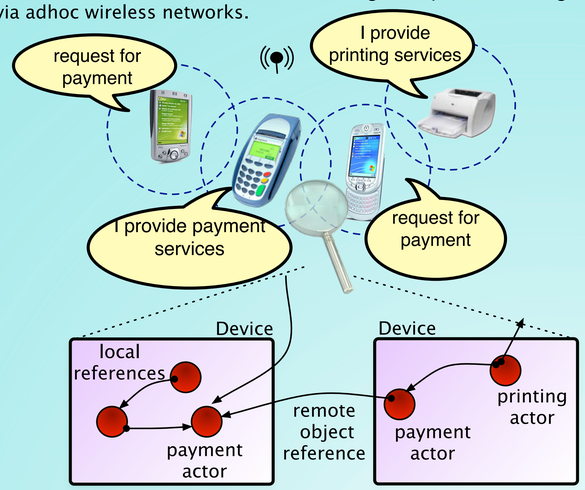
An Annotation System for Semi-Automatic Garbage Collection

Elisa Gonzalez Boix, Tom Van Cutsem, Stijn Mostinckx, Jessie Dedecker, Wolfgang De Meuter
Programming Technology Laboratory - Vrije Universiteit Brussel - Belgium

Motivation

Communication

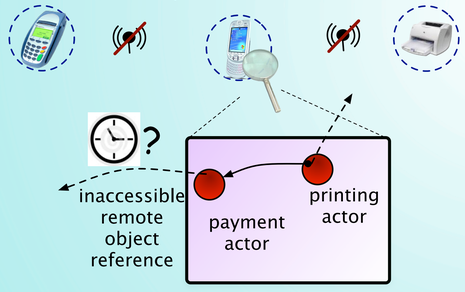
Autonomous concurrent devices roaming freely collaborating via adhoc wireless networks.



Devices consist of a set actors publishing and requiring services holding remote object references to actors on other devices.

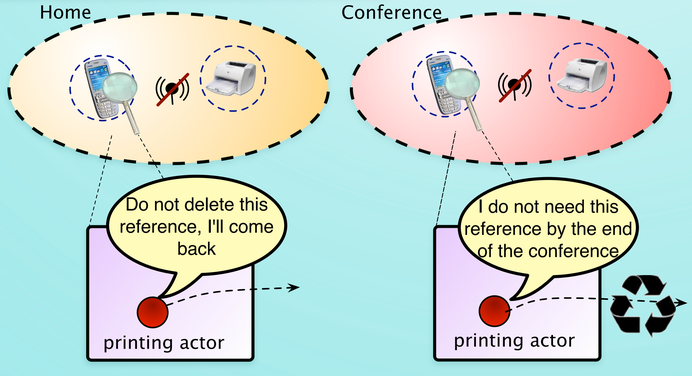
Failures

Disconnections are frequent



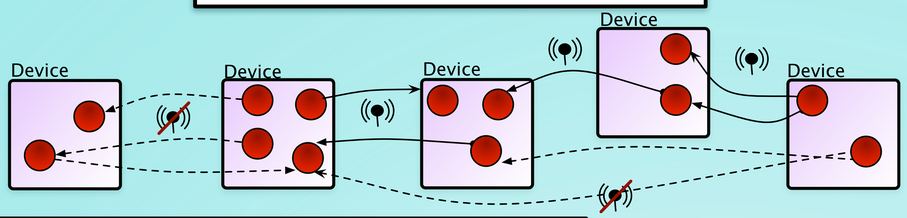
The system can never know if a remote reference is temporally lost or if it will be no longer accessible.

Context-Aware Applications



Semantics of the application are required to clear objects.

Problem Statement



Who is responsible for garbage collection?

Developer ▶ ~~Manual approach~~

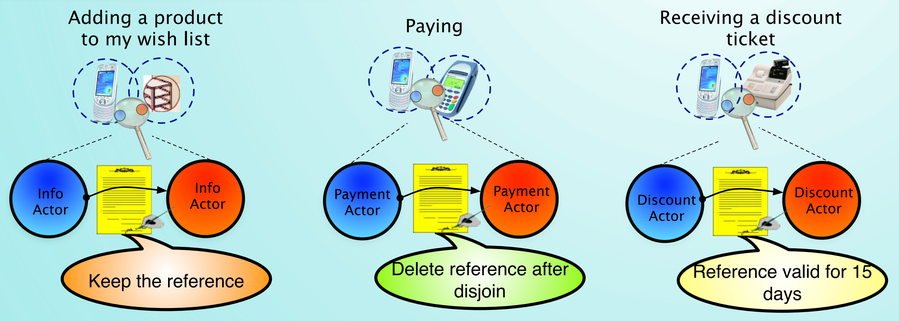
The System ▶ ~~Automatic Transparent Garbage Collection~~

A combination of both:

Semi-Automatic Garbage Collection

▶ The developer uses annotations to assist the collector to determine what to do with inaccessible references.

Ubiquitous Shopping Scenario




```
ShoppingScenario( customerId, guiActor ) ::
actor( object ({

  discount: vector.new();
  cashier: void;
  shopDesk : DeleteUponDisjoinRef(discover("shop"));

  requestPayment():{
    cashier: DeleteUponDisjoinRef(discover("payment"));

     when( ticket = cashier<-processPayment(customerId),
       {
        guiActor<-show(ticket);
        ticket.save();
        shopDesk<-requestDiscount();
         })
  }

  receiveDiscount(discount) :: {
    discounts.add( TimeBoundRef(discount.getExpiryTime(),
                                discount.getProduct()));
  }

  receiveInfo(from, info) :: {
    guiActor<-showText(from, info)
  }

  addWishList(productId):{
    productlist.add( KeepAlwaysRef("productId"))
  }
});
```

▶ Marking every remote object reference with annotations is cross-cutting.

Solution

Different events in relation to the remote object references (Annotations) ▶ pointcut

Actions to be taken with the remote object references (Contract) ▶ advice

Annotation System as an Aspect

Ubiquitous Shopping Scenario

pointcut ▶ advice

```
DeleteUponDisjoinRef()
  ForAll{
    call(discover("shop")
  }
  Ref{not( event(disjoint))
       && var(status, "checkOut")
  }

DeleteUponDisjoinRef()
  ForAll{
    call(discover("payment"))
  }
  Ref{event(disjoint)
       &&
      not(var(status,"InProgress"))
  }

TimeBoundRef()
  ForAll{
    call(discounts.add(discount))
  }
  { when(
  time = requestResource(
       discount.getExpiryTime()),
    Ref{dueTo(time)})
  }

KeepAlwaysRef()
  ForAll{
    call(productList.add(data))
  }
  Ref{}
```

▶ Idiomatic expressions used in the contract conditions.

Domain-specific language for garbage collection.

http://prog.vub.ac.be/amop