WHAT ARE COMPONENTS, ANYWAY?

Introduction to CBSE

Jaroslav Bauml,
June 1, 2009

CONTENT

- Motivation
- Definition(s)
- Features
- Examples – practical component programming
- Conclusion

MOTIVATION

Motivation comes from human qualities.

1. **laziness** – Why implement one functionality twice?
2. **cupidity** – Create it once, sell it N-times. N >> 1 ☺
3. programmers are **kittenish**.

MOTIVATION – MORE SERIOUS VIEW

Motivation comes from

1. **Real world analogy**,
   other engineering fields has deployable & integrable units...
   ...but all parables to real world fails in details.
2. **Software nature -** need for:
   - Composition - reusable software integration
   - Decomposition
   - Packaging & Transfer
   - Deployment
   - Standardization of sw products

DEFINITIONS - COMPONENT

Nomen est omen: *Components* are for composition.

Def 1 (bachman00):
- an opaque implementation of functionality
- third-party composition
- *conformant with a component model*

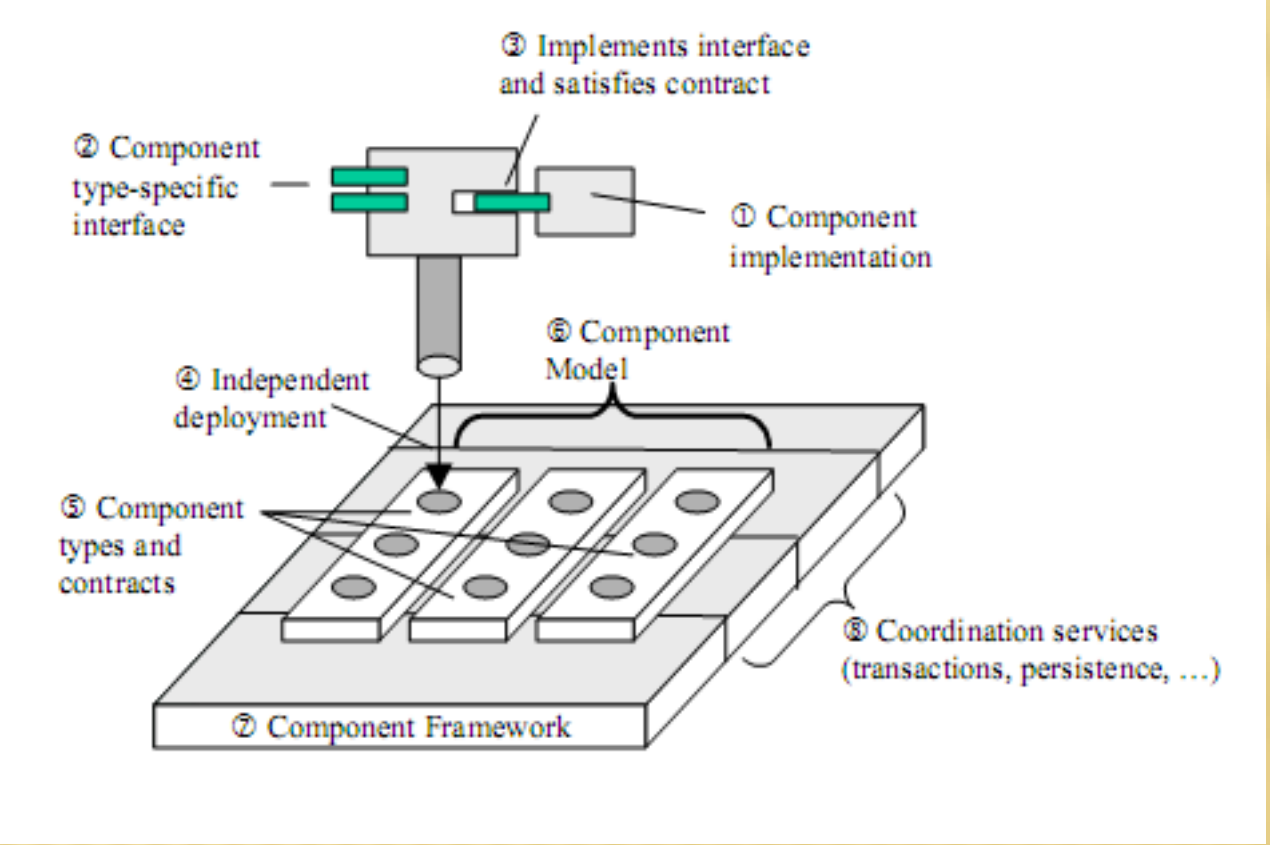
Def 2 (szyperki99):
- unit of independent deployment
- third-party composition
- no observable state

Def 3.

...

ELEMENTS OF COMPONENT SOFTWARE

- Component
- Component Interface
- Component Model
- Component Framework
- Component Types



COMPONENT MODEL

"A component model is the set of component types, their interfaces, and, additionally, a specification of the allowable patterns of interaction among component types. "

Defines:

- Component types
- Interfaces
- Interaction

COMPONENT FRAMEWORK

"A component framework provides a variety of *runtime services* to support and enforce the component model. In many respects component frameworks are like special-purpose operating systems, although they operate at much higher levels of abstraction. "

- Runtime enviroment
- Implements model
- May implement more than one model
- Could be componentized

INTERFACE & CONTRACT

Interface - IDL vs. language
Surface description of component
What component can do and what it needs

Contract - between two or more parties

- Parties often negotiate the details of a contract before becoming signatories.
- Contracts prescribe normative and measurable behaviors on all signatories.
- Contracts can not be changed unless the changes are agreed to by all signatories.

COMPOSITION & BINDING TIME

Components are for *composition*.

WHAT

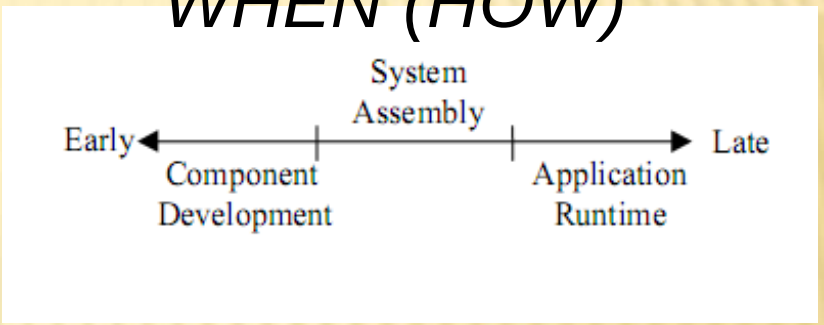
Classes of interaction

- Component–Component (C–C)
- Framework–Component (F–C)
- Framework–Framework (F–F)

Compositional Forms

- Component Deployment
- Framework Deployment
- Simple Composition
- Heterogeneous Composition
- Framework Extension
- Component (Sub)Assembly

WHEN (HOW)



"Interestingly, the design constraints needed to support late binding of component resources represent a kind of early binding of design decisions, in particular those that deal with how components coordinate their activities with each other. This early binding of design decisions is consistent with the overall “architecture first” philosophy of CBSE, and leads to systems with properties that are more readily analyzed and predicted prior to system assembly."

OSGI TECHNOLOGY

Component model is defined by OSGi Specification
Framework is an implementation of this model.
Two component types:

- **Bundles** – jar files with manifest
- fragments (degradated bundle)

Interface - Java language level

- Exp. & Imp. Packages
- Services - Java interfaces

Contracts

- Package level - Early
- Service level
  - Normal OSGi Service – Late
  - Declarative Services – Earlier

SPRING FRAMEWORK

Not a single-purpose technology - huge framework of best practices.
Component model is part of it – IoC framework
Components:

- **SpringBeans** – Java beans with defined rules **+++**

Interface - Java language level

Contracts
Beans wired together by IoC container in way described in xml descriptor

Time of binding - late

REFERENCES

Szyperski: Component Software, Beyond Object-Oriented Programming

Bachman: Volume II: Technical Concepts of Component-Based Software Engineering, 2nd Edition

OSGi Alliance: OSGi Specification 4.1