# Q-CAD: QoS and Context Aware Discovery Framework for Mobile Systems

Licia Capra, **Stefanos Zachariadis**, Cecilia Mascolo

## Outline

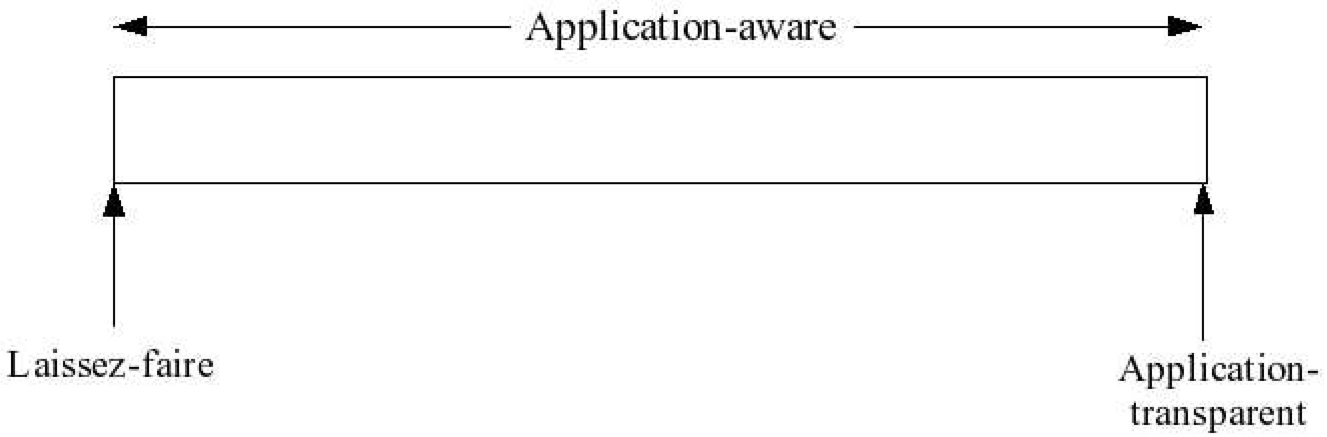
- Background / Motivation
- Model
- Discovery Protocol
- Architecture
- Implementation
- Conclusion

# Background

- Pervasive Computing Environment
  - Users reason in tasks
- Task:
  - "I want to print a picture"
    - Binding to a remote service
    - Getting the code to talk to the remote service
- Many choices for each task to be made
  - Context
  - QoS

## Dealing With Choice

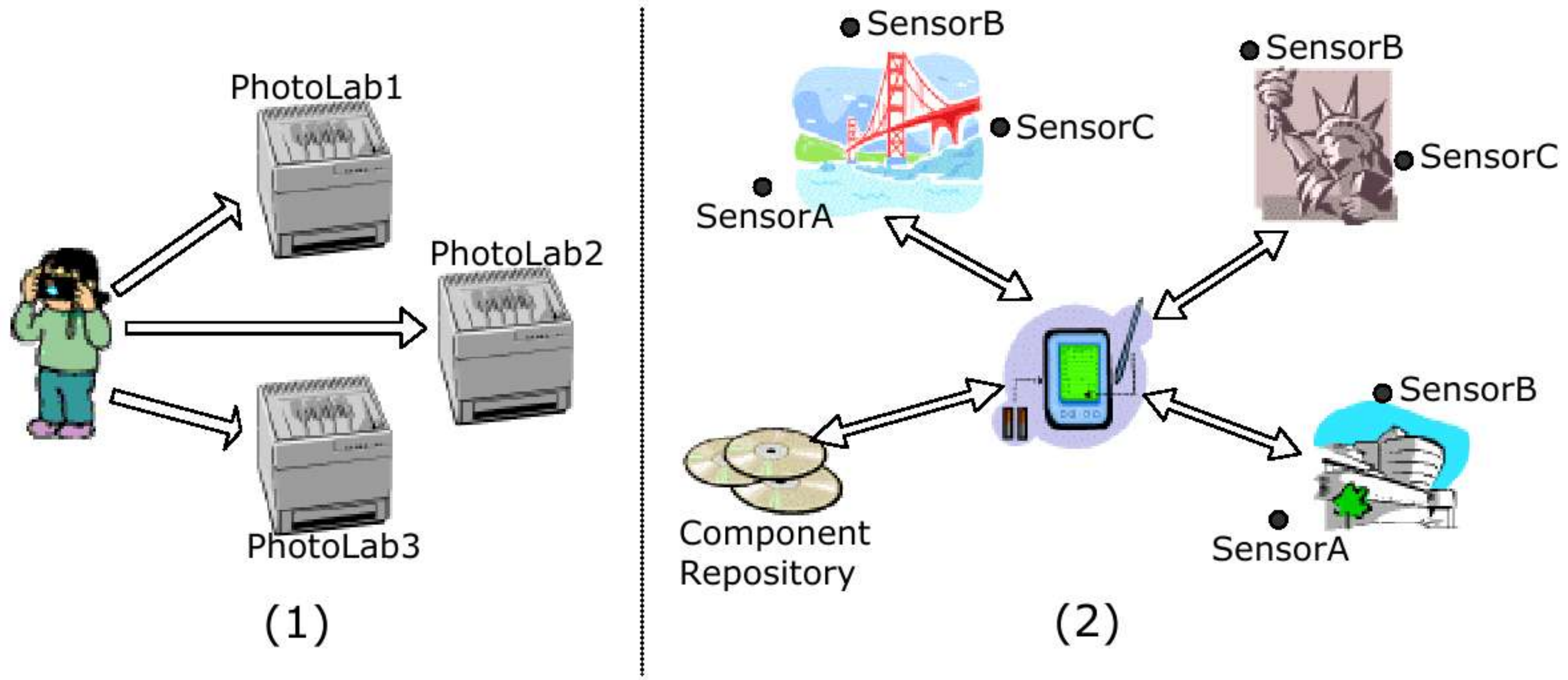
- Black Box Approach
  - System automatically decides
- Open Approach
  - User/Programmer decides



## Q-CAD

- Application – Aware framework
- QoS-based Context – Aware Resource Discovery
- Context
  - Application Profile
- QoS
  - Utility Function

## Case Study / Example



Proactive vs Reactive

# Q-CAD Model: Assumptions

- Component-based system
- Remote Resource
  - Service, component, sensor
  - Identified by URI
  - Resource Descriptor
    - Named key, value pairs
- Binding
  - Association of remote resource with local component
  - Deployment of remote component, locally

# Resource Descriptor

```
(ID, QCAD:displayVideo)
(type, component)
(code, display800600.jar)
(resolution, 800x600)
(version, 2.1)
(platform, JVM2)
(size, 70KB)
(cost, $10)
(memory, 2)
(battery, 4)
```

# Q-CAD Model: Assumptions (2)

- Context
  - Remote, Local
- Proactive and Reactive Discovery
  - Model and protocol same
- Independent of underlying SDP
  - Not quite true :-)
- Modeling of Requirements in Application Profiles and Utility Functions (CARISMA)

# Q-CAD Model: Application Profiles

- Defines what to do
- Context-Aware Discovery
- Each Session Has:
  - Trigger
    - Local/Remote
  - Where to Bind (Remote Resource)
  - Where to Bind To (Local Component)
- Different Checks at Different Stages

## Application Profile: Proactive

```
 <LOCAL_CONTEXT/>

 <REMOTE_CONTEXT/>

 <BIND>

     <BIND_RESOURCE name="printPicture">

         <REMOTE_CONTEXT id="1">

             <CONDITION name="diskSpace" op="greaterThan"
value="100MB"/>

         </REMOTE_CONTEXT>

     </BIND_RESOURCE>

 </BIND>
```

## Application Profile: Proactive (2)

```
<ADAPT>
    <ADAPT_COMPONENT id="1">
        <LOCAL_CONTEXT id="2">
            <CONDITION name="battery" op="greaterThan" value="30%"/>
        </LOCAL_CONTEXT>
        <REMOTE_CONTEXT/>
        <ATTRIBUTES>
            <ATTRIBUTE key="protocol" op="equals"
value="encryptedUpload"/>
        </ATTRIBUTES>
    </ADAPT_COMPONENT>
</ADAPT>
```

# Application Profile: Reactive

```
<LOCAL_CONTEXT id="1">
    <CONDITION name="battery" op="greaterThan" value="30%"/>
</LOCAL_CONTEXT>
 <REMOTE_CONTEXT id="2">
        <ATTRIBUTES>
            <ATTRIBUTE key="sensor" op="equals" value="videoSensor"/>
            <ATTRIBUTE key="resolution" op="equal" value="800x600"/>
            <ATTRIBUTE key="format" op="equals" value="jpeg"/>
        </ATTRIBUTES>
</REMOTE_CONTEXT>
<BIND>
    <BIND_RESOURCE name="videoSensor"/>
</BIND>
```

## Application Profile: Reactive (2)

```
<ADAPT>

    <ADAPT_COMPONENT id="3">

        <LOCAL_CONTEXT/>

        <REMOTE_CONTEXT/>


        <ATTRIBUTES>

            <ATTRIBUTE key="type" op="equals" value="displayVideo"/>

            <ATTRIBUTE key="cache" op="greaterThan" value="1024KB"/>

            <ATTRIBUTE key="resolution" op="greaterThan"
value="800x600"/>

        </ATTRIBUTES>

    </ADAPT_COMPONENT>

</ADAPT>
```

# Utility Functions

- Suppose many resources match the conditions
- Need to Select
- Criterion: QoS requirements
  - Encapsulation as Utility Functions
  - Executed against Resource Descriptors
    - Locally or Remotely
- Automation vs Application Input

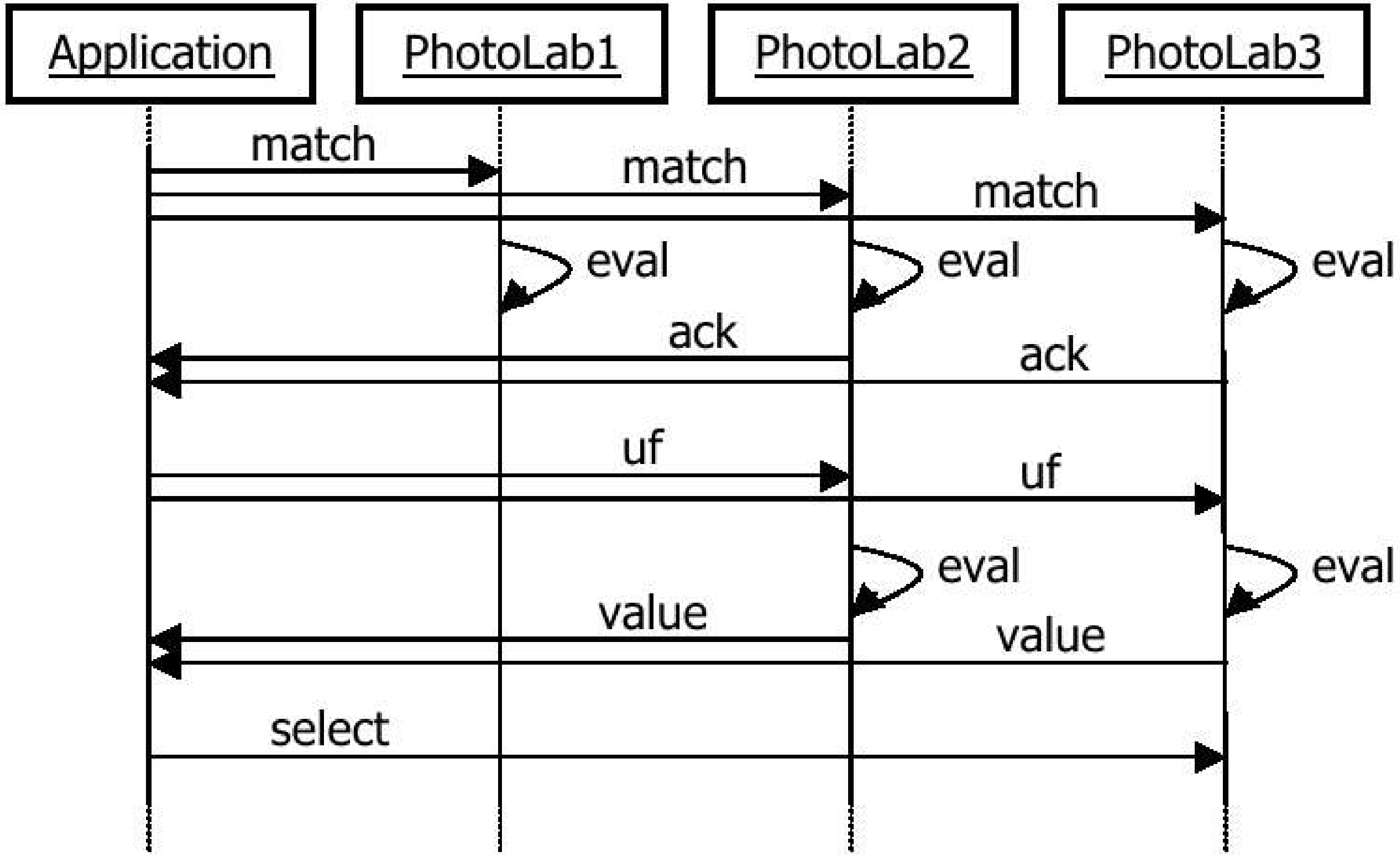
# Utility Function

```
<RETURN>
    <EVALUATE>
        <ATTRIBUTE key="cost" op="greaterThan" value="10$"/>
    </EVALUATE>
    <FILTER>
        <ATTRIBUTE key="cost"/>
    </FILTER>
</RETURN>
<MAXIMISE>
    <ATTRIBUTE key="battery" weight="10"/>
    <ATTRIBUTE key="memory" weight="5"/>
</MAXIMISE>
```

# Discovery Protocol

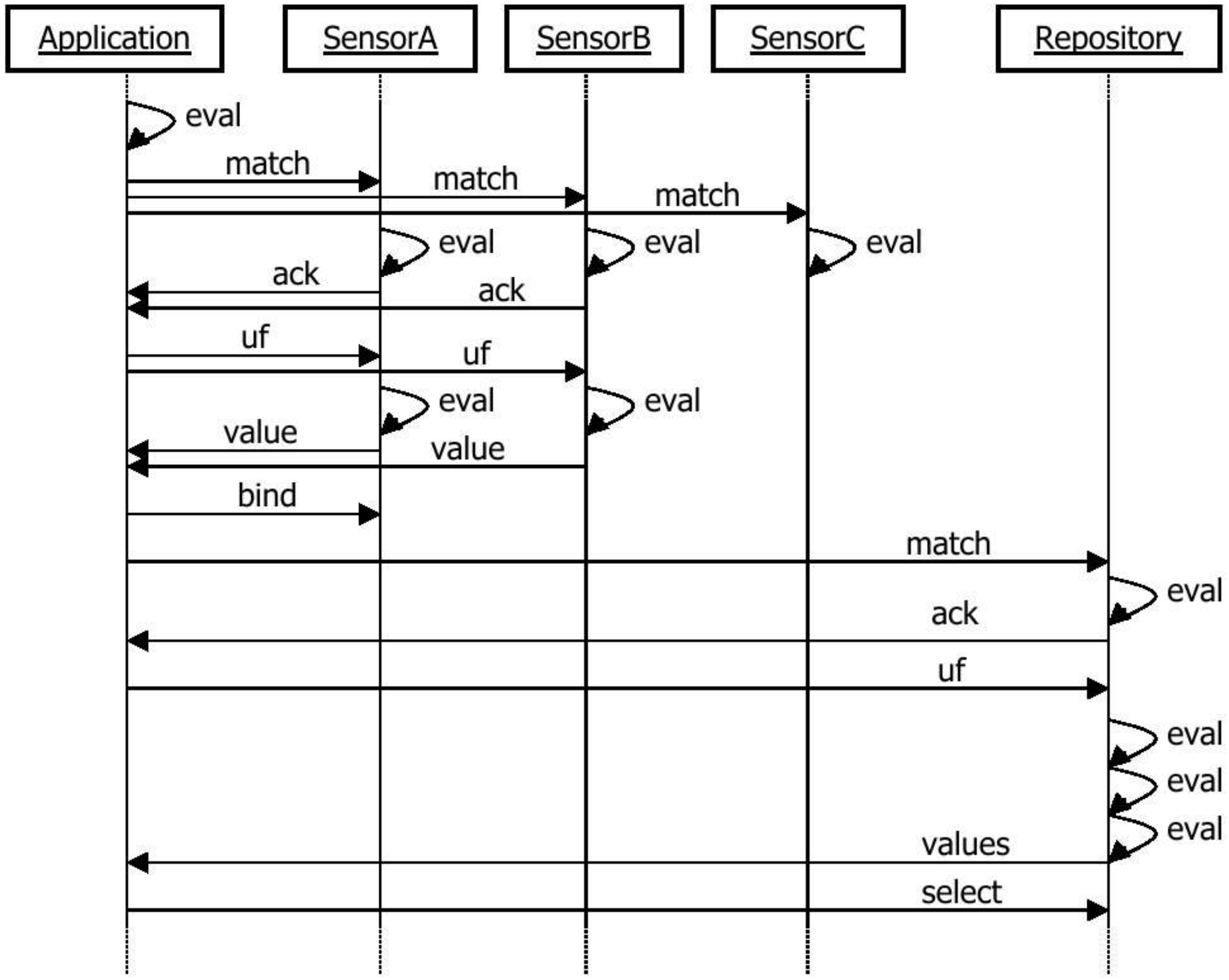
- 3 Step Protocol
  - Matching
  - Evaluation
  - Selection

# Discovery Protocol Sample



Proactive Discovery

# Discovery Protocol Sample (2)



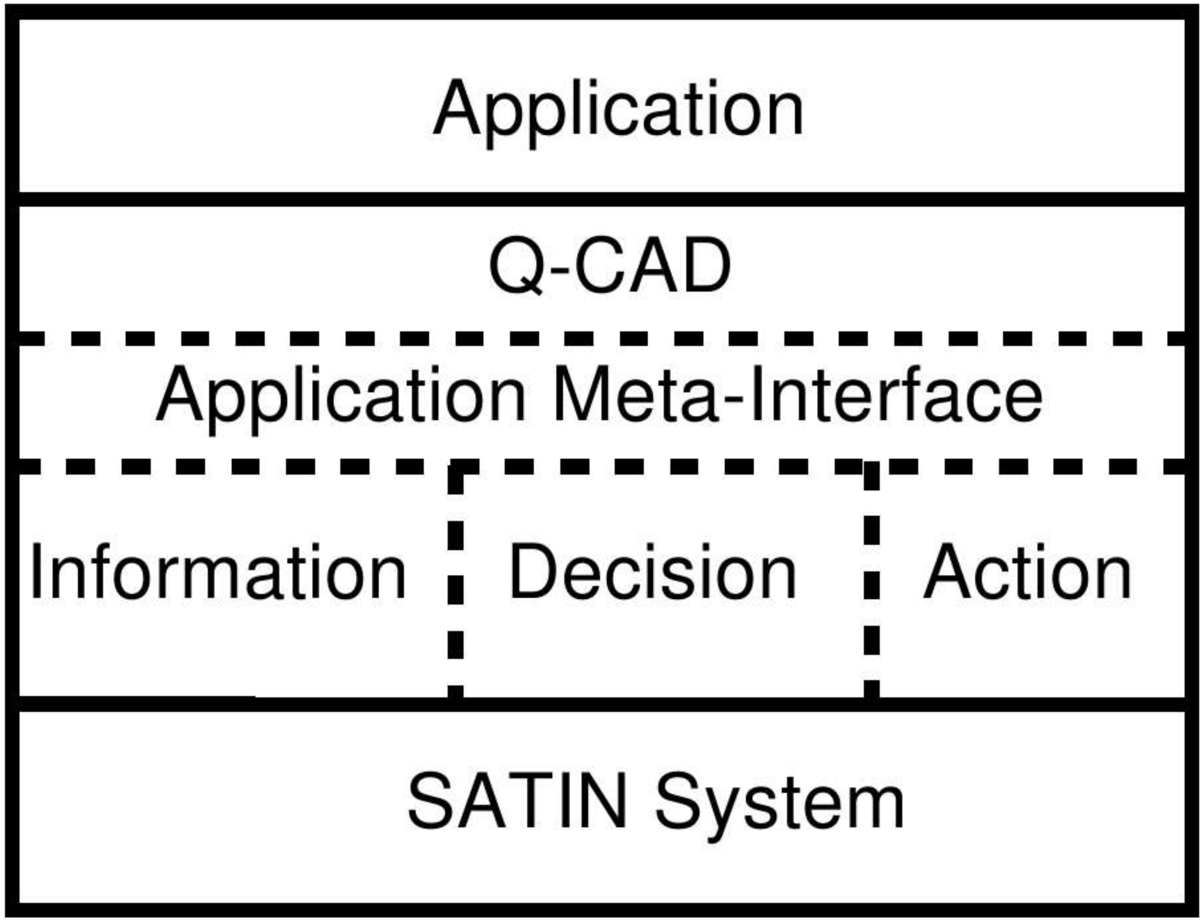
Reactive Discovery

# SATIN

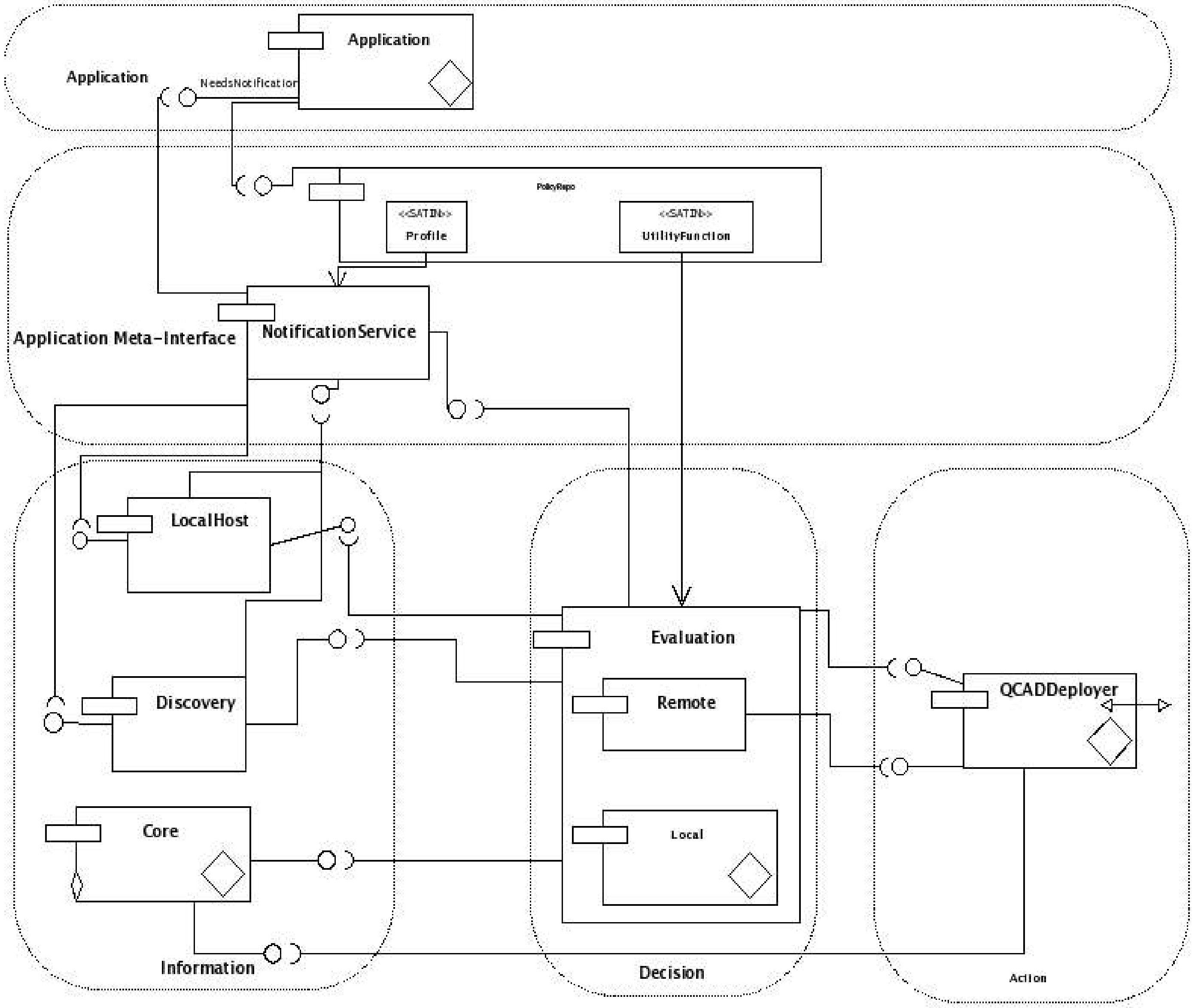
- Local component metamodel
  - instantiated as middleware system
- Logical Mobility as 1st class citizen
- Uses key,value attributes for reasoning
  - locally and remotely
  - Uses dynamic code to match attributes
- Pluggable Advertising and Discovery Framework
- Provides means but not decision logic (laissez - faire)

# Q-CAD Architecture Outline

- Engineered using SATIN



# Q-CAD Architecture



## Implementation

- BSc Thesis
- Using Multicast and Publish Subscribe
- Preliminary results available
- More work (Afra) during the summer

## Future Work

- Ontology Translation
- Trust
- Message Routing

## Related Work

- Directory based
  - UPnP, Jini
- Decentralised
  - SSDP, DEAPspace, Lanes, JXTA
  - Q-CAD can be built on top
- Semantic Routing
  - Q-CAD richer

## Conclusion

- Q-CAD
  - QoS and context aware framework for resource discovery
    - Component, Sensor, Service
  - Application Profiles
  - Utility Functions
- Q-CAD is current