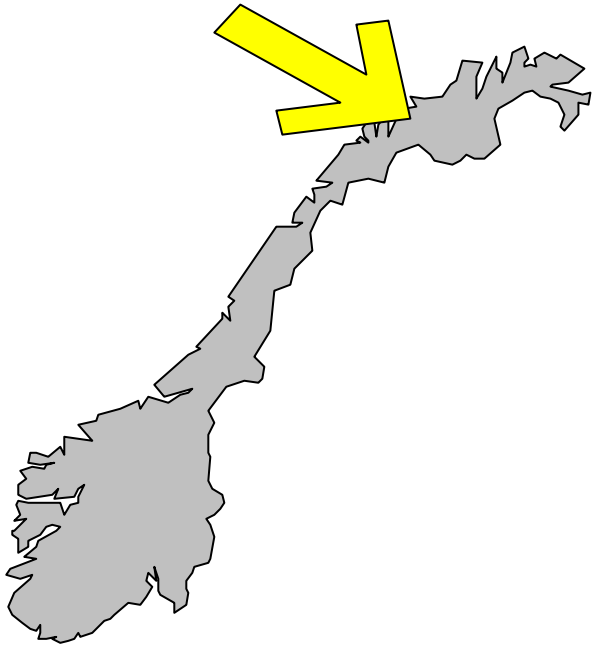
FlexiNet Binding Framework

Oyvind Hanssen*

14 October 1997

(*) *Seconded to the ANSA programme by the University of Tromso.*

(*) *Supported by NATO Science Fellowship, through the Norwegian Research Council grant no. 116590/410*



Motivation

We want flexible, policy-governed bindings!

- The FlexiNet Open ORB framework - an “extensible microkernel”:
  - Building blocks, core binding architecture
  - Special binding support may be added as extensions.
- Extensions that support:
  - Alternative policies for when bindings are activated, Lazy Binding
  - Explicit binding, QoS, non-functional properties
  - Policy-governed bindings in a wider sense

- Concepts -

Bindings and Activations

- Bindings:
  - The association of a “proxy” to the client program. . . .
  - A proxy know how to “reach” the remote interface
- Activation of binding:
  - Resources allocated to the object and the communication path between the client and the object so that invocations can be carried out.
  - Done according to some policy

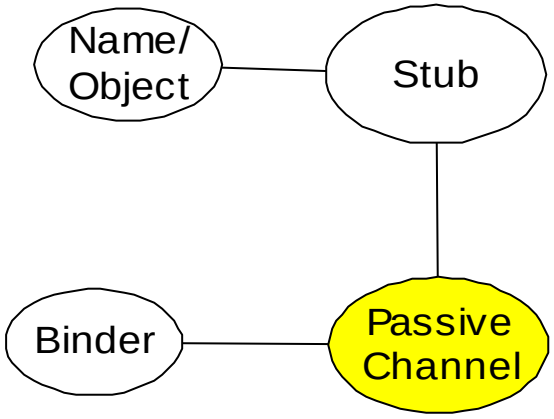
- Concepts -

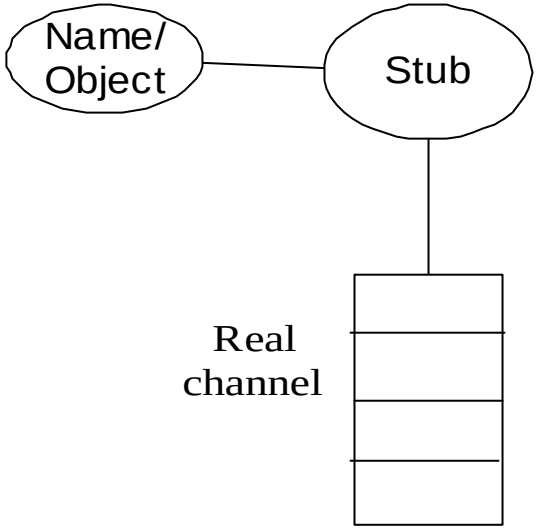
Policies and Metapolicies

- Policy:
  - Choice of protocols, resource management strategies etc.
  - Provides a certain set of non-functional properties (QoS)
- Metapolicy:
  - How activations of bindings are managed
  - Choice of and (possibly) dynamic replacement of policies
- Policy trading:
  - Declarative QoS requirement + environment description --> Policy
  - Service that looks up a software component . . .

- Engineering Model -

Simple Lazy Binding





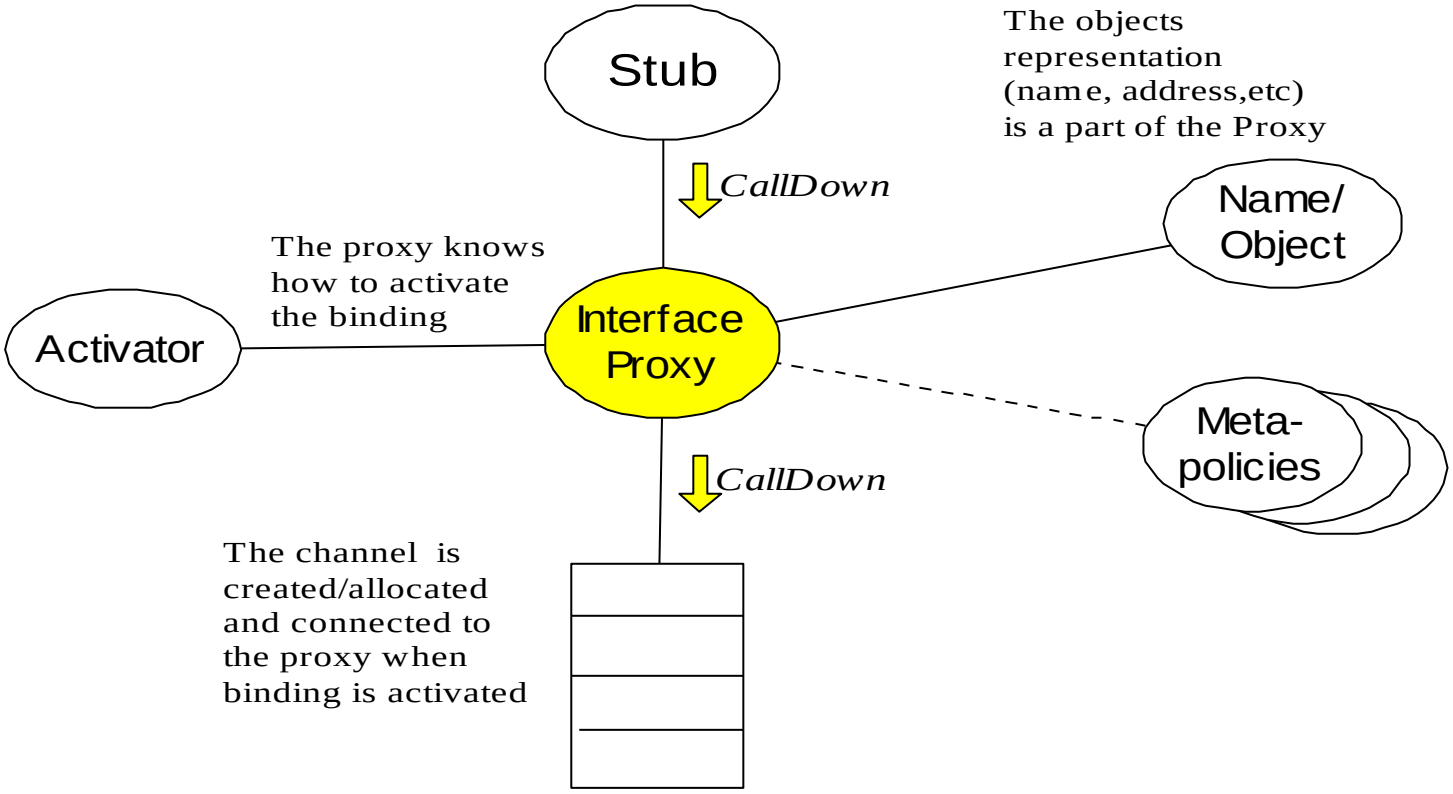
- Engineering Model -

A more flexible approach

- Interface proxies - represents the bindings
  - Transparency layer
  - Forwards downcalls to "channel", manages activation of channel
- Reflection: Most of the semantics of the proxy is defined in replaceable meta-objects.
  - Activator object -> Represents the binding policy. Sets up "channel".
  - Meta objects -> Represents meta-policies
- Object proxies - more than one binding to one object …
- Server interface proxies ...

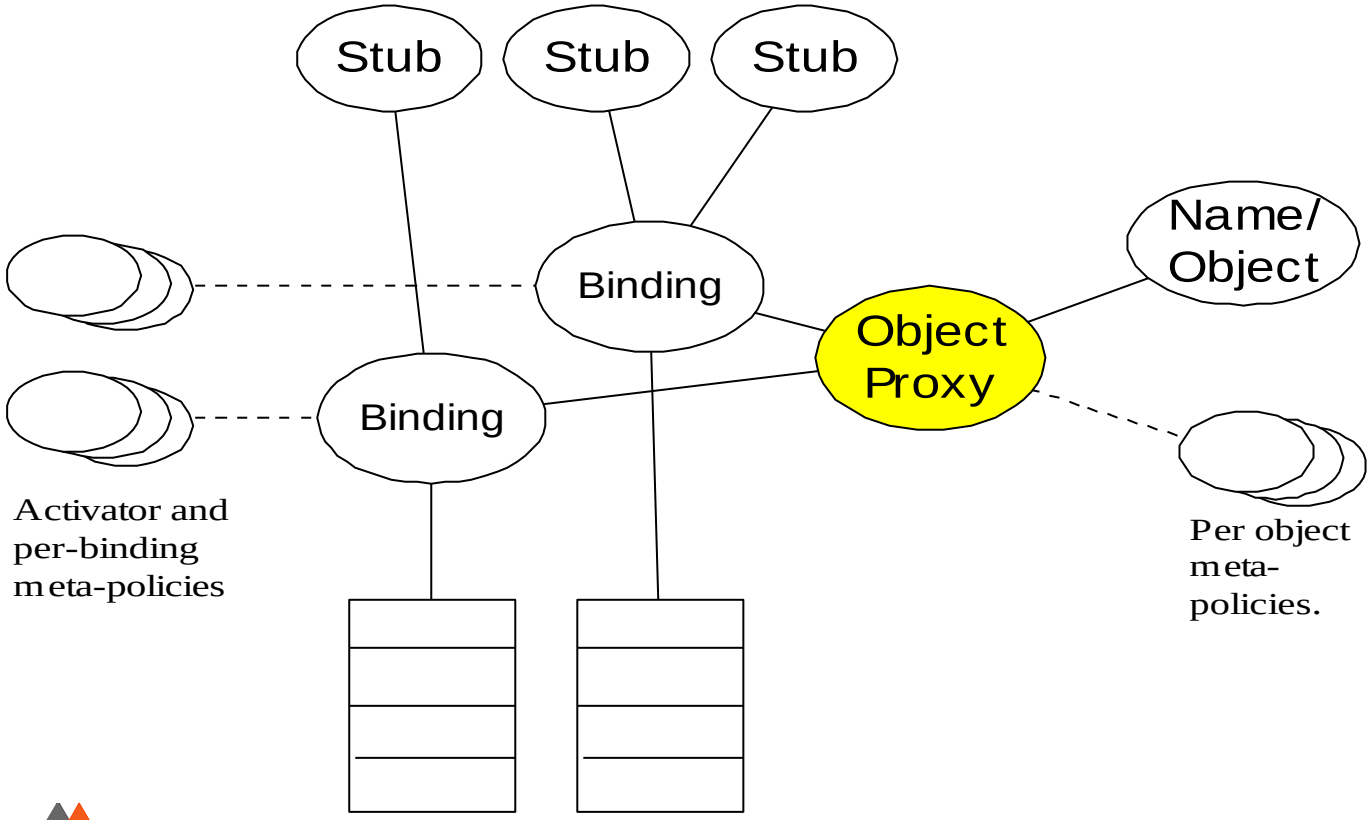
- Engineering Model -

Interface Proxies



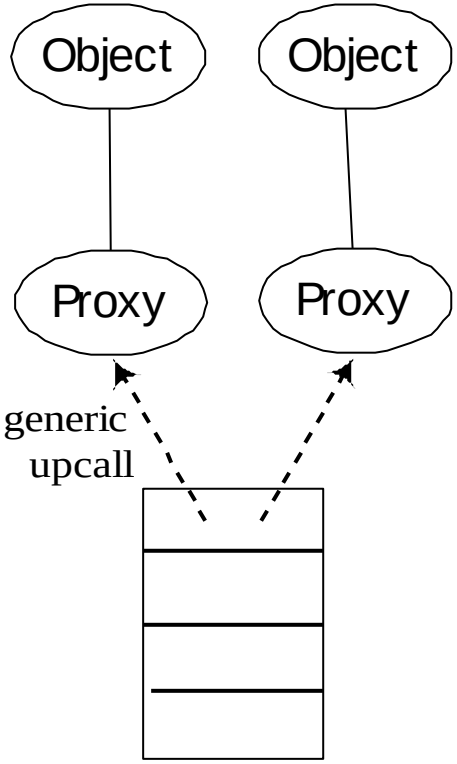
- Engineering Model -

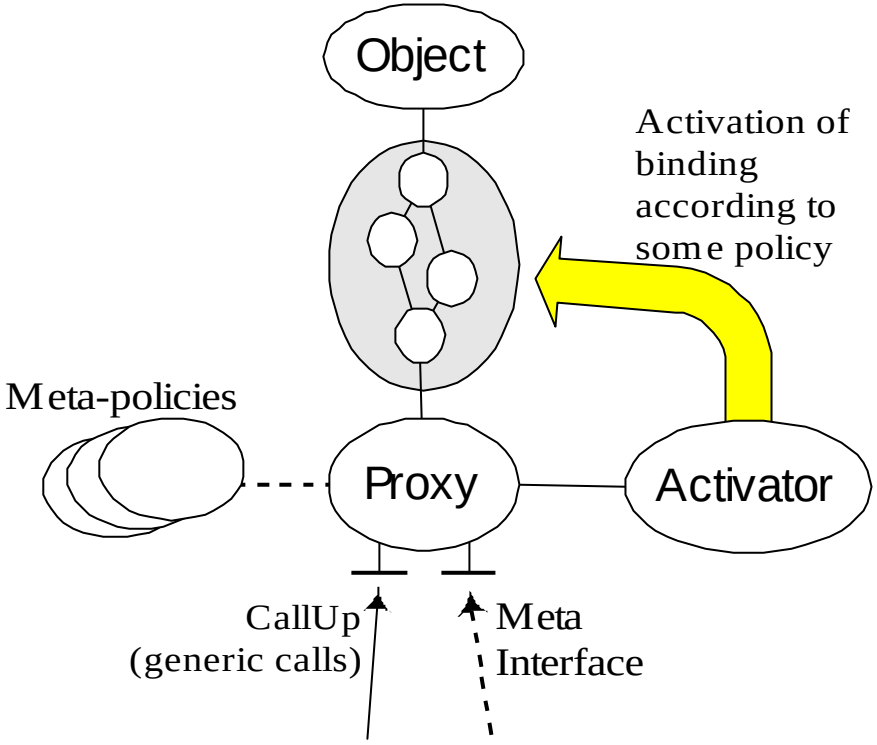
Object Proxies



- Engineering Model -

Server Interface Proxies



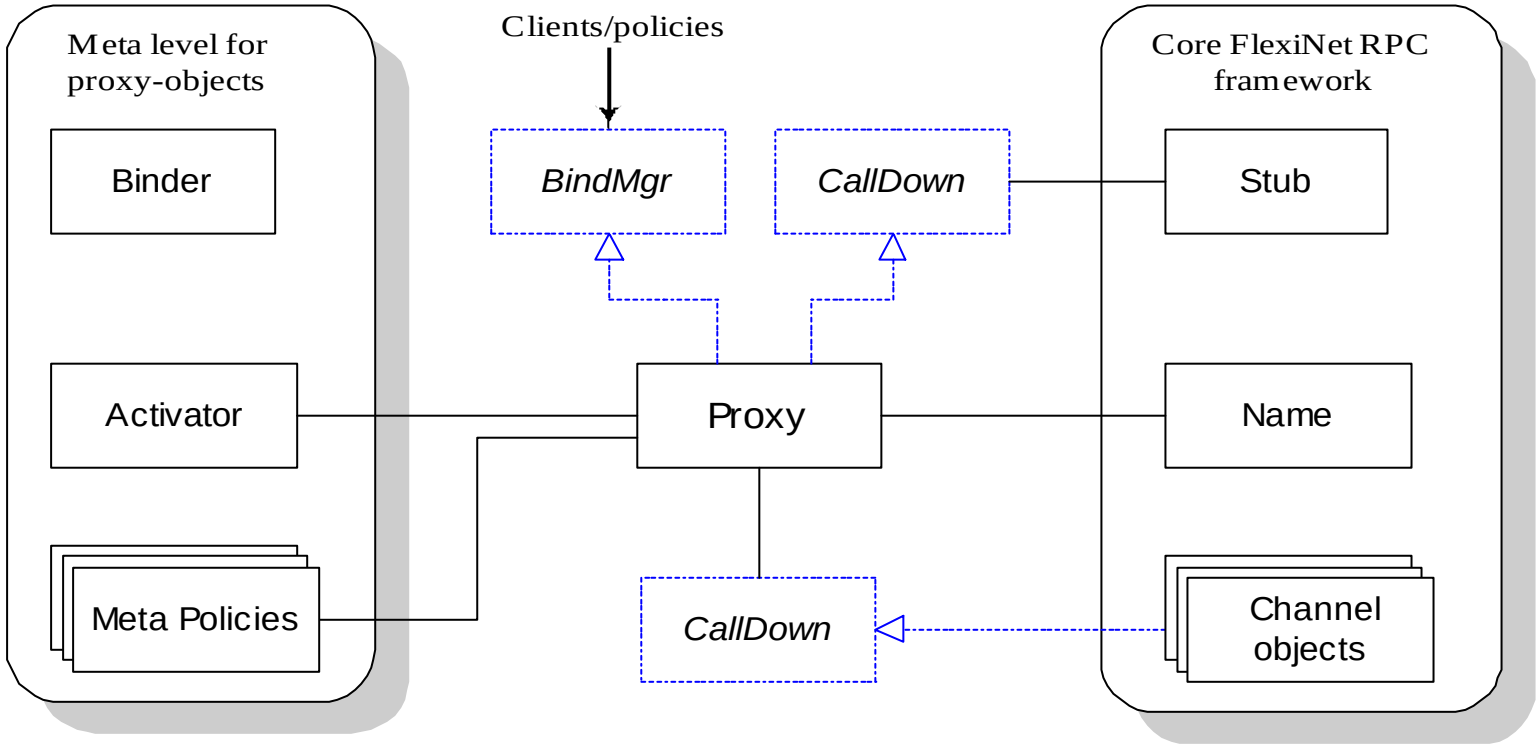


An Experimental Framework

- We have partly implemented the ideas ...
  - The Proxy class is the core
  - Binder and activator framework
  - Environment framework
- Focus at client side
- Will also do server side experiments ...

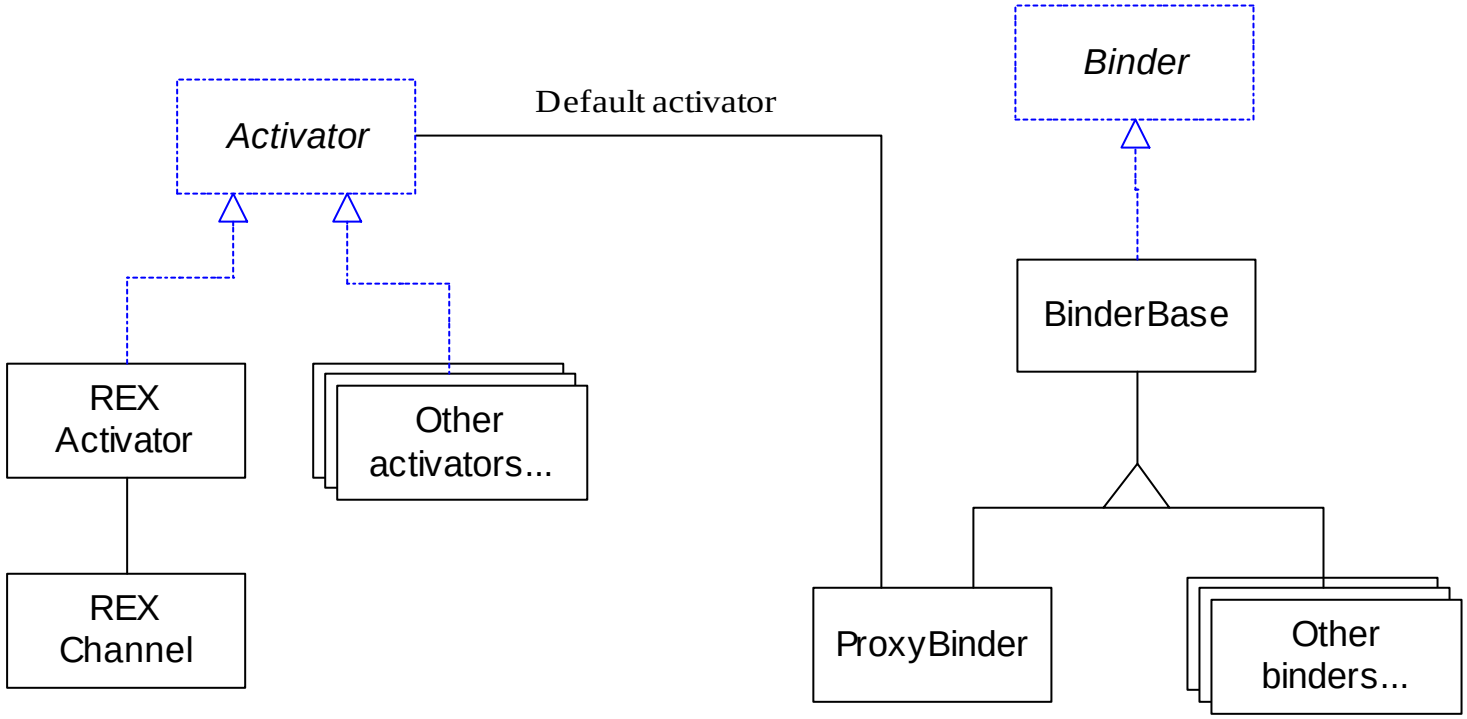
- Experimental Framework -

Core Design



- Experimental Framework -

Binders and Activators



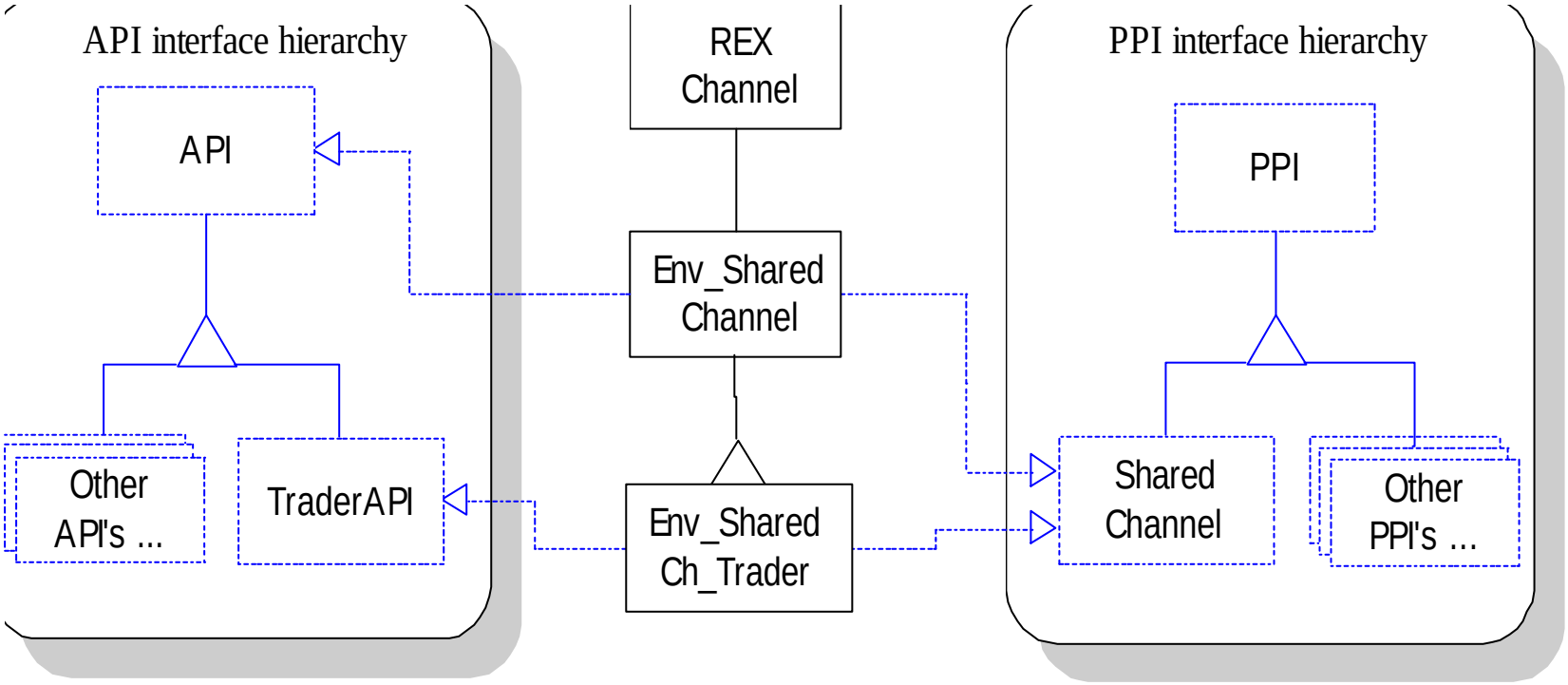
- Experimental Framework -

Environment Objects

- Different environments provide:
  - (Sligthly) different API's
  - Different sets of services/resources for use by policies (activators)
- Encapsulate into environment objects each with two interfaces:
  - API
  - Policy Programmer Interface (PPI)
- Two (interface) type hierarchies

- Experimental Framework -

Environment Objects



Summary

- Concepts
  - Binding, activation
  - Policy, metapolicy, policy-trading
- Engineering/design
  - Proxies represent bindings and provide transparency
  - Environments (API/PPI interfaces)
  - Policies:

	Instantiation	Run-time
Policy	Activator	Channel-config
Meta-Policy	Binder	Meta-object

Further Research

- Policy selection based on declarative statements …
  - Policy trading
  - Automatic construction of policy implementations
- Languages for declarative specification of policies . . .
  - Automatic generation from these ...
- End-to-end policy binding including client/server negotiation