

Thinking about metadirectory !?

Presentation at EuroCAMP 2005

Porto, Portugal

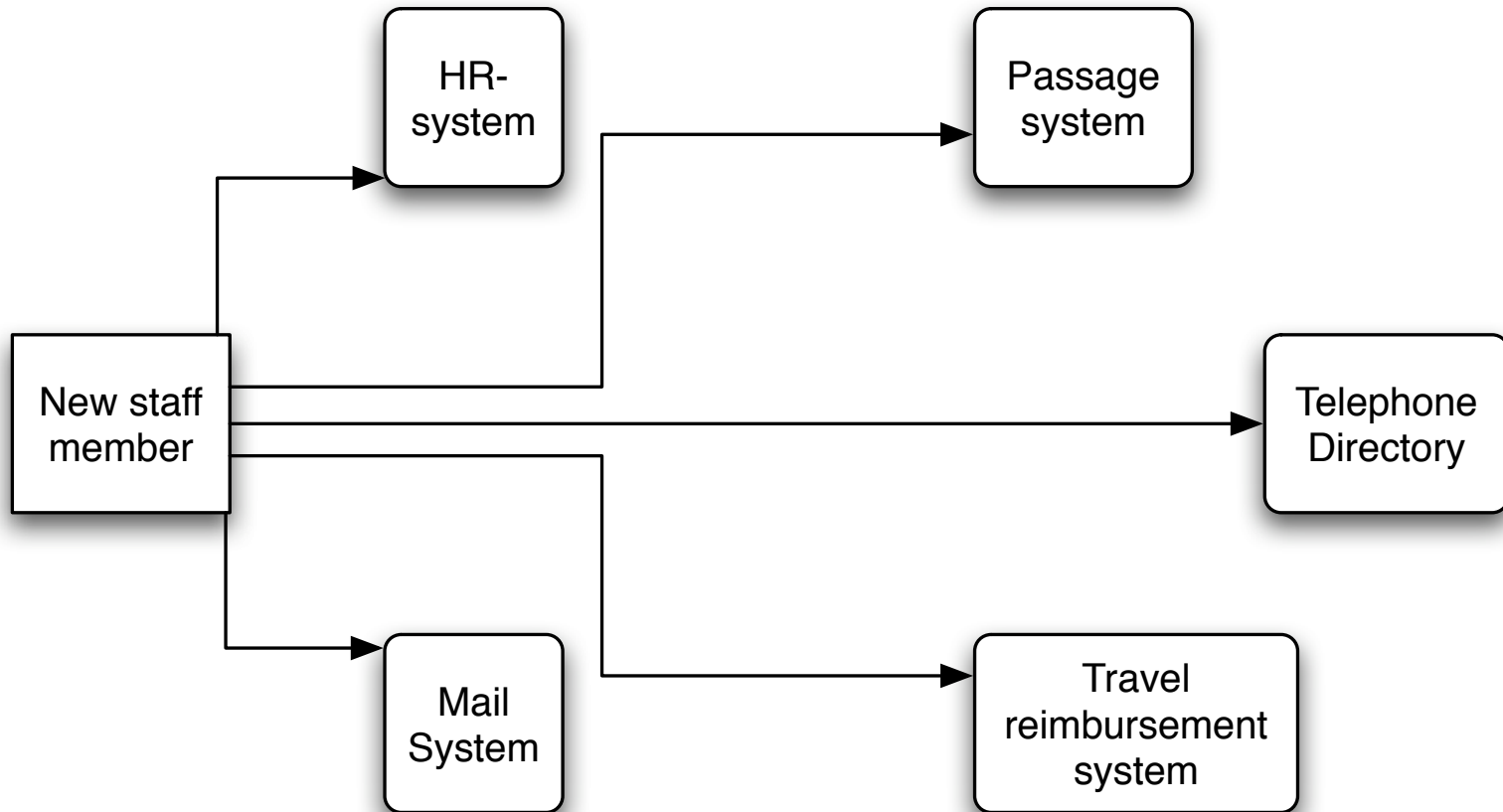
Roland Hedberg <roland.hedberg@adm.umu.se>

Definitions

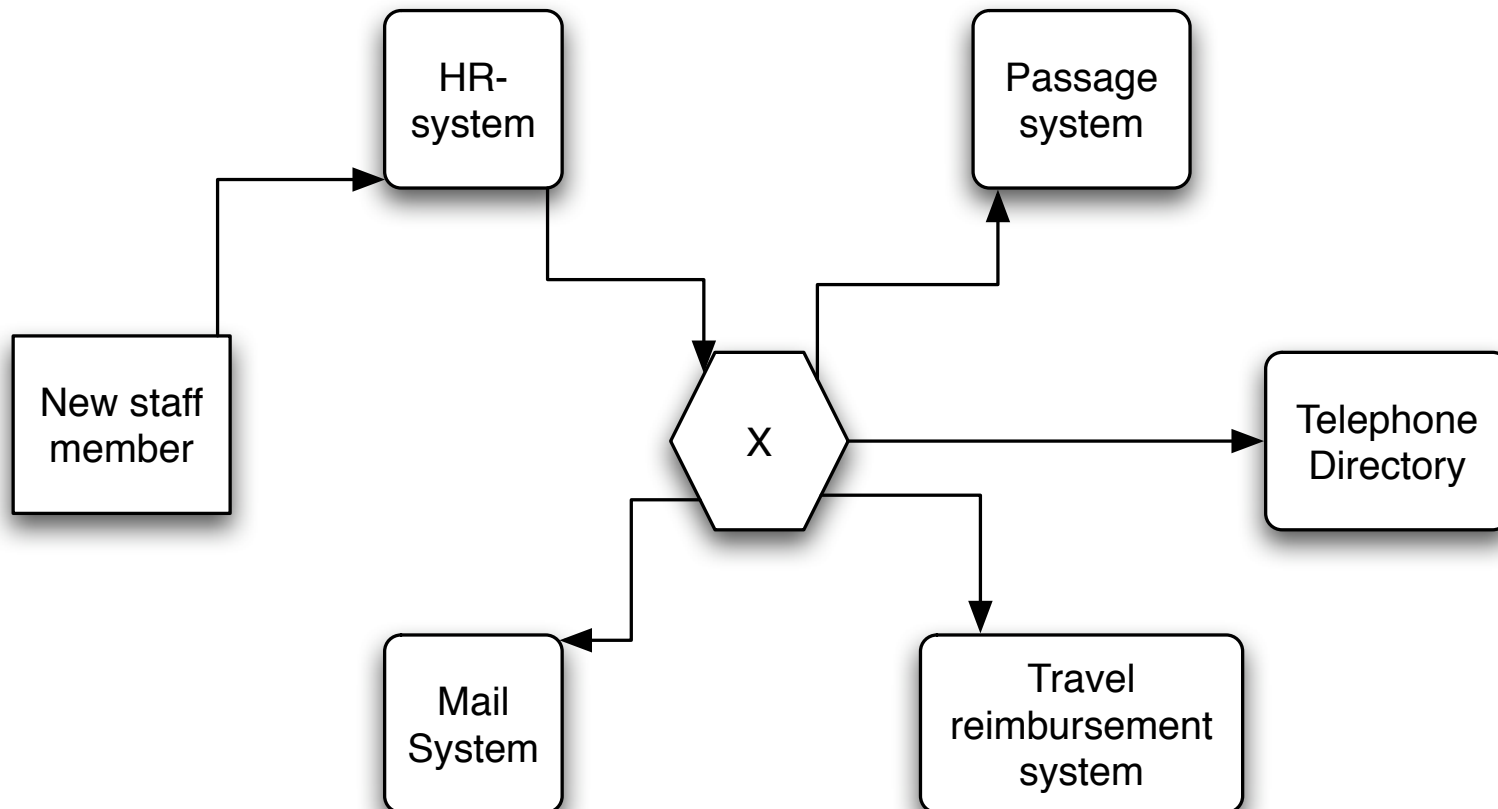
A ***metadirectory system*** provides for the flow of data between one or more directory services and databases, in order to maintain synchronization of that data, and is an important part of identity management systems. The data being synchronized typically are collections of entries that contain user profiles and potentially authentication or policy information.

User provisioning refers to the creation, maintenance and deactivation of user objects and user attributes, as they exist in one or more systems, directories or applications, in response to automated or interactive business processes.

Before



After



How to get there!

Item 1

◆ Draw the map !!

- Who is using/managing what about what (WAW)?
- Who needs WAW ? Not just internally but also externally !






Item 2

◆ You must have a Globally Unique Object Identifier for every object !!!

- Global == within the organization
- MUST NOT be changed, reassigned, or retired
- Basis for joining data from different sources
- Realize that external system wants access to the GUID

Item 3

The Attributes/WAW

-  Syntax
-  Semantics/Representation
-  Mapping tables
 -  sources => meta directory => sinks
 -  Constructing/deconstructing

Item 4

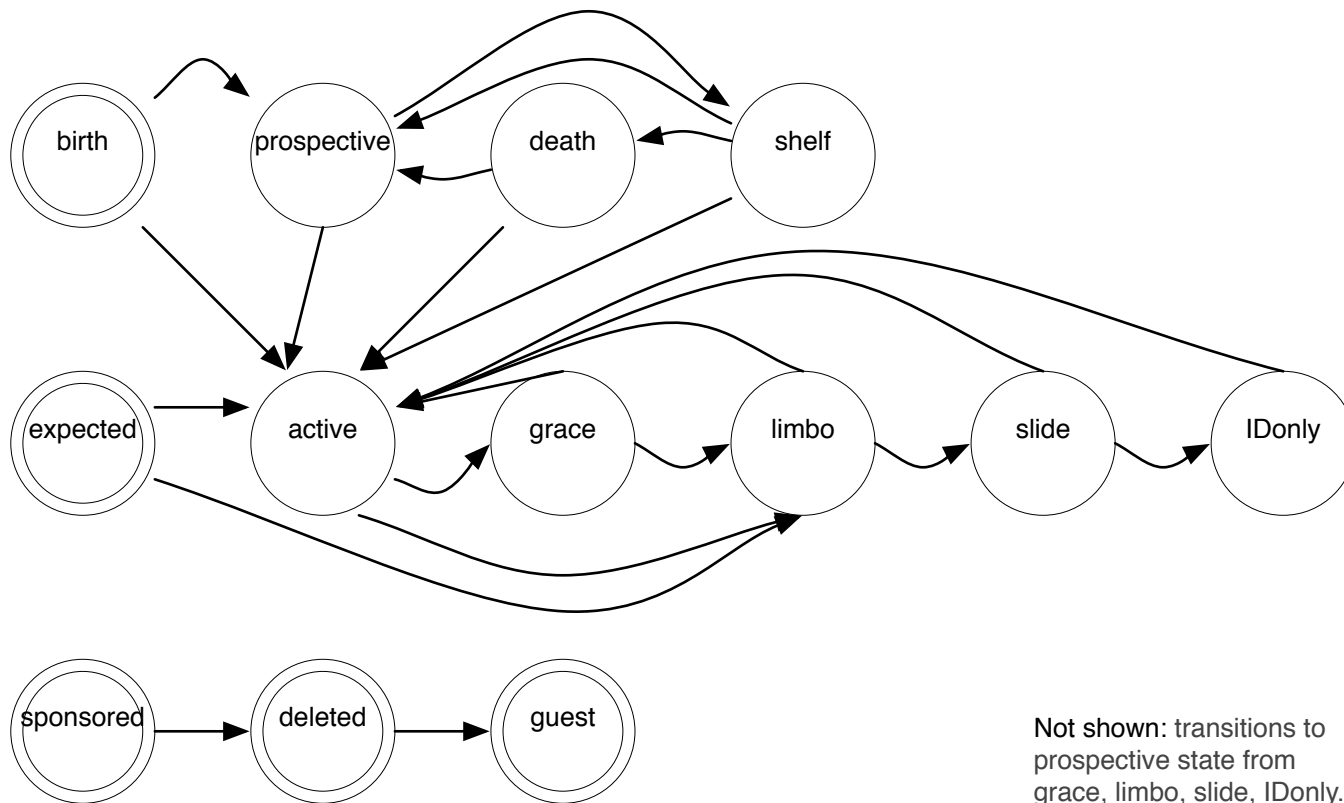


Policies

- Who may publish WAW ?
- Who should be the authoritative source for WAW !
- Who should get access to WAW ?

Item 5

◆ The life of an object (according to Tom Barton)



Item 6

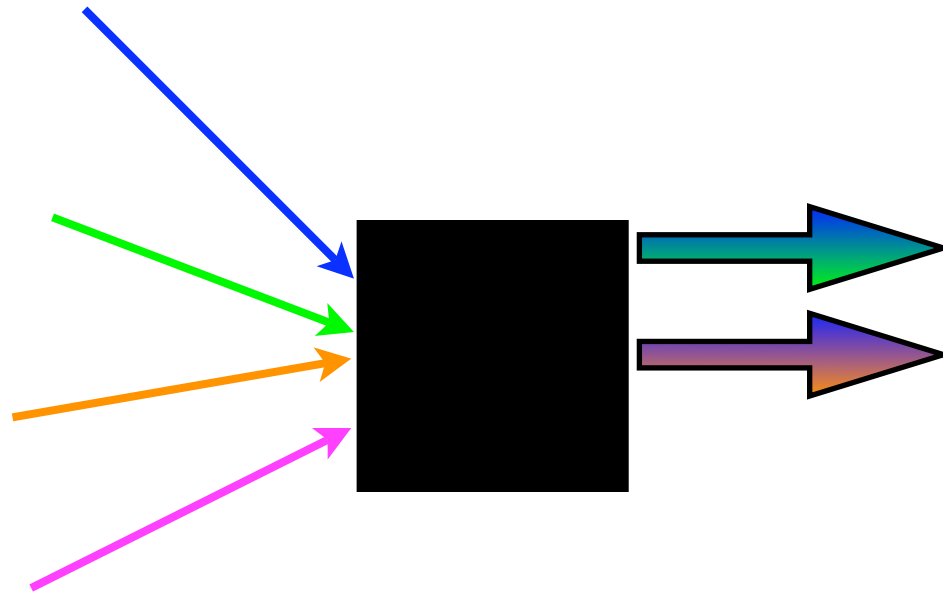
External access

- You need at least one official access point for 'external' systems. Possibly one or more LDAP directory
- You may want to 'export' views

Accessing information

- ◆ Ldap access is to one dimensional !!
- ◆ We are more and more moving to web service based access
- ◆ Who gets access to what, is not infrequently dependent of evaluation of some logic

The view so far!





*Open
Metadır*

OM features

- Represents events as RDF graphs
- Allows for dynamic routing of information
- Keeps a history of everything that has happend.
- High probability that it will not loose packets.
- Written in Python

More on OM

- ◆ The whole system is regarded as a virtual database on which one can perform certain operations

- ADD
- DELETE
- MODIFY

- ◆ There does not have to be one place where all information about one object can be found

OID server

- ◆ Constructs
 - omID (40 hexdigits)
 - umulD (64-bit int)
 - UID
- ◆ Stores maps between all oids for a object
- ◆ Allows read/write

RDF Graph example

```
<om:add> <om:src> "primula" .  
<om:add> <om:eid> "6648" .  
<om:add> <om:oid> "NIN:20051012-8575" .  
<om:add> <om:person> _:a .  
_:a <omat:norEduPersonNIN> "20051012-8575" .  
_:a <omat:givenName> "Per Arne" .  
_:a <omat:sn> "Nilsson" .
```

Modify add

```
<om:person> <om:oid> "NIN:19760808-8527" .  
<om:person> <om:eid> "2" .  
<om:person> <om:src> "primula" .  
<om:person> <om:add> _:guise .  
_:guise <om:guise> "anst2" .  
_:guise <om:guise> _:a .  
_:a <omat:umuSeOrgUnitID> "5602" .  
_:a <omat:eduPersonAffiliation> "faculty" .  
_:a <omat:umuSePersonEmploymentExtent> "100" .
```

Metadata

```
<om:person> <om:src> "AdminGUI" .  
<om:person> <om:eid> "I" .  
<om:person> <om:oid> "UID:anhe000 I" .  
<om:person> <om:add> _:a .  
_:a <om:guise> "anst I" .  
_:a <om:guise> _:b .  
_:b <omat:telephoneNumber> "+46 90 786 1234" .  
_:b <omat:telephoneNumber> _:c .  
_:c <omat:access> "intern" .
```

Routing

Matching rule (RDF => S-expression(spocp)):

(<om:add>

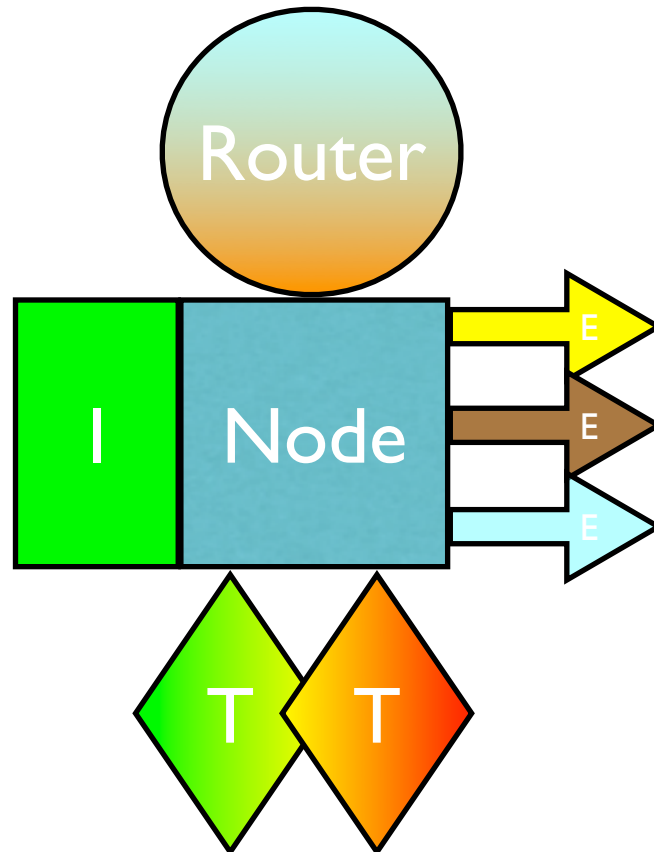
(* set

(<om:person>)(<om:eid>)(<om:src> primula)
(<om:oid>))

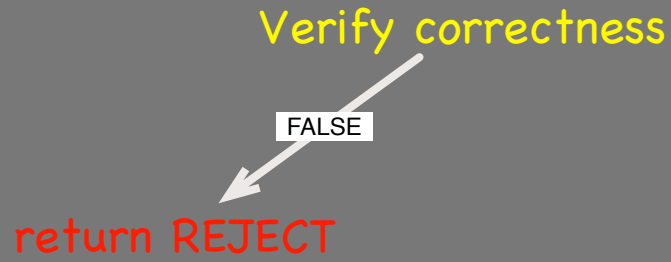
Return info:

"file:///to_abc om://dweller.catalogix.se:7890"

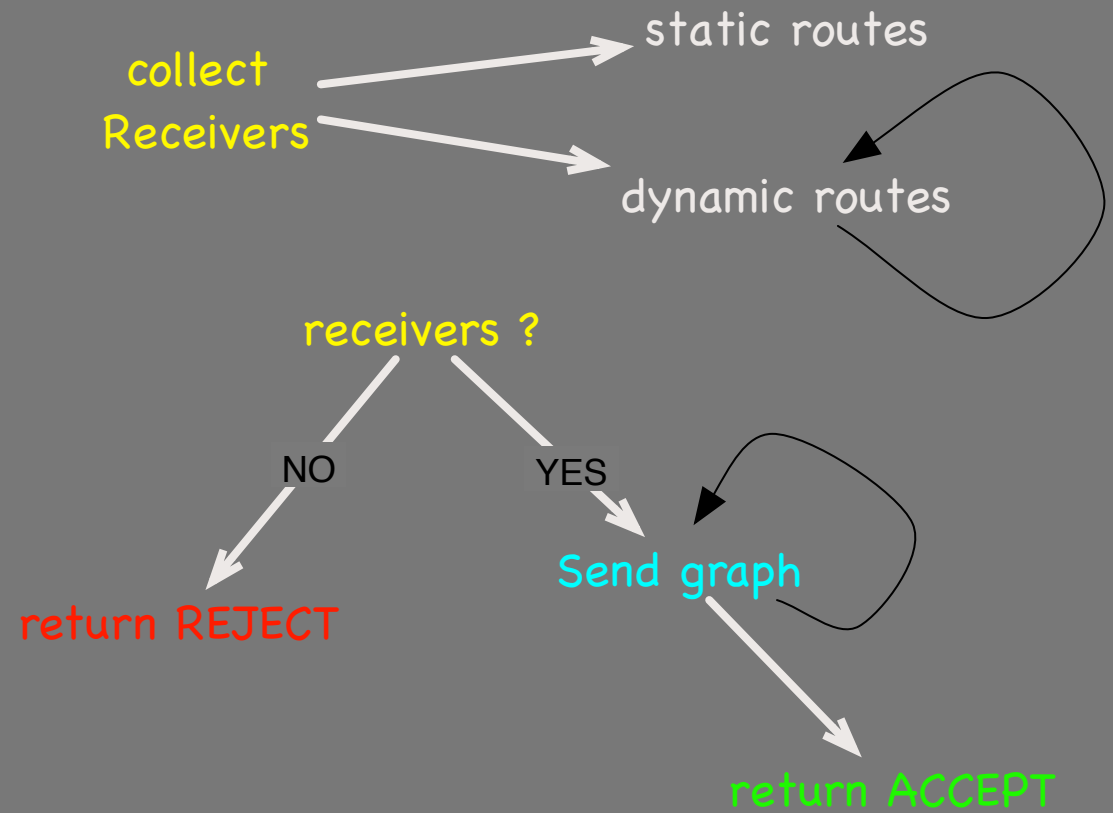
System architecture - Node = base component



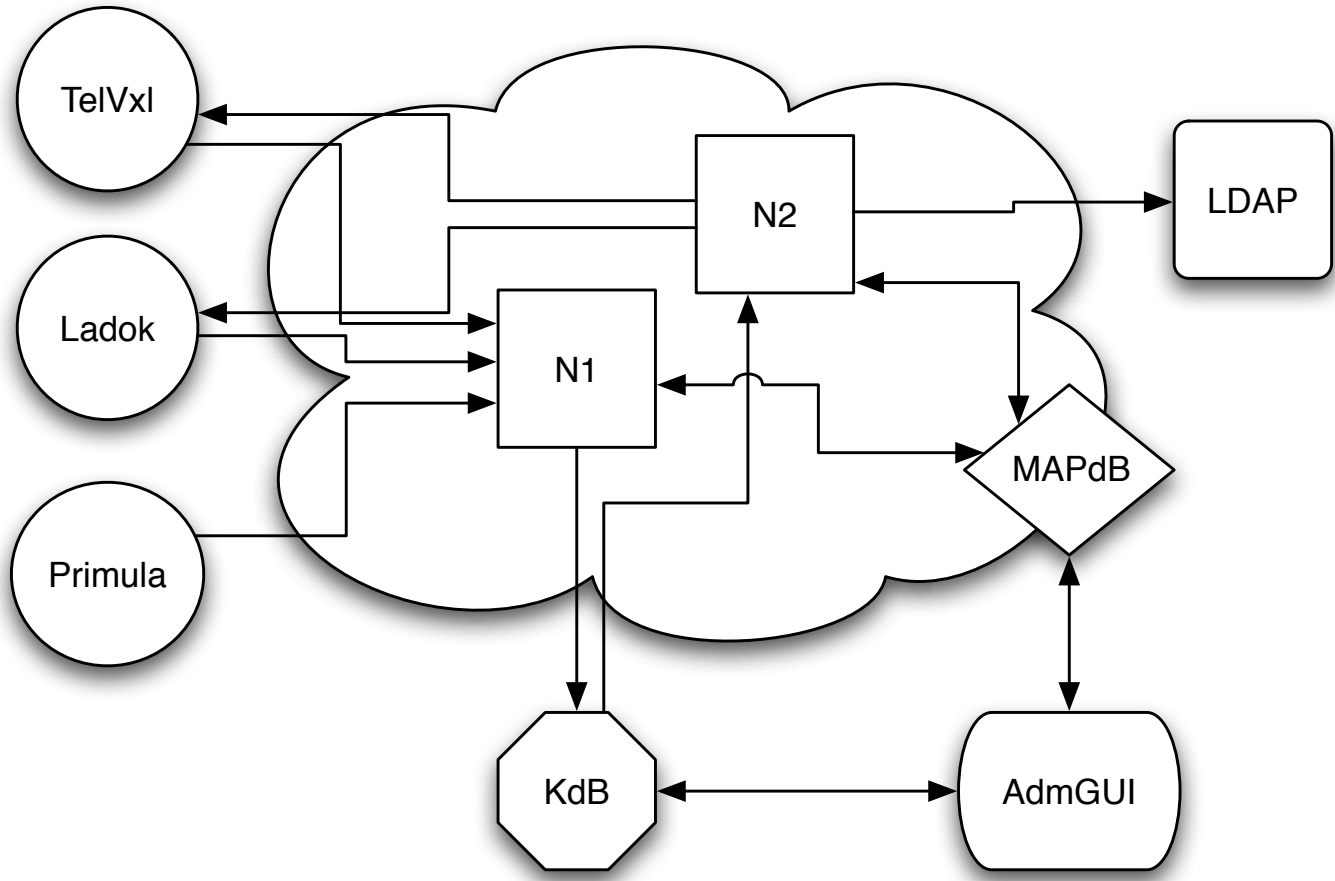
Incomming Graph



Transform ?



K2





What do you needed to do to use OM ?

- Adapters for your sources and sinks
- Decide on which nodes/modules to use and then to configure them

Configuration example

```
system:
  name: "primulaimp"
  logfile: "primulaimp.log"
  sleep: 5
  pstore: "primula_store"
importer:
  1:
    url: "file:///data/from_primula"
    type: oldest
transformer:
  1:
    url: "oidmap://localhost:8080"
    namespace: "http://dweller.catalogix.se/oidservice"
    to: omID
    map:
      UMUID: "<omat:norEduPersonLIN>"
      UID: "<omat:uid>"
  2:
    url: "orgoidmap://localhost:8080"
    namespace: "http://dweller.catalogix.se/oidservice"
    mustSucceed: yes
exporter:
  1:
    url: "file:///to_kdb"
    localroot: yes
    createDir: yes
```

Conclusion

-  Getting a MetaDirectory in place takes a lot of work, and thought, most of it not technical.
-  OM is a message based information router