Dynamic Adaptation and Deployment of Distributed Components in Hadas

Israel Ben-Shaul, Ophir Holder, and Boris Lavva
IEEE Trans on Software Engineering, Sep 2001
Presented by: Zhenxiao Yang

Outline

• Motivation
• Component Model
• Component Deployment & Configuration
• Evaluation
• References

Motivation

• We need distributed components
  – to share resources
    • Computing power
    – not affordable
    – not convenient
    – not possible
    • Human resources
    – components independently developed and maintained
    • Other resources
    – power supply
    – peripheral devices
Motivation – cont’d

• We need distributed components to
  – easily access services and resources
    • we need to access multiple resources
    – anytime, anywhere
    – during business travel: on a plane
    – when hiking, or backpacking: in the wild
    • sometimes, it’s impossible to carry all of them around
    • distributed component technology allows us to easily
      access resources remotely
      – only keep simple status locally

Motivation – cont’d

• We need dynamism in distributed components because
  – Internet-based wide-area computing is different from traditional distributed computing
    • large deployment space:
      – diversity of components
      – may be autonomously operated and maintained

Two Key Elements of Comp-Tech

• Component model
  – Basis for distributed component technologies
    • What does a component look like
    • How to construct such a component

• Component deployment & Configuration
  – deployment: what to do after we already have a component
    • distribute them over the network
  – configuration: what to do after a component is already deployed
    • configure it, start it, make it available
Two Key Elements – cont’d

- Recent trends....
  - Recent development in
    - autonomous computing
    - self-healing systems
    - adaptive software
  - Requires distributed components to be
    - dynamically deployed and redeployed
    - dynamically configured and reconfigured

Where Are We

- Motivation
- Component Model
- Component Deployment & Configuration
- Evaluation
- References

CORBA Component Model

- Component model designed for
  - interoperability
    - across platform
    - across language
- Component is
  - a meta-type in CORBA
  - an extension to the object meta-type
  - specified in IDL
  - represented in the interface repository
  - denoted by a component reference
- Component definition is
  - specialization and extension of interface definition
  - completely hidden from users
**DCOM Component Model**

- DCOM component object model
  - is designed for interoperability
  - cross language only
  - defines a binary interface between client and object, using Microsoft IDL
  - uses three ways to access servers
    - in-process, local calls
    - local-object proxy, lightweight RPC
    - remote object proxy, DCE RPC, supported by DCOM
  - each component has a globally unique id(GUID)
  - 128bit
  - globally unique

**Java/RMI Component Model**

- Designed for interoperability
  - cross-platform, not cross-language
- A Java/RMI component is
  - a regular java object extending the Remote interface
  - implemented in Java
  - located through RMI naming service

**Hadas Component Model**

- The mutable component model
  - design for (future) change
    - platform-specific, language-specific
  - how to change
    - Instance level
    - Class level
    - Hybrid approach
      - fixed and extensible
      - fixed part: shared by all instances
      - extensible part: instance-specific
  - Meta methods and meta-invocation
    - check for meta-methods before method invocation
Hadas Component Model – cont’d

• The fixed and the extensible

```java
public class Component {
    private Hashtable<String, String> fixed;
    private Hashtable<String, String> extensible;
    public Component() {
        // Constructor implementation
    }
}
```

Hash Table (fixed state - Hadas Data Item (CMS.createId()));
initialize and initialize, add and delete, delete Database
HashBehavior.put(getDatabase(), new getDatabase());
initialize and initialize, add and delete, deleteDatabase
HashBehavior.put(getDatabase(), new getDatabase());
```java
public final void baz(int arg1) throws String, methods, Exception, Exception{
    // hadas implementation of baz method serialization
}
```

Hadas Component Model – cont’d

• Meta-methods and meta-invocation

```java
public int op1(int arg1) {
    return arg1;
}
```

- metaTable.hasKey(arg1)
  - lookup the method
    - component-specific invocation
      - get and using foo loop and method(s)

Where Are We

• Motivation
• Component Model
• Component Deployment & Configuration
• Evaluation
• References
**CORBA Component Deployment and Configuration**

- Server: object manager
  - creation
  - manage
  - removal
- Client: CORBA name server
  - declare and bind to name server
  - get a reference to remote object
  - invoke a method

CORBA Component Deployment Architecture [4]

---

**DCOM Component Deployment and Configuration**

- A component registers itself on server side
- DCOM creates stub in server process space
- DCOM creates proxy in client process space
- Identified by GUIDs
  - each GUID corresponds to a record in the registry
    - server address
    - version
    - other fields

---

**JAVA/RMI Component Deployment and Configuration**

- Component server
  - bind a component
- Client
  - create a stub of the component
  - call remote methods
- Reconfiguration
  - recompile the component and stubs
Hadas Component Deployment and Configuration

- Hadas' universe
  - A collection of sites
  - Naming
    - Hadas://sys-techinon.ac.il/siteB.Home.ComB1
  - Ambassador
    - similar to Java in CORBA and RMI
    - dynamically deployed
    - able to evolve

Hadas Component Deployment and Configuration – cont’d

- Hadas configuration model
  - Step 1: site ambassador deployment
    - connection protocol
  - Step 2: component ambassador deployment
    - import protocol
  - Step 3: ambassador evolution
    - update protocol

Hadas Component Deployment and Configuration – cont’d

- Step 1: site ambassador deployment
  - Connect protocol
    - is performed between a pair of site objects
    - is issued from client site (A) to server site (B)
    - is used in the dynamic arrival and deployment of Amb
      - created at site B by traversing all components
      - only contains component place holders
**Hadas Component Deployment and Configuration – cont’d**

- **Step 2: component ambassador deployment**
  - Import protocol
  - Put component ambassadors into place holders

---

**Hadas Component Deployment and Configuration – cont’d**

- **Step 3: ambassador evolution**
  - Update protocol
    - Initiated by components
    - Applied to deployed ambassadors
    - Transparent to host site
    - Uses the observer design pattern

---

**Evaluation**

- **Strengths**
  - Designed for change
    - Component model
    - Dynamic deployment and configuration

- **Weaknesses**
  - Scalability
    - Ambassadors are heavyweight
  - Popularity
    - The power of distributed component middleware comes from the wide usage
References


CORBA Architecture

A Request Being Sent Through ORB [3]

CORBA Architecture – cont’d