Experiences in Implementing a Cross-domain Use Case by Combining Semantic and Service Level Platforms

Vesa Luukkala,Dirk-Jan Binnema,Mihaly Börzsei (Nokia)

Alessio Corongiu (Centro Ricerche Fiat)

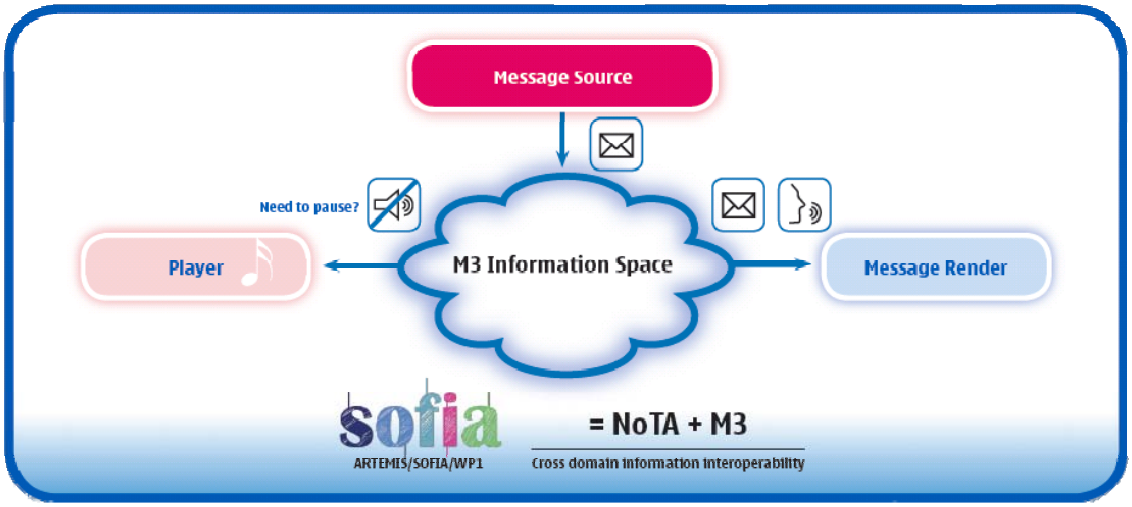
Pasi Hyttinen (VTT)

ISCC'10, June 22. 2010, Riccione

NOKIA

Motivation – our use case

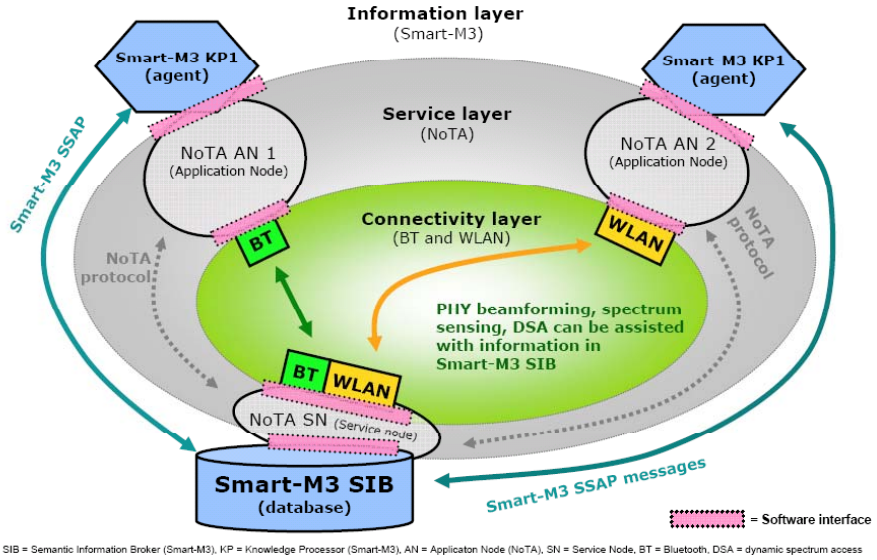
- Information interoperability enables access to embedded information and mashups
- Two use cases at the same time: Music Follows User, Read aloud message



NOKIA

Service and semantic level

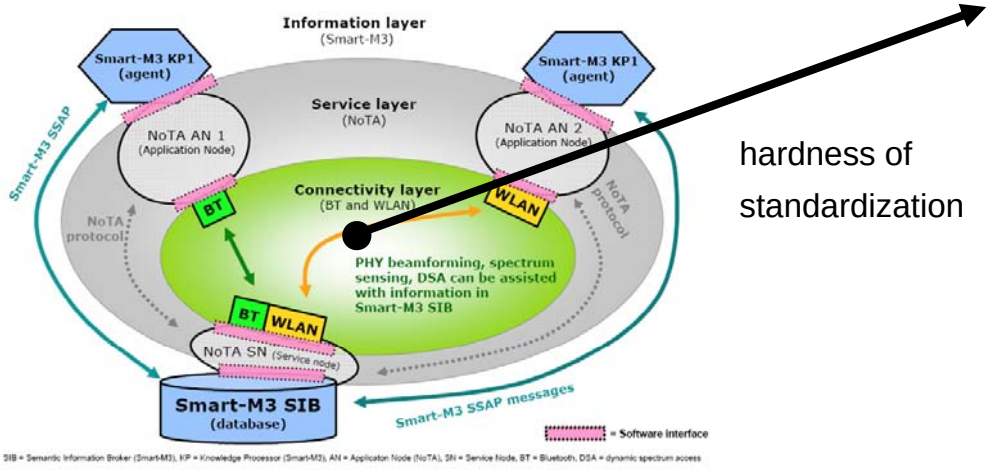
- Service level: interfaces to service implementations
- Semantic level: access to semistructured, self-describing information (RDF)



NOKIA

Semantic level problem

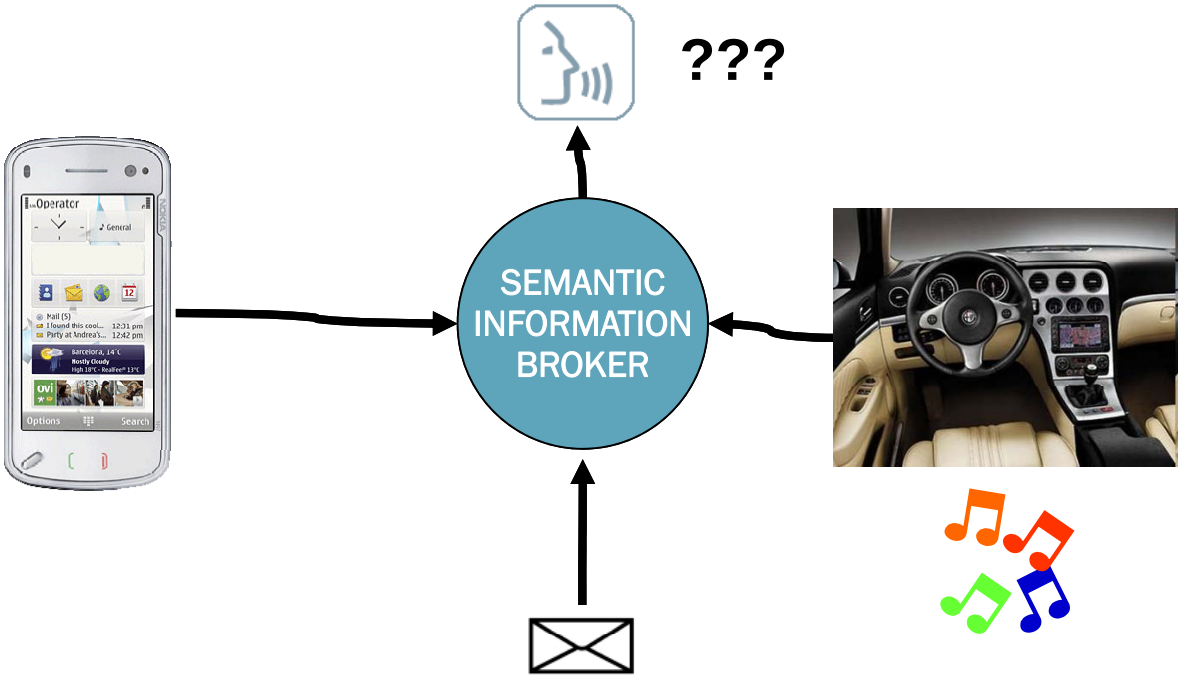
- Mash ups easy when just writers and one reader
- Truism: two way communication requires an agreement or protocol
- Standardization slow and heavyweight -> at information level standardize on RDF



NOKIA

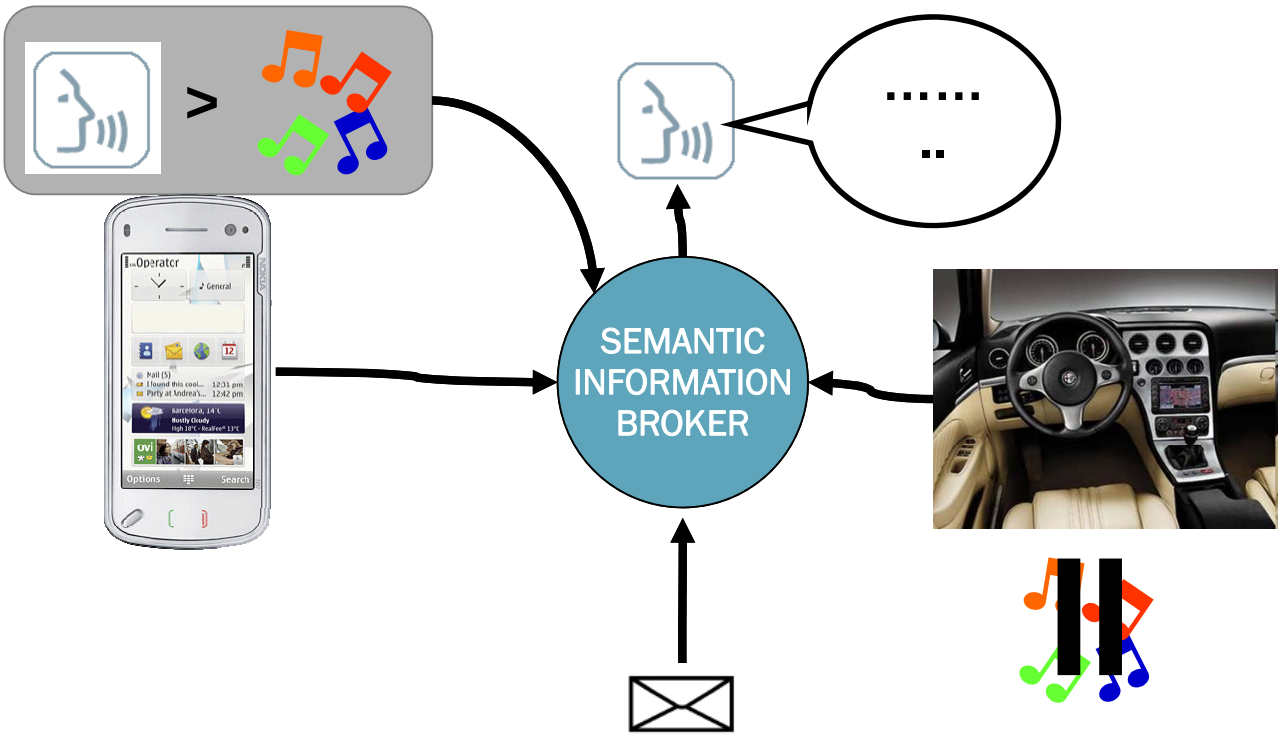
Semantic level problem for use case

- Who gets to use the **limited resource** of noisemaking?



NOKIA

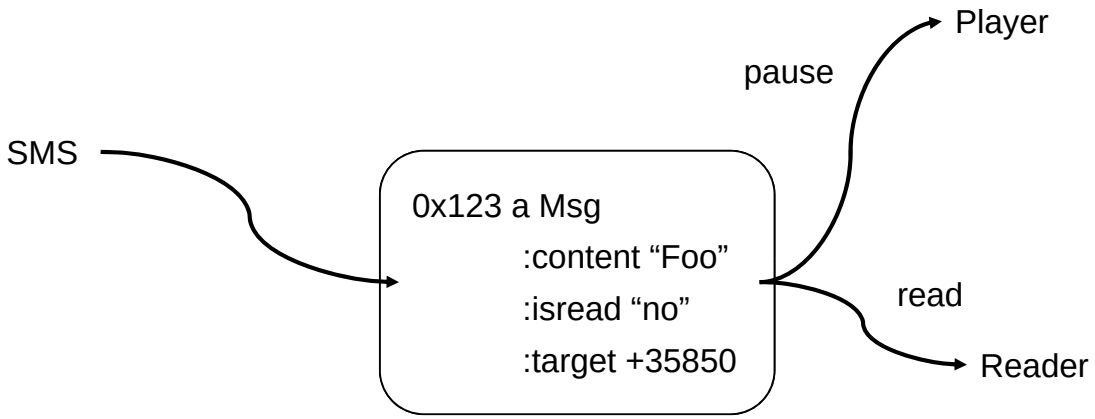
Semantic level problem for use case



NOKIA

Straightforward Attempt

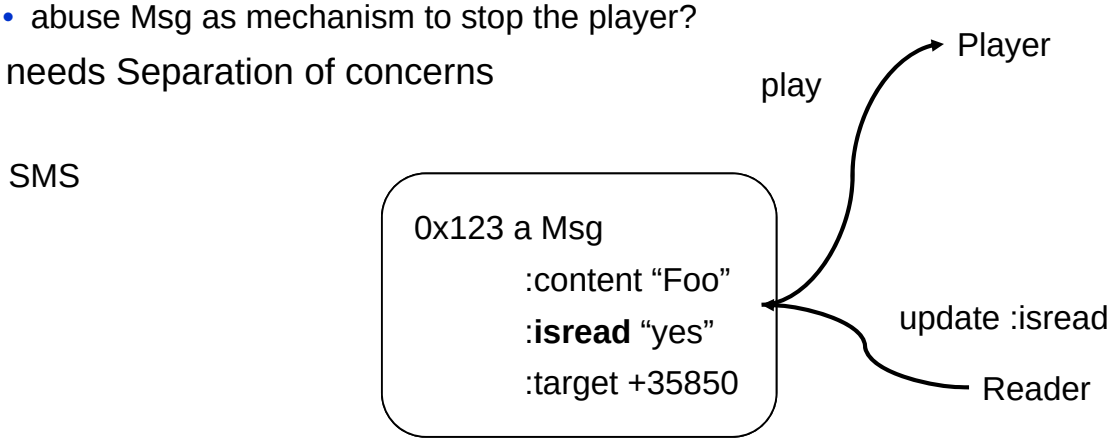
- Simply subscribe to the message



NOKIA

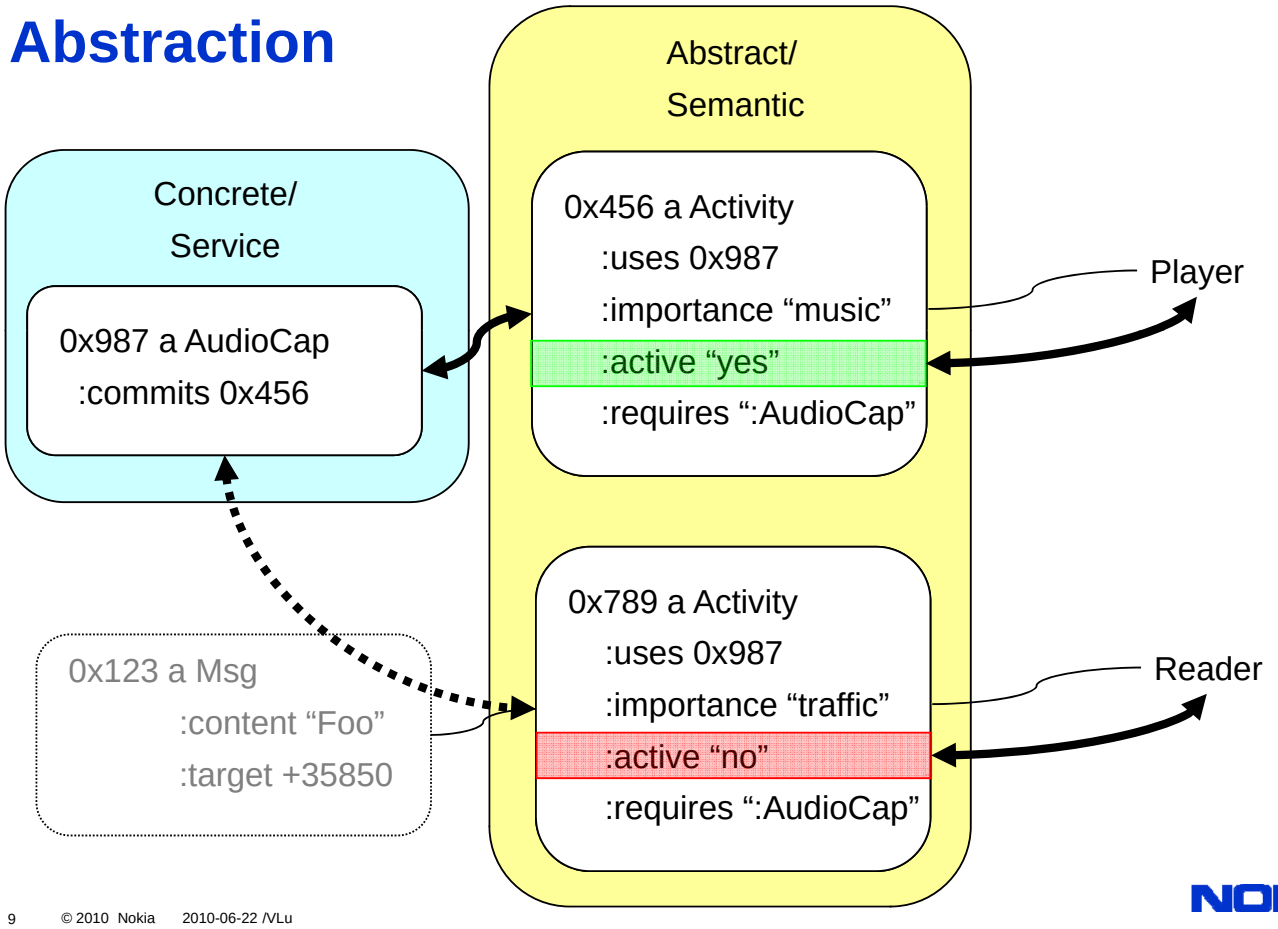
Straightforward Attempt

- Update and synchronize on a property, shared between player and reader
- What if player should be paused by something else than Msg
  - modify player to understand the new thing?
  - abuse Msg as mechanism to stop the player?
- -> needs Separation of concerns



NOKIA

Abstraction

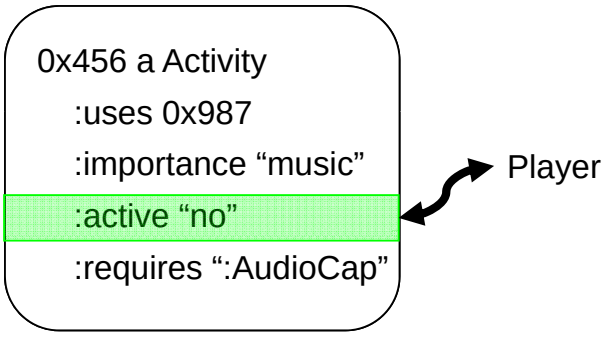


NOKIA

Abstraction For the “user “

- User creates an instance and promises to observe and honor “:active”
- It targets resources by :uses properties
- It may require **types** of resources

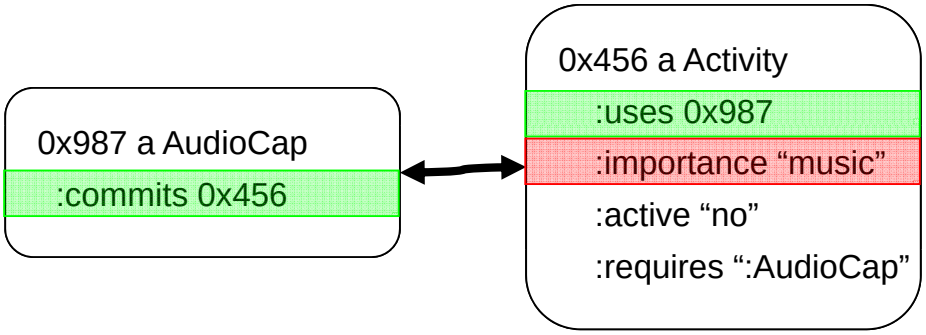
- An Activity instance is “virtual”



NOKIA

Service view

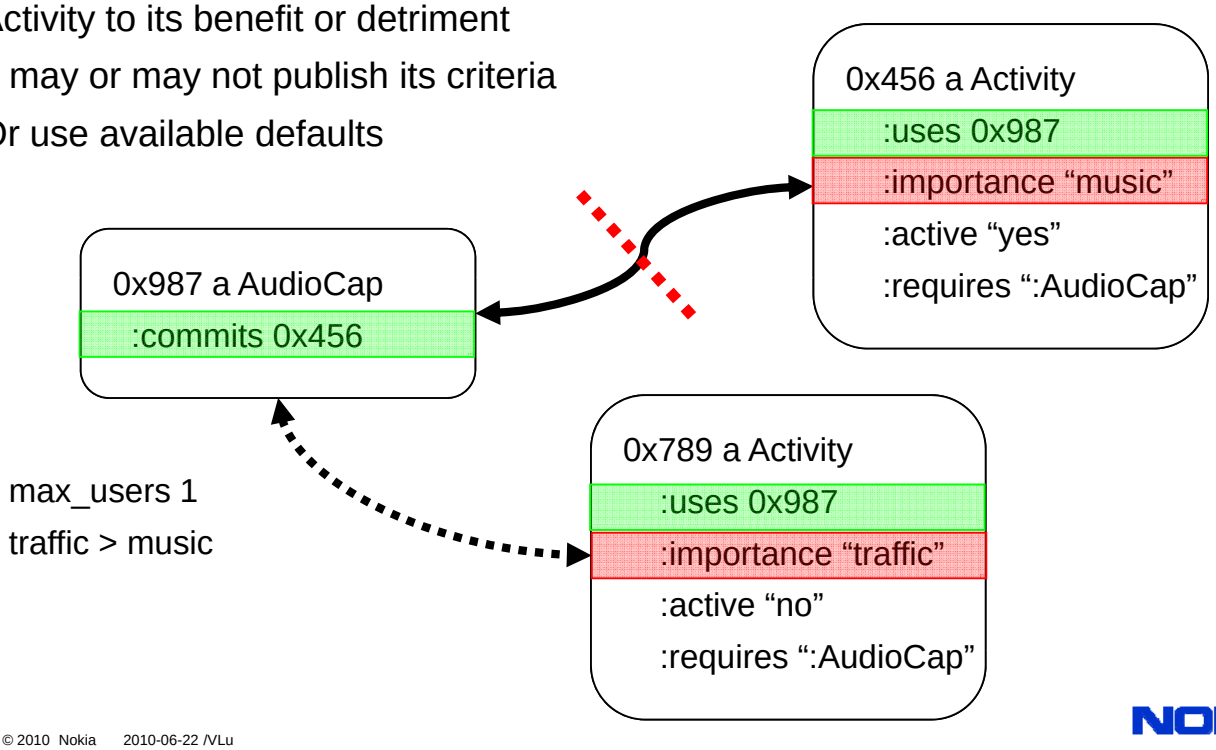
- A service implementation must represent itself semantically
- It may be pointed by an activity with :uses, to which it can reply with a :commit
- The service implementation may accept or deny :uses **locally**



NOKIA

Service view

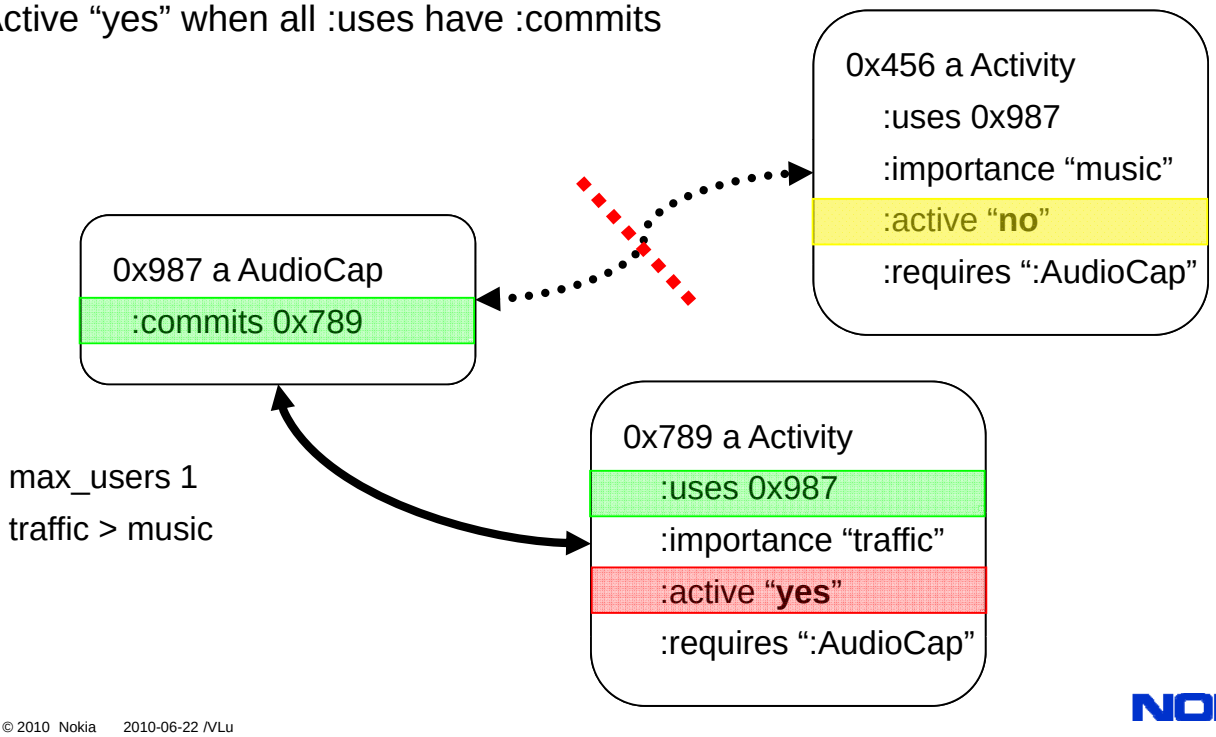
- A service implementation can interpret the additional information provided by Activity to its benefit or detriment
- It may or may not publish its criteria
- Or use available defaults



NOKIA

Local and co-operative scheduling

- An activity is reponsible for setting itself active and inactive
- Active “yes” when all :uses have :commits

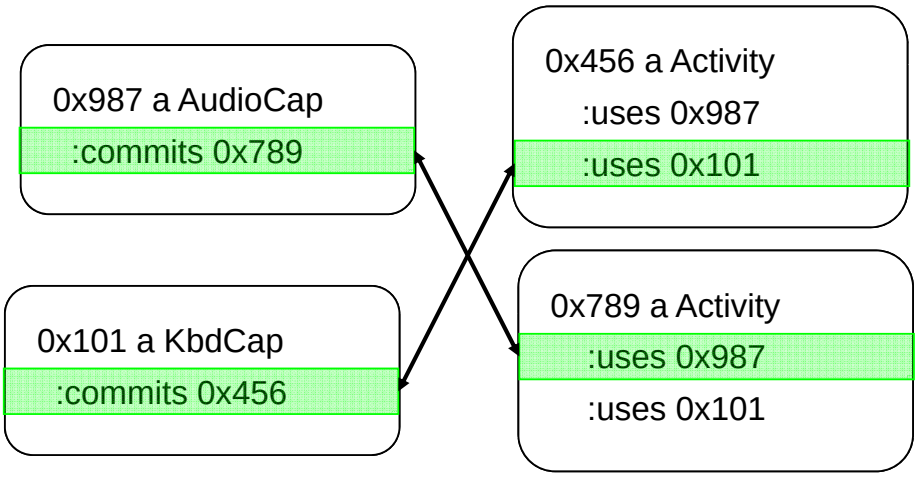


NOKIA

Local and co-operative scheduling

- Local decisions
  - Service implementations automatically yield to higher priorities
  - Activities locally monitor their status
- Are not always enough:

- Timeout and retry for Activities is a localized solution
- A global arbiter can also be implemented, but may not be scalable
- May not be fair…



NOKIA

The End -- Thanks

- We hope:
  - reasonable compromise between standardization and ad-hoc approach
  - lightweight and simple enough for Node and application implementations
  - saves our own work in implementing new use cases and scenarios
- Answer set programming rules implementing described arbitration available at http://sourceforge.net/projects/ssls
- We expect:
  - identify (and implement) other similar patterns
  - contribute to core ontologies along with behaviour implementations
- Question:
  - Here deliberate separation of Activity from Content (e.g. SMS)
  - RDF enables same instance to be interpreted as different Classes
  - Would it make sense always combine the Content and the Activity as same instance?

NOKIA