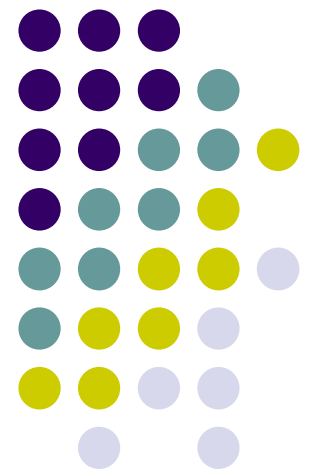
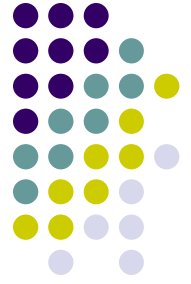


# Components in Java

---

Sebastian Pavel  
OBASCO Project – EMN/INRIA





# Plan

- **ArchJava**
- **Dynamic Configuration in ArchJava**
- Asynchronous Communication Experiments
- Symbolic Transition Systems for Components



# ArchJava

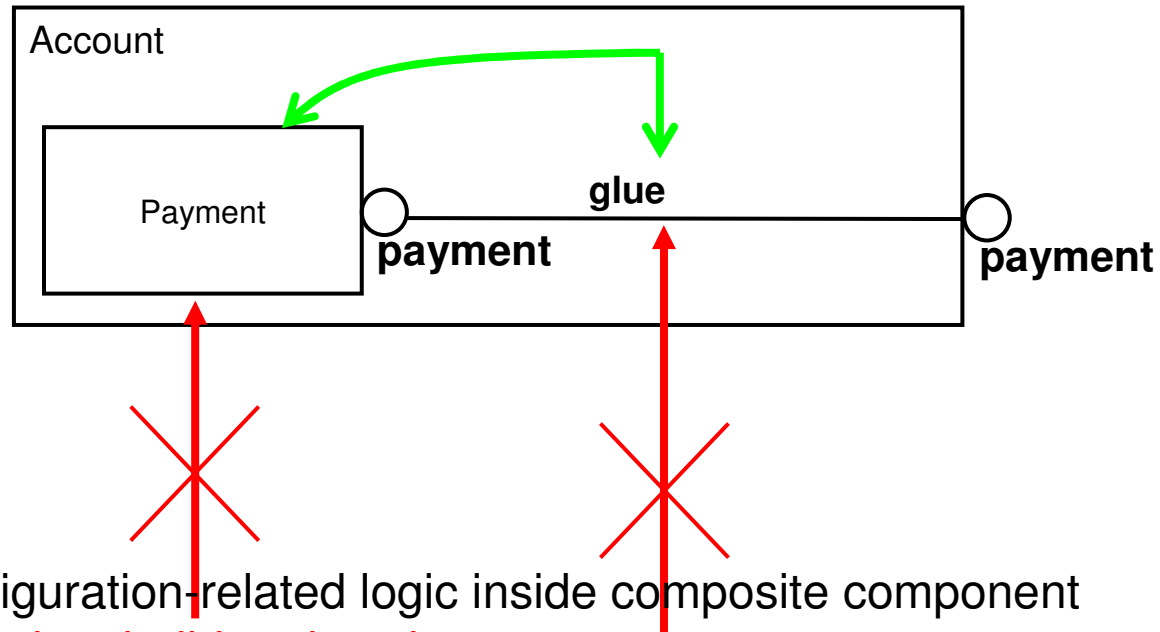
- Specifies architecture within Java code
  - **Small extension** to express hierarchical components and connections
- Verify control flow preserving **communication integrity** consistency property
  - Code and architecture evolve together
- Supports **dynamically changing architectures**
- Allows **common implementation techniques**



# Dynamic Configuration

- Context SPL (Software Product Lines)
  - Families of Software Products
- Communication Integrity
  - + Good architectural property
  - + No hidden communication channel
  - + Implementation conforms to architecture
  - No component classes in ports
  - Black-box components

# Communication integrity restrictions

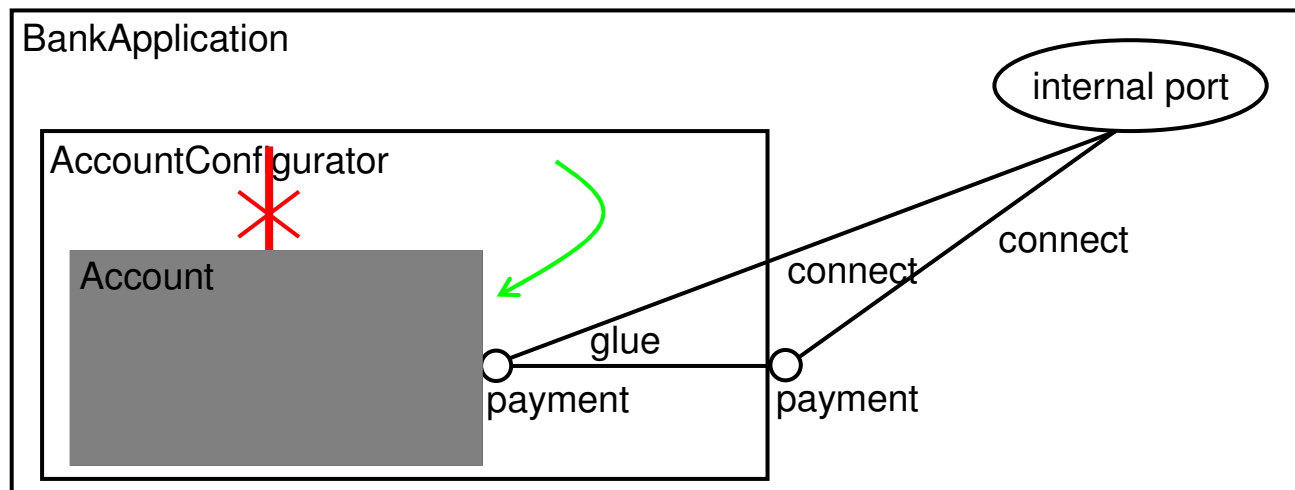


- Configuration-related logic inside composite component
- **Difficult to build and evolve**



# Configuration Pattern

- Configuration knowledge out of composite components



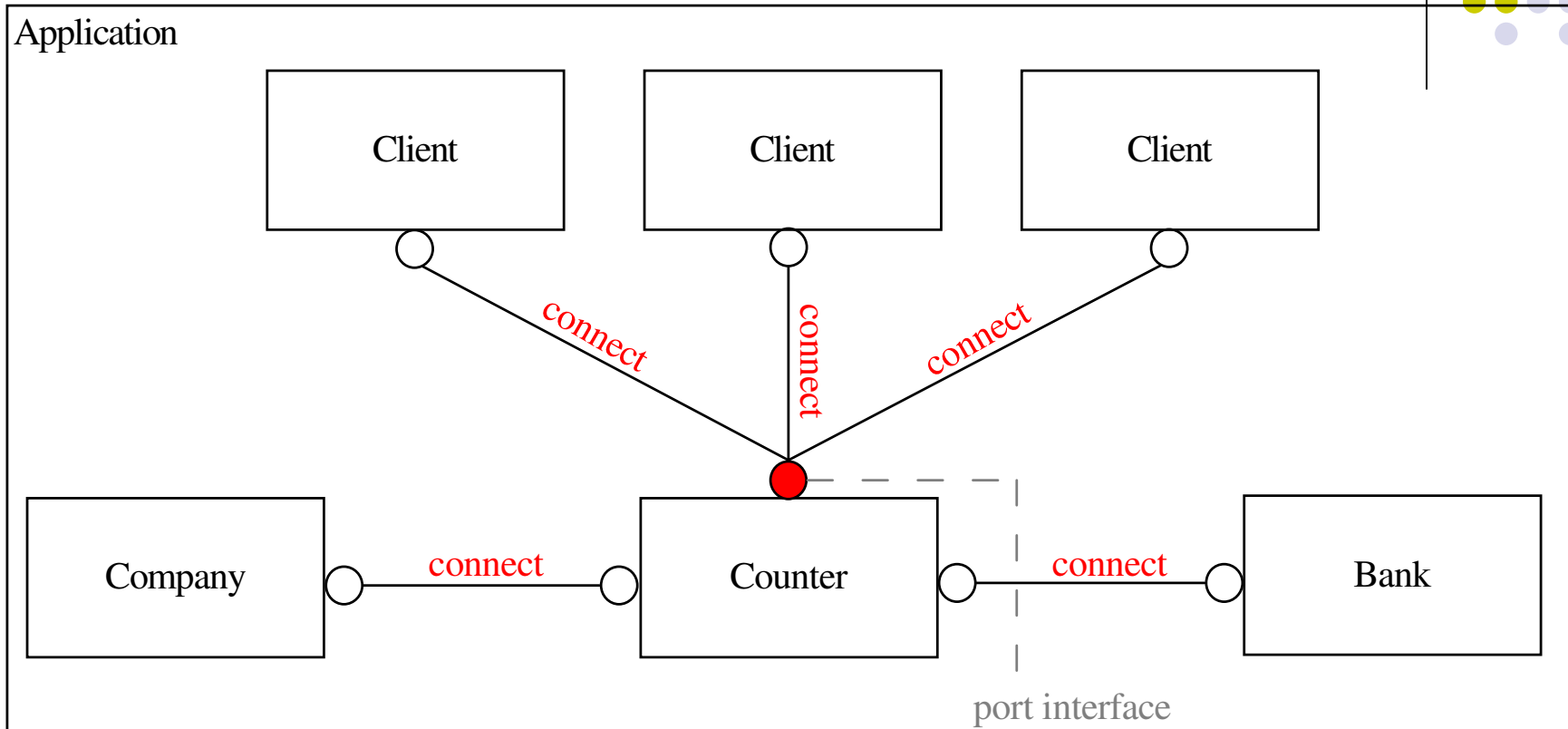
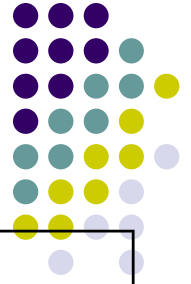
- AccountConfigurator = dynamic generator + placeholder



# Plan

- ArchJava
- Dynamic Configuration in ArchJava
- **Asynchronous Communication Experiments**
- Symbolic Transition Systems for Components

# Ticket reservation system

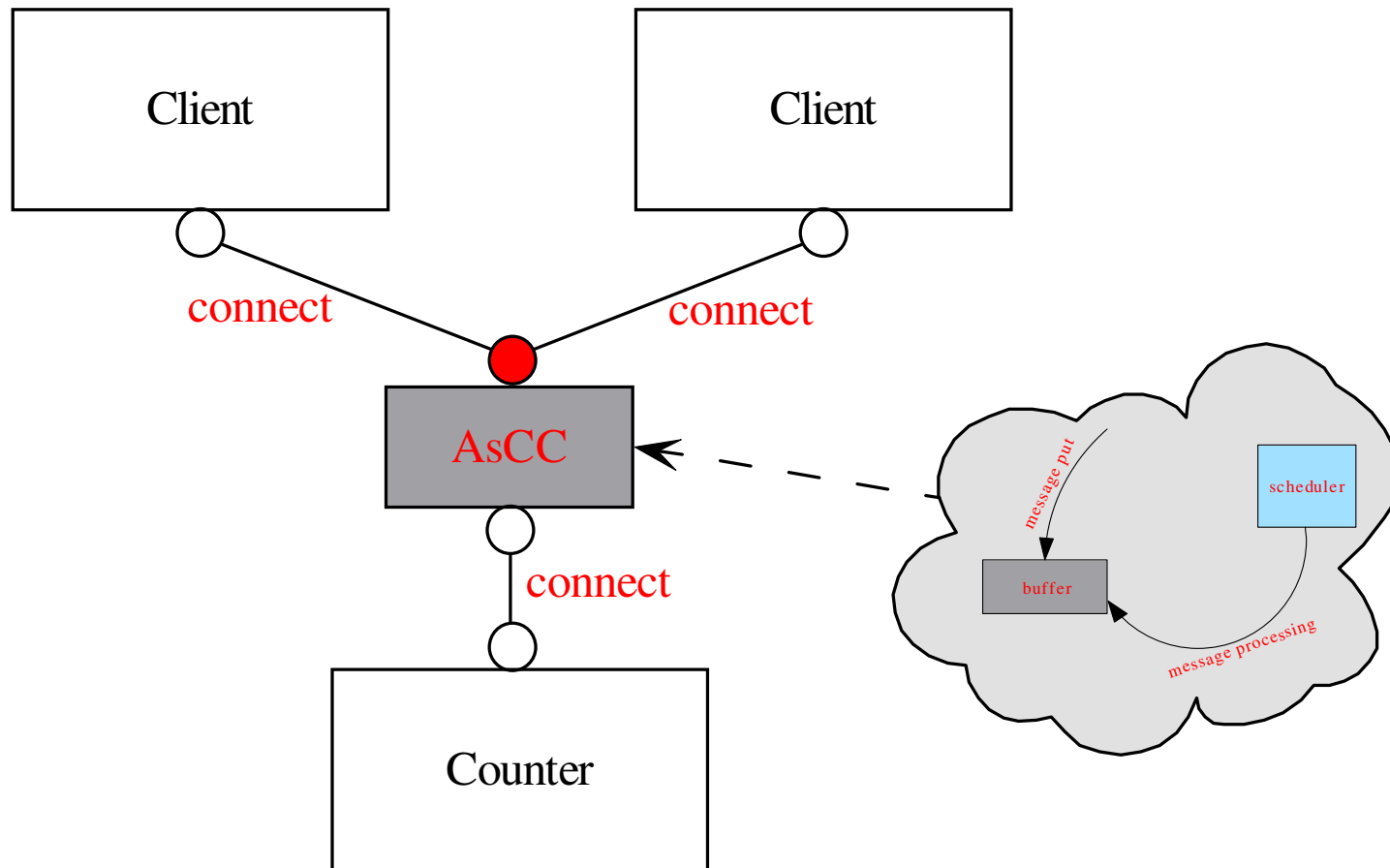


## Implementation Solutions:

- buffer inside receiver
- connectors
- AsCC (**A**synchronous **C**ommunication **C**omponent)

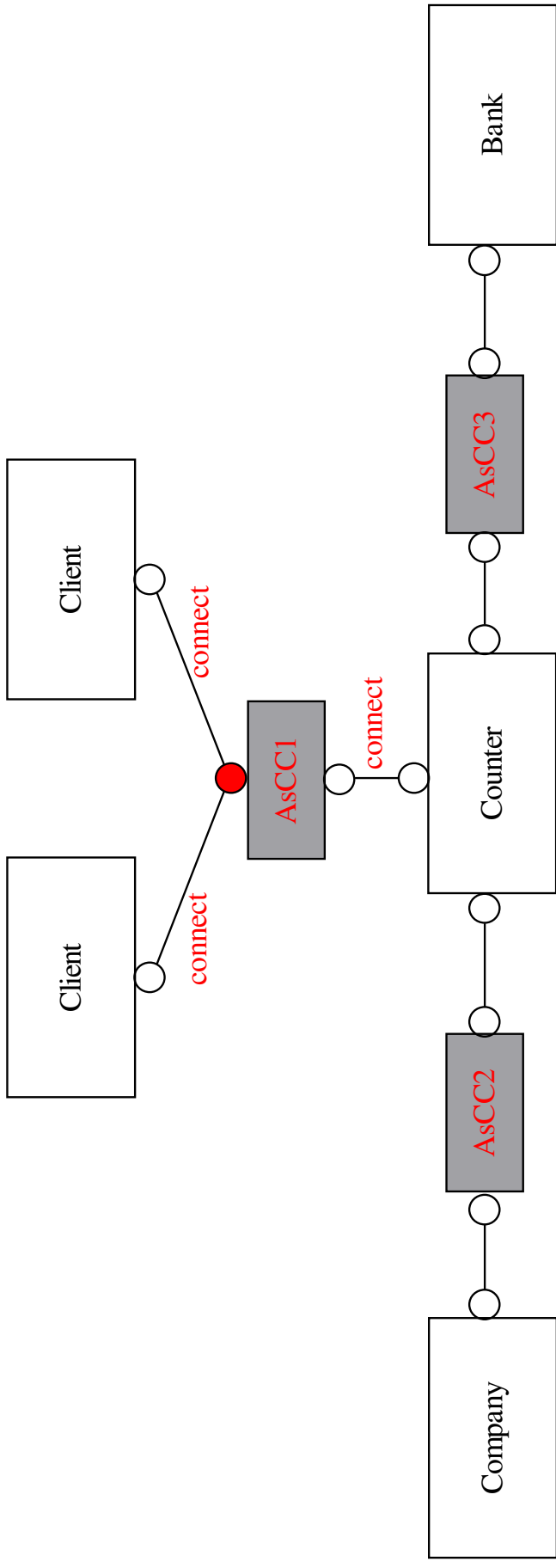
© 2004-2006, September 2004

# AsCC - Asynchronous Communication Component



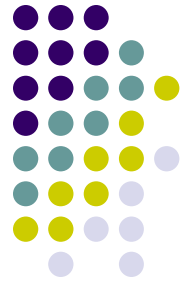


# General Overview

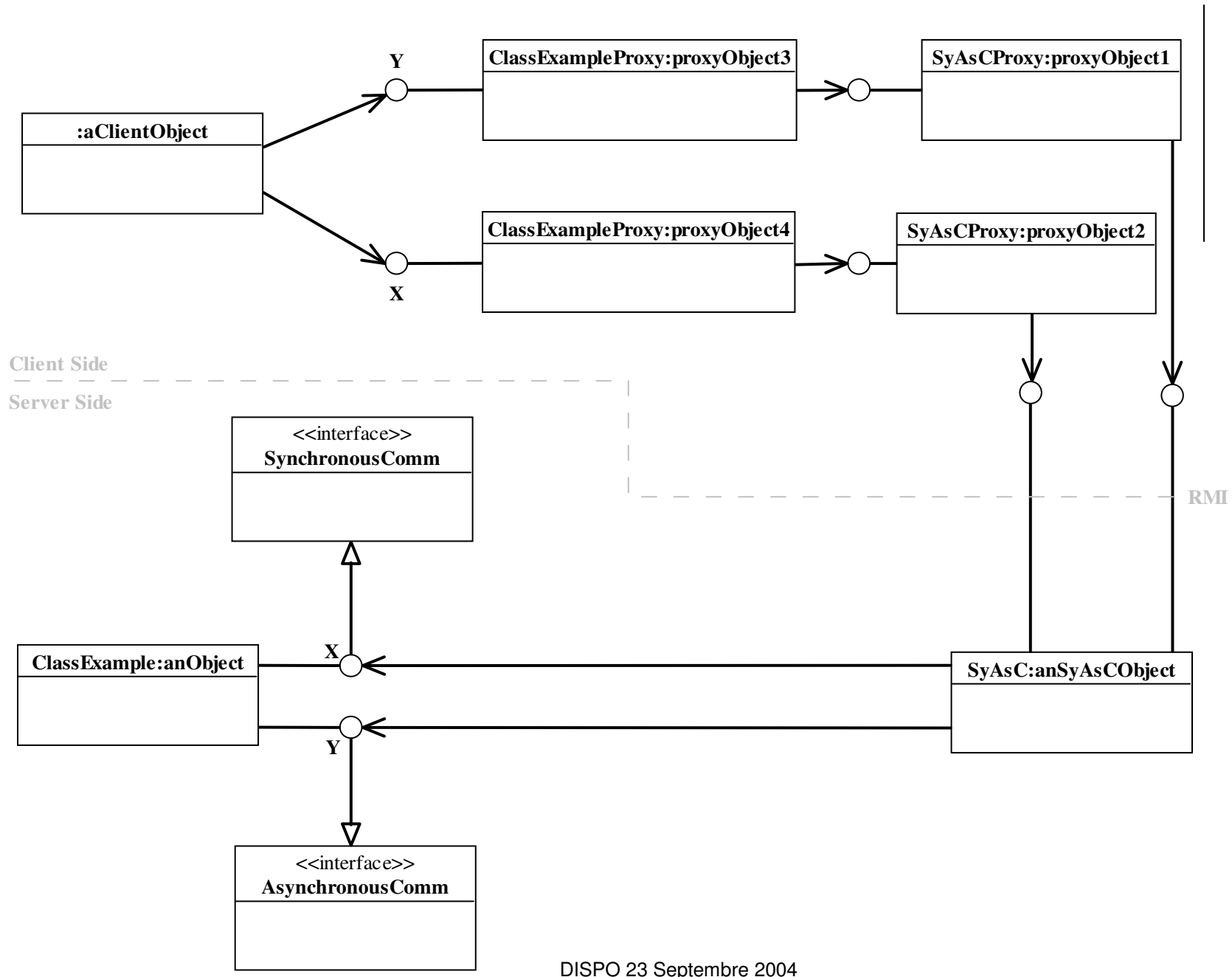
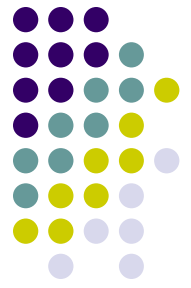


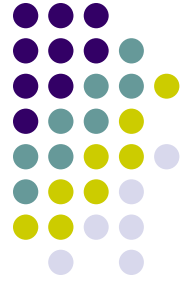
Too complicated ... we need some implicit asynchronous communication mechanism

# MultipleCommunication Package



- ProActive
  - Reflective mechanisms to transform
  - Passive objects into active objects
  - Use Futures for synchronous communication
- MultiComm
  - (A)Synchronous remote communication
  - Transparent use of RMI registry/proxies/stubs (thanks to java 1.5.0)





# Plan

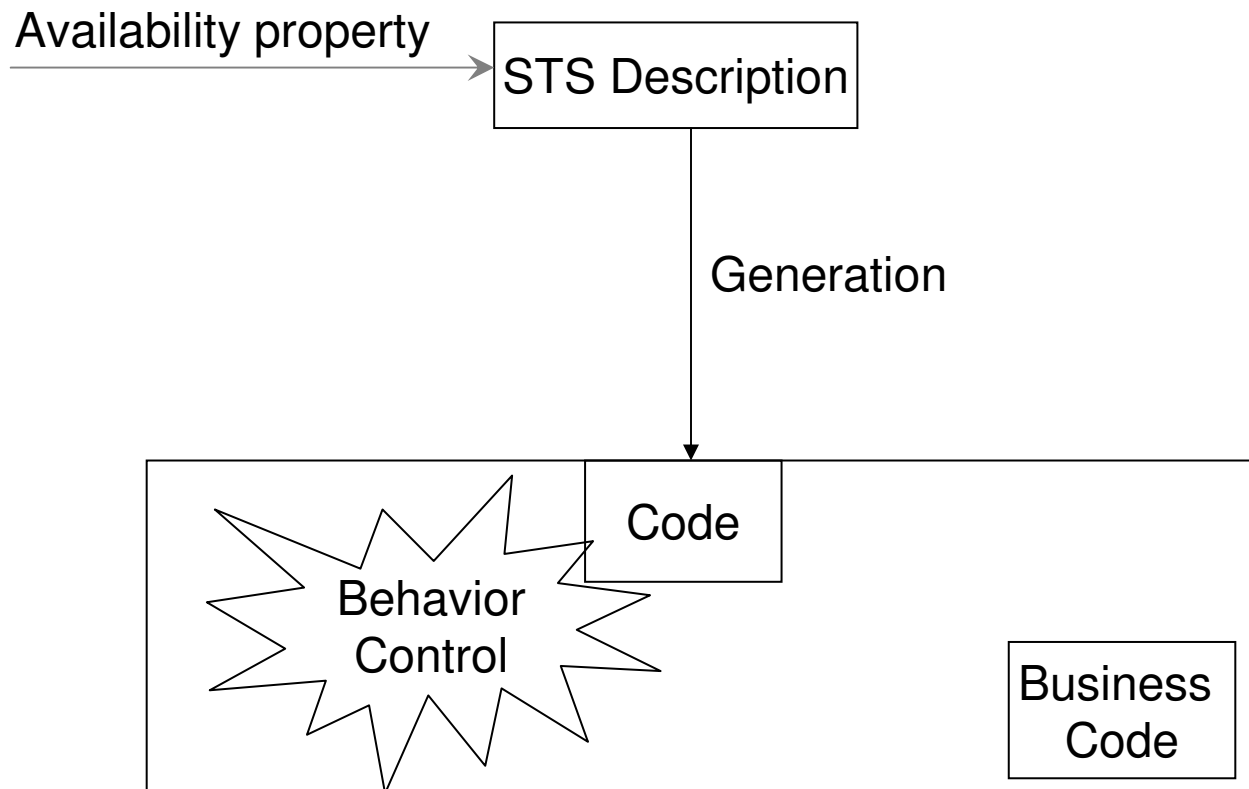
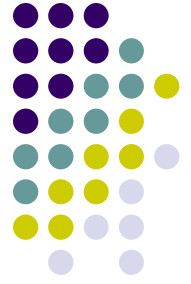
- ArchJava
- Dynamic Configuration in ArchJava
- Asynchronous Communication Experiments
- **Symbolic Transition Systems for Components**



# STS for Components

1. STS definition
  - Graphical
  - Textual for manipulation
2. Set of rules for translation into
3. Code: **Java**, ArchJava, others

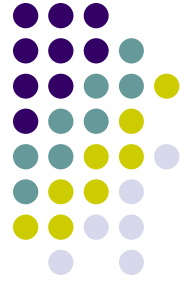
# STS for Components



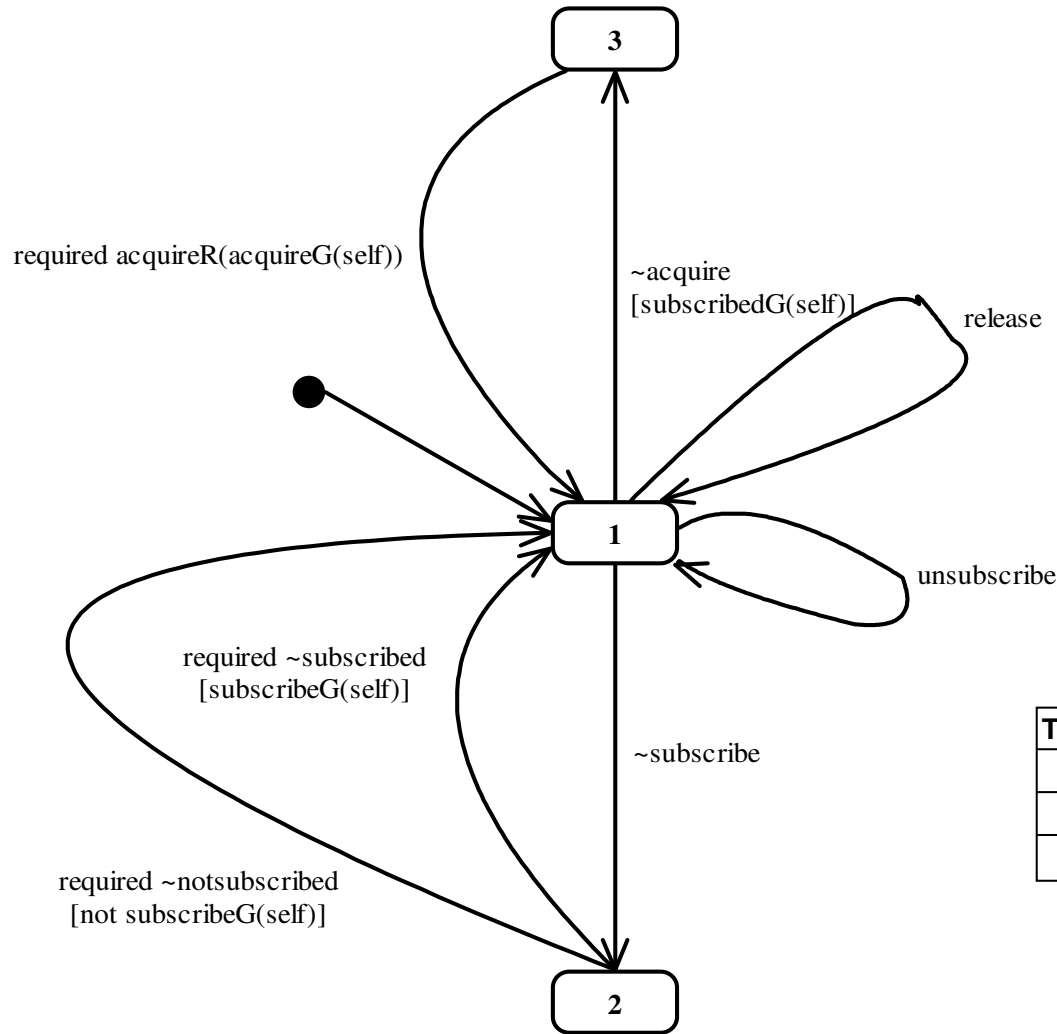
# Current Work and Perspectives



- Resource Allocation Case Study
- STS definition
  - Synchronous/asynchronous communication
  - Availability Property?
- Generation Process
- Code Conformance



~operation ::= asynchronous operation



Types	
Name	Description
Client	...
Ressource	...

Operations			
Name	Parameters	Type	Description
subscribe	?idClient:Client	public/provided	
unsubscribe	?idClient:Client	public/provided	
subscribed	void	public/required	
notsubscribed	void	public/required	