

Anatomy of component technologies

Jean-Marie Favre
Equipe ADELE
Laboratoire LSR-IMAG
Université Grenoble I

Component definitions

Component definitions (1)

- « A component is a non trivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interfaces. »

✦ Philippe Krutchen, Rational Software

Component definitions (2)

- « A runtime software component is a dynamically bindable package of one or more programs managed as a unit and accessed through documented interfaces that can be discovered at runtime »

✦ Gartner Group

Component definitions (3)

- « A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to third-party composition »

✦ Clemens Szyperski, Component Software

Component definitions (4)

- « A business component represents the software implementation of an « autonomous » business concept or business process. It consists of the software artifacts necessary to express, implement and deploy the concepts as a reusable element of a larger business system »

✦ Wojtek Kozaczynski, SSA

Component definitions (5)

- "A Java Bean is a reusable software component that can be manipulated visually in a builder tool. ... First a bean must be capable of running inside a builder tool. Second, each bean must be usable at run-time within the generated application."

♦ Java Bean Specification, Sun Microsystem

Component definitions (6)

- « A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard »

♦ G. Heineman, W. T. Council,
« Component-based Software Engineering »
Addison Wesley, 2001

Component definitions (7)

- **Components :**
 - are reusable units
 - have clear and explicit boundary and dependencies
 - can be customized
 - can be assembled
 - are units of distribution
 - are units of substitution
 - are units of deployment
 - have certified properties

♦ Software Engineering Institute

Current software component technologies

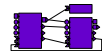
- In-house component technologies
 - ♦ e.g. Dassault Systèmes « Object Modeler »
- Standard component technologies
 - ♦ Microsoft' COM / DCOM / COM+ / .NET/C#
 - ♦ Sun' JavaBeans, Enterprise Java Beans (EJB)
 - ♦ OMG' Component Model (CCM)
 - ♦ OSGI
 - ♦ ...
- From internal use to external use

Difficult to study and compare

- Different applications domains
 - ♦ DS' OM : cad/cam
 - ♦ JavaBeans : graphical user interface
 - ♦ EJB, .NET, .CCM : internet-based applications, client-server
 - ♦ OSGI : set top boxes, small devices, ...
- Industrial solutions to solve day to day industrial problems
 - ♦ Technical solutions, No clear concepts
 - ♦ Complex (to very complex) technologies
 - ♦ Incremental evolution

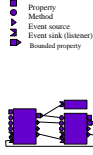
Anatomy of a (software) component technology

- A development process
 - ♦ actors, activities, business model, etc.
- A component model
 - ♦ defines the essential concepts and their relationships
- An implementation model
 - ♦ provides implementation techniques to implement components
 - ♦ coding standard
- A set of tools
 - ♦ assembly tool(s), deployment tool(s), code generator(s), wizards, ...
- A (set) of component infrastructure (s)
 - ♦ runtime infrastructure / assembly time infrastructure / testing time infra...
- Documentation
 - ♦ standard specifications, APIs description, tutorial, ...



Component model

- Define essential concepts
 - e.g. :
 - components
 - event sources
 - receptacles
 - facets
 - ...
- ... and their relationships ...
 - e.g. a component « implements » one or more interface, ...
 - component composition rules ...
 - protocol of interaction with the infrastructure, with the tools, ...
- ... all this at a conceptual level.

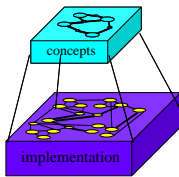


Component model vs. component language

- All component technology defines new concepts
- ... but usually no new language
 - (CCM and .NET/C# are exceptions)
- Defined in terms of underlying implementation patterns
 - No separation between important concepts and technical details
 - Sometimes defined only through obscure APIs and few samples (« how to ? »)
 - No stable definition
- Issues in
 - implementing components
 - using components
 - understanding and maintaining component-based software

Conceptual level vs. Implementation level

- No direct mapping
- A single concept -> many implementation entities



- different implementation strategies
- implementation spread in many different artefacts (e.g. source code, XML files, raw files, ...)
- use of low level mechanisms (e.g. macro directives)

Concept vs. Implementation

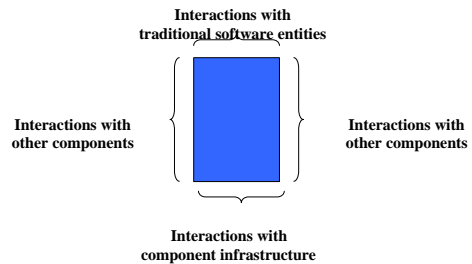
```

private void fireKeyListenerKeyTyped(java.awt.event.KeyEvent event) {
    keyListener.keyTyped(event);
}
// Add a VetoableChangeListener to the listener list.
@param 1 The listener to add.
public void addVetoableChangeListener(java.beans.VetoableChangeListener
    vetoableChangeListener, addVetoableChangeListener (i);
// Removes a VetoableChangeListener from the listener list.
@param 1 The listener to remove.
public void removeVetoableChangeListener(java.beans.VetoableChangeList
    vetoableChangeListener, removeVetoableChangeListener (i);
// Getter for property temperature.
@Structure Value of property temperature.
public int getTemperature() {
    return temperature;
}
// Setter for property temperature.
@param temperature New value of property temperature.
@throws PropertyVetoException
public void setTemperature(int temperature) throws java.beans.Property
    vetoableChangeListener.fireVetoableChange("temperature", new Integer
    this.temperature = temperature;
    propertySupport.firePropertyChange("temperature", new Integer (oldT
    
```

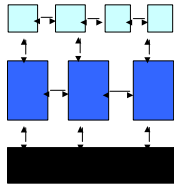
Different points of comparison

- Interface of a component
 - external view of a component (client's view)
- Composition of components
 - how component are assembled together ? (assembler's view)
- Implementation of a component
 - how component are implemented ? (implementer's view)
- Life cycle
 - Life Cycle
- Infrastructure and services
 - interaction between components and infrastructures

Graphical Conventions (1/2)



Graphical Conventions (2/2)



Traditional software entities

Components

Component Infrastructure

Example

