component is a unit of composition with contractually specified interfaces and fully explicit context dependencies that can be deployed independently and is subject to third-party composition"
- D'Souza - Components
 - Reusable part of software, independently developed
 - May be adapted, but not modified
 - Composition of components into larger units
- Microsoft
 - "A piece of compiled software, offering a service"

What is a component

- Basically
 - Self-contained sub-system of a larger system
 - Reused in different contexts
 - Clear *interfaces*
- Separation of interface and implementation
 - Visibility only through the interface
 - Well understood interfaces

Component technology

- Model
- Architecture
- Repository
- Composition
- Run-time environment



Component technologies

- No standard
 - Many standards?
- Three widely used component technologies
 - EJB, JavaBeans - Sun Microsystems
 - .NET, COM/DCOM - Microsoft
 - CORBA, CCM - Object Management Group (OMG)
- Not suitable for embedded system

Issues for real-time/embedded systems

- Real-time systems must satisfy extra-functional constraints
- Important that extra-functional properties are predictable
- Embedded Components
 - Unit of composition with contractually specified interfaces
 - Attributes for extra functional properties
 - Fully explicit context dependencies
 - Deployed before runtime
 - Composed before runtime

Embedded Components

- Szyperski C.
 - Components should be delivered in binary form
 - Deployment and composition be performed during run-time
- Embedded
 - May be delivered in binary form (*introspection necessary?*)
 - Can often not be deployed or composed during run-time
- Szyperskis model seems suitable for non-real-time, and less resource-constraints applications

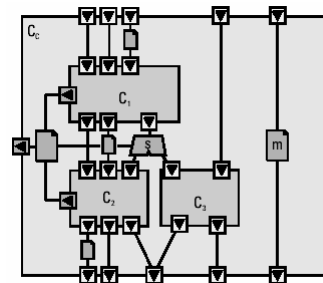
Component technologies for embedded systems

- | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> • IEC 61131-3 (IEEE) <ul style="list-style-type: none"> • Application is divided into a number of blocks • Several features referring to the underlying hardware • properties • Port based • No/Little support for extra-functional • Successfully used in industry (Bombardier) | <ul style="list-style-type: none"> • Rubus Component model (Arcticus) <ul style="list-style-type: none"> • Persistent state • Port based • Programming close to the OS • Components = Tasks • Implicit interfaces • Attributes for extra-functional properties • Successfully used in industry (Volvo CE) |
| <ul style="list-style-type: none"> • Koala (Philips) <ul style="list-style-type: none"> • Component hierarchy • Explicit interfaces • For customer electronics devices • Port based • No/Little support for extra-functional properties • Successfully used in industry (Philips) | <ul style="list-style-type: none"> • PECOS (IST) <ul style="list-style-type: none"> • Component hierarchy • Explicit interfaces • For field devices • Port based • Attributes for extra-functional properties • Research project |



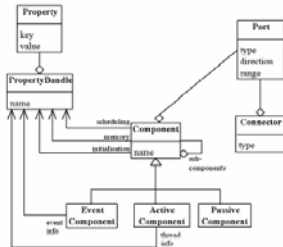
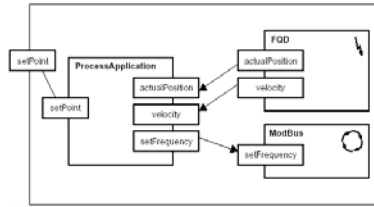
Koala

- Resource constrains environment
- Interfaces
 - Requires
 - Provides
 - Interfaces must fit
 - Diversity interfaces
- Component
 - Interact through explicit interfaces only
- Light-weight
- No support for extra-functional properties
- No support for analyzing real-time properties

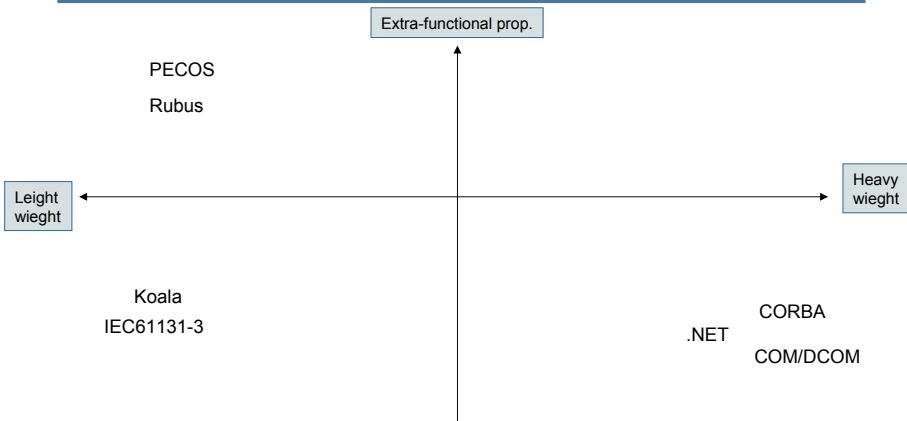


PECOS

- Resource constrains environment
- Interfaces
 - Ports
 - Each port has a name, direction & range
 - Connected with same type and compl. direction
- Component
 - Active, Passive and Event
- Light-weight
- Extra-functional properties
 - Worst case execution time
 - Cycle time
 - Etc.
- Synchronization and timing analysis



Component technologies



Our current research (SAVE)

- Safety-critical VEhicular systems
- SSF/IT programme involving
 - MdH, KTH, LiH, UU
- Industrial partners
 - ABB, Volvo, Bombardier, Saab, Scania
- Component technologies for safety-critical embedded systems

