

Imperial College
London

Service Composition: From Models to Self-Management

Dr. Howard Foster
Distributed Software Engineering
Department of Computing

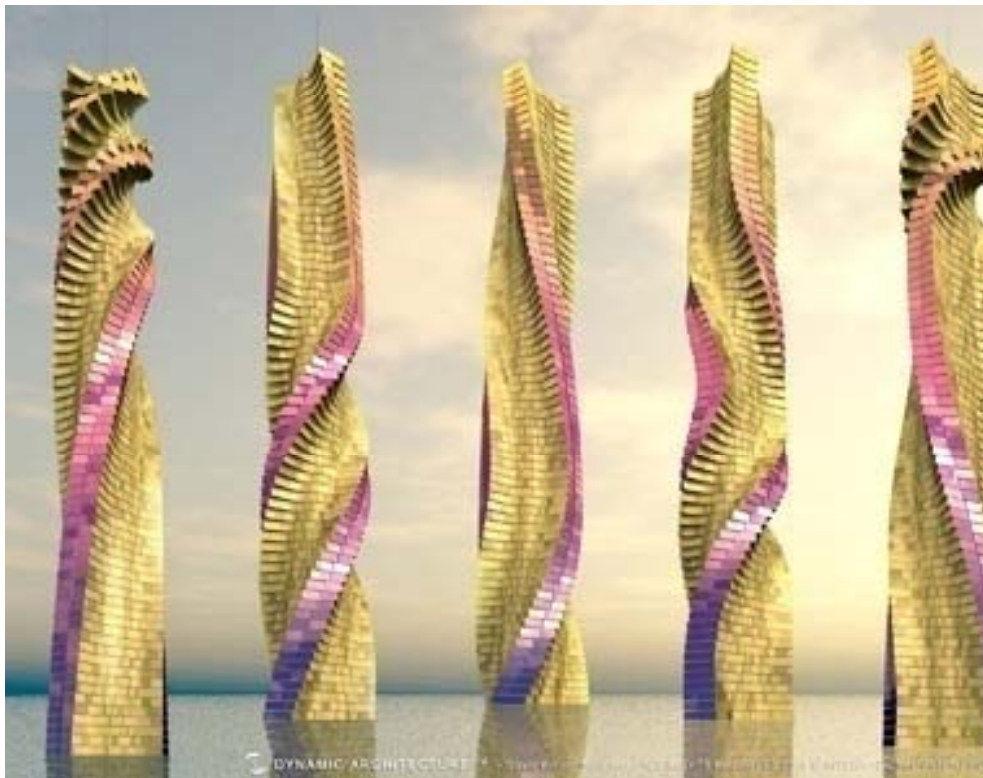


100 years of living science

100

We live in a composition world...

Dynamic Architectures...



Dynamic Architecture, www.dynamicarchitecture.net, 2008.

The building blocks...

For the rotating tower...

- 90% modules pre-constructed
 - One floor installed every 3-6 days
 - Floor types:
 - Office, Hotel, Apartment, Penthouse...
 - Component types:
 - Kitchen, Toilet, Bedroom, Lounge etc....
- Each floor spins, moves and rotates 360degs independent of one another
- Future: **Rotating City!**

The problem is in the plumbing...

- Composition of Software
 - Technology
 - System Integration
 - Protocols
 - Reuse (Function)
 - Features, Processes, Semantics and Properties
 - Consistency and Correctness (Testing)

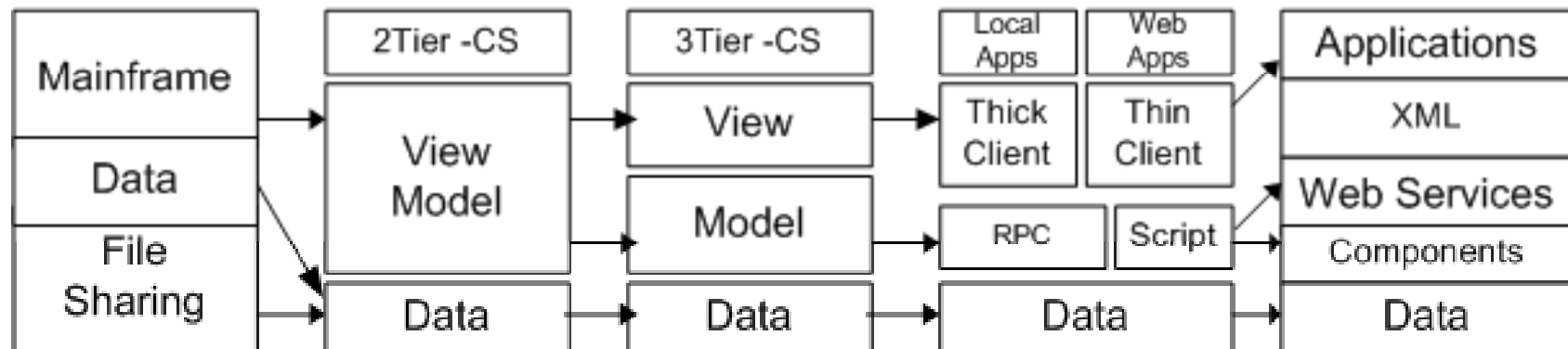


Challenges in Service Development

Services interface components...

- Development
 - Validation (is it the right solution?)
 - Verification (did we build it correctly?)
 - Maintaining the above with change
- Distributed Computing Emphasises
 - Coordination, Transactions and Concurrency
 - Change Control
 - Quality of Service

From Mainframe to (Web) Services



- Main Focus has been on Architecture
- Also need Behaviour and Policy
- Standards for describing integration have relieved some of the technology issues

Standard ways to describe things

Layer		Standards (W3C/OASIS)	Software Process	
Compositions	Policy	WS-Policy, XACML	Verification (Properties)	Validation
	Choreography	WS-CDL, WS-CI	Behaviour (State)	
	Transactions	WS-Transaction		
	Orchestration	BPEL, WSCI		
Services	Interface	WSDL	Identity	
	Message	SOAP		
	Format	XML Schemas	Input and Output	
	Data	XML		

Models provide an abstraction for..

- Service Configurations
 - Architecture – Component Configurations
 - Behaviour – e.g. Orchestration in WS-BPEL
 - Policy – e.g. Choreography in WS-CDL
- Assurance through Formal Analysis
 - Design vs. Implementation
 - Interactions and Obligations
 - Runtime (Execution vs. Design)

Gain greater assurance of correctness

- Behaviour Analysis
 - BPEL, CDL, WSIF, UML to FSP [Foster 03-08]
 - BPEL to Petri-Net [Hamadi04, Ouyang05]
 - BPEL to LOTOS [Salan 04]
 - BPEL to Promela / SPIN [Fu04]
- Properties
 - Safety Analysis (Deadlock Freedom)
 - Liveness Analysis (Uphold Assertions)
 - Composition of Models? A challenge...

Composition of Models

Lets consider behaviour analysis...

- Take a 2-dimensional view

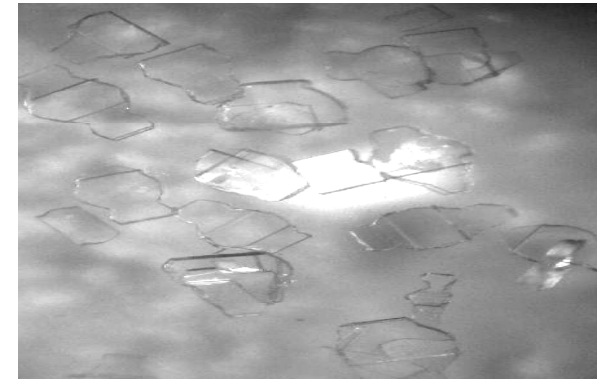
<i>Artifact</i>	<i>Design</i>	<i>Impl.</i>	<i>Arch.</i>	<i>Deploy.</i>
Orchestration	MSC	WS-BPEL	Process	Engine
Interface	Function	WSDL	RPC	Host
Choreography	MSC	WS-CDL	Policy	Engine
Resources	UML**	Threads	Runtime	Server

- Goal: Compare each of these with each other using model verification and validation (e.g. animation)

A Challenge: Composing Models for Analysis

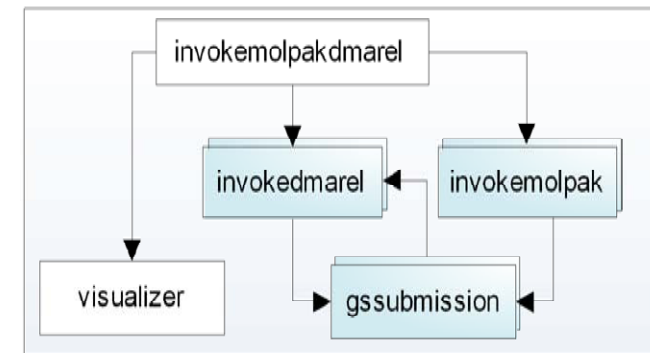
■ UCL Chemical Grid Comp.

- **Predict polymorphs (defects) in organic crystal structures**
- **Molpak** – analysis of 200 structures for 38 types of calculation
- **Dmarel** – calculates physical props. of structures



■ Detecting polymorphs

- clients **invoke** search and analyse structures
- **Submits 200** jobs for 38 types
- Up to **7,600 concurrent** interactions (ranging 5mins-1hr in duration)



W.Emmerich et al. **Grid Service Orchestration using BPEL4WS, Journal of Grid Computing, 3(3-4), 283-304. 2005*

A Test Case

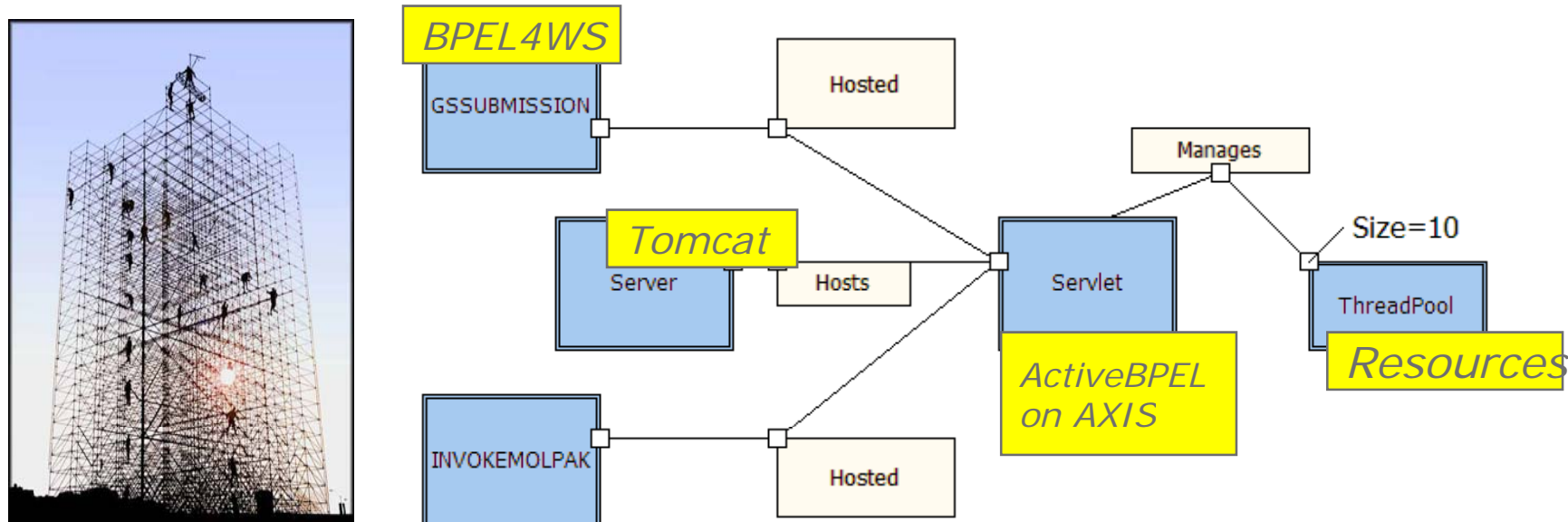
- We model checked each service
 - **Individually** for process deadlock freedom
 - In a **choreography** (for interactions between services)
 - **Deployed** solution to one web server and service engine host
- After some time, **DEADLOCK!**
 - An exhaustion of thread pool resources caused by a **complex interplay** between orchestrations, thread mechanisms, web service synchronisation and co-locations.
- Services Architecture Verification*
 - Emphasises **multiple-service hosting** and **intensive process requests** and **resource sharing**.

Howard Foster, W.Emmerich, Jeff Kramer, Jeff Magee, D.Rosenblum and Sebastian Uchitel. **Model Checking Service Compositions under Resource Constraints, ESEC/FSE 2007, Dubrovnik, Croatia. 2007*

Step 1. Model The Architecture

■ Abstraction for Architecture

- A **set of components** and **stereotypes** (in xADL2* or UML)
- **Attributes** (name of process, size of threadpool etc)
- **Relationships** between host (servlets), processes and resources

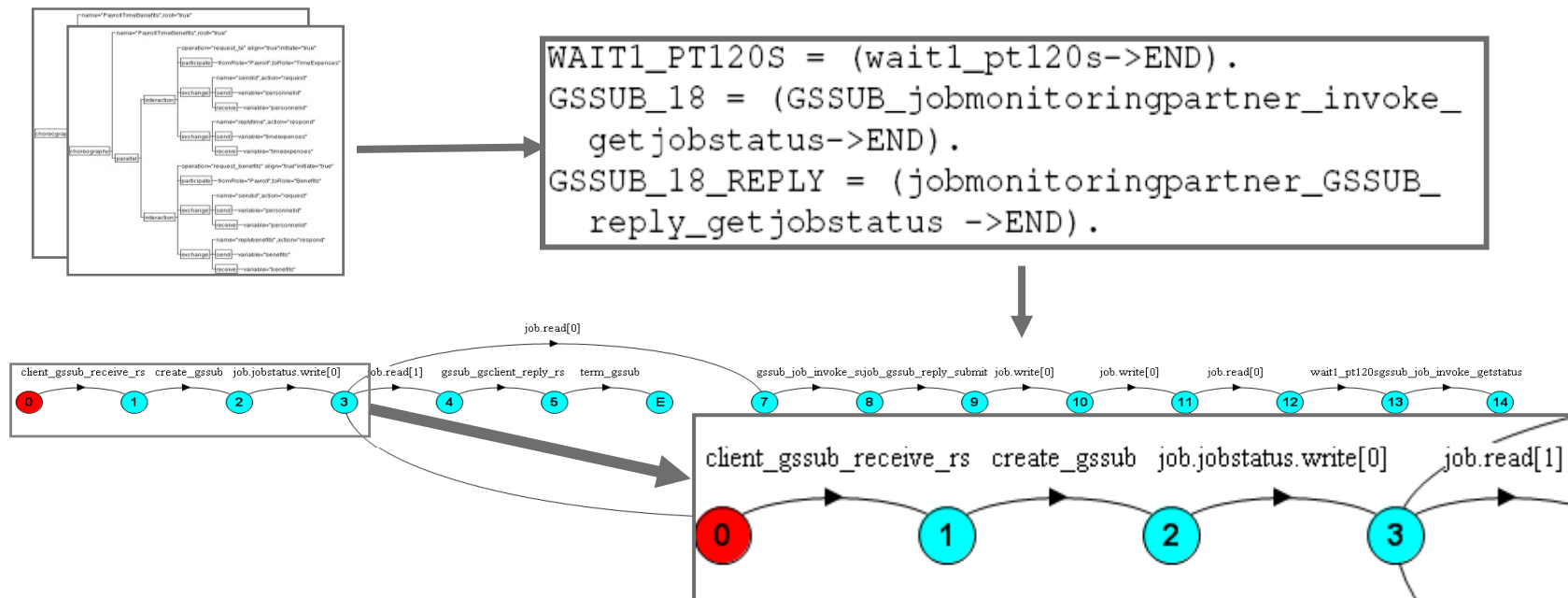


**E.M.Dashofy et al, xADL2 – A highly-extensible XML-Based Architecture Description Language, 2001.*

Step 2. Model The Processes

■ Abstraction for Behaviour

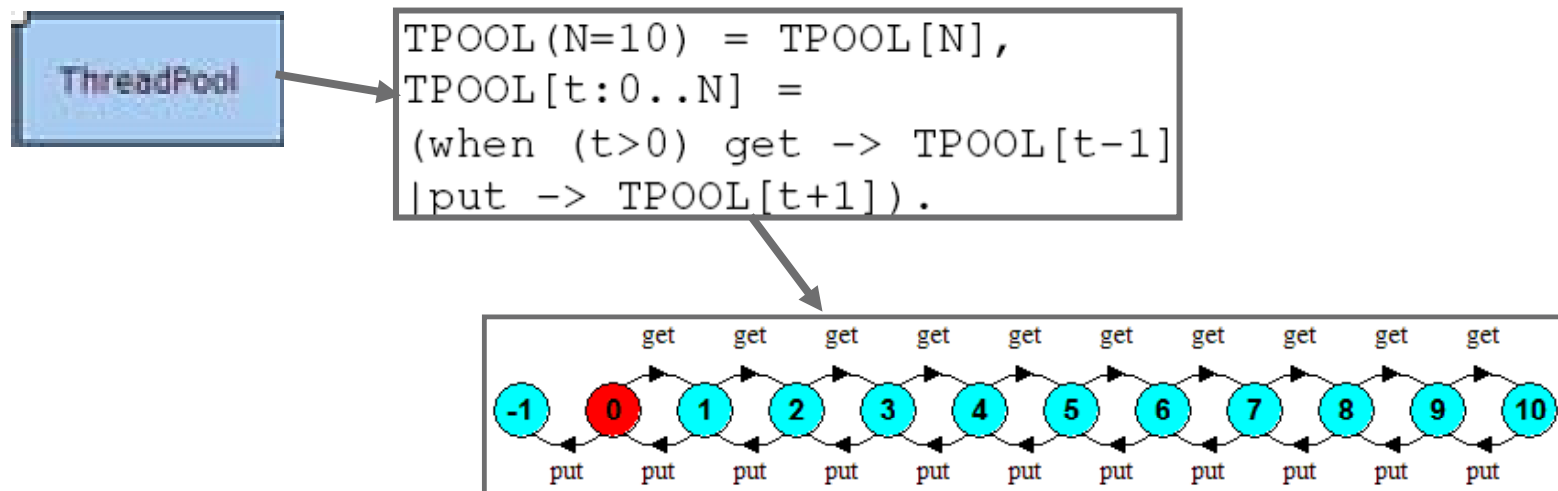
- Elements of service workflows (from WS-BPEL)
- **Each workflow translated to Finite State Processes (FSP)***
- **FSP compiled to Labelled Transition System (LTS)***



*Jeff Magee and Jeff Kramer, *Concurrency – State Models and Java Programs – 2nd Edition*, John Wiley & Sons Inc., 2006.

Step 3. Model The Resources

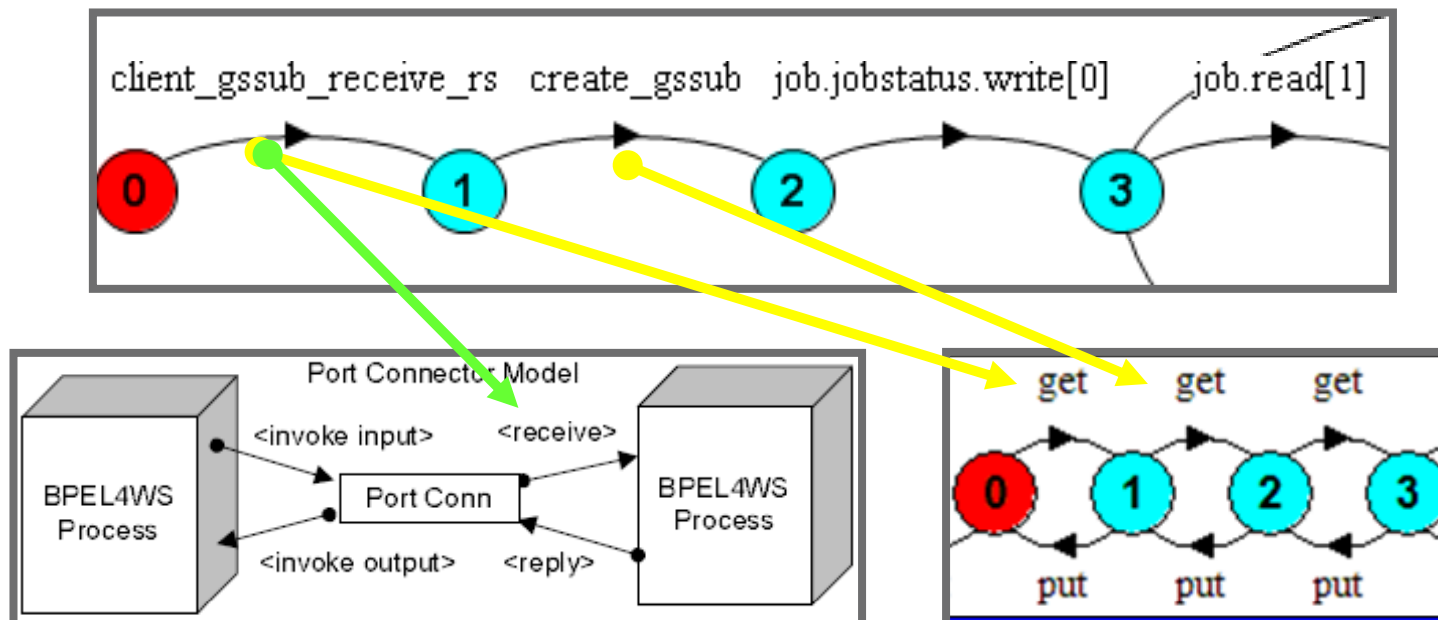
- Abstraction for Resource Usage
 - Elements of resource management (e.g. thread alloc/dealloc)
 - **A thread pool resembles a stack type (FSP process)**
 - **Get/put operations to acquire and release a thread**



- **But, need to abstract thread pool for practical size!**

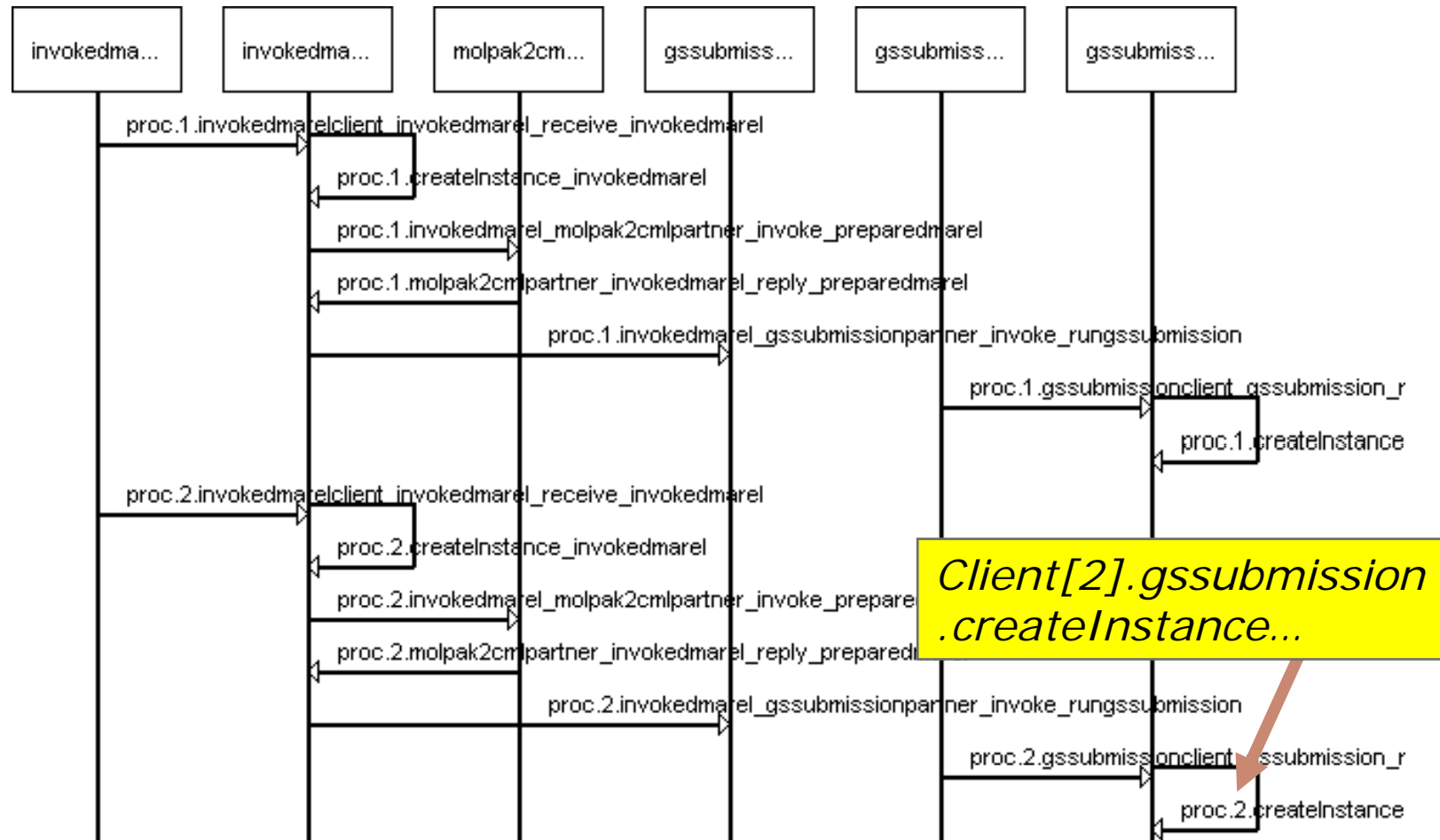
Step 4. Build the Analysis Model

- Link Models for Analysis
 - Workflow choreography
 - Behaviour and resource management (activities to get/put)
 - **Add some final links (clients, process instances etc).**



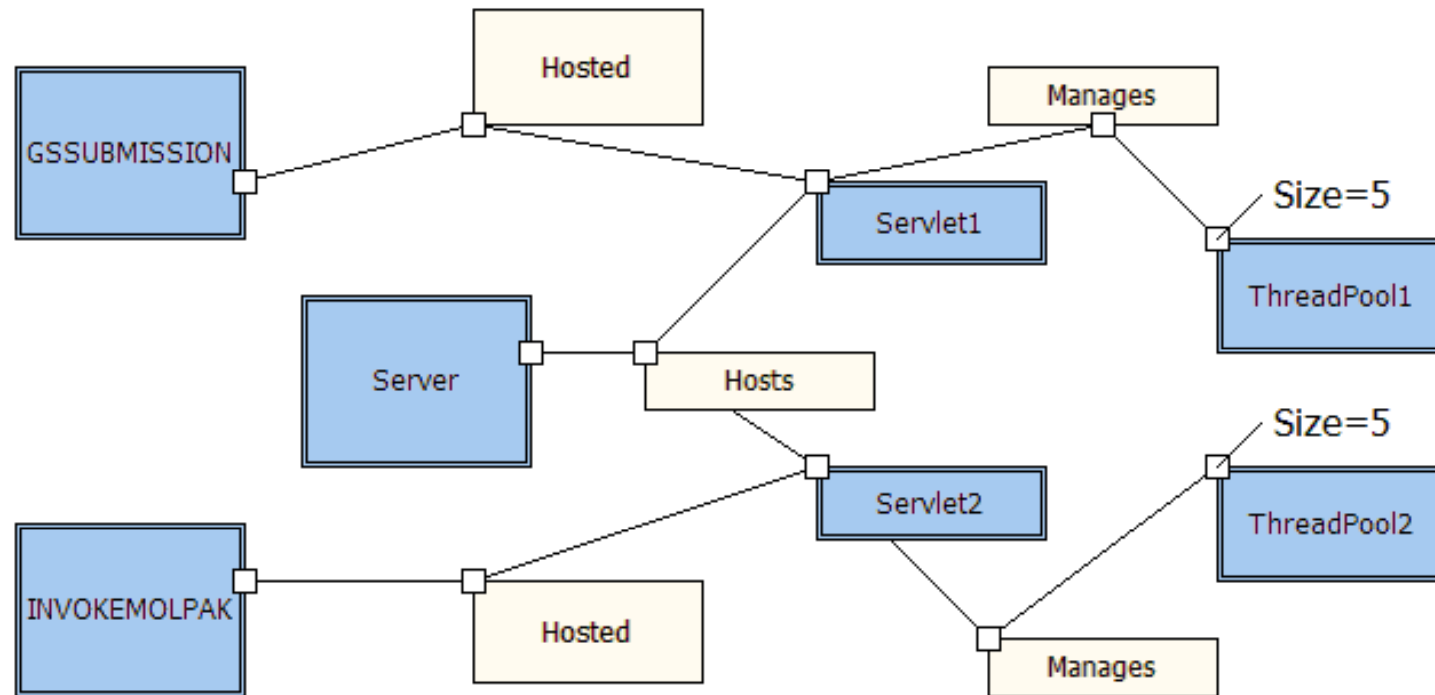
Step 5. Analyse the Model

- Run safety check for deadlocks



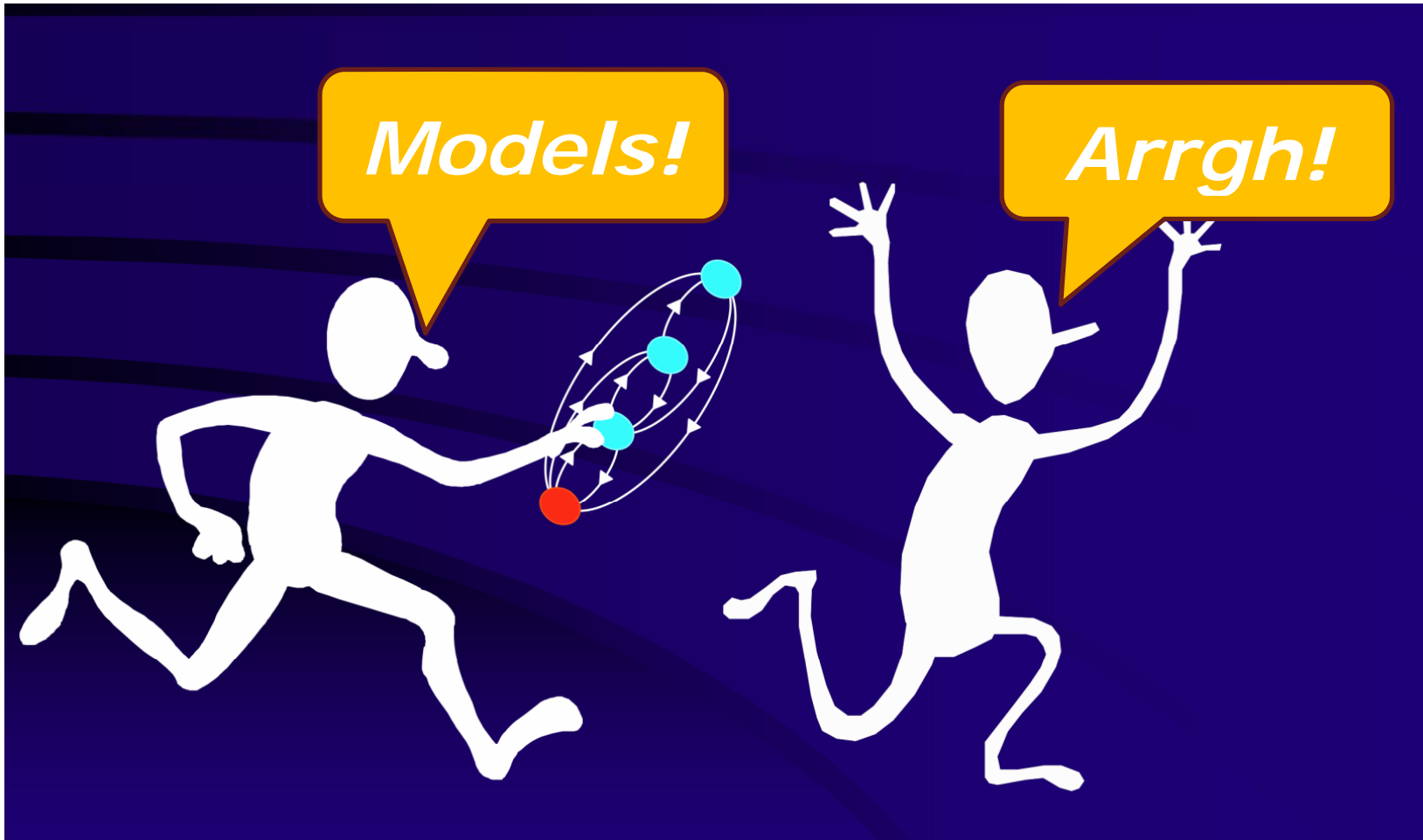
Towards a Solution

- Increase Resource Pool Size?
 - or reconfigure the architecture?



```
||DEPLOY_ARCH2 = (Client[c]:Arch_Comp || p1:TPOOL(5)|| p2:TPOOL(5))
```

Motivation for Tool Support and Demonstration



WS-Engineer & LTSA Eclipse

■ LTSA

- Labelled Transition System Analyser
- Developed at Imperial College London
- Concurrent software architecture analysis

■ WS-Engineer

- Built on LTSA for analysis of: WS-BPEL/CDL/IF etc
- Design, Interactions, Choreography, Architecture, Deployment

The screenshot displays the Eclipse IDE in the LTSA Perspective. The top window shows a transition system diagram with nodes 3, 4, 5, E, 7, and 8. The LTS Animator window on the right lists various process actions, with 'proc.1.createInstance_invoked' checked. Below this, the hMSC diagram shows interactions between 'proc.1' and other components. The BPEL Editor window at the bottom shows the XML code for a BPEL process named 'GSSubmission'.

Reading Material

Test and Analysis of Web Services

Baresi, Luciano; Di Nitto, Elisabetta (Eds.)

ISBN: 978-3-540-72911-2

- Monograph of 15 papers
- Analysis
 - Conversations, Choreography...
- Testing
 - Unit, Discovery, Regression...
- Monitoring
 - Run-time, WS-Agreements...
- Reliability, Security and Trust



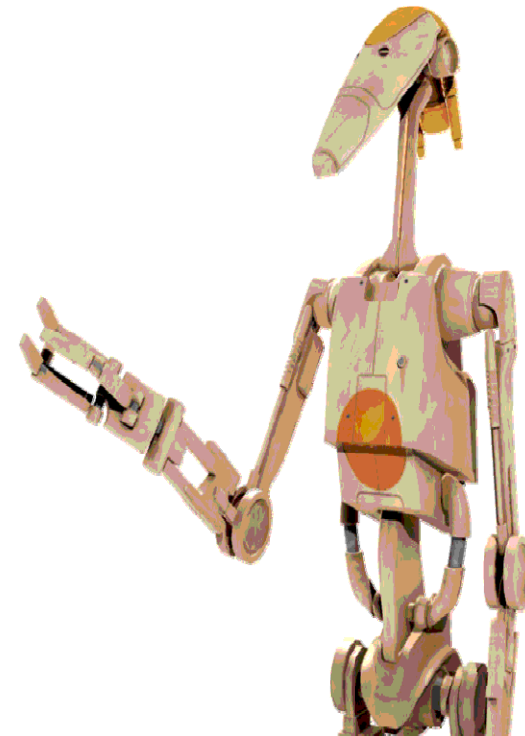
**Howard Foster, Jeff Magee, Jeff Kramer and Sebastian Uchitel.*
***WS-Engineer: A Model-Based Approach to Engineering Web Service
Compositions and Choreography***
(chapter 1: Analysis)

From Models to Self-Management

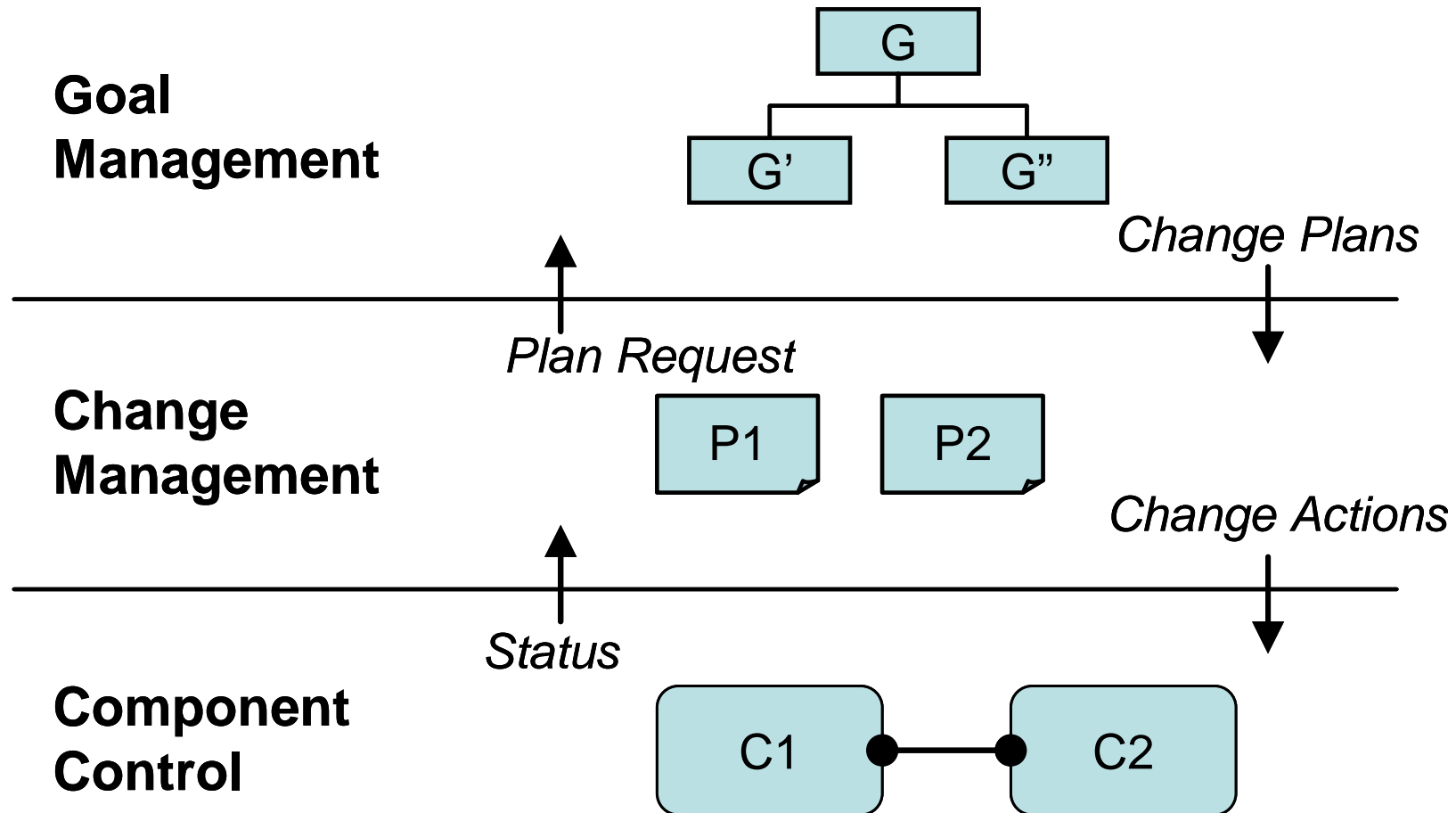
- Simple View
 - A single architecture of service composition
- Dynamic Multi-Arch View
 - Re-configuration
 - Changes to architecture and behaviour
 - Adaptive
 - Re-acting to events in service environment
 - Autonomous
 - Follow some goals (policy) towards a solution (expecting change)

Control, React and Deliberate

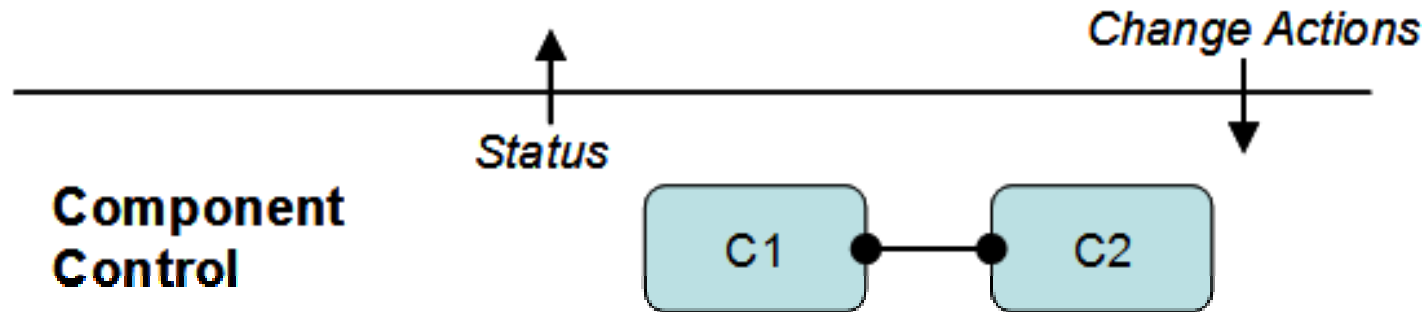
- Service systems should:
 - Change/update behaviour dynamically in response to changes in goals & environment without operator intervention
- Problem: Specification!
 - Self-
 - assembling
 - healing
 - optimising



Three-Layer Architecture for Self-Management

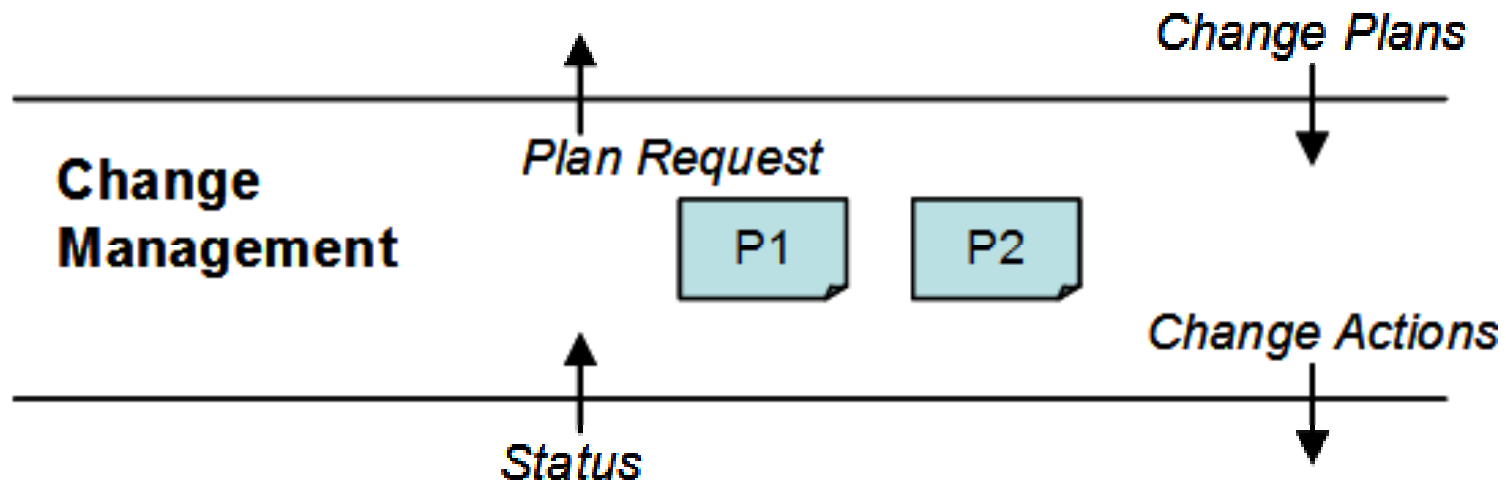


Component Control



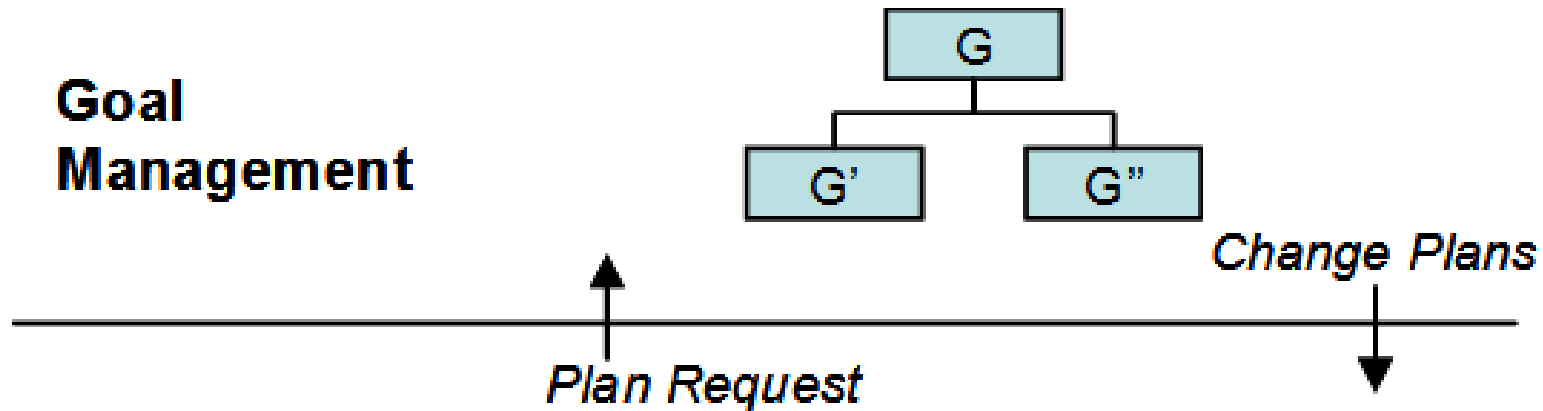
- Service Layer Supports
 - Service creation, requests, deletion.. Etc
 - Event / Status reporting during change
 - Probes and Effectors (controllers)
- E.g. Services can be self-optimized
 - Dynamic Service Brokering and Bridging
 - Quality-of-Service monitoring (status)

Change Management



- Service Layer Supports
 - Plan execution in response to predicted events / state changes in component layer
 - Plan update and execution in response to changing goals.

Goal Management



- Service Layer Supports
 - Execution of services in respect to Goals
 - Planning considers system capabilities and goals (output to Change Plan)
 - Addition and removal of goals

Planning Techniques

- Artificial Intelligence
 - Planning with Learning
 - Improving with experience at some task
 - Improve over task **T**
 - With respect to performance measure **P**
 - Based on experience **E**
- For Services
 - Adapting composition for changing goals
 - Discovering services to fulfill goals
 - E.g. Semantic Web, other examples...
 - Learning from existing service use?

Research Challenges

- **Component Control**
 - Safe Operation During Change
 - Stable Conditions (Kramer&Magee86)
- **Change Management**
 - Verification during change (Zhang&Cheng06)
 - Scalability in distributed change management and de-centralisation
- **Goal Management**
 - Precise specification of goals
 - Application, system, architecture constraints
 - Planning techniques (Pistore et al.05)

SENSORIA....



- Service-Engineering for Service-Oriented Overlay Computers
 - 13 Universities, 2 Research Institutes and 4 Companies from 7 EU Countries
 - 12 Work Packages (languages, models, properties, deployment, case studies etc)
 - **London Software Systems (LSS)**
 - Imperial College London
 - University College London
 - “Enhanced Specification and Deployment Mechanisms for Service Interactions, Composition and Self-Management”

Towards Self-Management in SOA

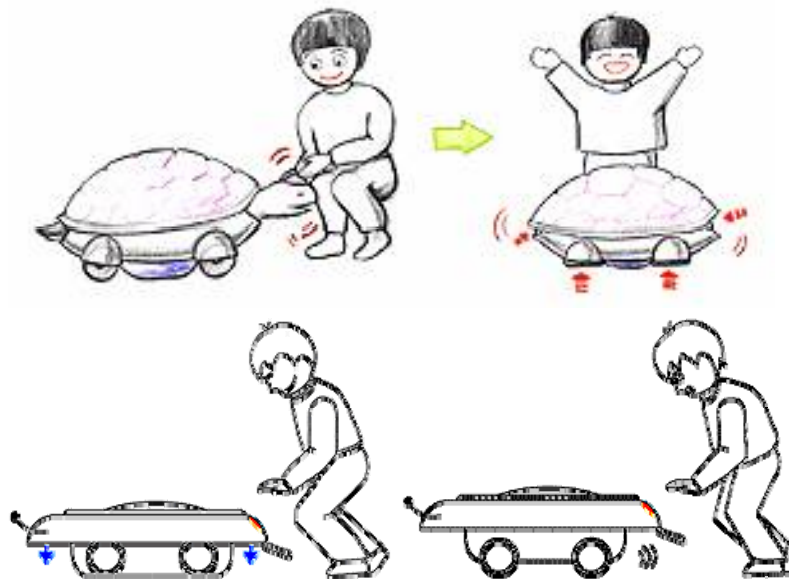
- We follow the notion of Modes *
 - Architecture, Behaviour and Policy
 - A set of Architecture Models
 - A set of Behaviour Models
 - Constraints (e.g. Mode Changes, QoS)
- Objectives
 - Combining different modes of operation
 - Verification of switching modes (behaviour)
 - Useful for SOA Deployment Artefacts?

**Dan Hirsch, Jeff Kramer, Jeff Magee and Sebastian Uchitel.
Modes for Software Architectures, EWSA 06, Nantes, France*

■ What are modes?

- 1) a way of operating, living or behaving
- 2) an operational state that a system has been switched to...

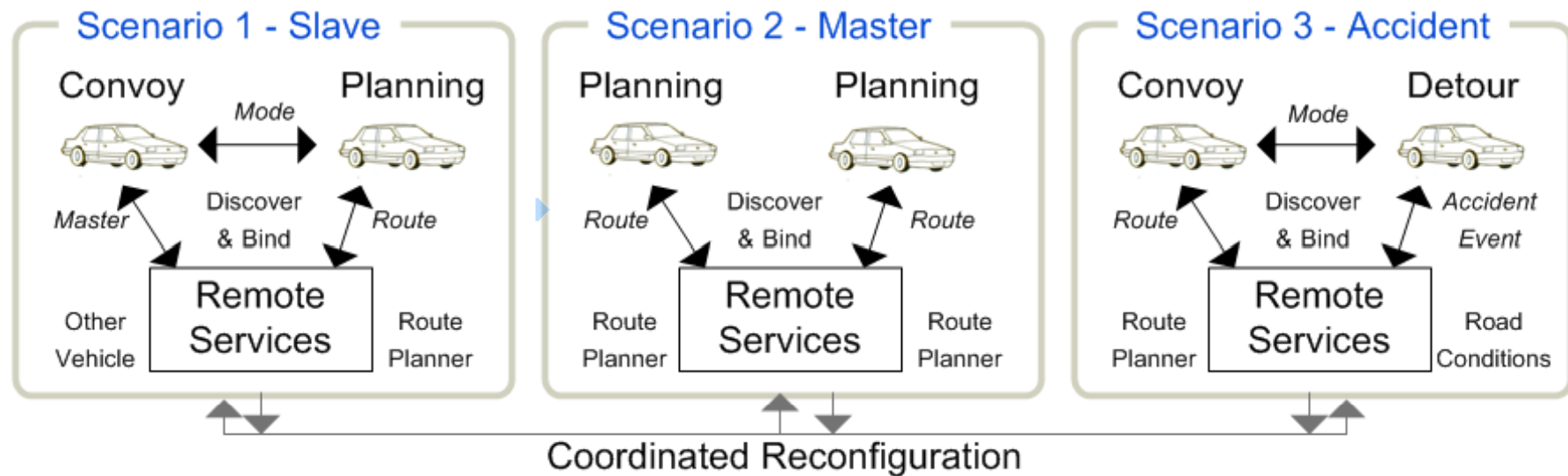
WHEN THE CHILD
INTERACTS WITH
THIS PRODUCT



<u>Mode</u>	<u>Event</u>	<u>Reconfiguration</u>
Cleaning	touch on head	stops
	pulls tail	stops and changes to Instrument mode
Finished Cleaning	End cleaning	Stops and changes to Finished Cleaning mode
	touch on head	Changes to Cleaning mode
Instrument	pulls tail	changes to Instrument mode
	Timeout	stops and changes to Cleaning mode

From "MusicCleaner", Illinois Institute of Design, 2007

Automotive Services Modes Example



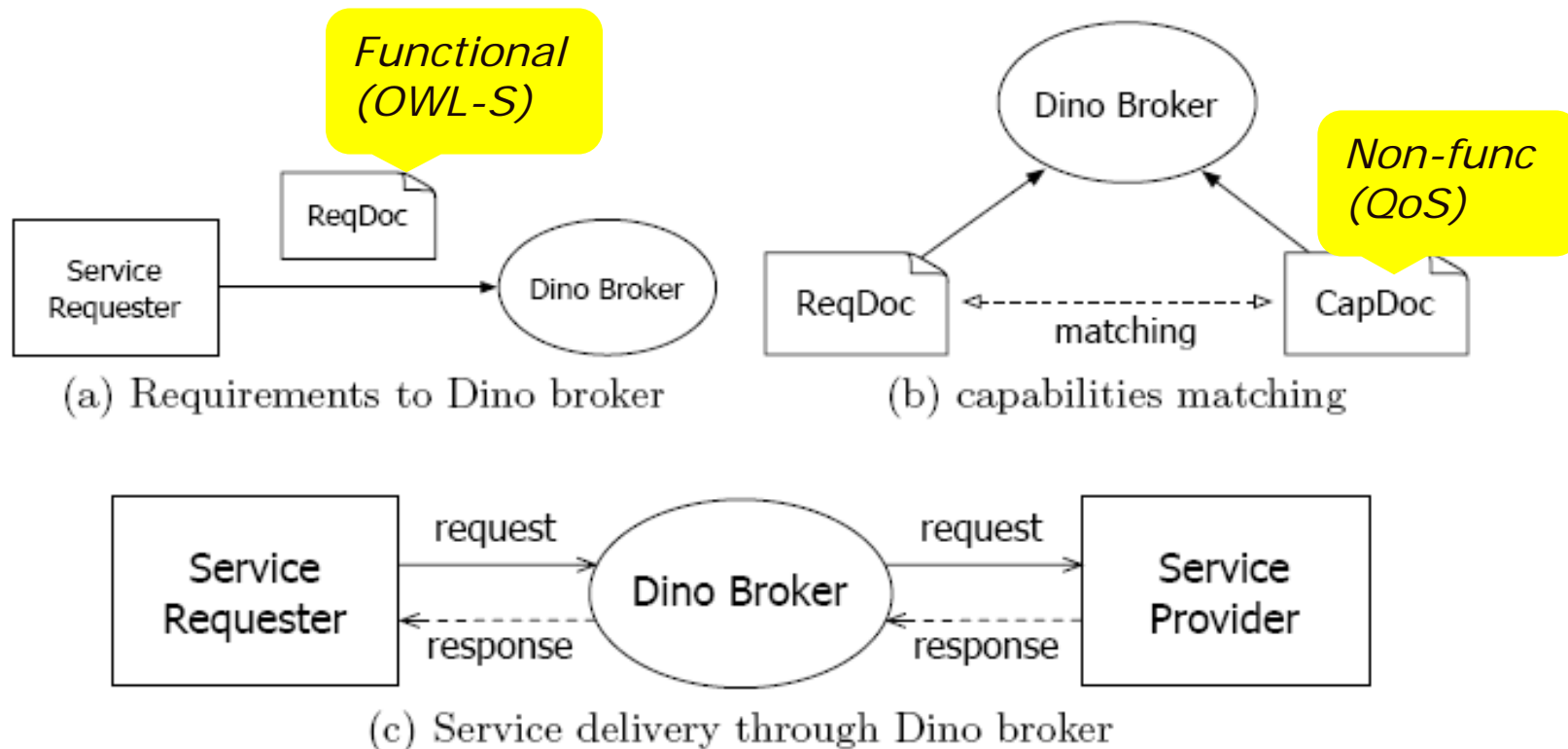
*Nora Koch and Dominik Berndt, **SENSORIA D8.2a: Automotive Case Study: Requirements modelling and analysis**, October 2007

What have we done so far?

- A Focus on Service Control and QoS
 - Modes
 - Practical example for adaptive service brokering
 - Seek to generate behaviour processes e.g. BPEL
 - Adaptive Service Brokering (UCL)
 - **Dino** - Dynamic and Adaptive Service Broker
 - Service Discovery, Selection, Binding + Management
 - Functional and non-functional adaptation
 - **Dino**+Modes – used to extract
 - Services required for a given mode
 - service capabilities required and provided
 - functional and non-functional Dino inputs (OWL-S/QoS)

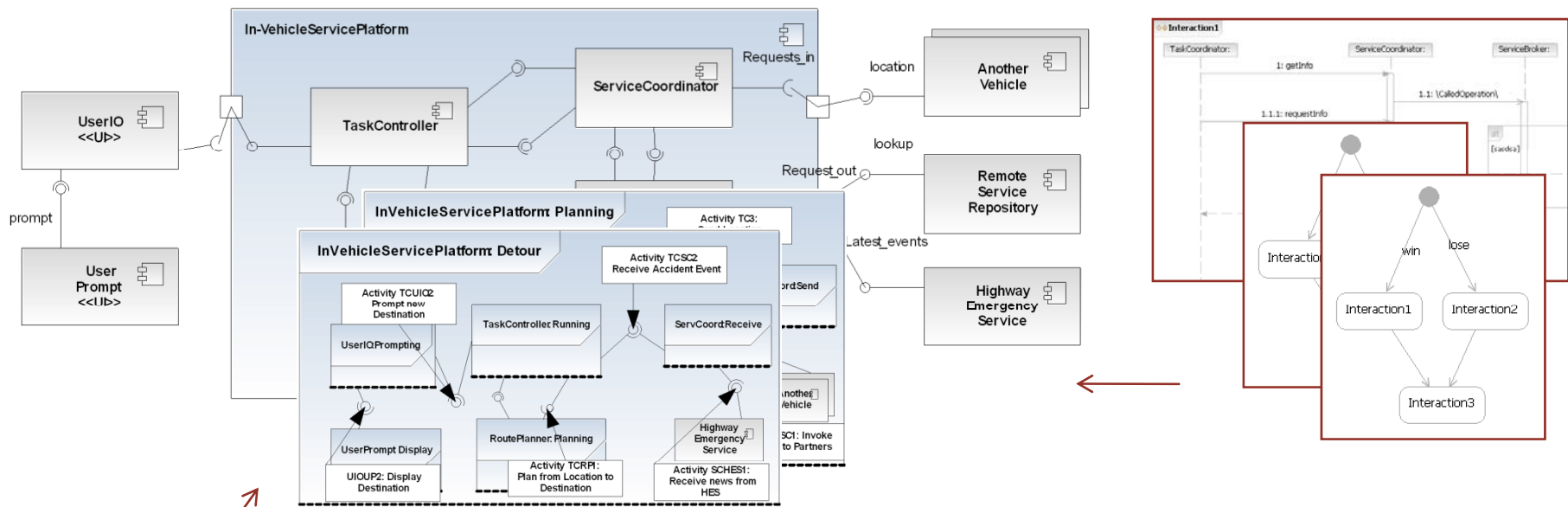
Dino – Dynamic and Adaptive Service Brokering

- Dynamic, Automated, De-centralised
 - Middleware service broker for service
 - Discovery, selection, monitoring etc

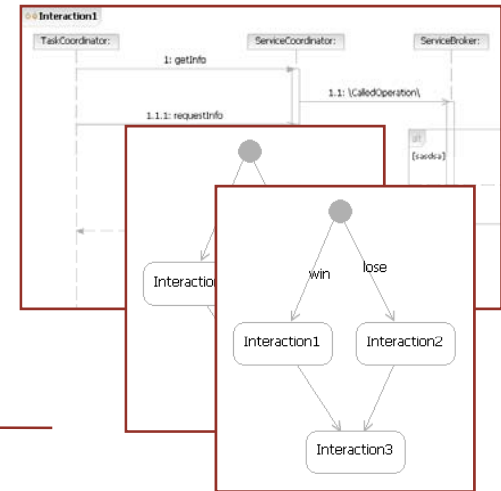


Modes Specifications

A Combination of Elements



Behaviour



Constraints

```

ModeChangeConvoy_Constraint
{context this::changemodeConvoy()}

ModeChangeConvoy_Constraint
{context this::changemodeConvoy()}

ModeChangeConvoy_Constraint
{context this::changemodeConvoy()}
pre InVehicleServicePlatform.Mode not convoy and
RoutePlanning.Mode not planning
post InVehicleServicePlatform.Mode = convoy
}
    
```







Distributed Management Policy

```

<!-- Create a policy -->
<create type="obligation"
event="/Event/toohigh"
active="true">
  <!-- We need this arg -->
  <arg name="msg"/>
  <!-- Do this action -->
  <action>
    <use name="/alarm"
alarm="on"
title="!msg;"/>
  </action>
</create>

<!-- Create a policy -->
<create type="obligation"
event="/Event/toohigh"
active="true">
  <!-- We need this arg -->
  <arg name="msg"/>
  <!-- Do this action -->
  <action>
    <use name="/alarm"
alarm="on"
title="!msg;"/>
  </action>
</create>
    
```

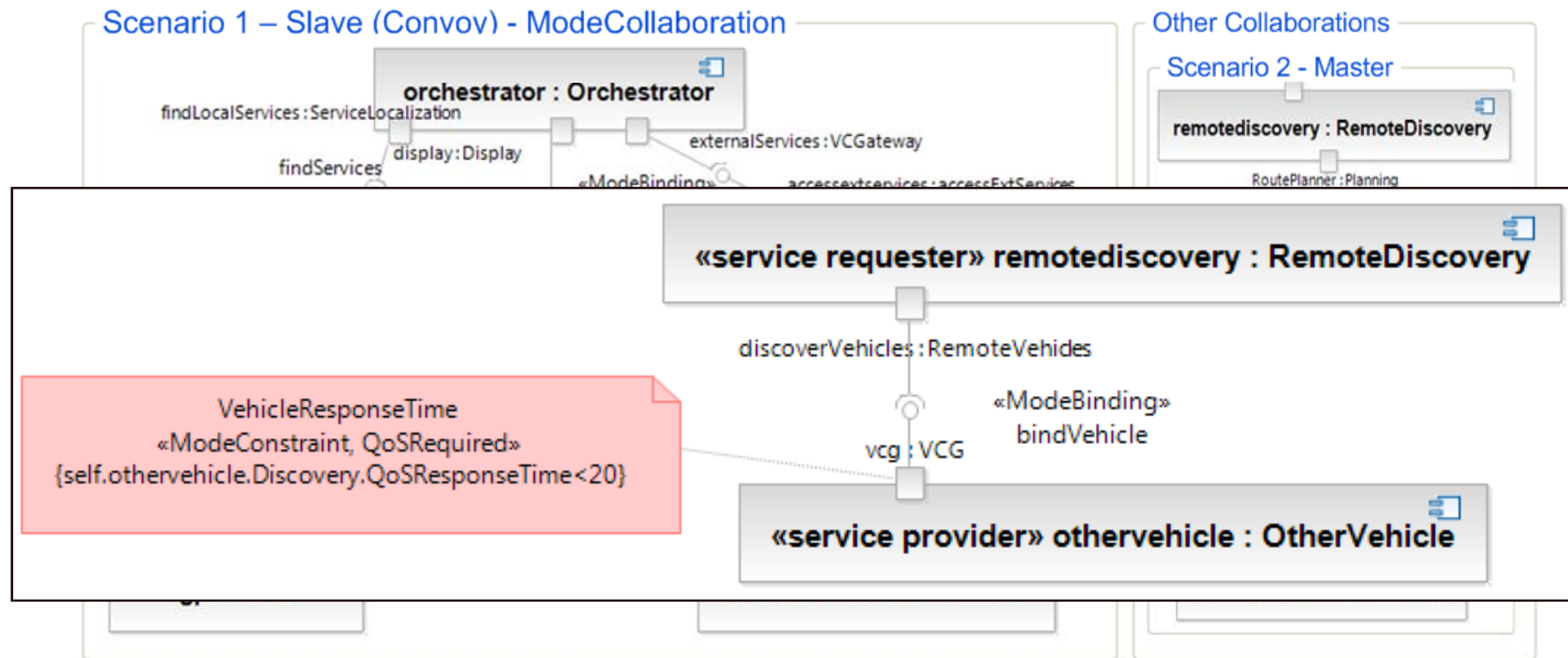
- Leverages
 - UML4SOA*
 - Provider
 - Requester
 - Types etc
- Mode- **
 - Collaboration
 - Bindings
 - Constraints

<i>Icon</i>	<i>StereotypeName</i>	<i>UMLBaseClass</i>
	ModePackage	Model, Package
	ModeBinding	Connector
	ModeCollaboration	Collaboration
	ModeInteraction	Sequence Chart
	ModeActivity	Activity
	ModeConstraint	Constraint

*N.Knoch, P.Mayer, R.Heckel, L.Gonczy and C.Montangero. **D1.4b: UML for Service-Oriented Systems**, Oct 2007.

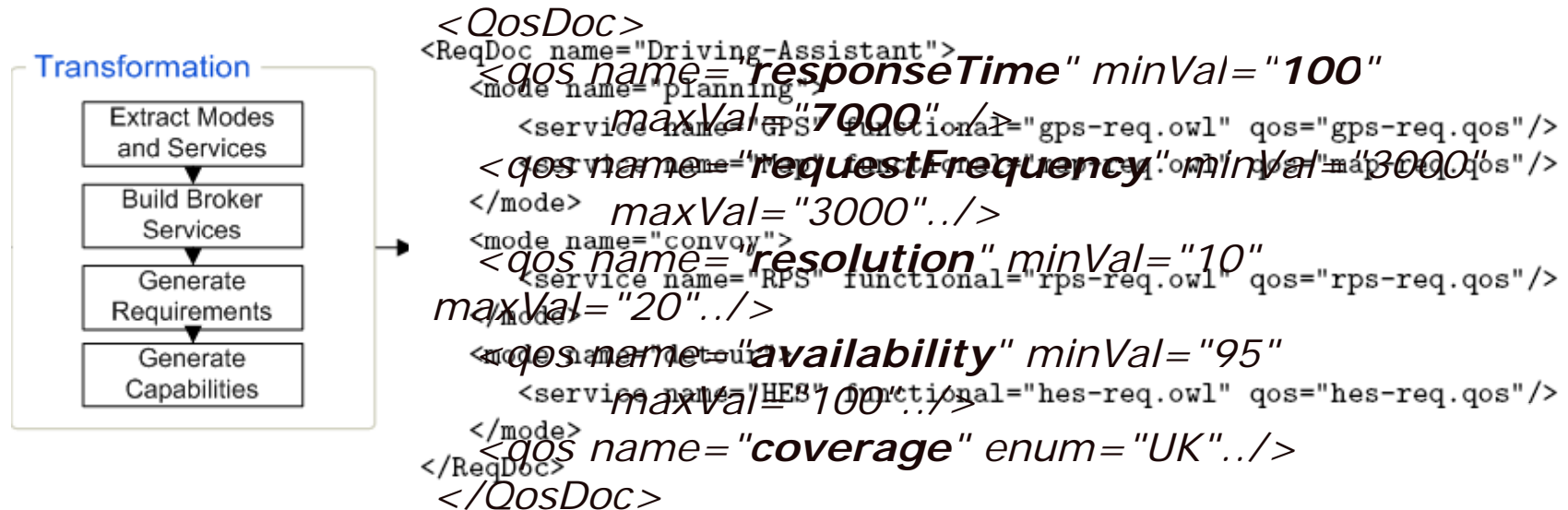
H.Foster, S.Uchitel, J.Magee and J.Kramer, **Leveraging Modes and UML2 for Service Brokering Specifications, MDWE 2008.

UML Profile for Software Architecture Modes



- For adaptive service compositions
 - Constrain bindings between service requesters / providers
 - Generate broker specifications for runtime

Generating Broker Requirements



- Generates Dino ReqDoc specification
 - The set of modes and the services required in each mode
 - Each service has functional and non-functional specification in a given mode

DinoModes Tool (at a glance)

The screenshot displays the DinoModes Tool interface, which is divided into three main sections:

- Package Explorer:** Shows a tree view of project files. The 'AutomotiveModes2' folder is expanded, listing various XML files such as 'AutomotiveCS.xmi', 'AutomotiveModes2.xmi', 'BM.profile.xmi', 'Default.profile.xmi', 'Deployment.profile.xmi', 'ModesProfile.profile.xmi', 'OnRoadRepair.xmi', 'ProfileBase.profile.xmi', 'RUPAnalysis.profile.xmi', and 'UML4SOA_09.profile.xmi'.
- DinoModes Browser:** Displays the 'Dino Modes Browser' window. It shows a 'Specification' section with dropdown menus for 'Capabilities' (Requirements), 'Mode' (Convoy, Detour), 'Required Services' (DriverVehicleUI, HWEmergency), and 'Functional/QoS' (HWEmergency.owl, HWEmergency.qos). The 'Dino Specification' section shows the XML content of the selected specification, including headers like '<?xml version="1.0" encoding="ISO-8859-1"?>' and various DOCTYPE and ENTITY declarations.
- Dino - Console:** Shows a table of sessions and a log of actions. The 'Sessions' table has columns for 'Session ID', 'Mode', and 'Action'. The 'Log' table has columns for 'ID', 'Action', 'Execution Time', 'Input', and 'Output'.

Session ID	Mode	Action
3d138763:11a44fb545c:-7ff9	default	-

ID	Action	Execution Time	Input	Output
0	Registering ReqDoc	5410626910	C:\dev\eclipse33\r...	
1	Selecting mode	2382984	Selecting mode: [d...	default
2	Invoking service	19251011022	priceFinder[http://...	file:/C:/Downloads...

Analysis of Software Architecture Modes

- Need a composition of service models
- Challenges in:
 - Generating new architectural configurations as goals change
 - Behavioural safety and stable conditions for change
 - Consistency of Policy (Rules/Triggers) etc

Analysis of Software Architecture Modes

- **Architecture** (e.g. Alloy)
 - valid structure and evolution of (mode) models
 - can be used to generate sample instances of architectural configurations which meet goals
- **Behaviour** (e.g. MSC->FSP, Darwin)
 - Translate to process algebra and check for safety
 - Translate to ADL and model check for consistency across bindings and modes
- **Policy** (e.g. Ponder2->FSP)
 - Are policy constraints achievable with all modes?
 - Where does my policy conflict in a scenario?

Assumptions and Limitations

- Specification is not exhaustive
 - i.e. partial specifications – “closed-loop theory”?
 - What about the “unknown in advance” situations?
 - How does this affect correctness and consistency?
- Service Semantics
 - Service semantics and context known in advance
- Generating deployment artifacts
 - Do we have sufficient detail to generate runtime artifacts?
 - Complexity of scenarios and modes to “workflows”

Imperial College
London

Thank you

Google™

“Howard Foster Imperial”



100 years of living science

100