# Are components really black boxes?

Premek Brada

for DSS seminar
1st June 2009

# Agenda

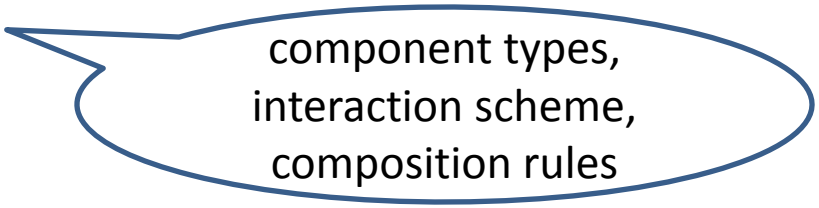
- Components and their properties
- The role of the black box
- How do we assess opacity?
- Case study 1: OSGi
- Case study 2: EJB
- Black Box vs Real Life

# Components and their properties

- Most cited definition (out of ~7 well known):

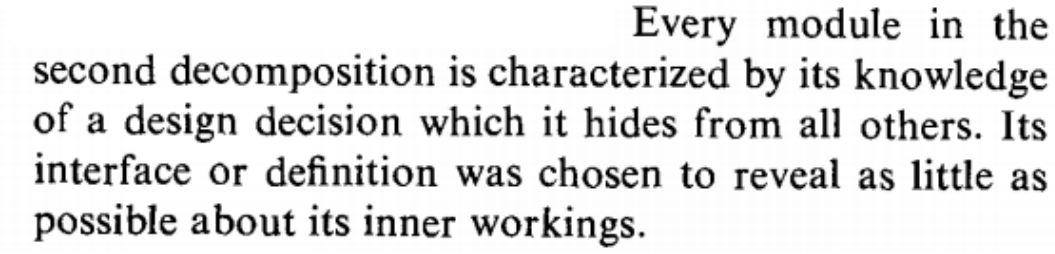
  *"A unit of composition with contractually specified interfaces and explicit context dependencies only [which] can be deployed independently and is subject to composition by third parties"*
  [Szyperski et al, 1999, 2002]

- General agreement:
  - opaqueness (black box, explicit specification)
  - compositionality
  - model conformance



# The role of the black box

- Remember David Parnas?


Every module in the second decomposition is characterized by its knowledge of a design decision which it hides from all others. Its interface or definition was chosen to reveal as little as possible about its inner workings.

» On the Criteria … 1972

modular programming -> programming to an interface
-> *software components:* information hiding is a must

- Goals and consequences
  - prevent property leaks
  - localize change effects
  - make software comprehensible
  - explicit specification needed

# How do we assess opacity?

- We need the border line, ie when to say “no” to claims “this is a component” from black-box point of view

- Explicit required role
- Completeness of specification
- Specification-implementation consistency
- Enforcement of black box by framework
- Ease of feature reconstruction
- Richness of contract types

# Case study 1: OSGi

- Explicit required role
- InCompleteness of specification
  - core: don't declare services… consequences below …
- Weak Specification-implementation consistency
  - package resolving only
- Moderate Enforcement of black box by framework
  - bind to declared packages and registered services only
  - class leaks from packages deprecated but easy
- Poor Ease of feature reconstruction
  - except for services in core model, and required features
- Low-level Richness of contract types
  - events implemented but not 1st class citizens

# Case study 2: EJB

- Explicit required role
- Moderate Completeness of specification
  - events for MDB, attributes for BMP
- Mixed Specification-implementation consistency
  - extremely poor for EJB 2.1
  - good for annotation style EJB 3
- Enforcement of black box by framework
- Moderated Ease of feature reconstruction
- Richness of contract types

# Black Box vs Real Life

- How much “little” should/can we know about the inner workings?
  - “Business” features
    - » interfaces, services, attributes, streams, …
  - Control features
    - » configuration
    - » lifecycle management
  - Implementation aspects
    - » logs produced, references passed


- Did framework authors really take notice of component research?