

# 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)

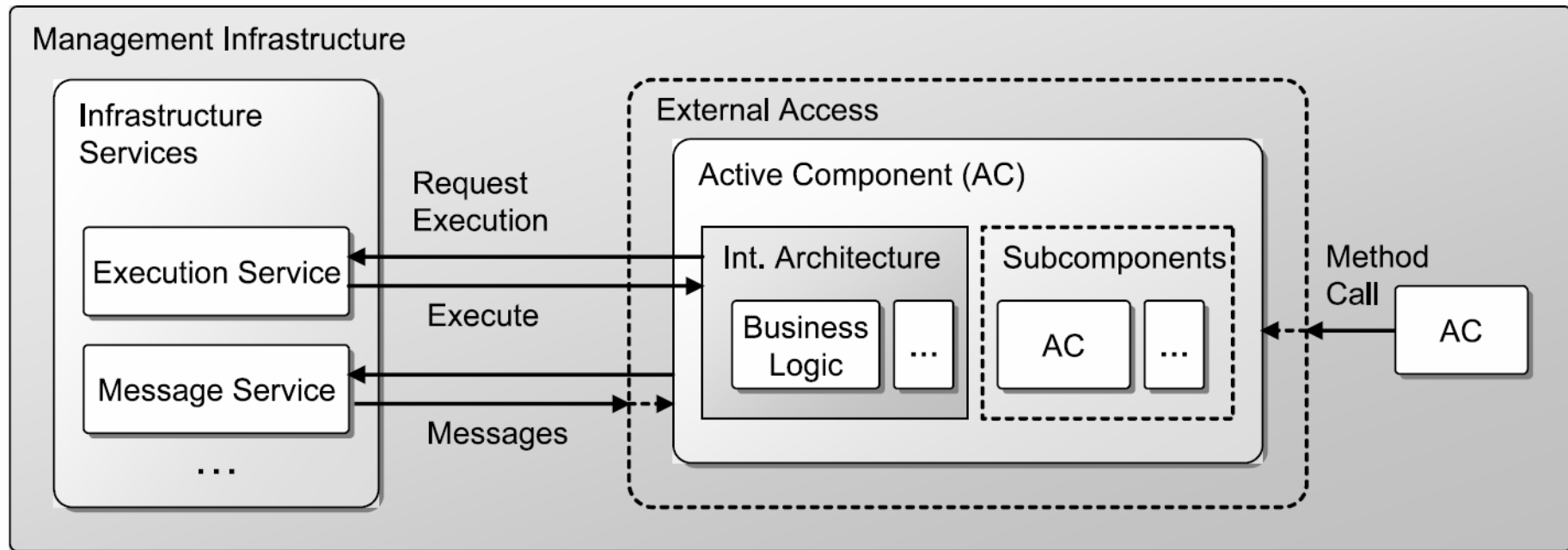
	structure		interaction		execution	
	hierarchical	internal architecture	message based	method call	autonomous	managed
agents	partially	yes	yes	no	yes	partially
active objects	no	no	no	yes	yes	no
components	yes	no	yes	yes	no	yes

# Overview

---

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

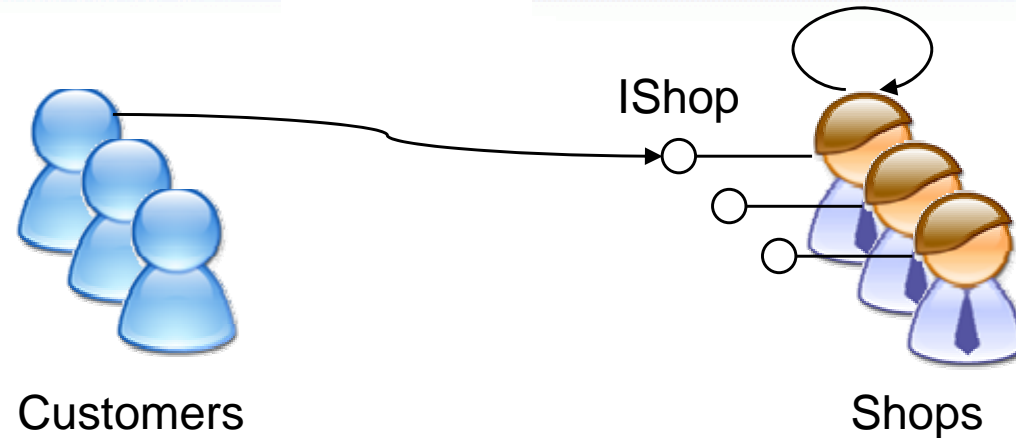
# Active Component Foundations



*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

---

```
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

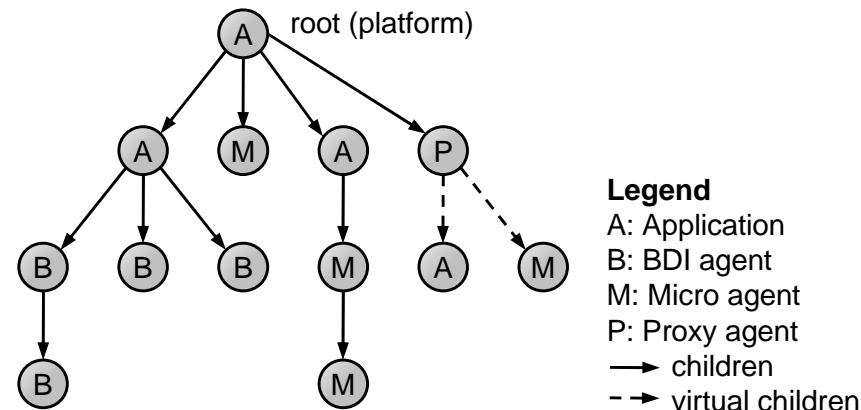
# Behavior Style: Internal Architecture

---

- 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



# Overview

---

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

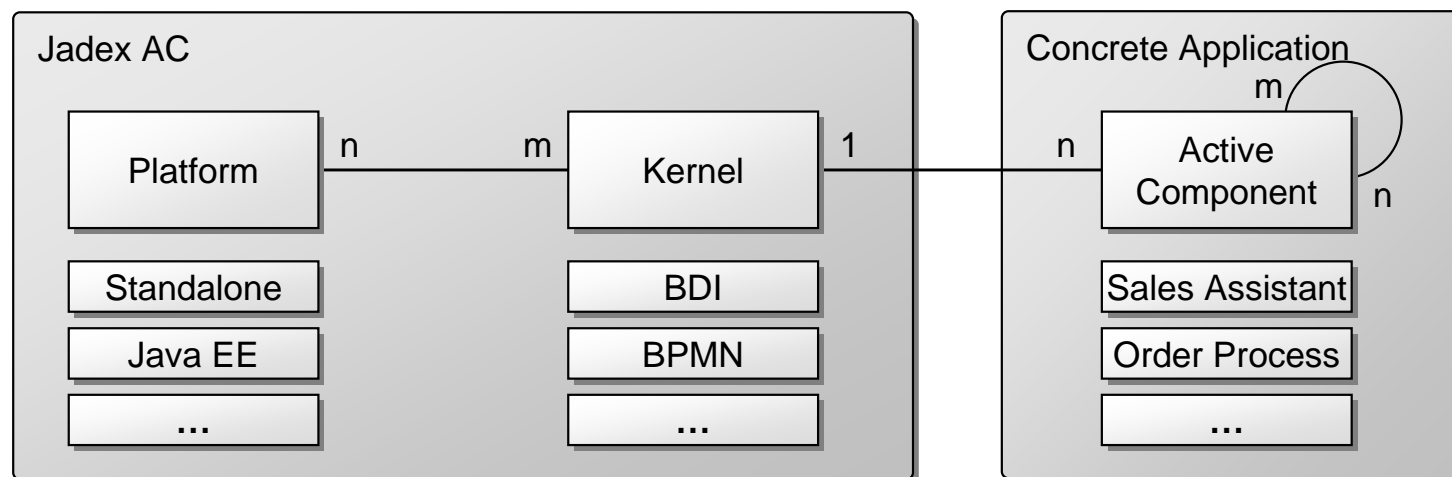
# Jadex V2 Architecture Goals & Concepts

## 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.”*

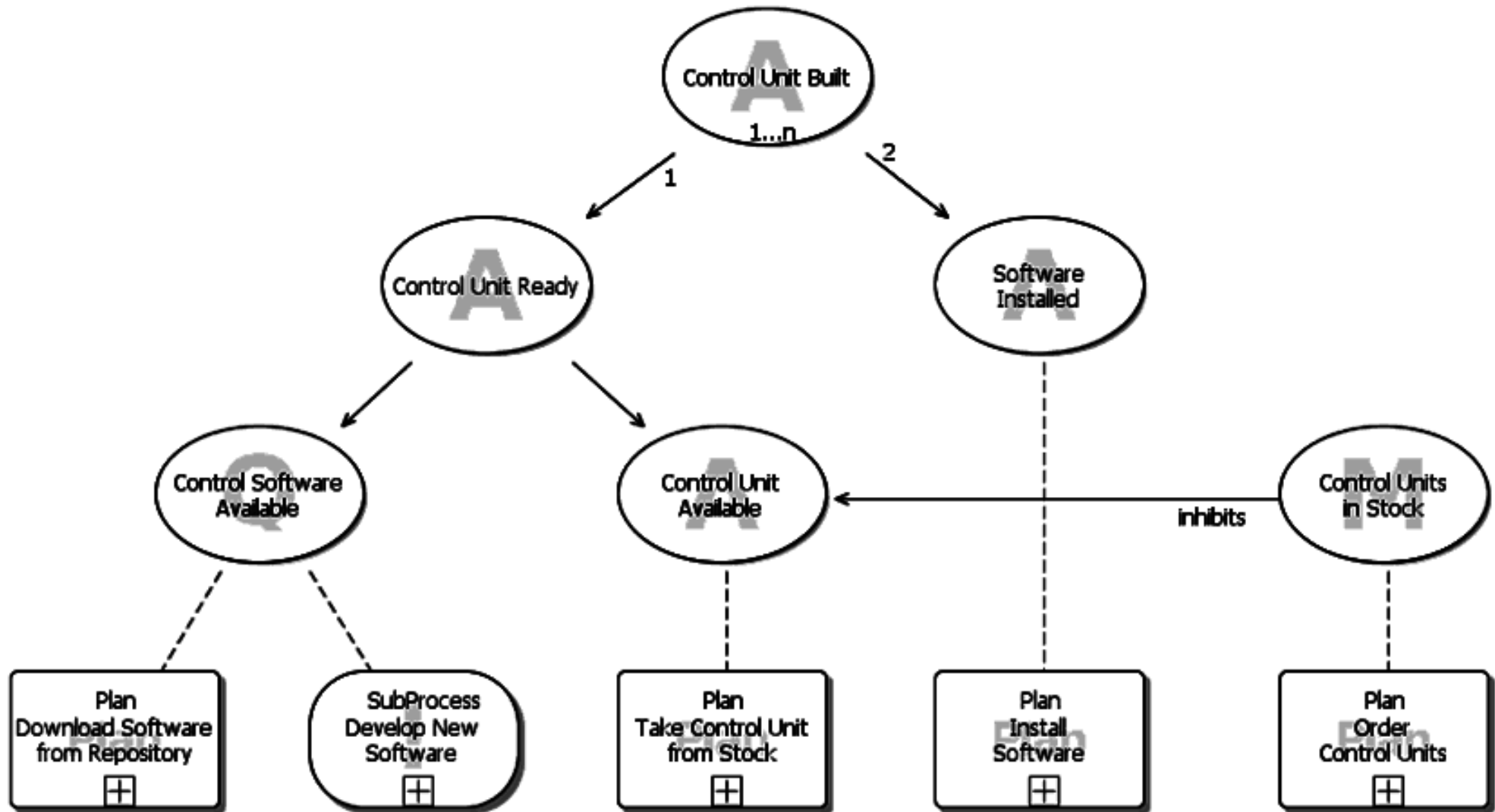


# GPMN Workflows

---

- 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

The screenshot shows the Eclipse IDE with two diagrams open. The top diagram is a BPMN diagram titled "E1\_CustomObjects.bpmn\_diagram" showing a process flow: "Input Customer Data" (Task) → "Create Customer Object" (Task) → a decision gateway labeled "Risk Taking" → "Sell High Risk Contract" (Task). A callout bubble labeled "BPMN Modeller" points to this diagram. The bottom diagram is a GPMN diagram titled "CleanerManufacturing.gpmn\_diagram" showing a goal hierarchy: "Control Unit Built 1..n" (Goal) is connected to "Control Unit Ready" (Goal) and "Software Installed" (Goal). A callout bubble labeled "GPMN Modeller" points to this diagram. The IDE interface includes a menu bar, toolbars, a palette, and a console window at the bottom.

The screenshot shows the Jadex Control Center 2.0-rc2 interface. The top panel is the "Process Inspector" showing a list of processes and a table of process details. The middle panel is the "History" table. The bottom panel is the "Com Analyzer" showing a sequence diagram of interactions between agents.

**Process Inspector Table:**

Process-Id	Activity	Pool	Lane	Exc...	Data	Status
ProcessThread_100	Task 2a II	Pool			{}	ready
ProcessThread_99	Task 1b	Pool			{}	ready
ProcessThread_98	Gateway 2	Pool			{}	waiting

**History Table:**

Step	Process-Id	Activity	Pool	Lane
0	ProcessT...	Task 1a	Pool	
1	ProcessT	Gateway 1	Pool	

**Com Analyzer Diagram:** Shows interactions between agents: Carry 3@lars, Carry 4@lars, Carry 5@lars, Produce r1@lars, and Sentry 0@lars. Messages include "inform (SpaceObject(id=6, type=target,...))", "inform (SpaceObject(id=4, type=target,...))", "inform (SpaceObject(id=1, type=target,...))", "request (RequestProduction(SpaceObject...", and "inform (SpaceObject(id=3, type=target,...))".



# 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

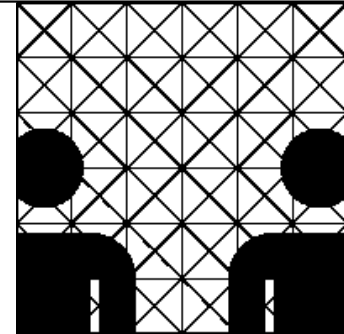
Active components

- 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