

# Could Message Hierarchies Contemplate ?

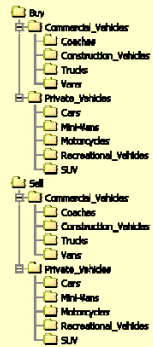


National University of Ireland, Galway  
Ollscoil na hÉireann, Gaillimh

Edward Curry – edward.curry@nuigalway.ie  
**Enterprise Computing  
Research Group**  
Department of Information Technology  
http://ecrg.it.nuigalway.ie

Author will be in attendance at the official poster session  
and at (most) coffee breaks during the conference

Author will be participating in the Doctoral Symposium  
**Channel Hierarchies**



Example Hierarchical Channel Structure

- Used in publish/subscribe messaging systems for static routing of messages
- Grouping of messages helps to manage large volumes of different messages
- Namespace schema needs to be well defined and universally understood
- Choosing a channel in to publish in is the responsibility of the publishing client
- *Channel selection logic* is integrated in the publisher's code, resulting in a tight coupling between the channel hierarchy and publisher

## Reflective and Adaptive Techniques

A reflective system is one that provides a representation of its own behaviour which is amenable to inspection and adaptation, and is causally connected to the underlying behaviour it describes

"Reflective flexibility diminishes the importance of many initial design decisions by offering late- and runtime-binding options to accommodate actual operating environments at the time of deployment, instead of only anticipated operating environments at design time."

- R. E. Schantz and D. C. Schmidt, "Middleware for Distributed Systems"

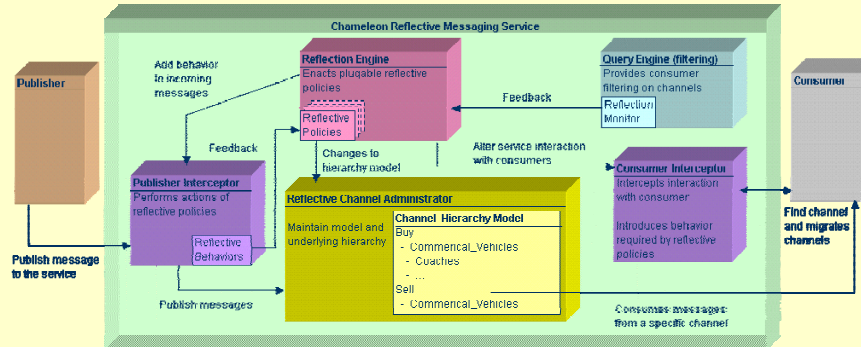


## Reflective Channel Hierarchies

A shift in responsibility of channel selection from the publishing client to the middleware service will allow the application of reflective techniques in channel hierarchies

Reflective and adaptive techniques can be used to dynamically grow and alter the hierarchy to meet the needs of its changing environment and operating conditions

Reflective Channel Hierarchies adapt the structure of their channel hierarchy to better represent its contents, and to meet the requirements of its users



## Chameleon Reflective Messaging Service Architecture Overview

### Reflective Channel Administrator

- Exposes a meta-model to express the structure of the channel hierarchy
- Meta-model is a causally-connected representation of the channel hierarchy; any alternations to the model will be replicated in the underlying hierarchy enabling its run-time inspection and adoption

### Reflective Engine

- Exposes a meta-interface for pluggable reflective policies
- Reflective policies control the reflective behaviour of the service. These intelligence policies contain the rules and strategy used in the adoption of the service. Policies can be designed with different motivations for the adaptation of the hierarchy, using different techniques to achieve them

### Publisher and Consumer Interceptors

- Mechanism used by the reflective engine to accomplish the work of its policies. Reflective policies can use these interceptors to dynamically introduce behaviour into the service

## Reflective Channel Hierarchies With Chameleon

- Publisher interceptor is responsible for publishing messages in the hierarchy
- Based on the rules of the active reflective policies the interceptor selects the most relevant channel for the message to be published in
- The resulting system is analogous to a post office: - messages are placed into a black box, how the messages are sorted (published) is of little concern to the sender

## Reflective Policies to Achieve Reflective Channel Hierarchies

Currently two policies have been identified to realise reflective channel hierarchies:

### - Consumer Filter Monitoring

Consumer filters are examined to expose common filtered fields/values. If a significant number of similar filters exist, the policy reacts by creating relevant channels for these messages

### - Message Traffic Pattern Analysis

Adapts the hierarchy based on a pattern analysis of messages submitted to the service



This work forms part of the V-LAB research project whose goal is to develop a system for the virtual brokerage, optimisation and management of road freight carriers. V-LAB combines diverse technologies to create a multi-agent software system that will facilitate companies competing for extra work convenient to their individual schedule. The system will be a real time broker that offers parcels up for delivery, and then allows companies to compete for these jobs by creating tenders for their delivery.

One aspect of messaging within V-LAB deals with job requests for the brokerage. A jobs location and completion time is an important factor for companies competing in this marketplace. Depending on the scenario of the deployment, the scales used to measure the location and times differ.

Continental Deployment	
<ul style="list-style-type: none"> <li>Germany                             <ul style="list-style-type: none"> <li>Berlin                                     <ul style="list-style-type: none"> <li>Within_1_Week</li> <li>Within_2_Days</li> </ul> </li> <li>Munich</li> </ul> </li> <li>Ireland                             <ul style="list-style-type: none"> <li>Dublin                                     <ul style="list-style-type: none"> <li>Within_1_Week</li> <li>Within_2_Days</li> </ul> </li> <li>Galway</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Country of origin and destination are of high priority to shippers, as not every company will service all the potential countries and regions</li> <li>• Completion time is measured in days and weeks</li> </ul>

Metropolitan Deployment	
<ul style="list-style-type: none"> <li>Over_10_KG                             <ul style="list-style-type: none"> <li>Within_1_Hour</li> <li>Within_2_Hours</li> </ul> </li> <li>Under_10_KG                             <ul style="list-style-type: none"> <li>Within_1_Hour</li> <li>Within_2_Hours</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Source and destination are less important as it is more likely for a city courier to service the entire city, criteria such as package size and speed of delivery become more important</li> <li>• Completion times in hours</li> </ul>

Alternative deployment scenarios with different domain specific criteria include Fruit, Oil, Gas, Automotive and several others.

A static channel hierarchy requires predefined criteria for every potential deployment scenario in advance. A reflective channel hierarchy allows the consumer to choose the criteria, enabling a channel hierarchy to evolve (or grow) and constantly adapt into one customised specifically for the deployment environment.

Starting from a single channel (seed) a customised channel hierarchy (tree) can grow based on the expressed requirements of its consumers.

All comments and feedback on this work is welcome, Please email the author or use the forms provided.



The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged  
Copyright 2002/2003 Enterprise Computing Research Group