

---

# Dynamic Adaptation

Minema Winter School 2009  
Gothenburg, Sweden

Paul Grace, Computing Department,  
Lancaster University

[p.grace@lancaster.ac.uk](mailto:p.grace@lancaster.ac.uk)



# Acknowledgements

- Gordon Blair
- Geoff Coulson
- Francois Taiani
- Nelly Bencomo



- Eddy Truyen
- Bert Lagaisse
- Wouter Joosen

# Outline of the Talk

## What is dynamic adaptation ?

- Principled software approaches
- Safety
- Types of adaptation

## Why is adaptation important?

- Characteristics of mobile computing
- Emerging fields

**PART ONE**

## Dynamic Middleware

- Reflective middleware
- (Dynamic AOP middleware)
- (Policy middleware)

## Future Challenges

- What are the next research areas?

**PART TWO**

# Further Reading

- Oreizy, P., Medvidovic, N., and Taylor, R. **Architecture-based runtime software evolution**. In Proceedings of the 20th international Conference on Software Engineering (Kyoto, Japan, April 1998)
- McKinley, P., Sadjadi, S., Kasten, E., and Cheng, B. **Composing adaptive software**. IEEE Computer 37(7): 56-64 (2004)
- Kramer, J. and Magee, J. **The evolving philosophers problem: dynamic change management**. IEEE Transactions on Software Engineering 16(11): 1293-1306 (November 1990)
- Kon, F., Costa, F., Blair, G., and Campbell, R. H. **The case for reflective middleware**. Communications of the ACM 45(6):33-38 (June 2002)
- Issarny, V., Caporuscio, M., and Georgantas, N. **A perspective on the future of middleware-based software engineering**. International Conference on Software Engineering (Minneapolis, USA, May 2007)

Definitions, Types & Motivation

# DYNAMIC ADAPTATION

# Everyone's Talking About It. Are you?

Talk 1 – Change the sensor network behaviour ...

Talk 6 – Self-healing mesh routes

Talk 10 – Lime reacting to changes in context

Talk 2 – Install new module on sensor

Talk 7 – Autonomous Gossip, evolution

Talk 11 – Autonomous driving

Talk 3 – Changing models of connectivity

Talk 5 – ANA, autonomic, Self\*

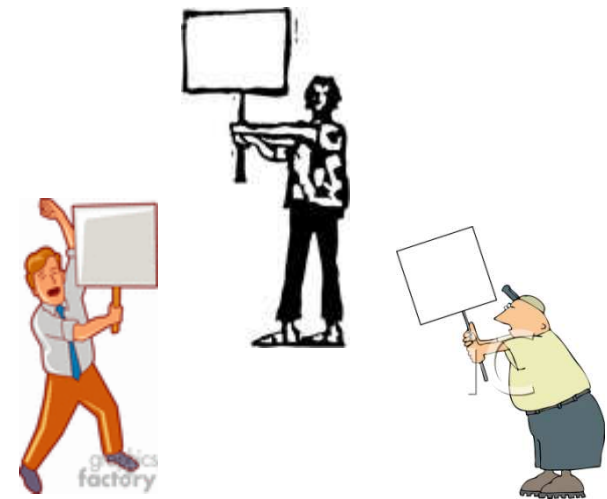
Talk 8 – change of subscriptions, change of nodes, change of connectivity

Talk 12 – reaction to context

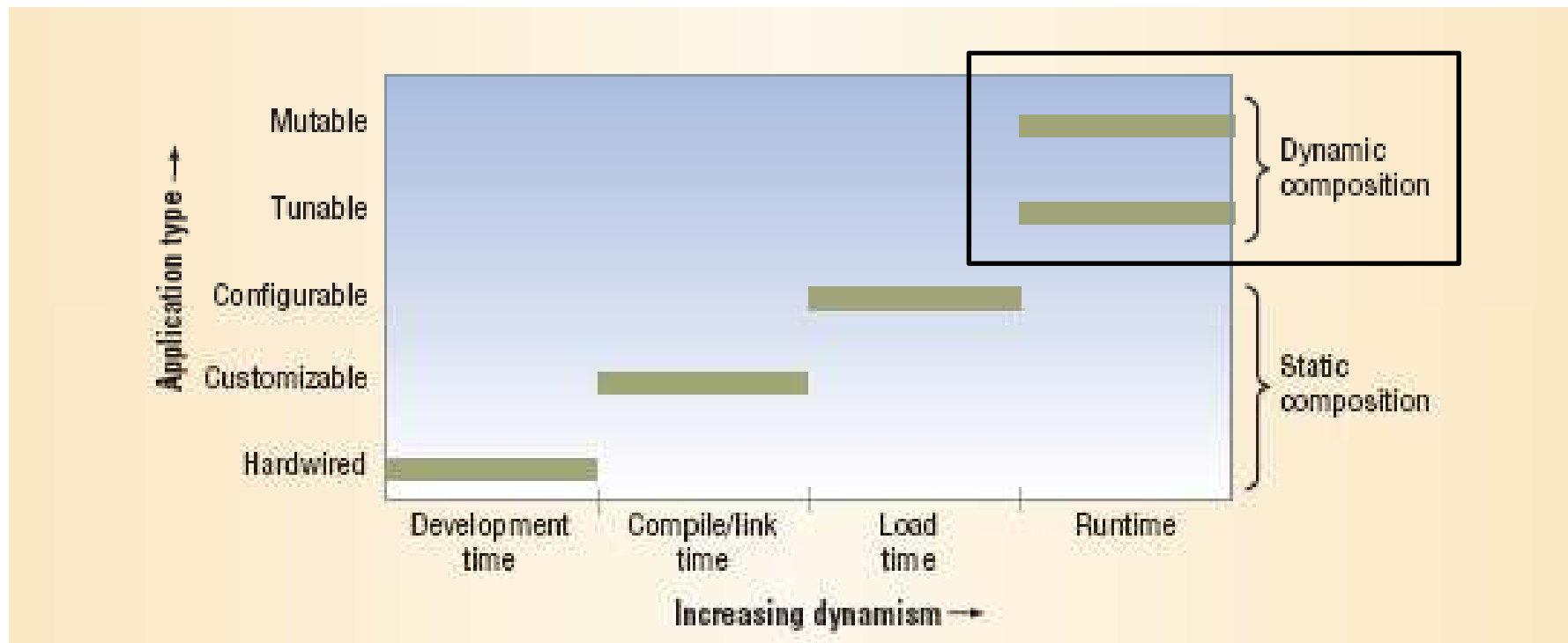
# Dynamic Adaptation

(or dynamic reconfiguration)

- ‘Changing the behaviour of an executing system at runtime’
  - A system is not taken off-line e.g. “Your update was successful and you need to reboot the system”
- Classifying Software Adaptation
  - Who makes the change
  - What behaviour is changed
  - How is the change made
  - When is the change made



# One Classification



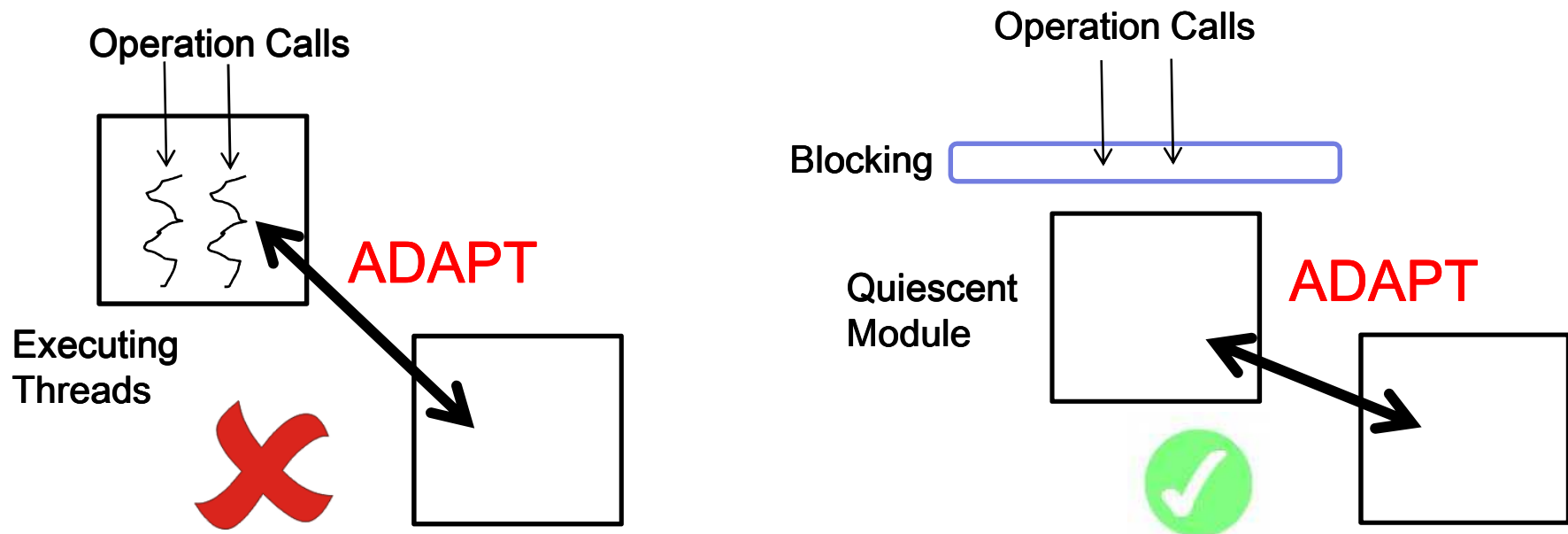
From McKinley et al., "Composing Adaptive Software"

# Additional Properties of Adaptation

- **Safety**
  - **Quiescence:** The system is placed in a state such that the adaptation does not cause erroneous behaviour
- **Consensus**
  - In a multi-party system the adaptation is agreed upon
- **Consistency**
  - Multiple adaptations (which may be concurrently executed) do not conflict
- **Rollback**
  - If the adaptation fails the system returns to the prior state

# Exploring Quiescence

- Seminal work by Kramer and Magee (Imperial College)



Wait for threads to complete; what are the problems with this ?

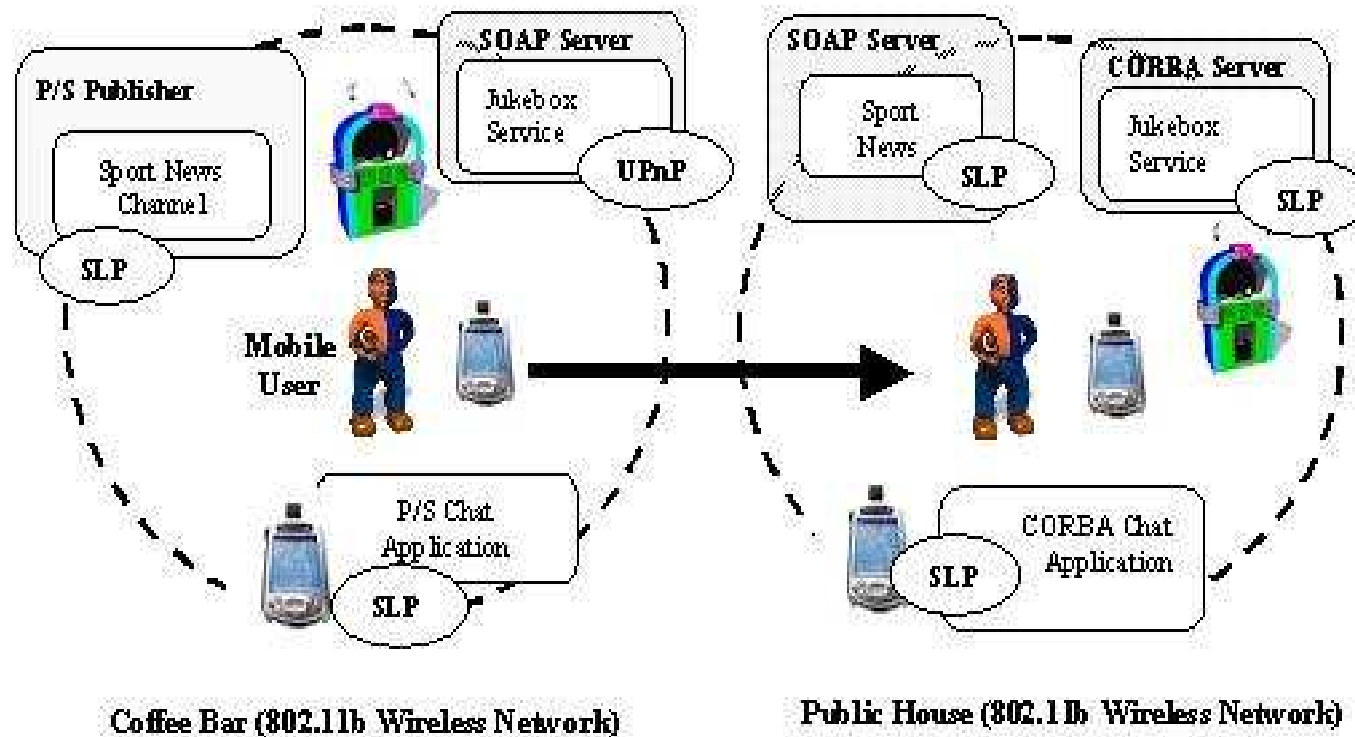
# Motivation: Mobile Computing

- The key characteristics of Mobile Computing are Change & Spontaneity
  - Changes in Context, Resource Availability, Fluctuating Network Quality of Service (QoS)
  - Applications must adapt, middleware must adapt, ...

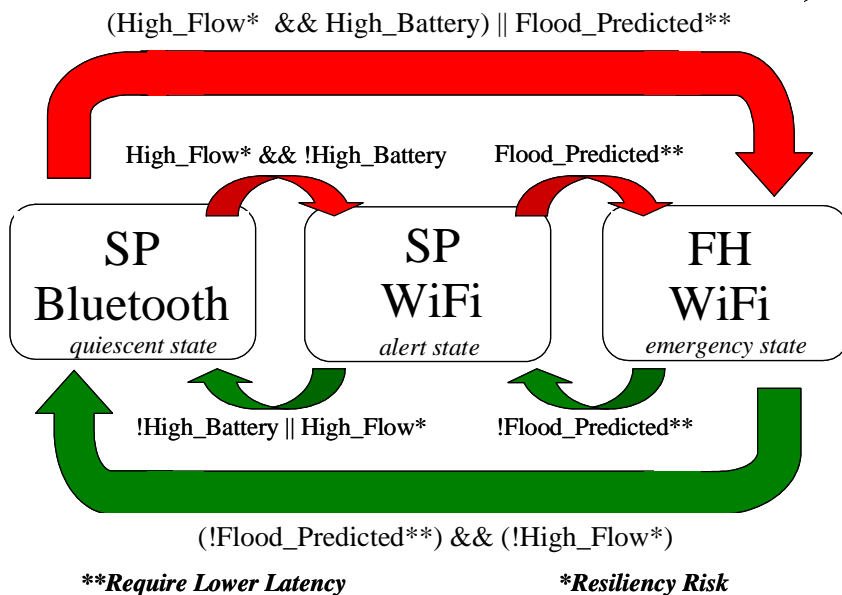
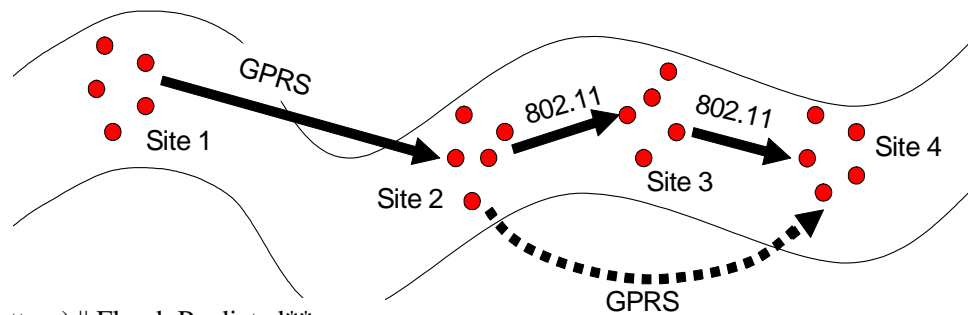


Images from the Runes project:  
<http://www.ist-runes.org>

# Case Study 1: Protocol Heterogeneity



# Case Study 2: Sensor Network Adaptation for Flood Monitoring



```

if (High_Flow && High_Battery
    || Flood_Predicted)
    replace Bluetooth with WiFi
    replace ST.sp with ST.fh
    
```

# Analysis

- Two distinct classes of adaptation
  - Node-local
    - Software adaptation within a single machine (or address space)
  - Distributed
    - Co-ordinated adaptation of modules across multiple machines
    - Introspection, Consensus, Safety ... remain open research challenges

Mechanisms: Reflection and Dynamic AOP

# DYNAMIC ADAPTATION

# General Approaches to Software Adaptation

- *Parameter Adaptation*

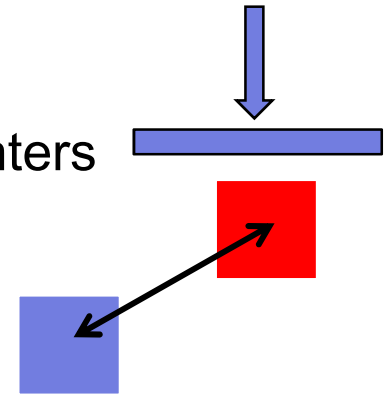
- Modifies pre-defined program variables that determine a system's behaviour
  - E.g. changing TCP retransmission rates in face of congestion
- Direct between existing strategies
- Cannot introduce new algorithms and behaviour

- *Compositional Adaptation*

- Dynamic recomposition of software modules
- Introduce new behaviour
  - *Add, remove, replace*

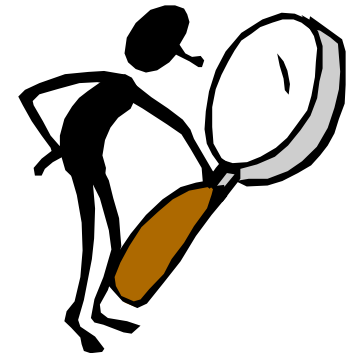
# Mechanisms for Compositional Adaptation at runtime

- Many potential mechanisms
  - Proxies, Interceptors, Strategy pattern, Functional pointers
    - Indirection is a common feature
- Here we focus on two composition types
  - Software Components
  - Aspect Oriented Programming (AOP)
- ... and we focus on two principled adaptation mechanisms
  - Computational Reflection
  - Dynamic AOP



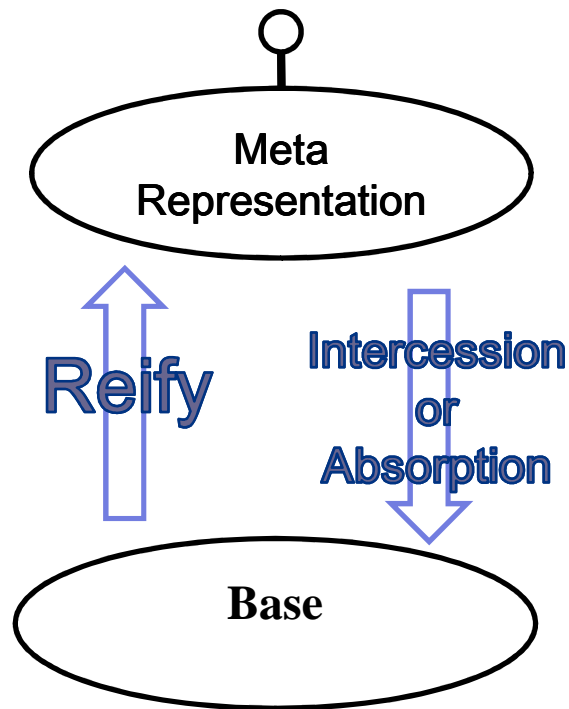
# Computational Reflection

- Reflection refers to the capability of a system to *reason about* and *act upon itself*
- A reflective system provides a representation of its own behaviour:
  - amenable to inspection and adaptation,
  - causally connected to the underlying behaviour it describes
- *Causal Connection*
  - changes made to the self-representation are immediately mirrored in the underlying system's actual state and behavior, and vice-versa



# Reflection Types and Concepts

## Meta Object Protocol

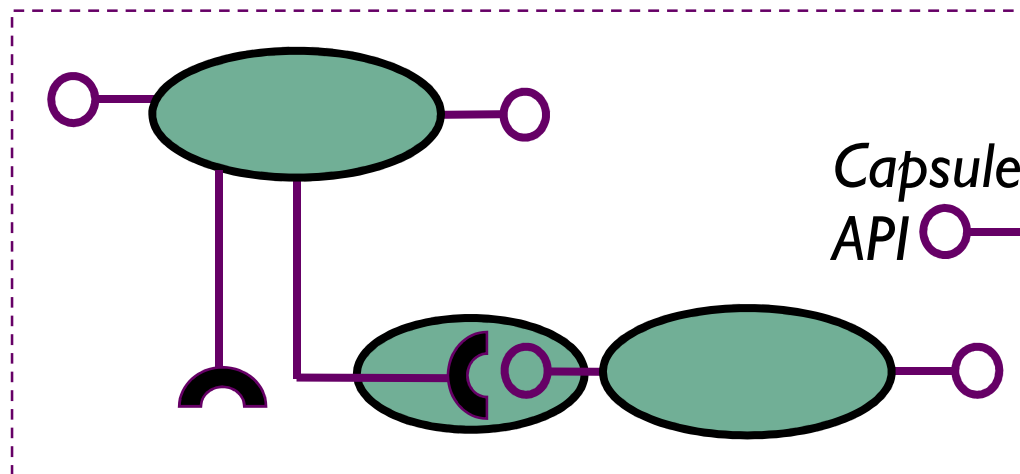


## Types

- Structural Reflection
  - Representation of *(static) structure* of the system
    - Classes, Components, ...
- Behavioural Reflection
  - representation of *ongoing activity*
    - Interceptors, Method Calls, ..

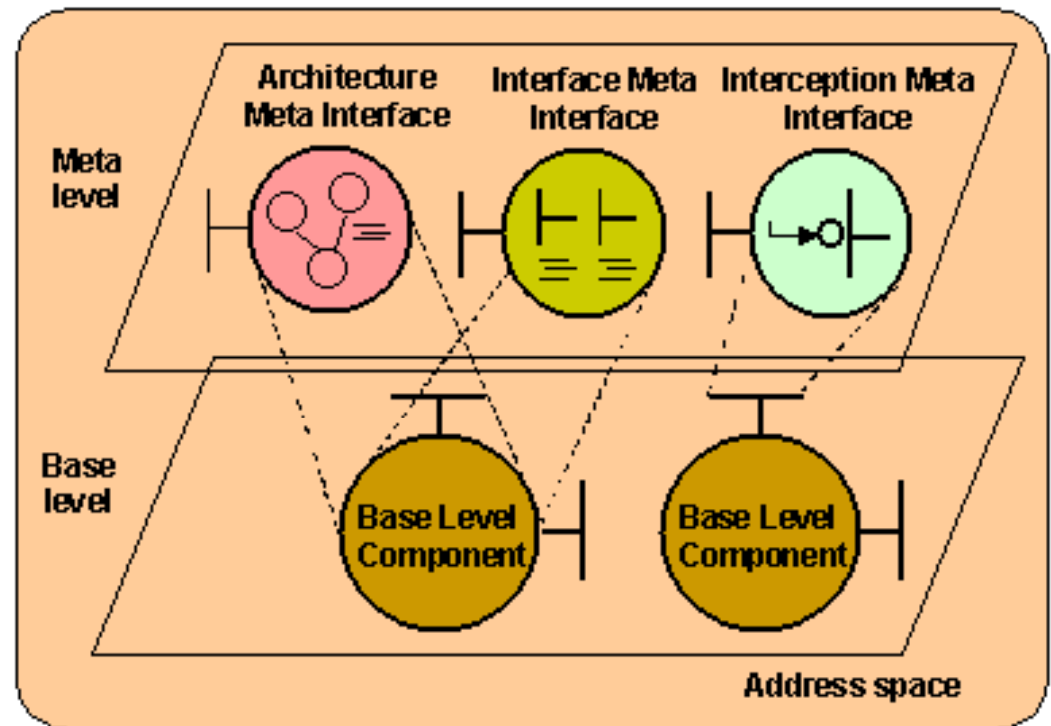
# OpenCom: A Reflective Component Model

- Tool for component based software development
  - Inherent support for performing node-local reflection
  - Fractal is an alternative
- Central concepts:  
component | capsule | interface | receptacle | binding



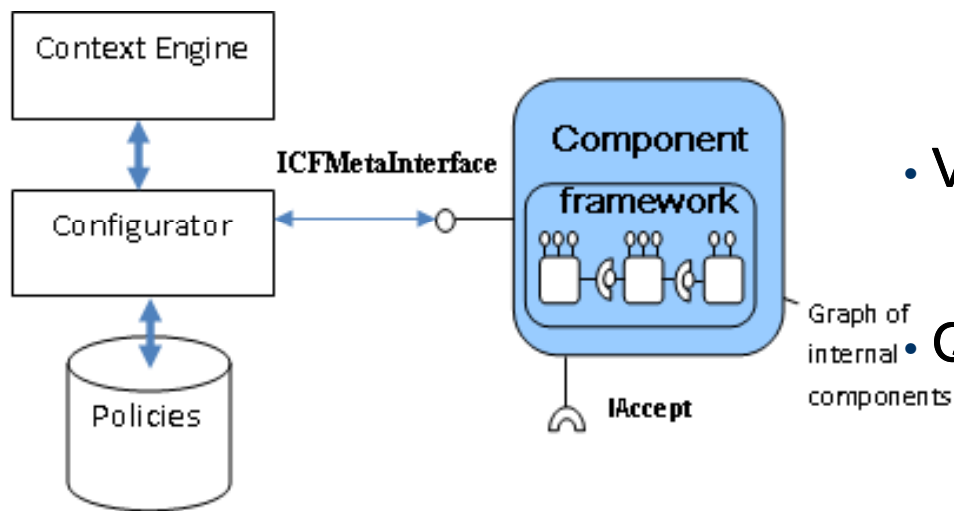
# A Multi-Model Approach

- **Architecture**
  - represent the topology of a composition of components within a capsule
  - 'graph-oriented'
- **Interception**
  - interpose interceptors
- **Interface**
  - dynamically discover details of a component's interfaces/ receptacles
  - dynamically invoke dynamically-discovered interfaces



# Component Frameworks

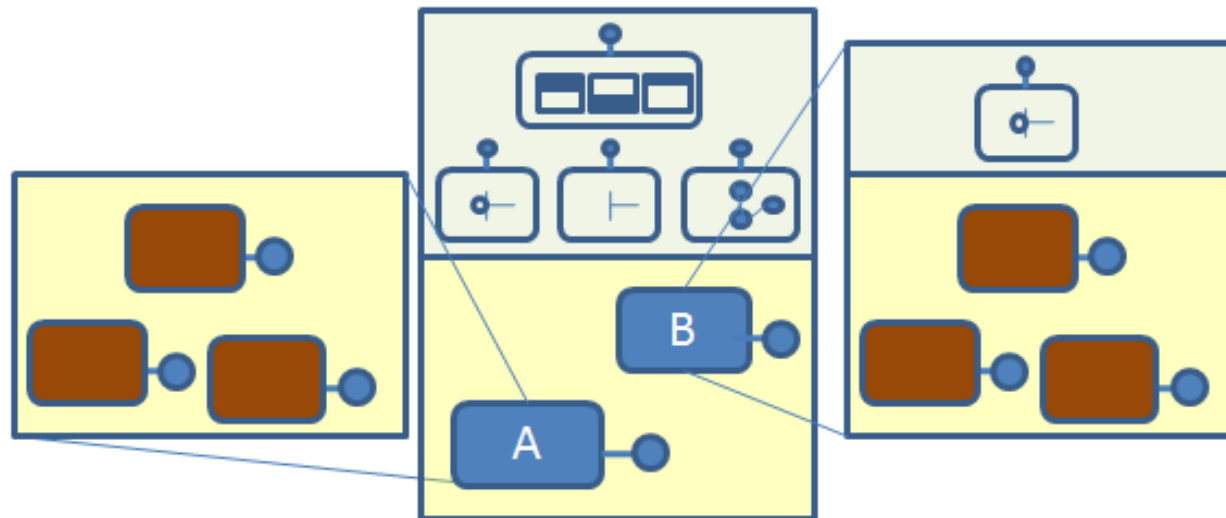
## Supporting Node-local Adaptation



- **Architecture Meta-Protocol**
  - Reflective operations to adapt component configuration
- **Validated Reconfigurations**
  - Constraint checking & roll back
- **Quiescence Management**
  - Ensure framework is adapt ready, no active threads
- **Policy-based reconfiguration pattern**
  - Context events trigger change

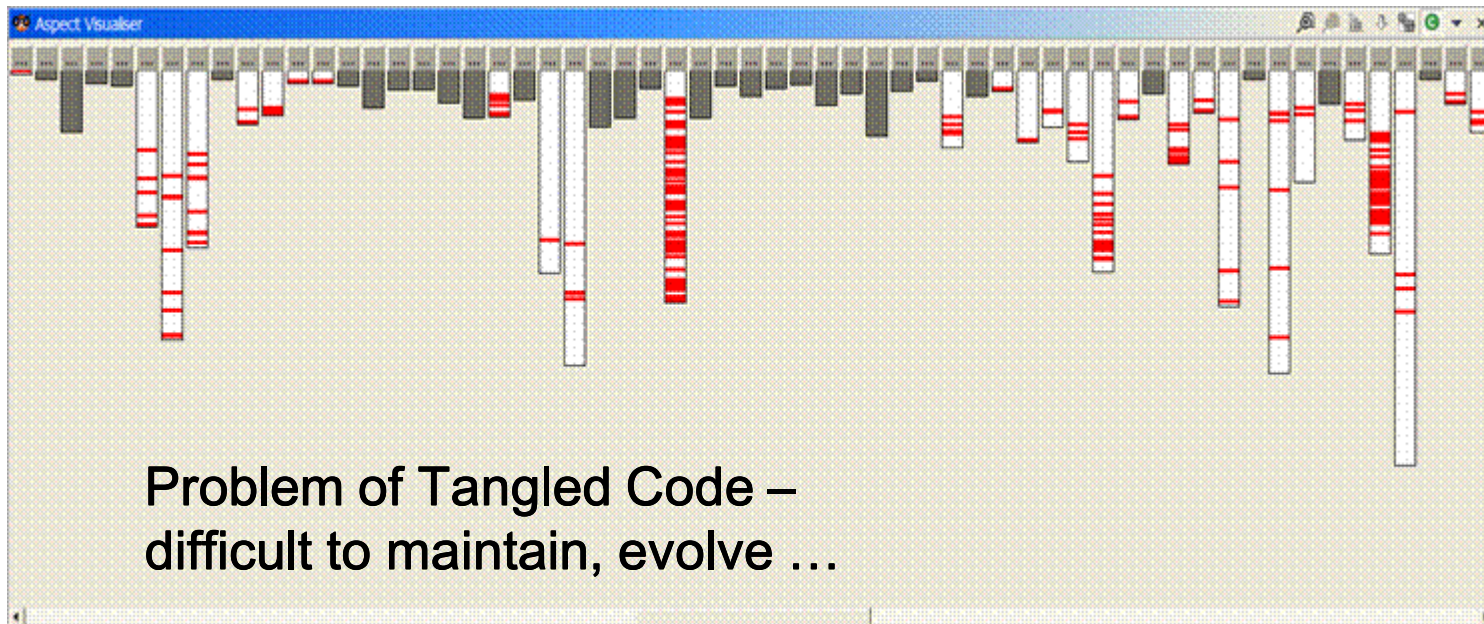
# The Cost of Reflection

- Reflection comes at a price
  - Structural = Increased memory consumption
    - Out-of-band
  - Behavioural e.g. Interception = Performance reduction
    - In-band
- *A solution = Partial Reflection*



# Aspect-Oriented Programming

- “There are some design decisions that are hard to cleanly capture because they *crosscut* the system’s basic functionality. We call the issues that these features address *aspects*...” [Kiczales, Aspect-Oriented Programming, ECOOP 97]



# Aspects

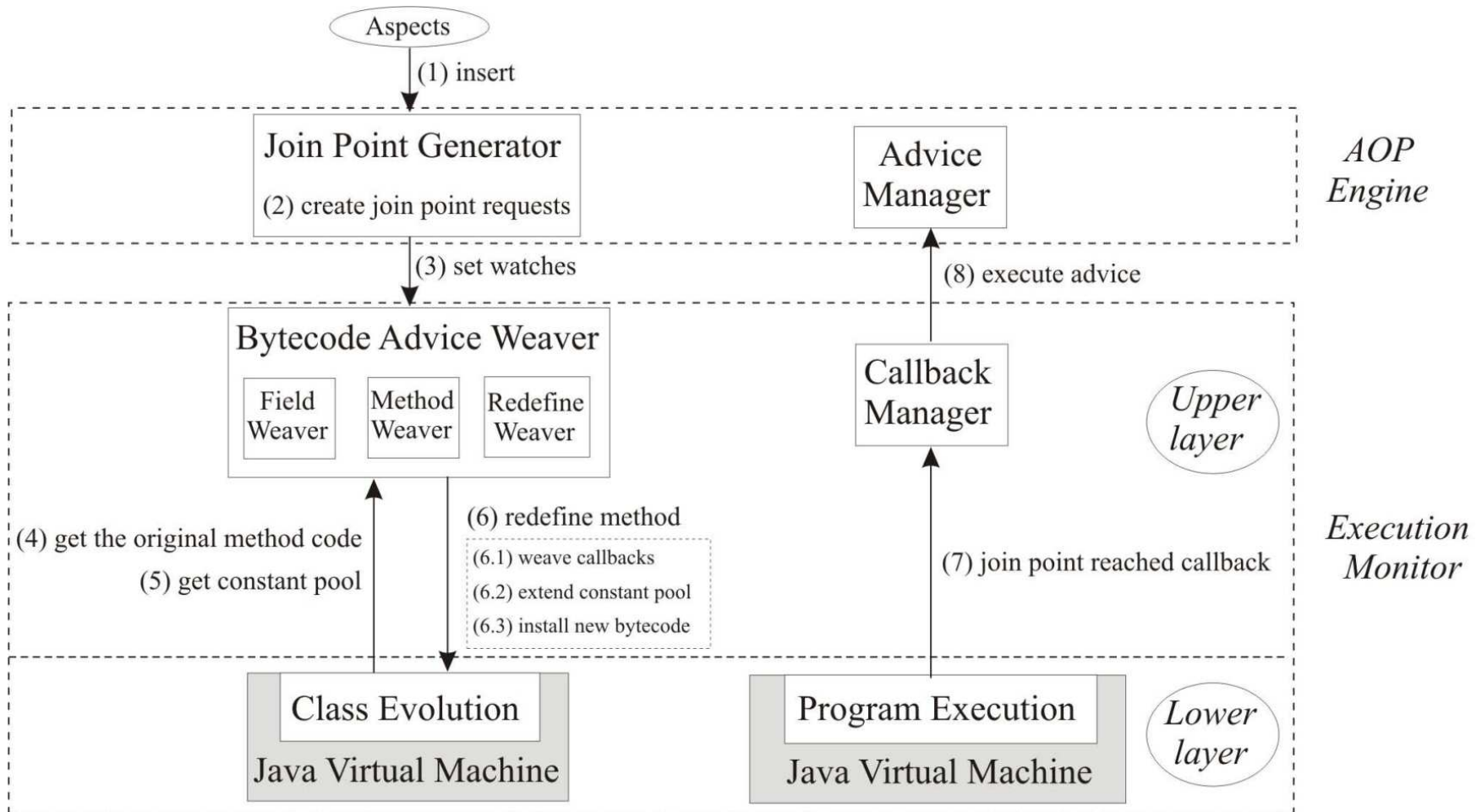
```
@Pointcut("(execution(**.*doEverything())  
|| execution(* *.*doSomething()))  
&& !within(*Test)")
```

- Aspect is a Unit of composition
  - Pointcut & set of advices
  - Woven into the base system
    - at compile time with early tools like the AspectJ
- Join point model
  - Pointcuts identify points (**join point**) where advice code should be executed
  - Model differs depending on the system type e.g. methods, fields in Object-Oriented language.
- Advice
  - Contains the actual code to implement the concern
  - Before, after, around execution points
  - Around utilises a **proceed** keyword

# Dynamic AOP

- Compile-time weaving is unsuitable for dynamic adaptation
- Dynamic AOP weaves and unweaves at runtime
  - However a large majority perform load-time weaving when the classes are loaded into memory e.g. the Spring Framework
  - Two common approaches
    - Invasive Weaving
      - Typically performed using byte-code transformation i.e. language specific
      - Prose
    - Non-Invasive Weaving
      - Typically performed using Interception
      - JBoss dynamic AOP, JAC, Lasagne

# PROSE



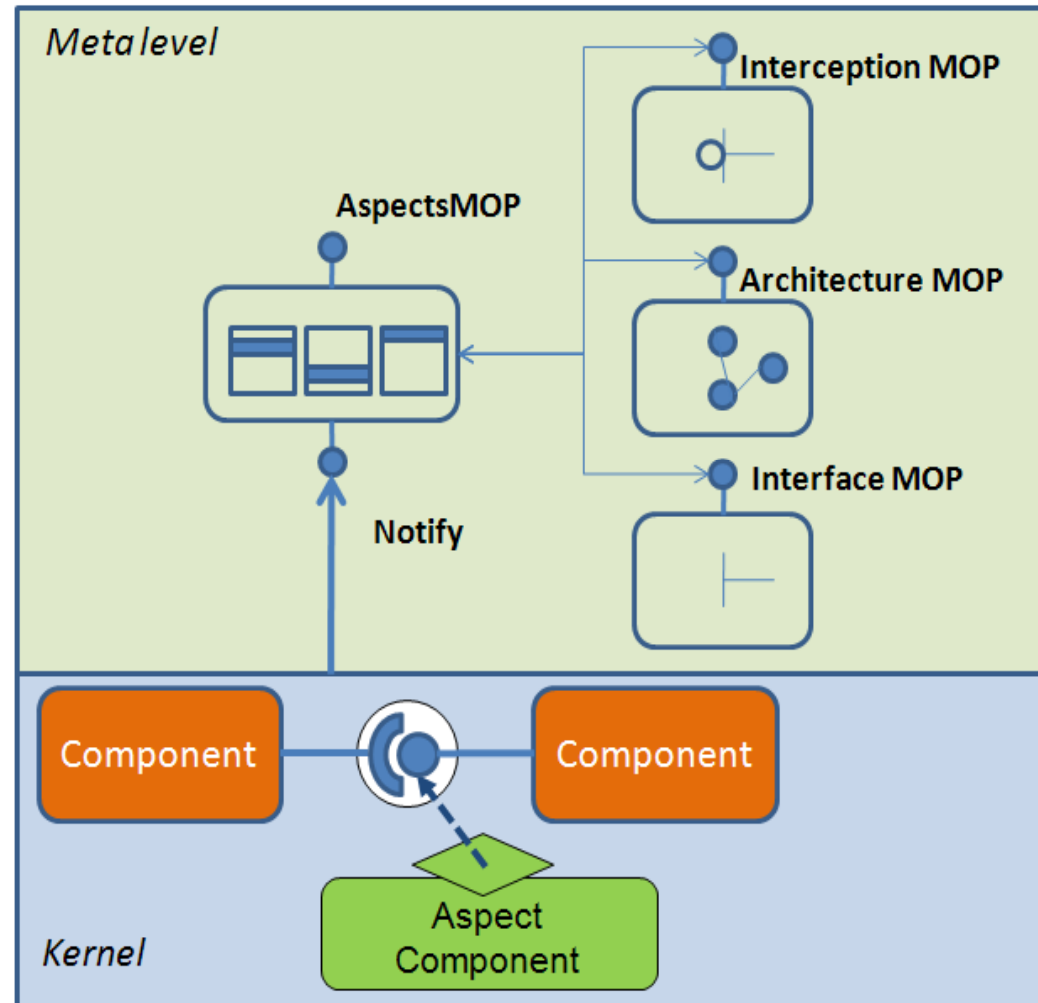
# Combining AOP and Reflection

Technology	Cross cutting	Adaptation	Self-Adaptation
Reflection	Poor	General	Yes
AOP	Strong	Aspects only (coarse-grained)	No

- Improve the management of crosscutting behaviour in reflective systems
- Build introspective self-adaptive aspect-based systems

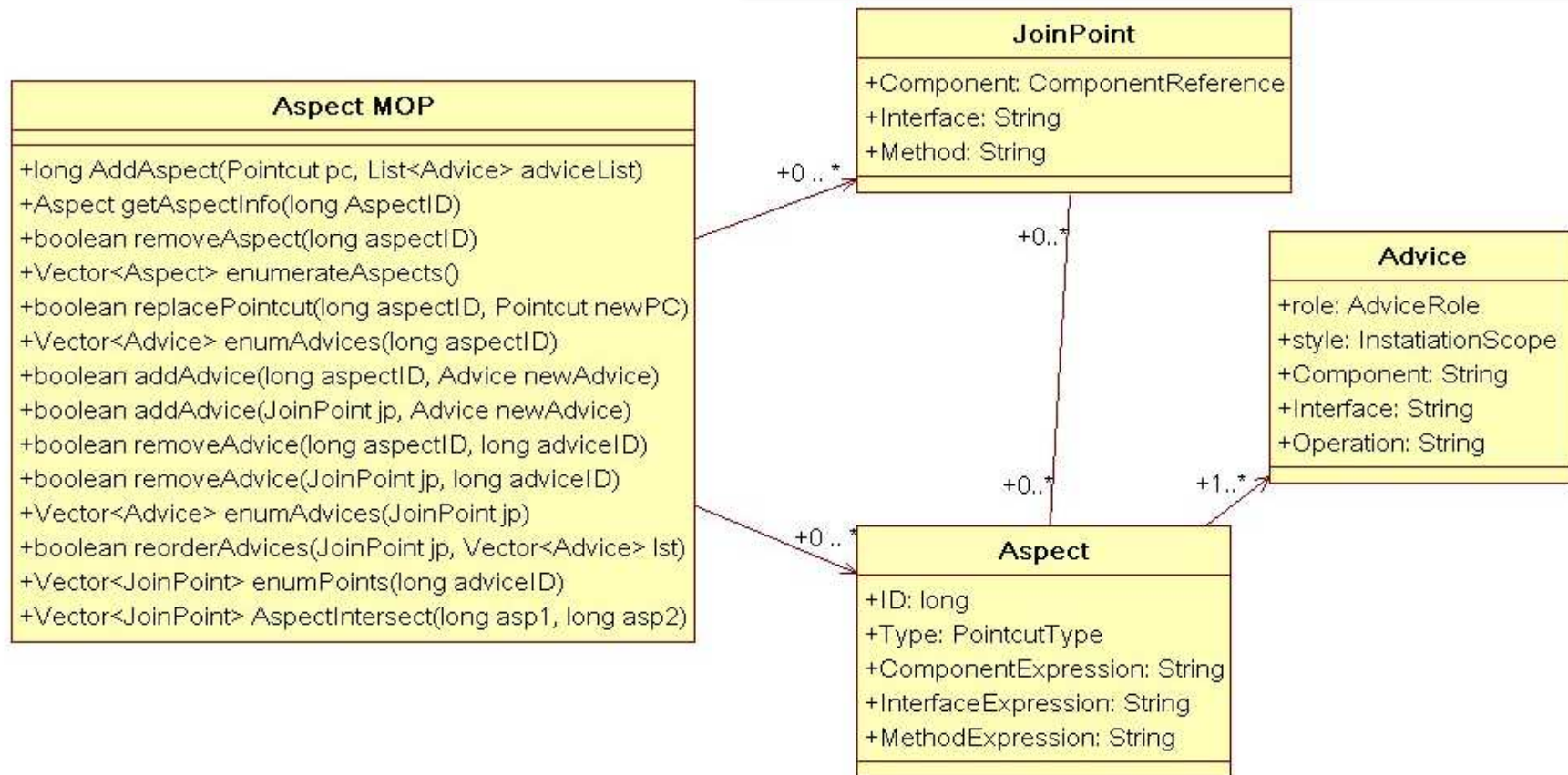
# AspectOpenCom

- Fine-grained introspection and adaptation of cross-cutting behaviour
- Aspects can be added to a system at run-time using the Aspect MOP as the implementation of this MOP is underpinned by existing reflective MOPs
- Reconfiguration aspects can be deployed using the Aspect MOP that abstract over reflective MOPs



# An Aspects Meta-Protocol

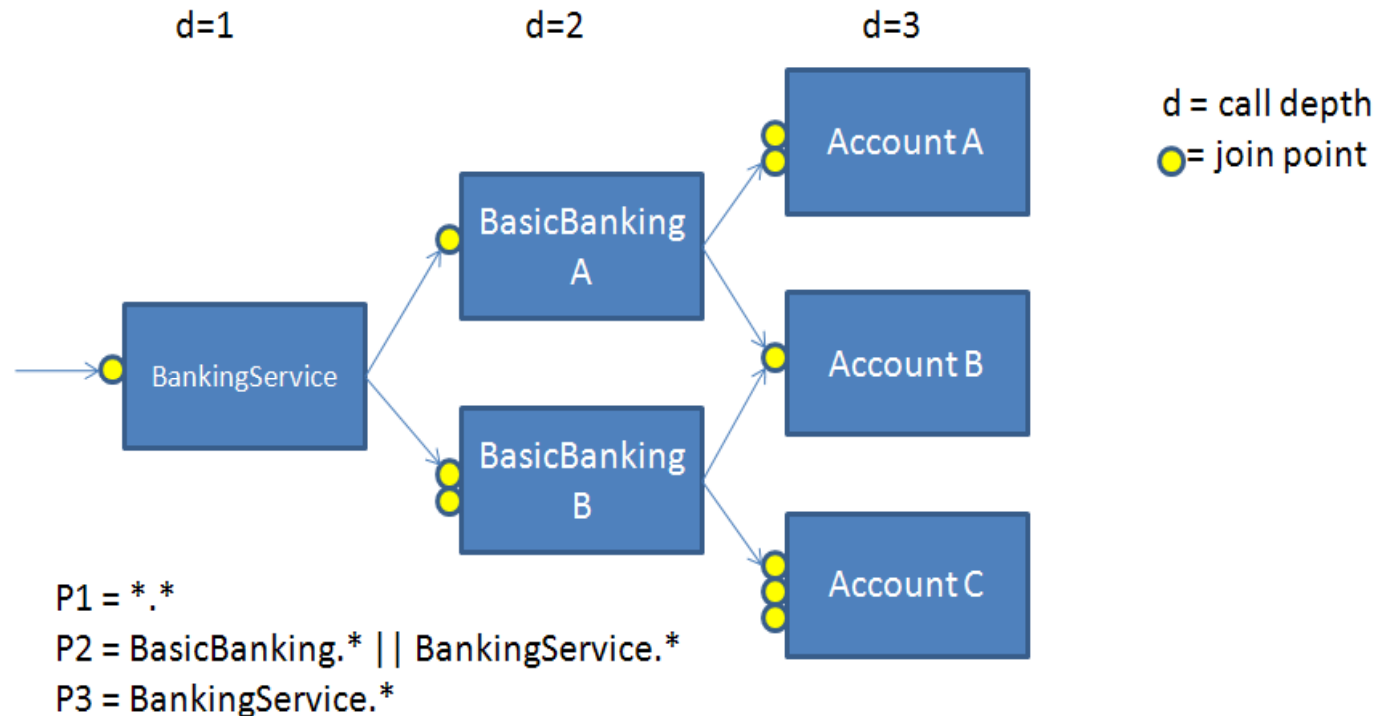
## META REPRESENTATION



# Join Point Set Adaptation

Adapt the Authentication Concern

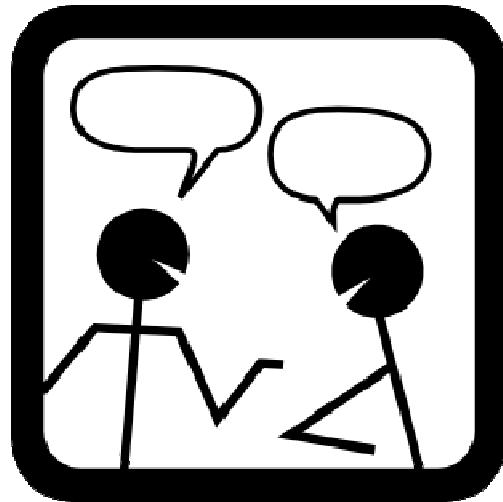
→ Deploy trace aspect at greater “depths” during suspicious behaviour



```
1 | Pointcut expr = new Pointcut(PointcutType.EXECUTION,  
2 |     "BasicBanking* || BankingService*", "*", "*");  
3 | aspectMOP.replacePointcut(traceAspect, expr);
```

## Question Point 1

*Any Questions about Dynamic Adaptation  
in General?*



# Summarising Dynamic Adaptation

- Focus on how to perform adaptation
  - Useful for developers of novel applications and adaptive middleware
- Examined tools and approaches for node-local adaptation
  - Limited support so far for developing adaptive distributed applications
- Investigated two important compositional mechanisms
  - Adaptation had similar properties that has and can be applied to other composition types

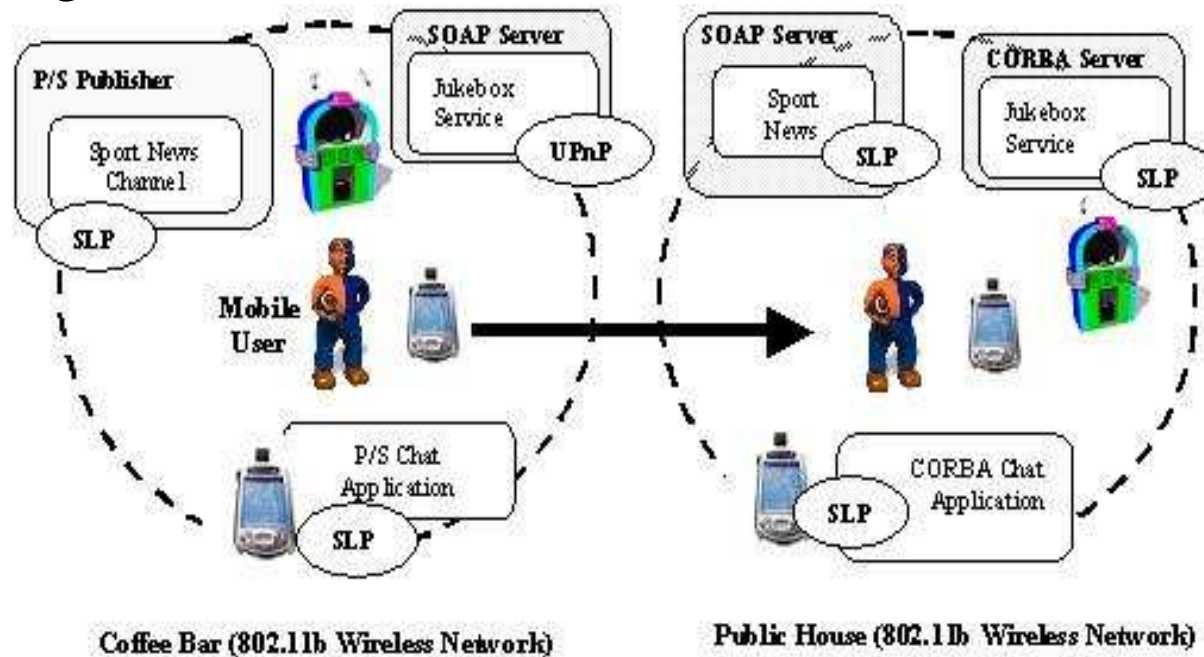
# DYNAMIC MIDDLEWARE

# The Case for Adaptive Middleware

- Middleware is ideally located to support adaptations
  - Provide mechanisms to underpin application adaptation
  - Transparently adapt distributed system behaviour to alleviate the challenges distributed application developers face
- Explore Reflective Middleware
  - ReMMoC, ExORB, GridStix, K-Components
- More about Dynamic AOP middleware, Policy-based Middleware in the book
  - DyRES, CARISMA, Micro-Protocol frameworks

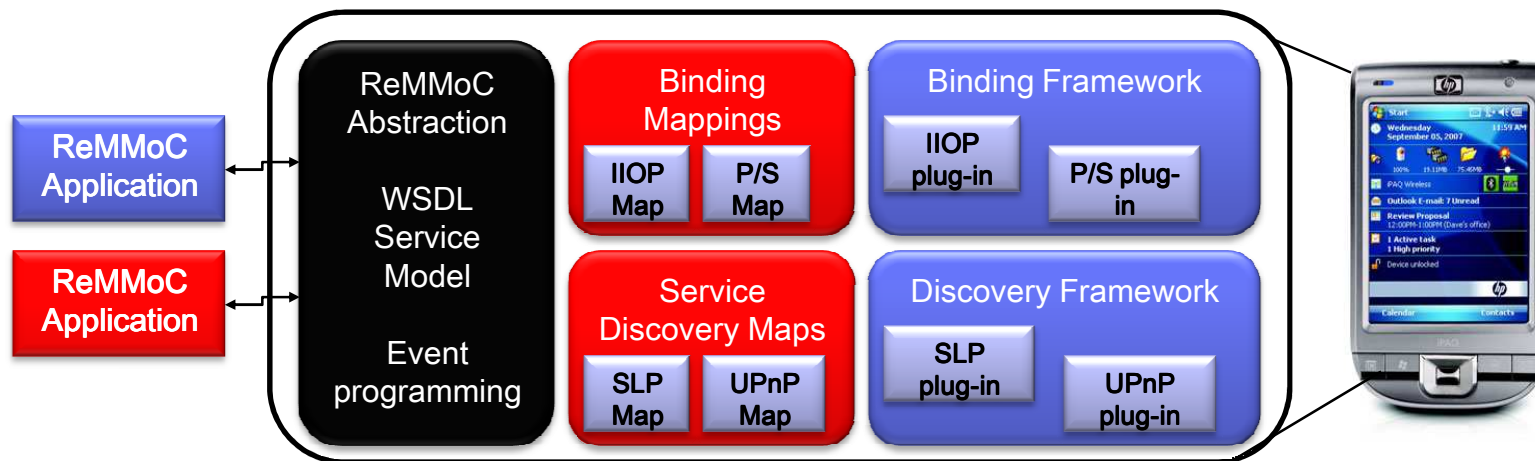
# ReMMoC

- A middleware for developing mobile clients that encounter heterogeneous services from location to location



# A Context-based, Reflective Solution

- What discovery protocols are in the environment ?
- What is the middleware type of the found service
- Dynamically adapt plug-ins to *Mirror* the environment
  - Component based implementations of the full protocol specification
  - Change of context changes the ReMMoC Personality
- Assumption: Point of agreement = A WSDL description

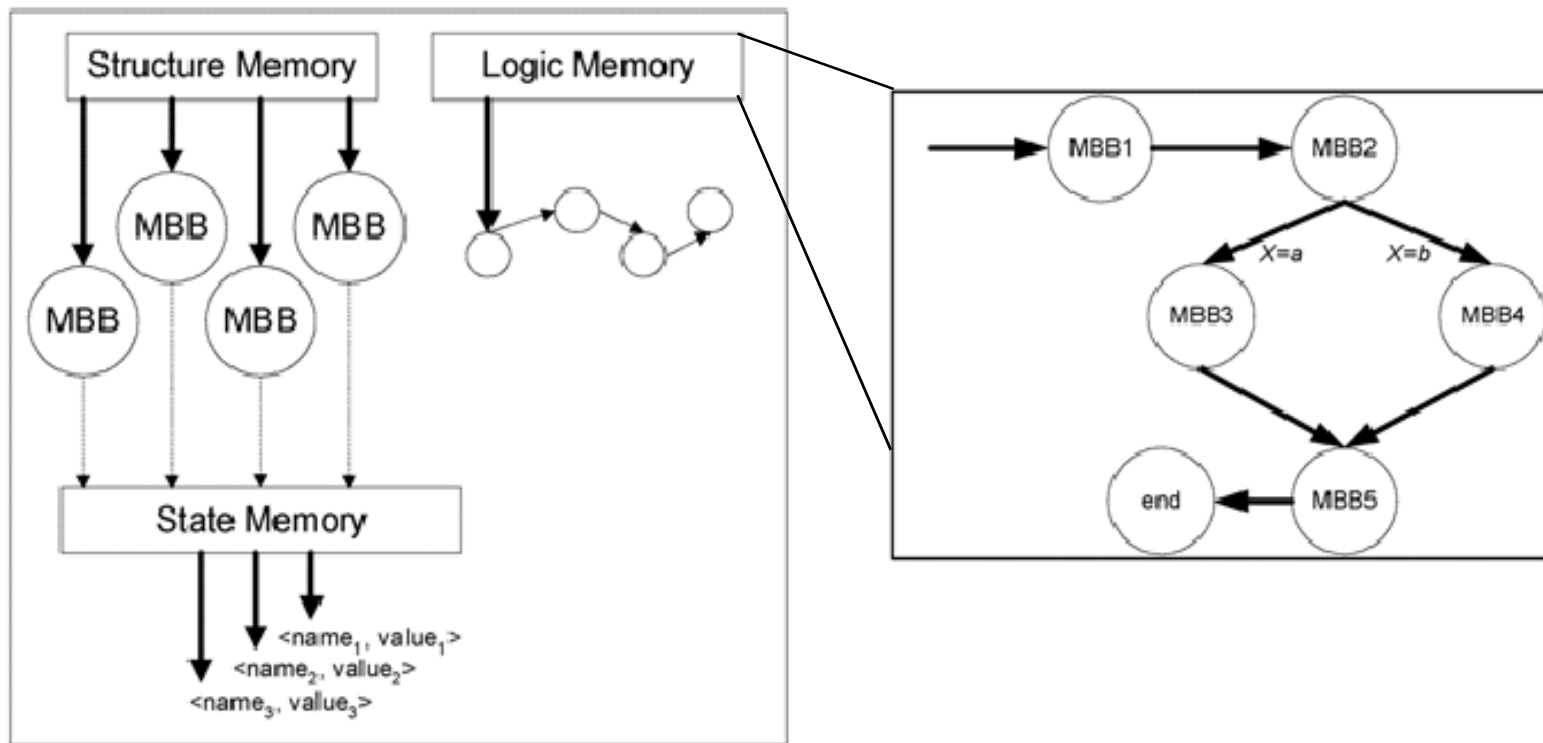


# ExORB

- Targeted towards mobile phones and their configurability, updating and upgrading
- Based on the principle of externalisation (CCSR)
  - Explicit denotation of platform's state, logic and component architecture
    - Micro Building Blocks (MBBs)
      - The smallest addressable functional unit in the system
      - State held elsewhere
    - Actions
      - Specifies the order in which MBBs execute
      - Represented as a deterministic directed graph
    - Domains
      - Represent collections of MBBs + actions + localised state
      - Supports hierarchical composition

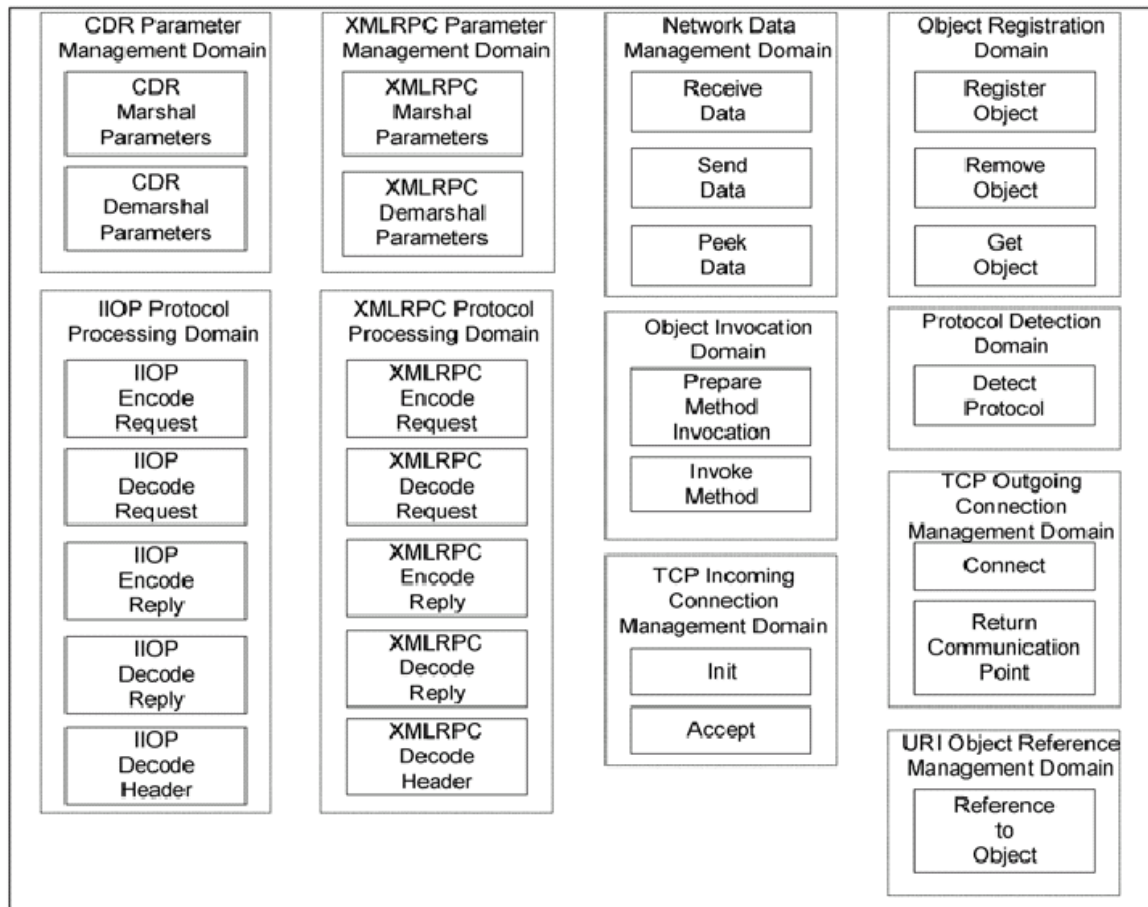


# Dynamically Programmable Reconfigurable Services



Adapt structure and logic (maintain state)

# ExORB: Using DPRS for a Reconfigurable ORB

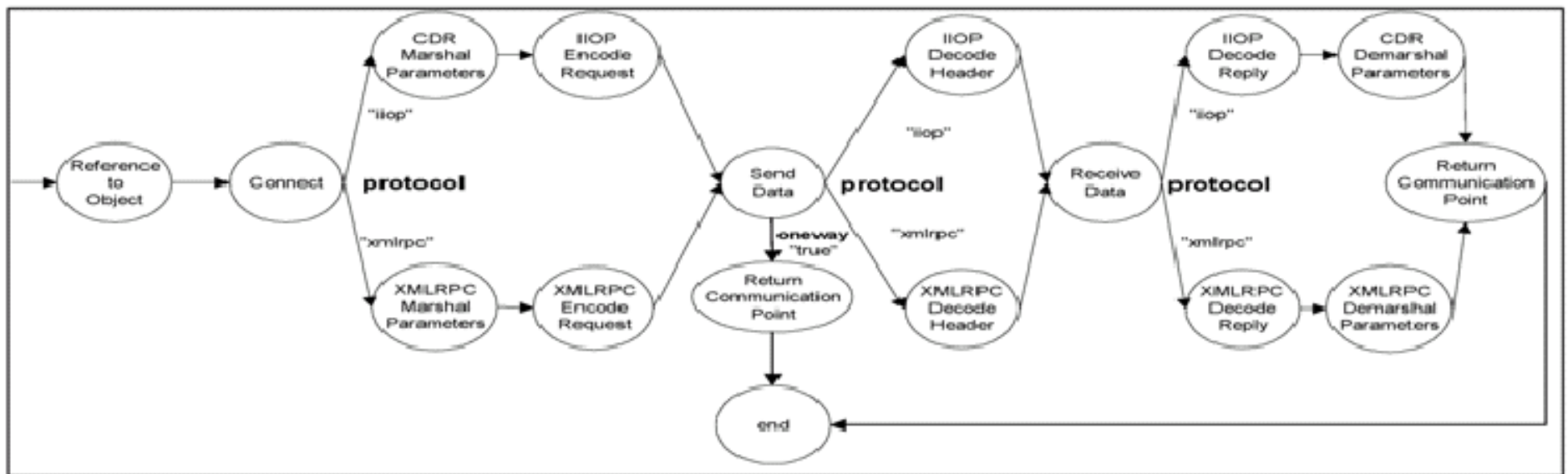


## Multi-ORB building blocks

- IIO Protocol
- XML RPC

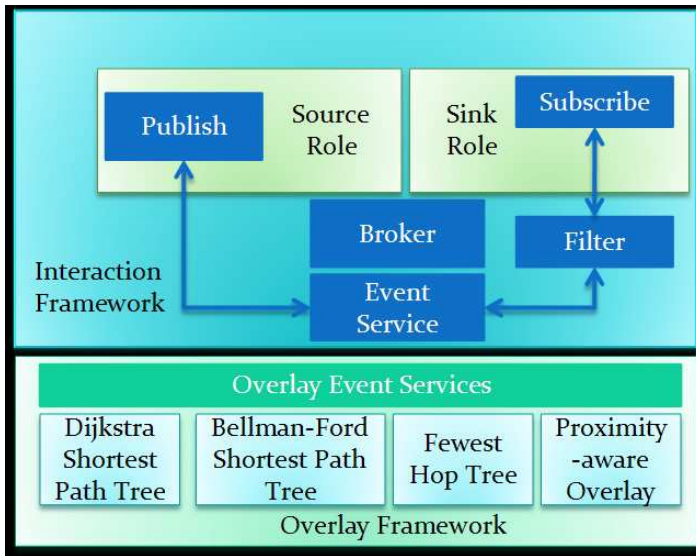
→ Change structure e.g. Replace a block with new version, add new blocks

# Logic: An Example encoding IIOP and XML RPC requests



Adapt the logic: New Protocol, New Behaviour e.g. logging

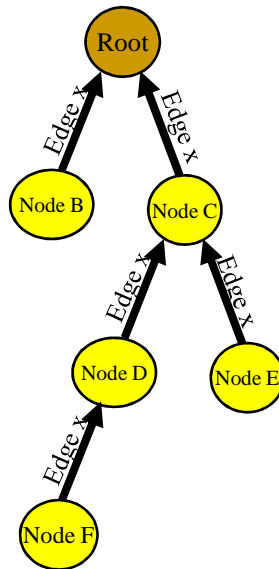
# Gridstix Middleware



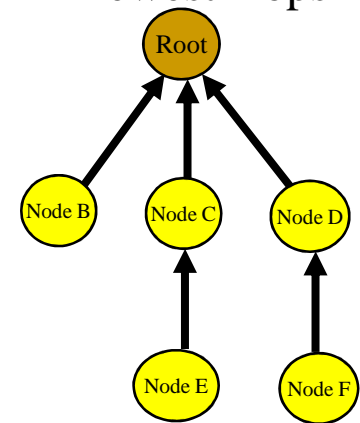
Per node middleware

Change the complete sensor network from one spanning tree to another

Shortest Path



Fewest Hops



**Trigger:**  
Flooding  
predicted

# Gridstix's Reflective Approach to co-ordinated adaptation

## •Architecture Meta-Protocol

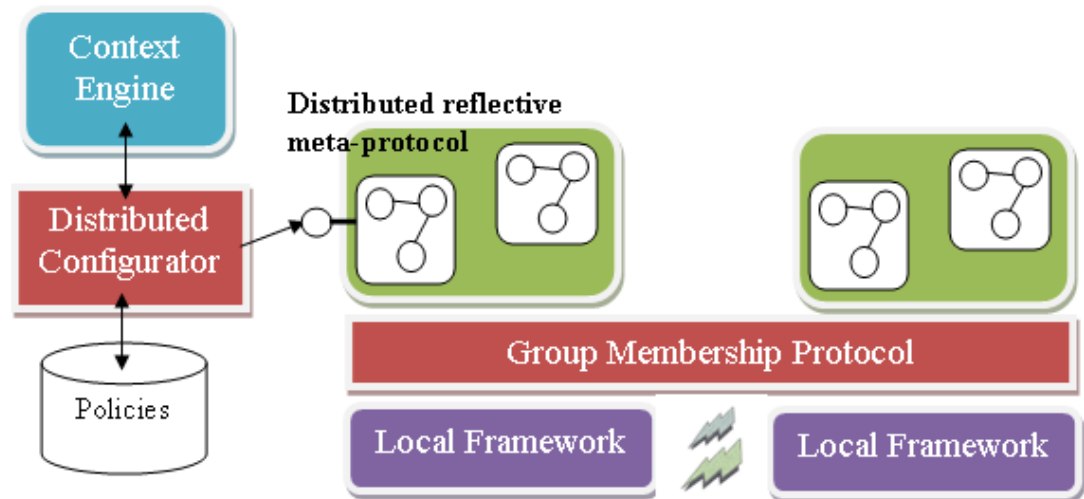
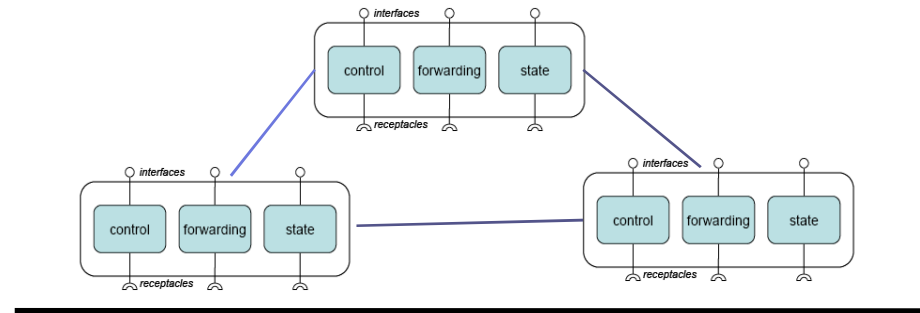
- Distributed view & adaptation

## •Quiescence Management

- Distributed algorithm to place nodes in safe state
- Centralised/Single configurator
  - Investigating decentralised

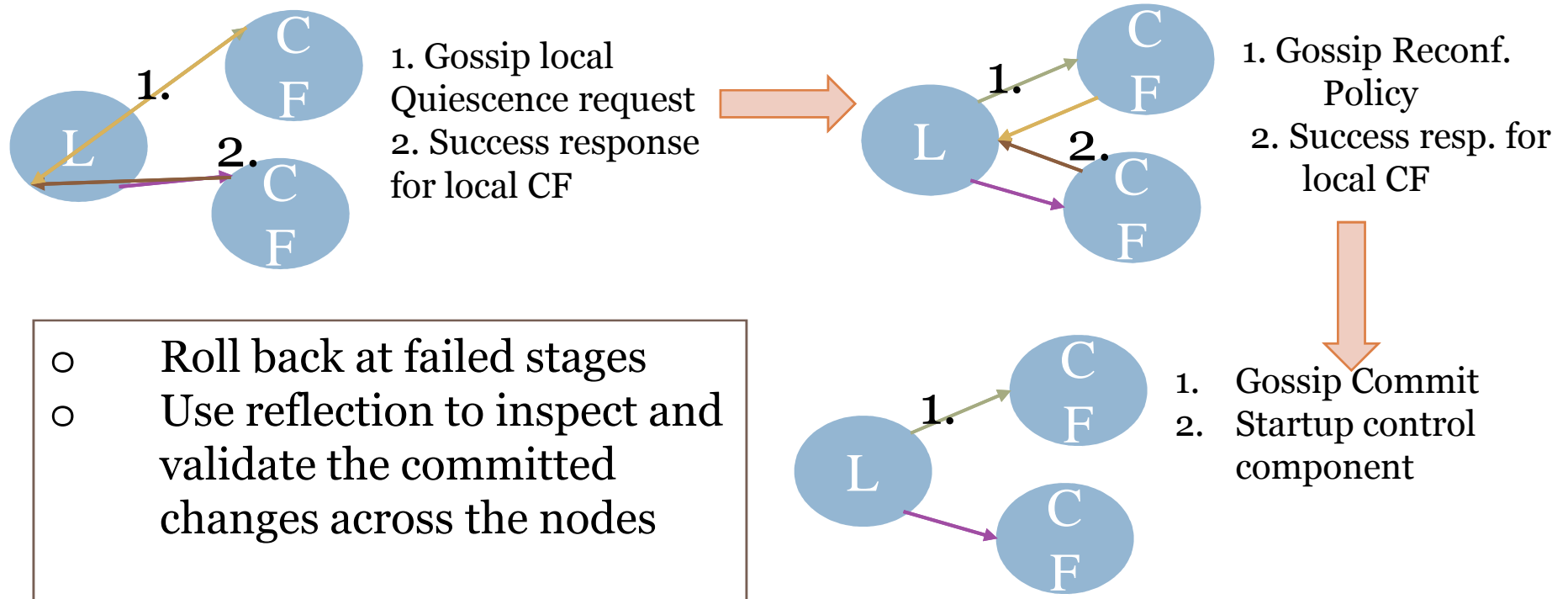
## • Policy-based reconfiguration

- One or more configurators
- Consensus on adaptation



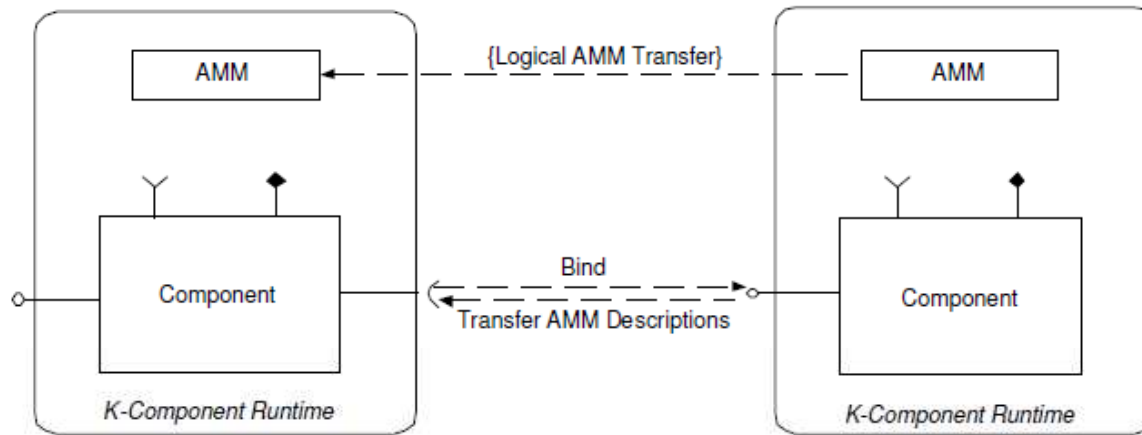
# Adapting a Sensor Network

- ⑩ Place all nodes in a safe state; replace the control component; rebuild the tree





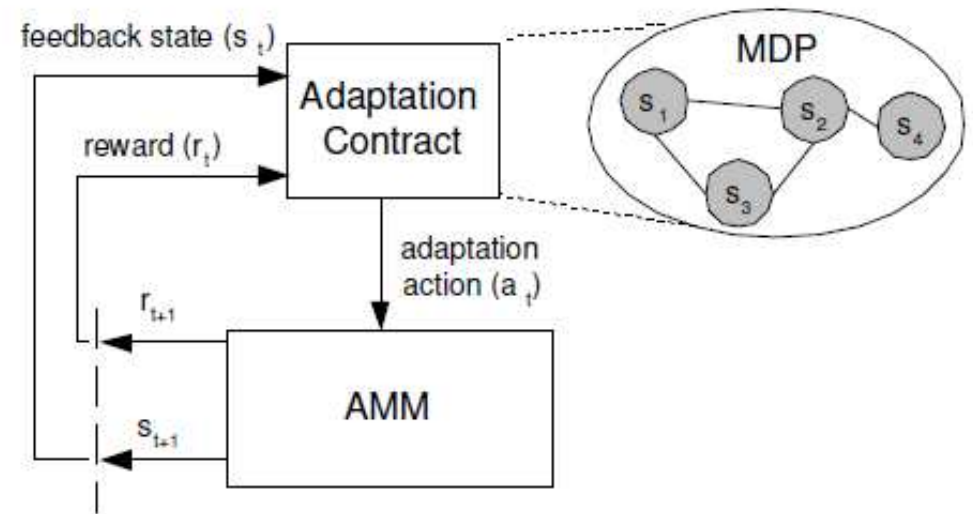
# Using K Components for Autonomic, Decentralized, Co-ordinated Adaptation



Share context, behaviour, structure to inform decisions

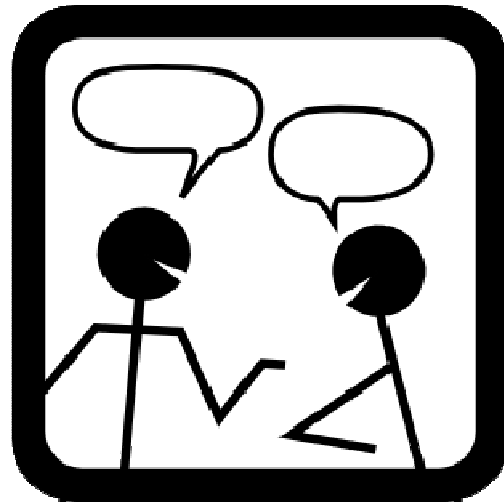
Build a reinforced learning policy

See SAMPLE – learns an ad-hoc routing protocol



## Question Point 2

*Any Questions about Adaptive Middleware?*



# Analysis

- Middleware built around node-local mechanisms and centralised distributed adaptation mechanisms
  - Initial solutions that do not fully support next generation domains
    - Ubiquitous, P2P, Autonomic, Large-Scale
- Need for flexible approaches
- Need for further investigations into maintaining the core properties of adaptation in decentralized fashion
  - K Components early proponent in this direction

# FUTURE CHALLENGES

# Challenge 1 (New Adaptation Approaches)

- Mechanisms for decentralised adaptation
  - E.g. Adaptation in ad-hoc networks, Large-Scale P2P networks, Hybrid domains
    - How to decide?
    - How to make a system safe?
    - Flexible algorithms
    - Combined compositional adaptation

# Challenge 2 (Engineering)

- Engineering complex adaptations
  - How to support developers describe and deploy adaptive systems
    - **Software Engineering community**
      - SEAMS - Software Engineering for Adaptive and Self-Managing Systems @ ICSE
    - **Modelling of Adaptations**
      - Models@Design Time
      - Models@Run Time
    - **Verification of Adaptive Systems**
      - ARAMIS - Automated engineerRing of Autonomous and run-tiMe evolving Systems Workshop @ ASE

# Challenge 3 (Applications)

- Real time adaptations
  - Introducing time dimensions into reconfigurations
    - Impact upon safety mechanisms
    - Impact upon decision mechanisms
- Cross-domain middleware
  - Ad-hoc, sensor, Internet, Grid, P2P
    - Adapt across the technologies

# Expected Outcomes

- Convinced you that dynamic adaptation is important
  - In mobile computing it is a fundamental requirement
  - Needed across a wide range of emerging fields
- Understand the problems faced when performing adaptation
  - Safety, Consistency, Consensus
- Provided knowledge how to perform adaptation
  - Mechanisms and tools
- Made you aware of the state of the art in middleware solutions in this domain

# Questions

---

- ⑩ Thank you for your attention

