Unifying Agent and Component Concepts
- Jadex Active Components

Alexander Pokahr, Lars Braubach, Kai Jander

Distributed Systems and Information Systems Group (VSIS)
Department of Informatics, University of Hamburg, Germany
{pokahr, braubach}@informatik.uni-hamburg.de
Overview

- Motivation for Active Components
- Active Component Concepts
- Active Components Platform & Example
- Summary and Outlook
Requirements

• Building distributed systems is hard due to several inherent characteristics such as
  ▪ Message communication
  ▪ Concurrency
  ▪ Non-functional aspects like scalability, fault tolerance

• Current technology trends further increase the demand for novel software technical concepts
  ▪ Increased hardware concurrency
  ▪ Delegation of tasks to computers
  ▪ ...

• We want the software paradigm to cope with most of the complexities
  ▪ Exhibit different kinds of entity behavior
  ▪ Having rich interaction styles
  ▪ Can act on their own
  ▪ Support non-functional characteristics
Paradigm Comparison (1)

Software agents:
- Autonomous, reactive, proactive, social, (mentalistic notions)
- Communication based on speech acts/protocols

Active objects (coming from actors):
- Objects that decouple caller from callee
- Method-based interaction

Software components:
- Passive but composable entities managed by an infrastructure
- Communication message and call-based
Paradigm Comparison (2)

Structure
- Support software engineering principles (hierarchical, modularization)
- Exhibit different kinds of entity behavior (internal architecture)

Interaction:
- Having rich interaction styles (message based, method call)

Execution:
- Can act on their own (autonomy)
- Support non-functional characteristics (management infrastructure)

<table>
<thead>
<tr>
<th></th>
<th>structure</th>
<th>interaction</th>
<th>execution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>hierarchical</td>
<td>internal architecture</td>
<td>message based</td>
</tr>
<tr>
<td>agents</td>
<td>partially</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>active objects</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>components</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Overview

• Motivation for Active Components

• Active Component Concepts

• Active Components Platform & Example

• Summary and Outlook
Active Component Foundations

Definition: “An active component is an autonomous, managed and possibly hierarchical software entity that is capable of interacting with other active components in different modes including message passing and method calls.”

- Invocation styles like agents and active objects
- Rich behavior styles like agent architectures or workflows
- Management infrastructure and composability of components
Invocation Style: Shop Example

Scenario
- Shops have an inventory and offer items for certain prices
- Customers can search stores and buy items in them

System design
- Shops define an interface IShop that allows customers to get the catalog of offered items and buy them
- Customers search for IShop providers and use the interface to issue buy orders
- The call is decoupled at interface level and executed asynchronously in the callee (external access)
Shop Example Interface

```java
public interface IShop {
    public String getName();
    public IFuture buyItem(String item, double price);
    public IFuture getCatalog();
}
```

- Simple shop interface offers methods for getting the shop name, buying an item and getting the catalog
- `getName()` is allowed as it is considered as constant, i.e. the value will be cached
- `IFuture` represents a value that is immediately returned but may provide the result in future
Behavior Style: Internal Architecture

- Many different internal agent architectures exist
  - reactive (e.g. Subsumption, Tasks)
  - deliberative (e.g. IRMA)
  - hybrid (e.g. BDI)

- No one-fits-all solution
  - Simple architectures for simple problems
  - Complex architectures for complex problems

- Active components widen the spectrum
  - Agent architectures: BDI, Micro, (Task)
  - Workflow engines: BPMN, GPMN
  - Applications (management of components and non-component functionality)
Composability: Component Hierarchies

• Active components are contained in a tree-like structure, whereby the platform component is the root

• Typically this hierarchy reflects the creation relationships
  • A component can start another component as its subcomponent
  • The hierarchy does not enforce any policy on components

• Component hierarchy is helpful for
  • Applying commands on subtrees of components (e.g. terminating, suspending)
  • Including remote platforms using proxy components

Legend
A: Application
B: BDI agent
M: Micro agent
P: Proxy agent
→ children
↔ virtual children
Overview

• Motivation for Active Components

• Active Component Concepts

• Active Components Platform & Example

• Summary and Outlook
Jadex V2 Architecture Goals & Concepts

Design Goals:
- Platform can execute different kinds of components
- Component kernels should be enabled to run on different platforms
- Applications should be platform independent
- Applications should be composable from arbitrary component types (heterogeneous applications)

Platform: “A platform is the management infrastructure for components, which is responsible for their execution as well as for providing administration capabilities like a messaging system or a component service registry.”

Kernel: “A kernel encapsulates the internal behavior definition of a specific active component type.”

![Diagram](image-url)
GPMN Workflows

- GPMN = Goal Process Modeling Notation
- Is based on research of Daimler AG and currently topic of a technology transfer project between the University of Hamburg and Daimler AG
- Research focus is the development of flexible and agile workflow concepts by employing agent technology concepts
- GPMN workflows are executed via model transformation as normal BDI agents
- GPMN workflows can be modeled via an eclipse-based editor similar to the BPMN version
- Behavior concepts
  - GPMN workflows consist of a goal hierarchy
  - The leaves of this hierarchy are concrete BPMN plans
GPMN Workflow Example
Modeling and Runtime Tools

BPMN Modeller

Debugger

GPMN Modeller

Com Analyzer
Overview

• Motivation for Active Components

• Active Component Concepts

• Active Components Platform & Example

• Summary and Outlook
Summary and Outlook

Active components as step beyond purely message-based agents aiming at easier pragmatic application construction
- Simplify agent concepts
- Enrich them with further concepts from active objects, and software components

Active components
- Support different invocation styles
- May be realized using different internal architectures
- Are hierarchical and managed entities

Jadex AC Platform with modeling and runtime tools: http://jadex-agents.informatik.uni-hamburg.de

Future work: embracing concepts from the services area (SOA)
Thank You!

{pokahr, braubach}@informatik.uni-hamburg.de