

# Reflektiiviset väliohjelmistot

Seminaari, syyslukukausi 2003  
Lea Kutvonen

## Sisältö

- ? Johdanto
- ? Väliohjelmistot
- ? Reflektiivisyys
- ? Reflektiivinen väliohjelmisto
- ? Seminaarin ohjelma

## Johdanto

### Trendejä

- › Hajautettuja / yhteentoimivia järjestelmiä
- › Sovellusten sopeuvuus erilaisiin ympäristöihin, tilanteisiin; räätälöitävyys
- › Sovellusten älykkyyys, havainnot käyttäjistä ja ympäristöstä vaikuttavat toimintaan

### Vaatimuksia

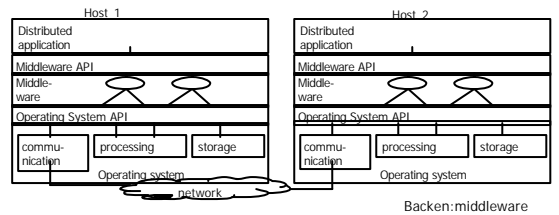
- › Ohjelmiston elinkaarikustannukset alas
- › Ohjelmistotuotannon vaativuus alas

### Keinoja

- › Väliohjelmistot korottavat abstraktiotasoa
- › Reflektiivisyys tukee erityispiirteiden tai muutosten soveltamista perussovellukseen tai toteutusympäristöön

## Defining middleware

Middleware is a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems. It is defined as a layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system.

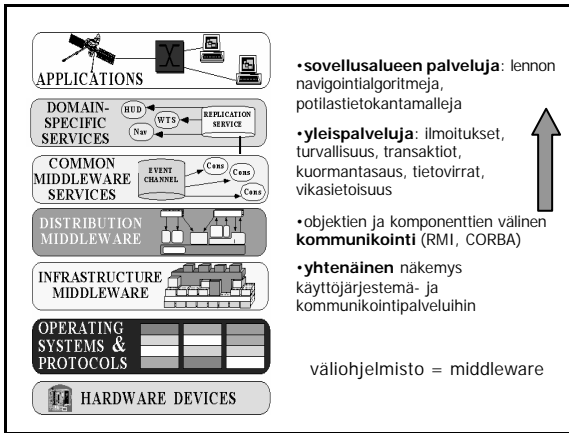


## Roles of middleware

- ? **Dual role**
  - Communication glue between partners
  - Bridge between programmer and processor
- ? **Dual tools**
  - Services in the runtime environment = infrastructure, platform
  - Software engineering tools
    - › Programming language concepts, like objects, components, RPC, transactions, operations/methods/invocations
    - › Descriptive language support: IDL, UML, WSDL, ebXML, ...
    - › Generative and reuse tools
    - › Software engineering processes, product lines, ...
  - In combinations: component frameworks, application servers, ...

## Types of middleware

- ? **Transactional middleware**
  - IBM CICS, BEA Tuxedo, Transarc Encina
  - communication, storage, limited facilities for processing
- ? **Message oriented middleware, MOM**
  - IBM MQSeries, Sun Java Message Queue
  - communication, no processing, limited storage
- ? **Procedural middleware**
  - RPC variations
  - communication, processing, no storage
- ? **Object and component middleware**
  - CORBA, DCOM, Java RMI, EJB, SOAP, .NET
  - communication, processing, and storage



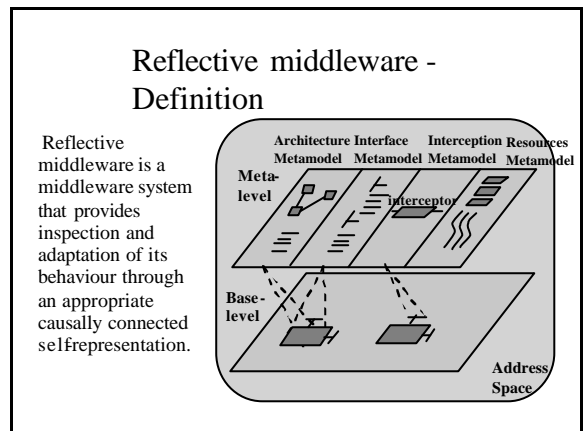
## Reflection - Definition

- 7 **Reflective system** is a system that is able to describe and manipulate itself or other related systems.
- 7 **Computational reflection** is the behaviour exhibited by a reflective system, where a reflective system is a computational system which is about itself in a causally connected way.
  - Maes 1997
- 7 **Metaprogramming** involves methods and tools for creating programs and methods for reading, writing and changing other programs.

- ## Methods of reflection
- 7 Foundations
  - 7 Metalogical reflection
  - 7 Managing self-references
  - 7 **Computational reflection**
    - Compile time: Shiba
      - 7 partial evaluation; static and dynamic
      - 7 runtime code generation; deferred dynamic compilation
    - Runtime: Smith, Maes
    - **Reflective programming languages with built-in metainterfaces**
      - Rewrite logic programming languages
  - 7 **Metaobject protocols and metaprogramming**
  - 7 **aspect-oriented programming**
  - 7 **open implementation style**

- ## Terminology
- Reify/kuvata – to explicitly describe the system behaviour and rules
  - Inspect/tutkia – to retrieve reified information
  - Reflect/päittää – to make decisions as a result of inspection
  - Adaptation/sopeutua – a change in system behaviour as a result of reflection
  - Causally connected self-representation – a change in the self-representation causes immediate change in behaviour
  - Metaobject reifies some aspects of an object
  - Metaobject protocol specifies how (protocol or interface) metaobject can be accessed

- ## Open implementation styles
- Boxes**
- black: generic solution, no user interference; heavy to run
  - white: user uses internal pieces at will; complex to use
  - open: common interface, user can change internals at will
- 
- Styles of open box**
- Procedural reflection – new code
  - Declarative reflection – select existing code by parameters
    - compilation time
    - runtime
  - Changes in interfaces & inspection



## Expectations on middleware

- ? **Middleware able to cope with change**
  - Applications are deployed in hostile and dynamic environments
    - Multimedia, Group communication, Realtime and embedded environment, Mobile and handheld computing
  - But there is a problem as conventional ORBs are static:
    - Fixed threading model, transport protocol: IIOP (over TCP/IP), security strategy (typically no security), scheduling
- ? **Middleware provides selfhealing systems**
  - Fault-tolerance, replication groups, ...
  - Self-monitoring by injected monitors; adaptation by strategy selectors reacting to feedback from monitors; alternative strategies implemented as separate components or aspects
- ? **Middleware provides for Context-aware applications**
  - Device awareness (memory, battery power, screen size, internal resources).
  - Environment awareness (bandwidth, network connection, location, services in reach, external resources).
  - User and application profiles
  - Service levels: best effort, ... guaranteed

## Solutions in reflective middleware

- ? **Metainformation management services in middleware provide meta-level data**
  - service trading, service types, system architectures
  - transparencies (location, migration, group, ...)
- ? **Base-level objects/components can be managed in a framework of components and connectors**
- ? **Middleware services provide functions and metadata repositories declarations for open box manipulation**

## Seminaarin ohjelma

- ? Reflektiivisyyden peruskäsitteet
- ? Metadatan hyväksikäyttö väliohjelmistoissa
- ? Perusmalleja
  - mobiliteetti, QoS, group management,
  - security
- ? Platformeja
- ? Arkkitehtuuripohjainen reflektiivisyys
- ? Yhteenvedoistunto - ryhmätyöskentelyä