Adding High-level Reasoning to Efficient Low-level Context Management: a Hybrid Approach

Daniela Nicklas¹, Matthias Grossmann, Jorge Mínguez, Matthias Wieland

Universität Stuttgart

¹from 31.3.2008: Carl von Ossietzky-Universität Oldenburg
daniela.nicklas@uni-oldenburg.de
Motivation: design of smart room application (Conference Guard) on top of a shared context model (Nexus)

Two different implementations:
- A: with distributed event service and local application logic
- B: with general purpose Semantic Service (OWL ontology and rule-based reasoning)

Stress Testing the Semantic Service
- big T-Box $\rightarrow$ small A-Box

Hybrid Approach: reduce the number of instances

Challenges
Nexus Smart Room Environment

- Ambient Orb
- Working area
- Meeting area
- Phonometer
- Motion sensor
Nexus abstraction: shared context model

Applications

Update(id, value)

continuous query (filter)

event (condition)

(shared) context model

Sensoren (Fusion)

physical World

Update(id, value)

Applications
Nexus context information model

- **Standard Class Schema**
  - Fundamental object classes
  - Needed by most applications
  - Ensures interoperability
  - Unique identifier for every object instance

- **Extended Class Schema**
  - For future applications and integration
  - Objects inherit from Standard Class Schema

**Diagram:**
- Standard Class Schema
- Extended Class Schema 1
  - Italian restaurant
    - pizza-price: ...
    - deliver: yes/no
- Other Extended Class Schemas

Universität Stuttgart
SFB 627
Dr. Daniela Nicklas
Conference Guard A

- Application on top of shared context model
- Uses context queries for service discovery
- Uses distributed event service to observe context changes
- Contains application-side logic to derive actions based on context changes

Diagram:

1. **MeetingArea. occupied = true?**
   - Yes
   - No
2. **Phonometer. soundlevel > 1000?**
   - Yes
   - No
3. **MotionSensor. motion = true?**
   - Yes
   - No
4. **AmbientOrb. color = green**
   - No
5. **AmbientOrb. color = yellow**
6. **AmbientOrb. color = red**
Conference Guard A Architecture (with Event Service)

- **Application Tier**
  - Conference Guard
  - Other Applications

- **Federation Tier**
  - Nexus Node
    - Context Queries
    - Event Service
    - Value Added Service
  - Federation (Shared Context Model)

- **Service Tier**
  - Nexus Node
  - Context Server
  - Sensor
  - Actuator
  - External System
  - Event Source

- **University of Stuttgart**
  - SFB 627
Conference Guard B: with Semantic Service

- Approach: use standard ontology technology (i.e., Protegé, OWL, Jena) to create a general (but configurable) Semantic Service for the Nexus architecture
- Semantic Service offers rule-based reasoning based on facts from Nexus shared context model concepts
- Implement Conference Guard scenario using rule set within Semantic Service
  - extend Nexus architecture by higher level context support
Semantic Service Architecture

- **SSQL (Semantic Service Query Language):** used to query derived (higher-level) information
- **NOML (Nexus Ontology Modeling Language):** used to represent results of the Semantic Service
- **Ontology Manager:** manages different ontology models
- **Rule-based Reasoner (Jena):** applies pre-defined rules to the ontology models
Simple context model with 6 entities and 9 properties

One instance of context model per observed room

Set of rules to derive properties

Different property types:

- sensed (^^): from real or logical sensor
- derived (*): determined by rule set
- active (a): change of property results in change in real world (actuator)
- static (no symbol): defined by context modeler
for all rules:
...check types and partOf relationships of context model
...get values of MotionSensor (ms).motion, MeetingArea (ma).occupied (bool) and
   Phonometer (p).soundLevel (int)
ruleGreen:
   equal(?ms_motionDetected_val, '0')
   equal(?ma_occupied_val, '0')
rulesYellow1:
   equal(?ma_occupied_val, '1')
   le(?p_soundLevel_val, 1000.0)
ruleYellow2:
   equal(?ms_motionDetected_val, '1')
   le(?p_soundLevel_val, 1000.0)]
ruleRed1:
   equal(?ma_occupied_val, '1')
   greaterThan(?p_soundLevel_val, 1000.0)
ruleRed2:
   equal(?ms_motionDetected_val, '1')
   greaterThan(?p_soundLevel_val, 1000.0)
Stress Test of the Semantic Service

- T-Box (Nexus classes): 143
- A-Box (Smart Room objects):
  - each AWM class and each AWM attribute results in one A-Box instance → 39 instances per smart room
- Results indicate an exponential growth

→ reduce number of instances!
Hybrid Approach

- Semantic Service: value added service in the Nexus architecture -> general reasoning service over the shared context model
- Goal: reduce the number of instances for the semantic service
- Approach:
  - Semantic Service at federation level populates its local ontology using context queries
  - Semantic Service registers events at event service to get notified when context changes (sensed attributes)
Conference Guard A Architecture (with Event Service)

Application Tier
- Conference Guard
- Other Application

Federation Tier
- Nexus Node
- Context Queries
- Event Service
- Value Added Service

Service Tier
- External Systems
- Context Server
- Sensor
- Actuator
- Event Source

Area Service Register
Conference Guard B Architecture (with Semantic Service)
Challenges and Future Work

- Heuristics to determine good context queries for A-Box population
  - here: one room is enough. But in general?
- Add (context-aware) cache replacement strategies
  - A-Box can be seen as partial cache of shared context model
  - When to remove unimportant entities?
- How many instances of Semantic Services do we need?
  - group by application (rule set) or by spatial layout (shared instances)?
- Full integration of actuators (active properties)
Thank You!

questions, remarks?