Reflecting on Advanced Distributed Programs

Elisa Gonzalez Boix

Software Languages Lab

Vrije Universiteit Brussel

Software Languages.Lab

- Programming Languages & Software Engineering
- 4,5 Professors; ± 30 researchers (± 6 Post-doc)
- 60^{+} PhD theses finished, ±20 ongoing
- 150^{+} Master theses

W. De Meuter

T. D'Hondt

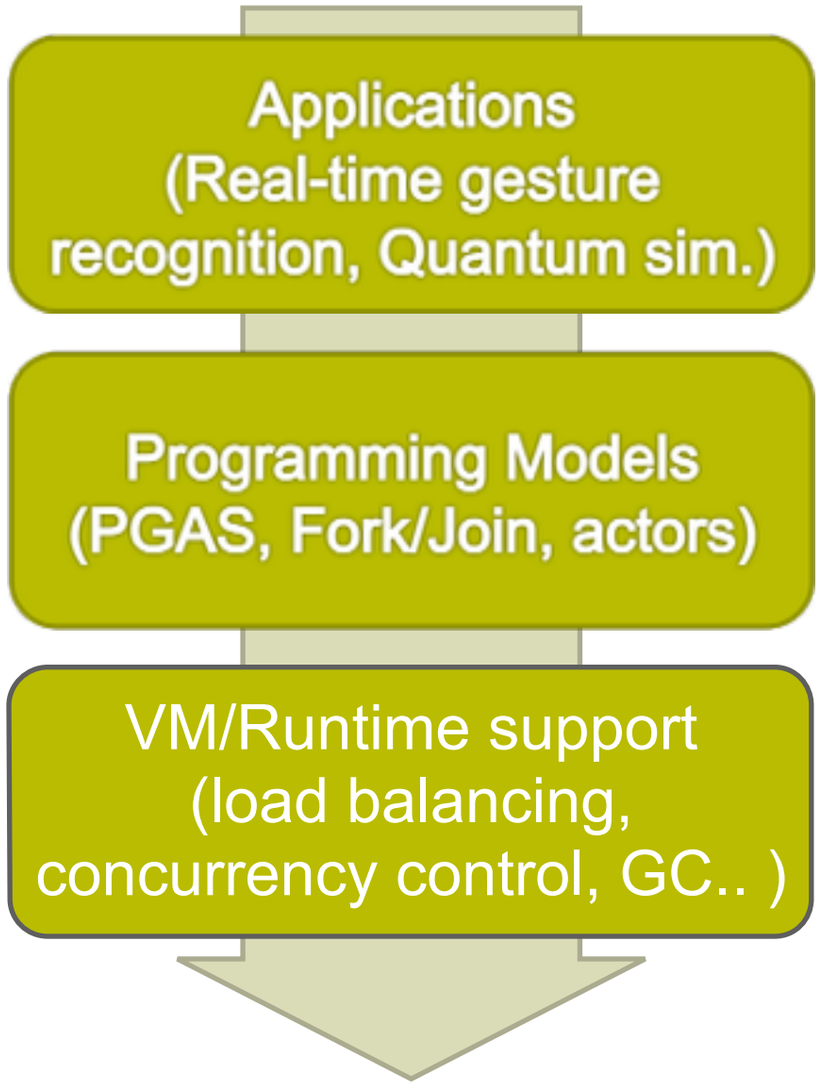
E. Gonzalez Boix

V. Jonckers

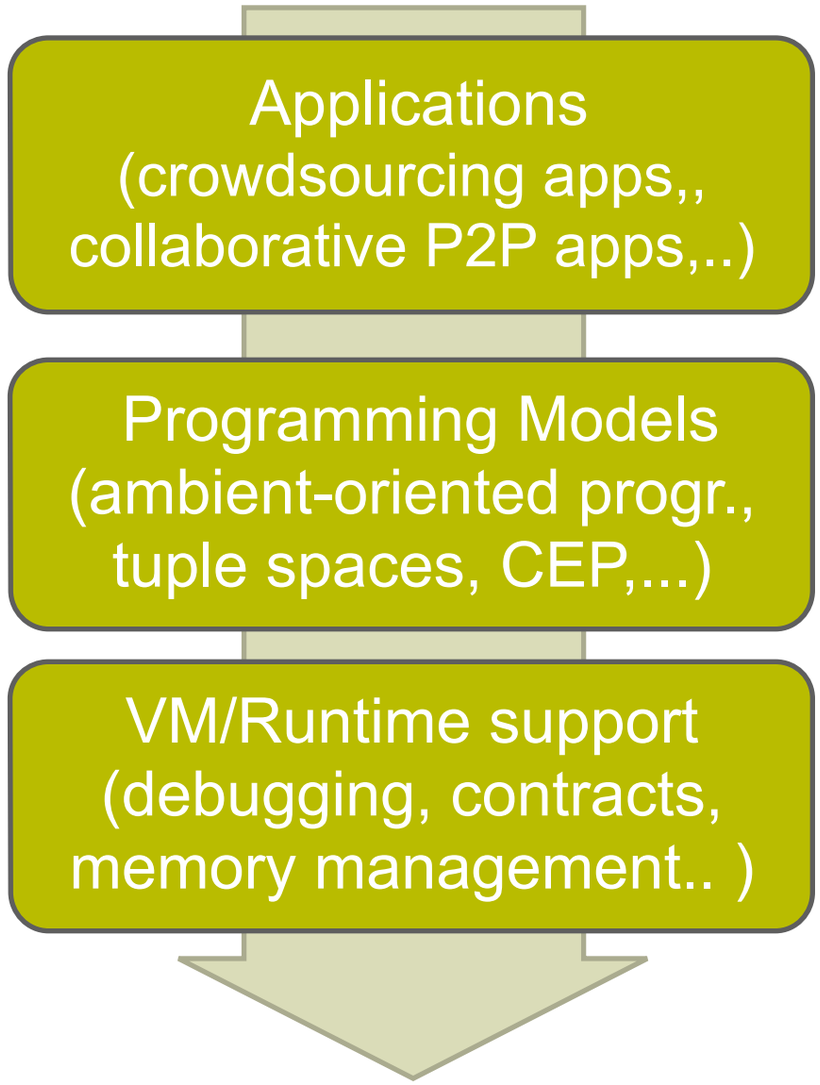
C. De Roover

Language Research

Parallel, Multicore & Exascale Computing



Mobile and Cloud Computing



Ambient & Cloud Computing @ SOFT

Prof. Dr. Wolfgang De Meuter

Prof. Dr. Elisa Gonzalez Boix

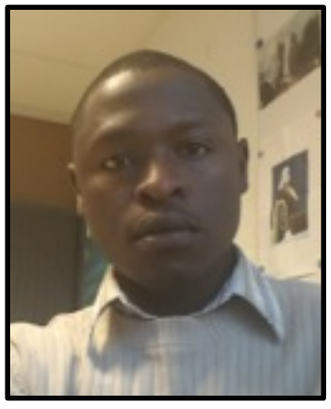
Dr. Christophe Scholliers

Dr. Ellie D'hondt

Dries Harnie

Kevin Pinte

Lode Hoste



Kennedy Kambona

Laure Phillips

Thierry Renaux

Simon Van de Water

Nathalie Oostvogels

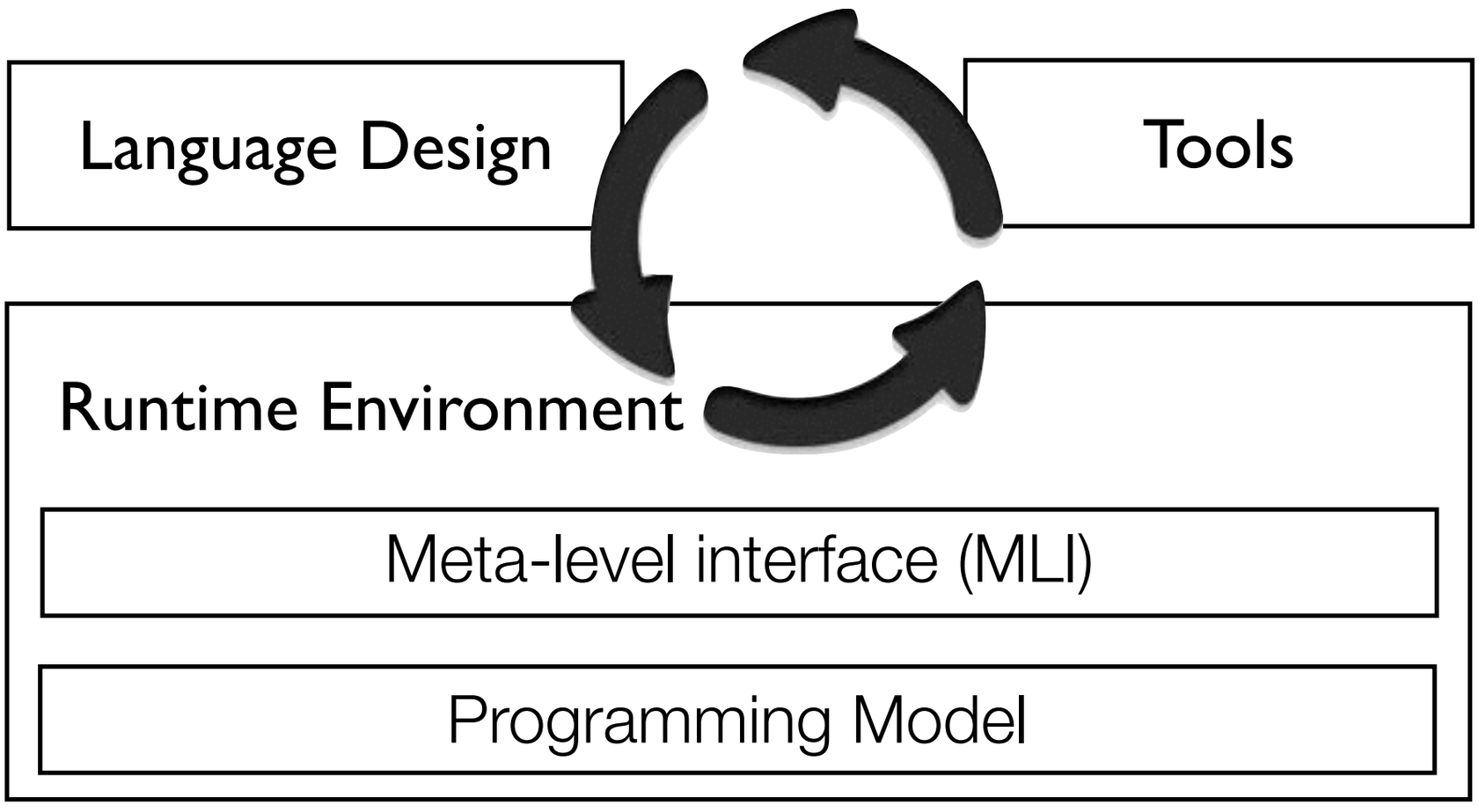
Jesse Zaman

Paul Blouet

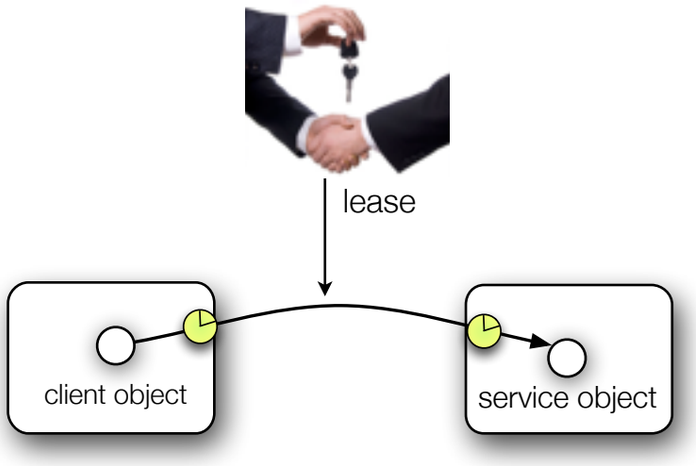
...

Vision

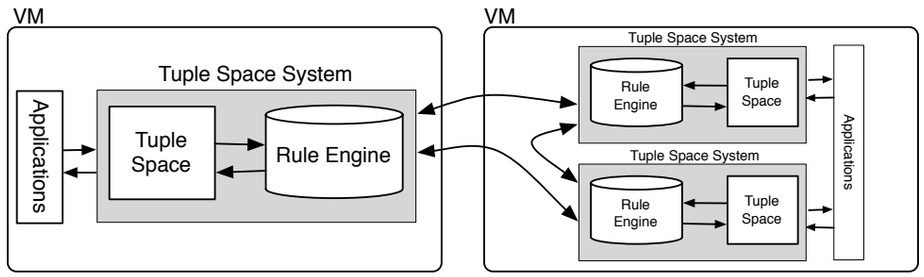
Novel programming languages need to be paired with the right meta-level engineering



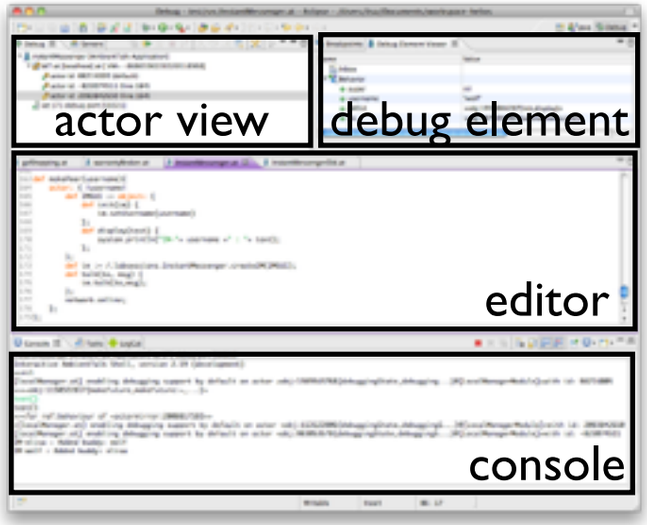
Mobile Computing



Leased Object References



Tuples on the Ambient (TOTAM)



REME-D

Failure Handling Abstractions

Distributed Debugger

AmbientTalk/M

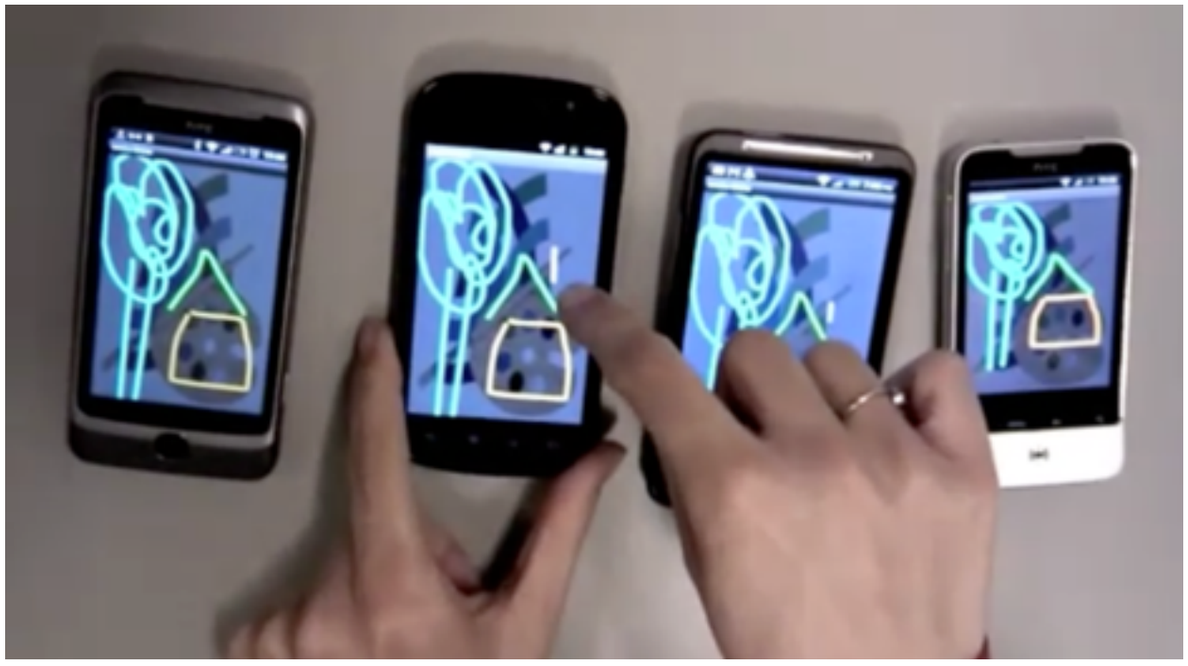
Transmitter-Receptor Meta Model

Ambient-oriented Programming

AMBIENTTALK

Mobile Ad hoc Networks

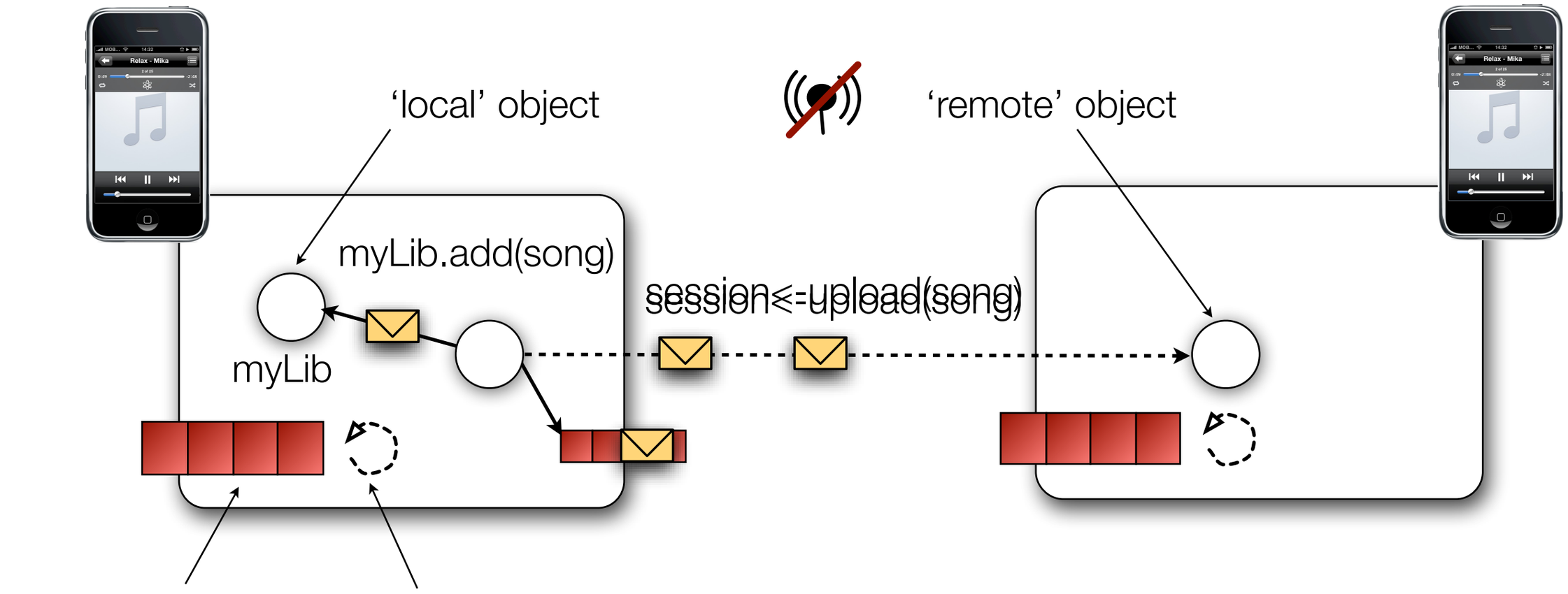
“Spontaneous Discovery & Collaboration”



Zero Infrastructure

Volatile Connections

Ambient-oriented Programming in a Nutshell



Ambient-Oriented Language Support

- How to reconcile asynchronous computation with return values?

  Non-blocking Futures

- How to discover and address a group of objects in the network?

  Ambient References
  [Van Cutsem,08]

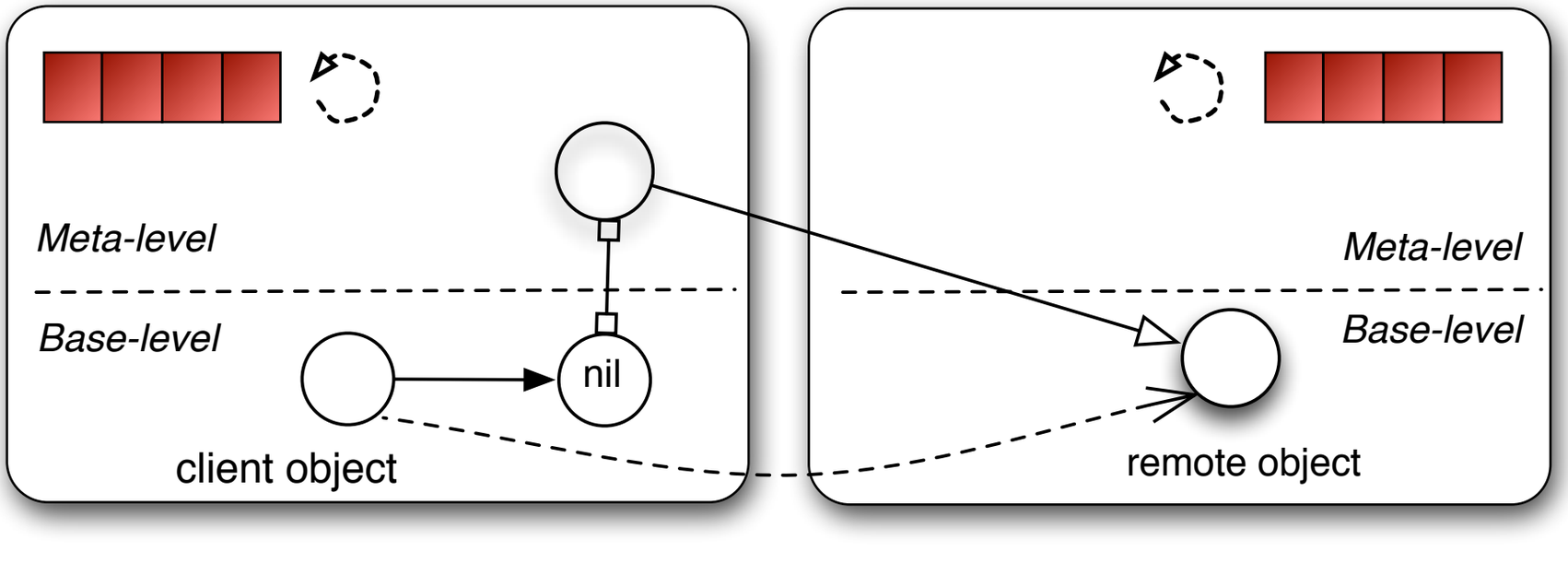
- How to provide high-level representation of failures?

  Leased References
  [Gonzalez Boix,12]

- How to map physical objects stored in RFIDs into software objects?

  Things
  [Lombide Carreton,11]

Custom Distributed Object References as Proxies

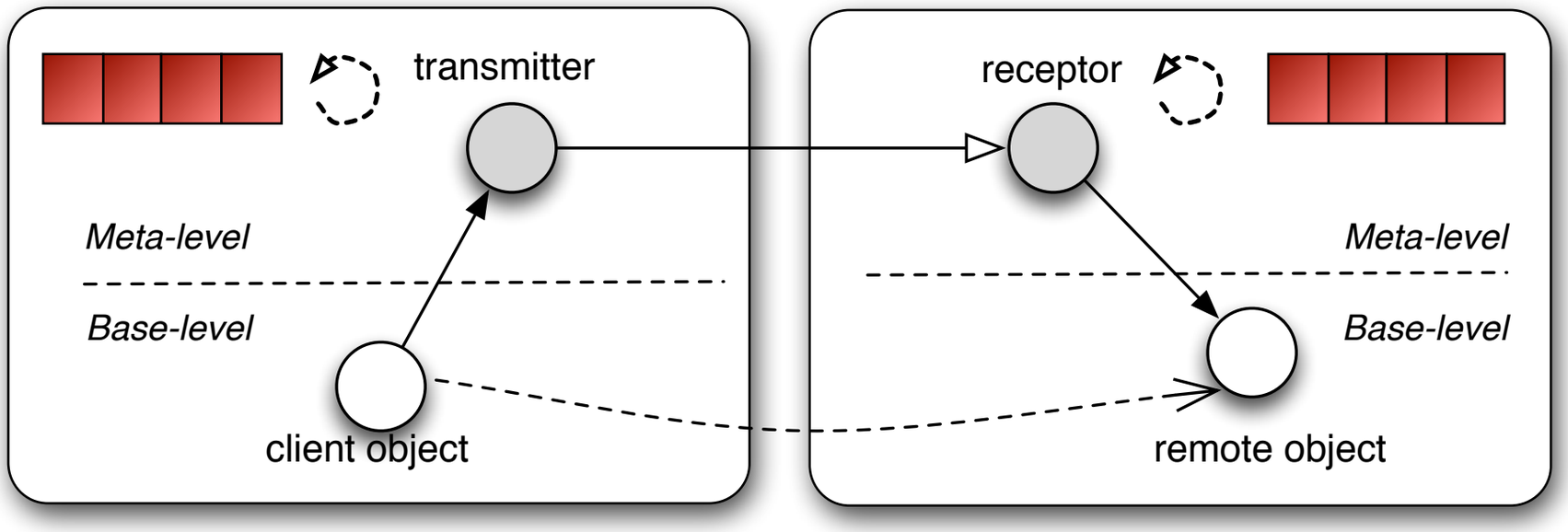


“Two-body problem”

Manual proxy manipulation for reifying both sides of communication

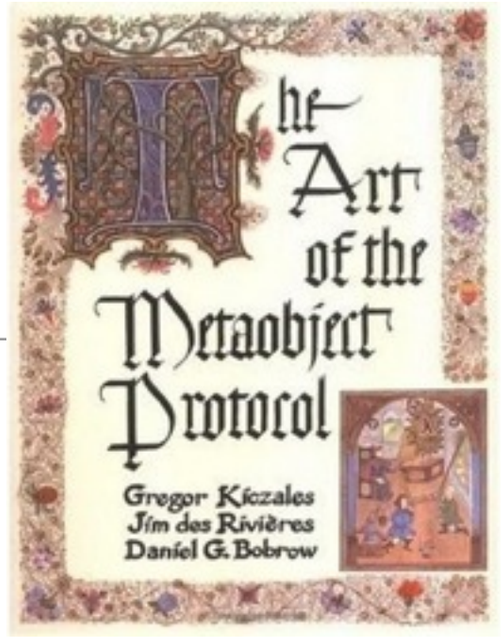
Transmitter-Receptor Meta Model

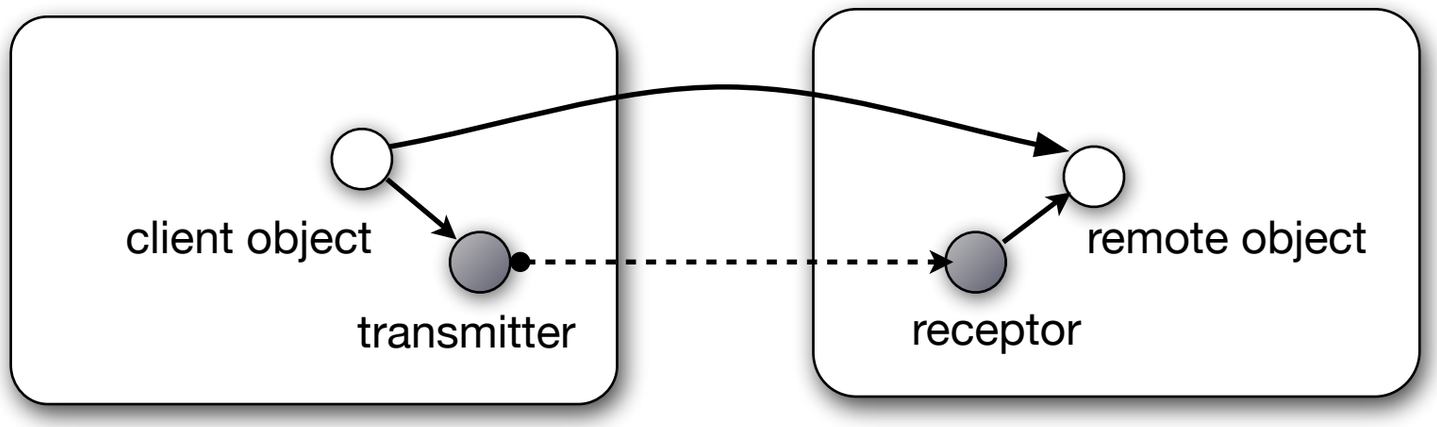
"Remote object references are exposed as **first-class** values with a **custom metaobject protocol** reified by two metaobjects, one at each end of the reference"



- Reify message sending and parameter passing
- Causally connected to base-level object for the dynamic extend of a distributed interaction

Transmitter-Receptor Meta Model





Asynchronous message process protocol

```
def schedule(rcv, msg);
def serve();
def transmit(msg);
def leave(msg);
def listOutgoingLetters();
```

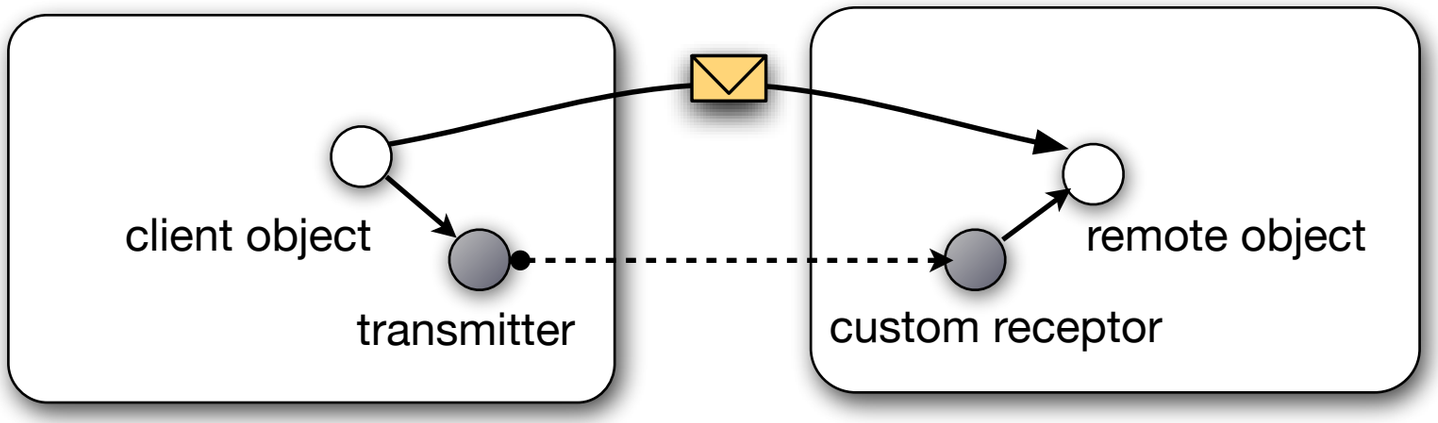
```
def accept(msg);
def sendMessage(rcv, msg);
def performInvocation(rcv,inv);
```

Reference Marshalling protocol

```
def marshallingStrategy();
def unmarshallingStrategy();
```

Implementing Safe References as a Custom Receptor

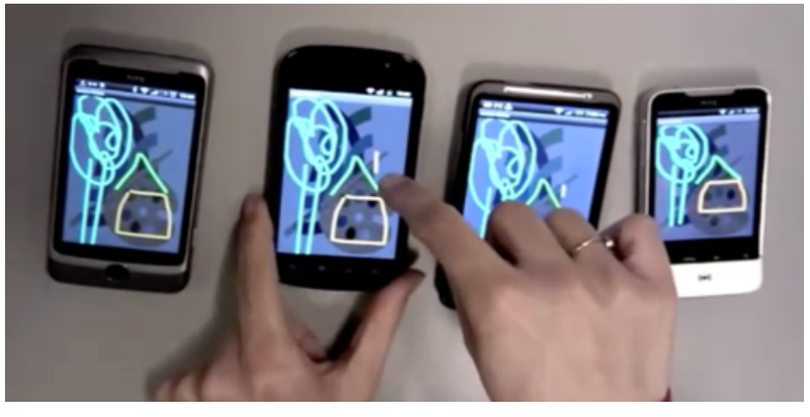
Akin to immutable references [Clarke et al.,08]




```
def makeSafeReference(){
  extend: defaultReceptor with: {
    def performInvocation(rcv, inv){
      if: !(invocation.selector.isAssignmentSymbol()) then:{
        super.performInvocation(rcv,inv);
      }
    }
  }
}
```

captures all method invocations resulting of the processing of an async message

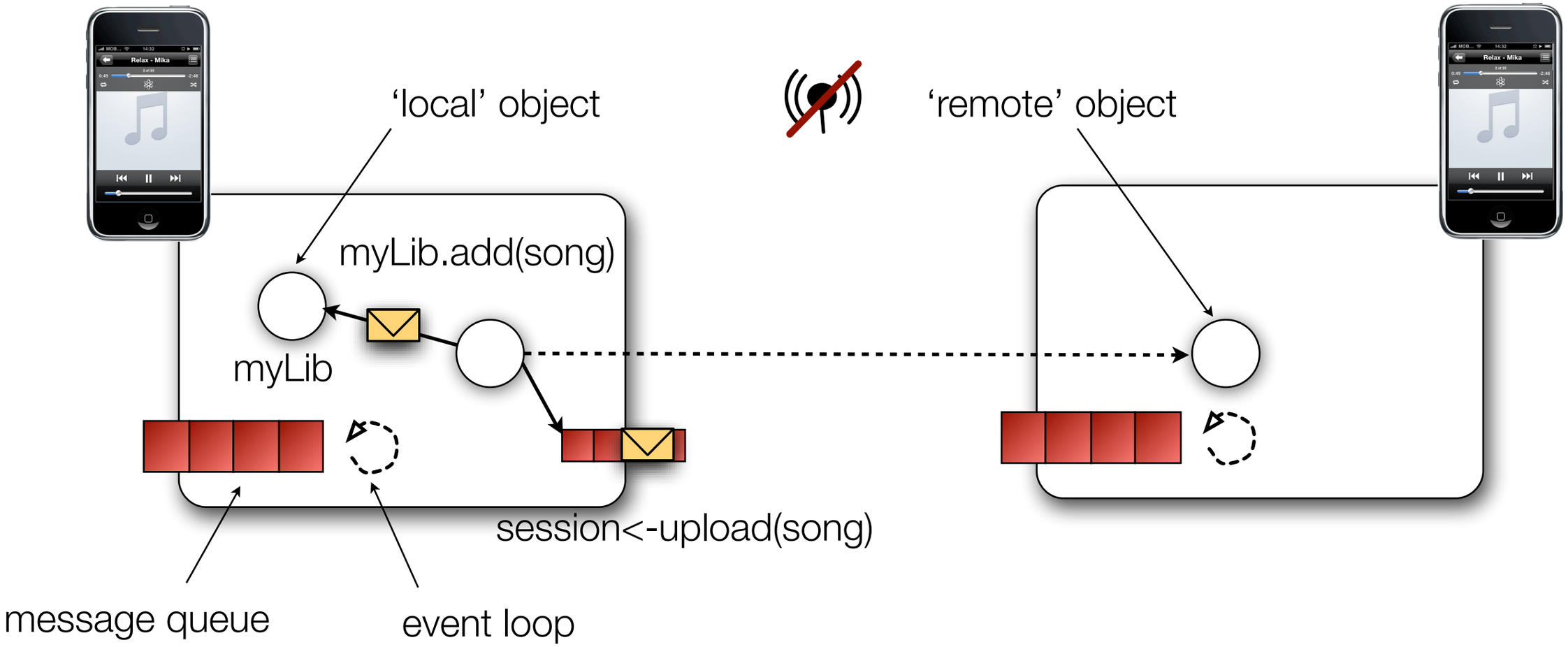
Implementing Ambient References as a Custom Transmitter



```
def nearbyDrawers := ambient: Drawer;
nearbyDrawers<-updateShape(shape);
```

```
 def ambient: typeTag {
   def transmitter := extend: defaultTransmitter with: {
      def receivers := Vector.new();
      def schedule(rcv, msg){
         receivers.each: { |receiver|
            receiver <+ msg;
         };
      };
      whenever: typeTag discovered: { |potentialReceiver|
         receivers.add(potentialReceiver);
      };
   } taggedAs: [typesModule.Isolate];
};
```

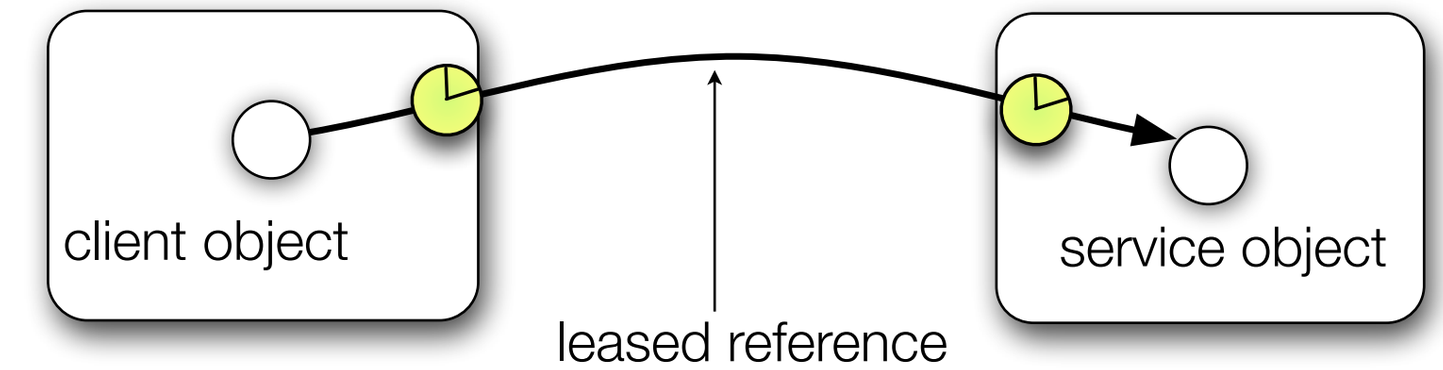
Supporting Failure Handling Abstractions

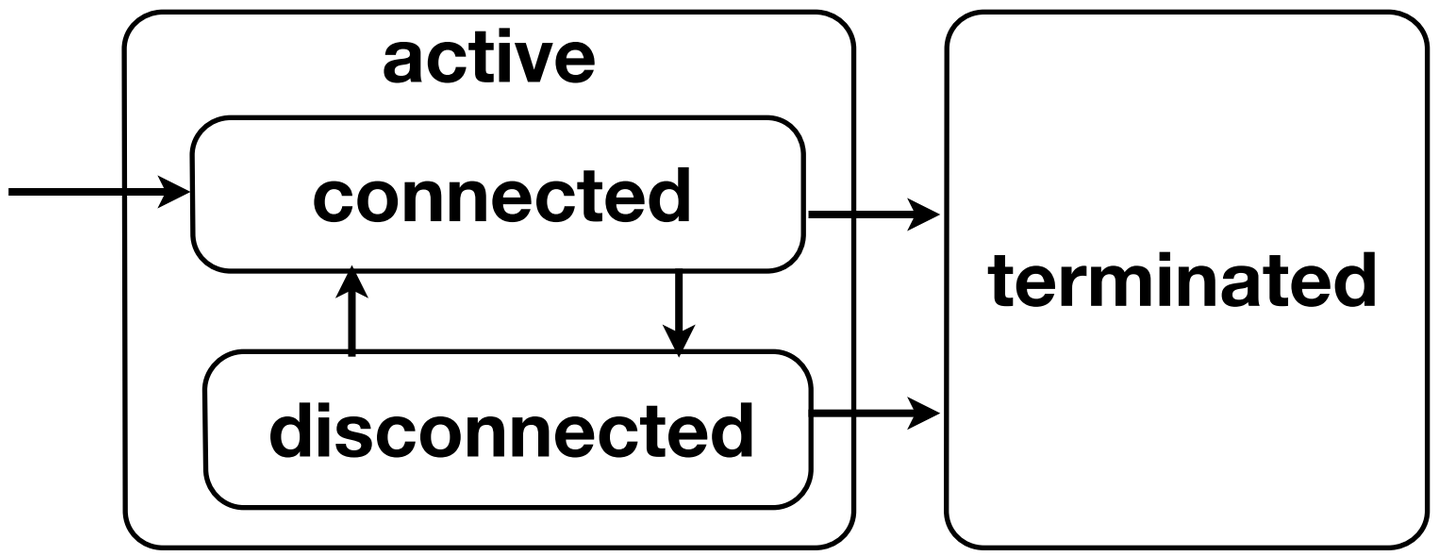


- ✔ no race conditions on objects
- ✔ tolerates transient failures
- ✘ memory management
- ✘ high-level representation of failures

Leased References

A **lease** denotes the right to access a **resource** for a specific duration **negotiated** when the access is first requested.



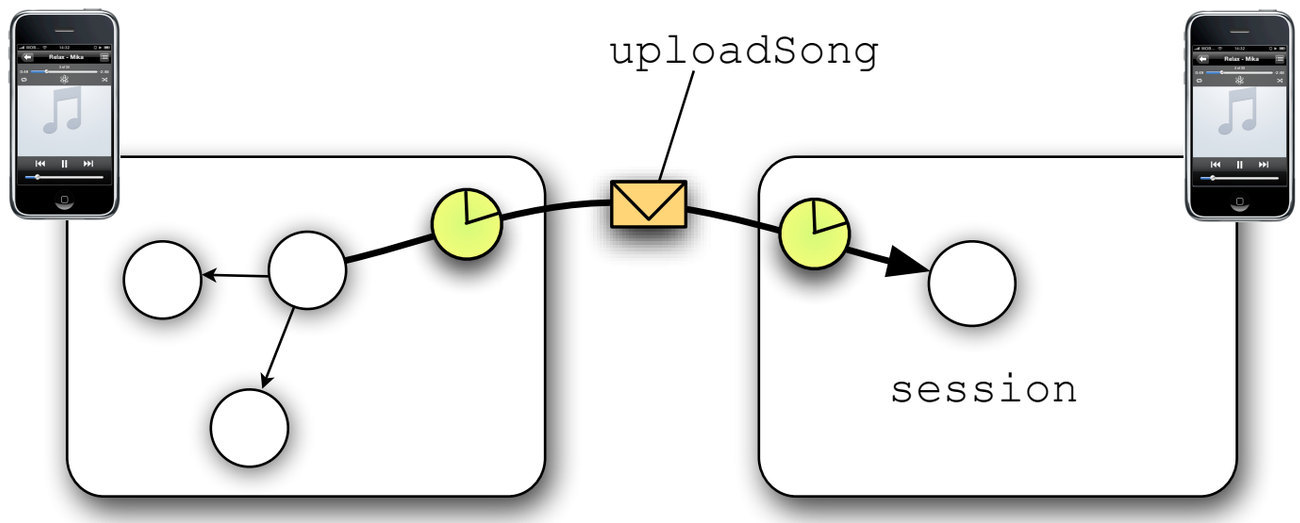


Leased References in AmbientTalk

Transparent for clients

Client object

```
leasedSession<-uploadSong(“Mika”,“Relax”,..)
```



Leased reference kinds

Event listener at each of the reference

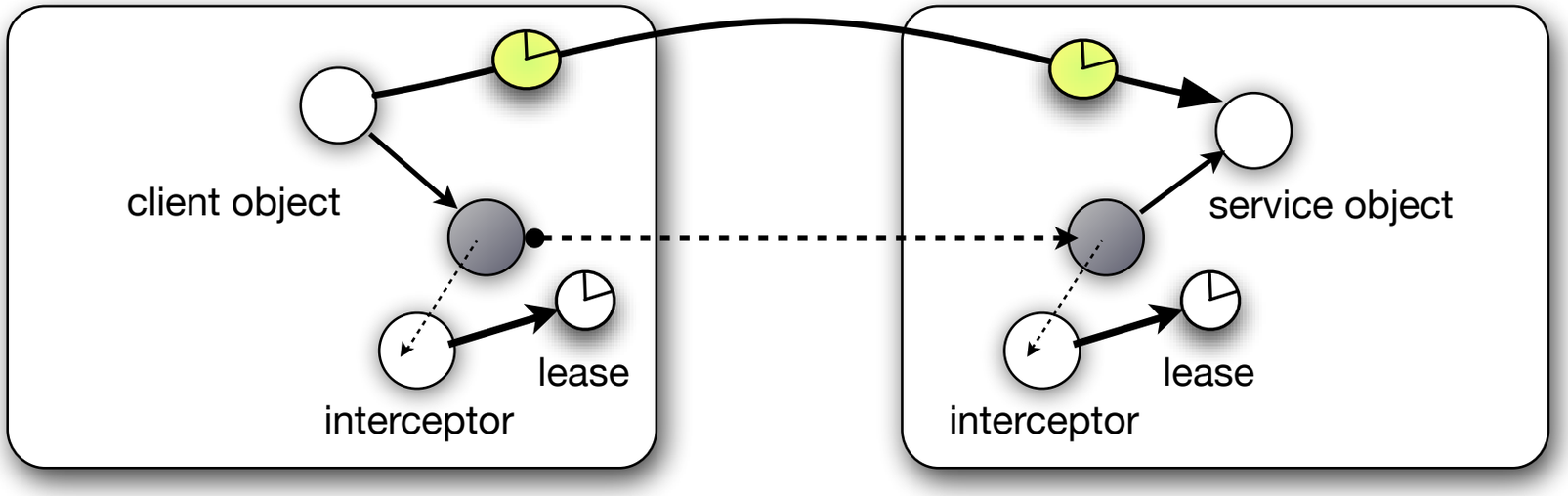
Service object

```
def session := object: {
  def uploadSong(song) {...};
  def endExchange() {...};
};

def leasedSession := renewOnCallLease: 10.minutes for: session;

when: leasedSession expired: {
  println(“session with remote music player expired”);
  // cleanup the partially received library
}
```

Domain-specific API for Leased Object References



Lease Object API

```
def renew(otherTerm);
def revoke();
def extend(otherTerm);
def sublease(otherTerm);
def activate(lr);
def getLeaseTermLeft();
def addListener(closure,stateType);
```

Interceptor API

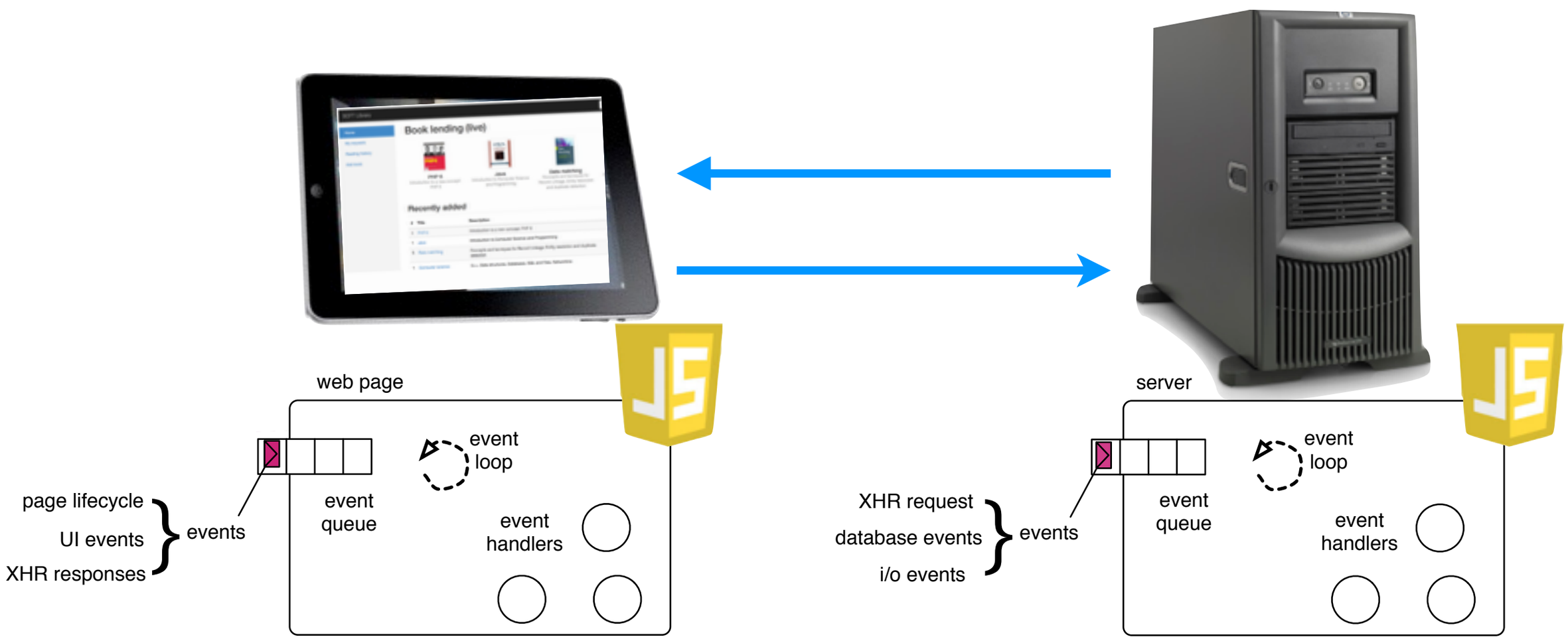
```
def onMessageReceived(msg,lease);
def onReferenceCreated(lease);
def onReferenceShared(lease);
```

From Mobile to Cloud Computing

Since Academic Year 2013 / 2014

with Jasper Tack, Felipe Caicedo & Dr. Carlos Noguera

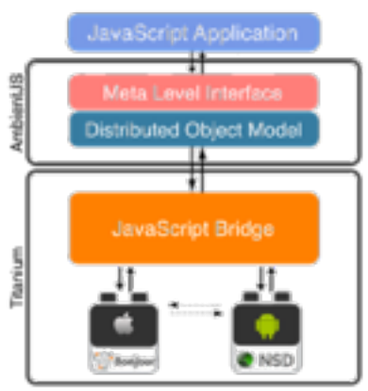
From Mobile to Cloud Computing



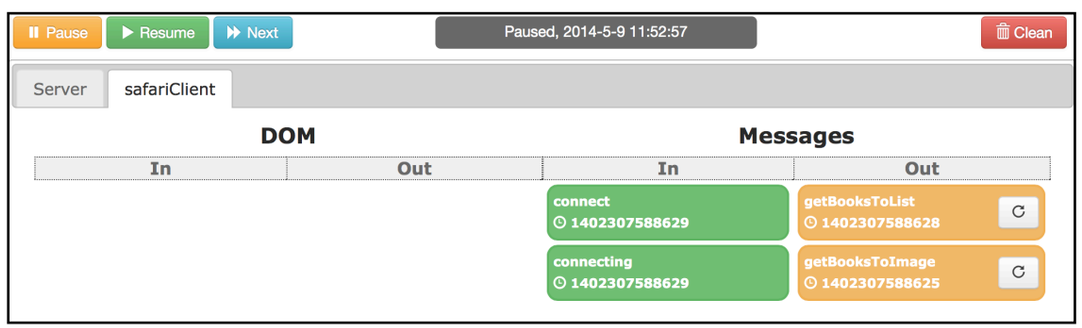
"What is the underlying mechanism that can control references to objects to build more secure systems?"

"How to build debugging support for intra event loop communication?"

Mobile Computing



AmbientJS



JAD

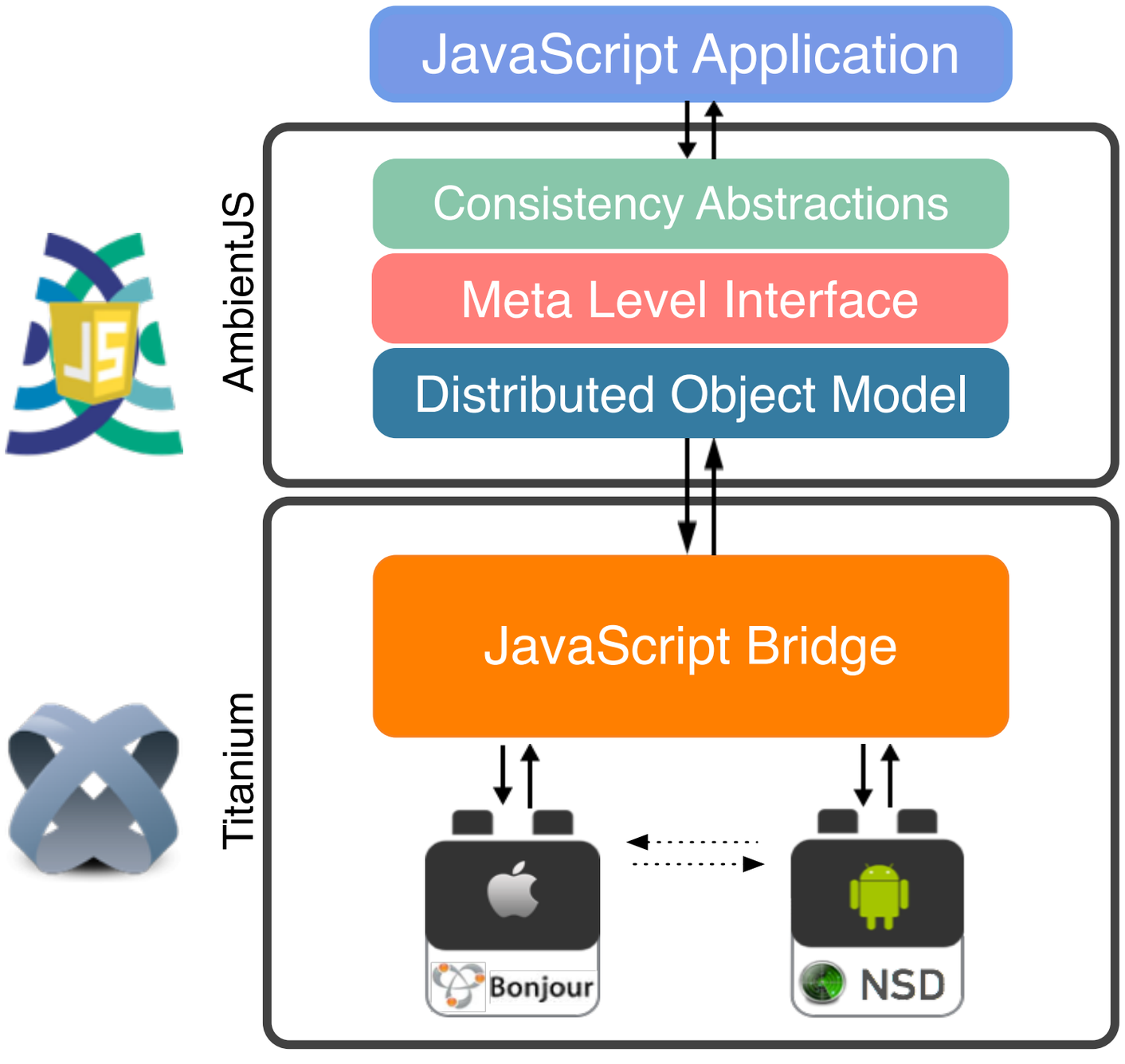
Consistency Abstractions

Distributed Debugging

Transmitter-Receptor Meta Model

Ambient-oriented Programming

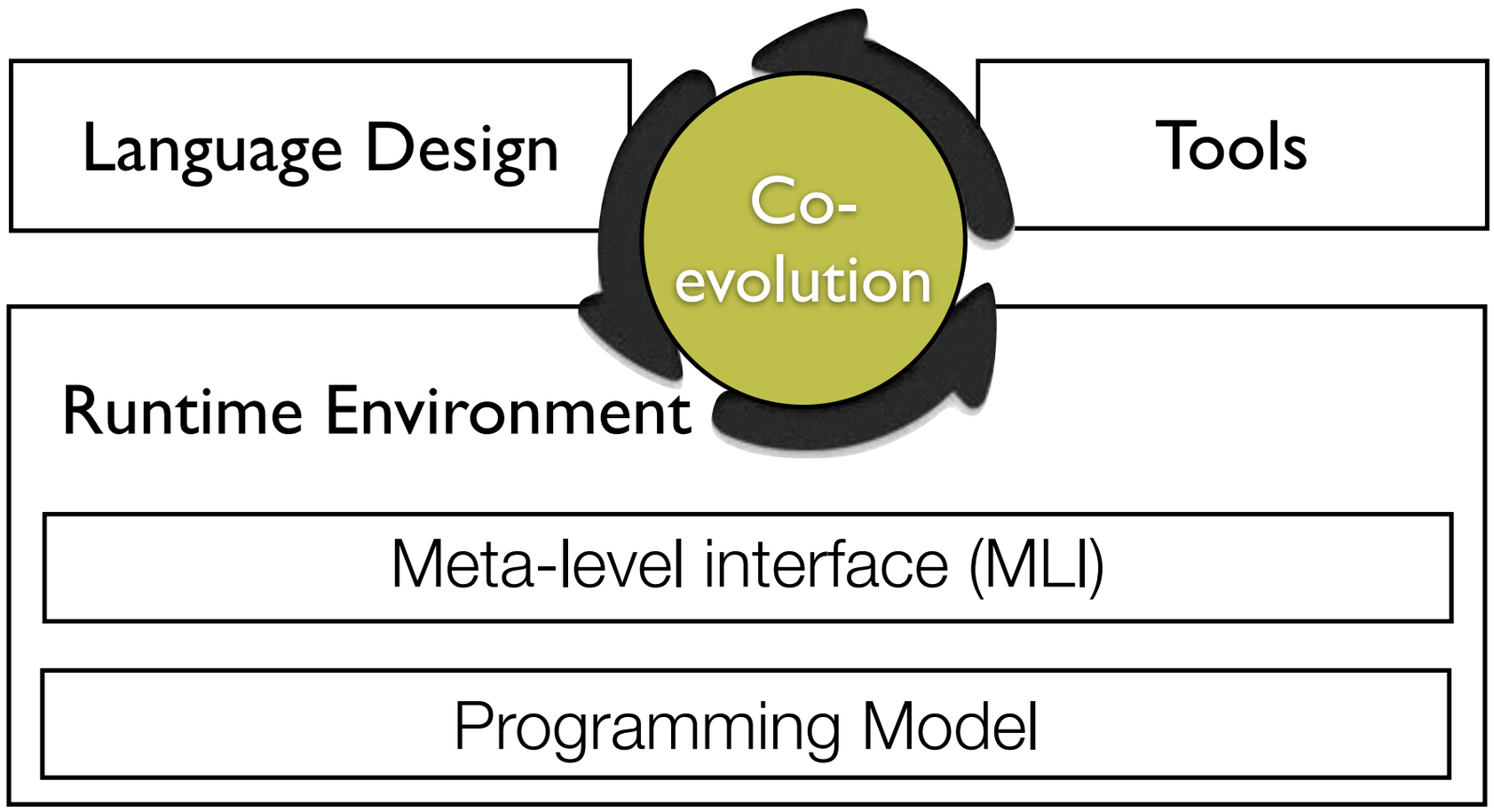
AmbientJS: Cross-platform Ambient-Oriented Programming



Consistency Objects as TR Pairs

```
1   var usedOffline = false;
2
3   function singleUseConsistencyObject(object){
4       var referenceProxy = AmbientJS.createReferenceProxy(function(delegate){
5           this.onReceive = function(msg) {
6               if(delegate.checkConsistency(msg)) {
7                   delegate.onReceive(msg);
8                   usedOffline = false;
9               }
10              else if (!usedOffline) {
11                  delegate.onReceive(msg);
12                  usedOffline = true;
13              }
14          }
15          this.onPassReference = ...
16      });
17      return AmbientJS.createObject(object,referenceProxy);
18  }
```

The Virtual Machines of Tomorrow



http://soft.vub.ac.be/soft/users/egonzale

https://code.google.com/p/ambienttalk/