On the Applicability of Rule-Based Programming to Location Inference

Katharina Hahn, Kirsten Terfloth,
Georg Wittenburg, and Jochen Schiller
Freie Universität Berlin

3. GI KuVS Fachgespräch "Ortsbezogene Anwendungen und Dienste", Berlin, Germany

Problem Statement

- Location inference is a fundamental building block for location-based services (LBS).
- Location inference algorithms may work on a variety of input data:
  - GPS coordinates + stored data (e.g. maps, ...)
  - Information from local radio transmissions (e.g. RFID, ...)
  - Information from sensors on the device
- Additionally, a location inference algorithm should
  - follow event-like semantics, and
  - run efficiently on resource constrained devices.

➢ How can location inference be implemented elegantly?

Idea / Motivation

- Rule-based programming naturally suits both
  - diverse input formats, and
  - provides event-like semantics.
- The FACTS middleware architecture provides a rule engine requiring no more than 8 KB.

➢ Implement a rule-based location inference algorithm on the FACTS middleware architecture.

Outline

- Related Work
- Short Introduction to Rule-Based Programming
- FACTS Middleware Architecture
- LBS on FACTS
- Example: Dating Service
- Conclusion

Related Work (1)

- Nimbus [1]
  - Platform for location aware applications
  - Mobile fraction
    - Integration of diverse positioning systems
  - Network fraction
    - Resides on decentralized, self-organizing, logically coupled servers
    - Enhancement of location information
  - Nimbus services
    - Provide semantic position (campus), meta-information (wood, building)
    - Location based services (trigger, geocast)
  - Architecture
    - Application layer
    - Service layer
    - Base layer

Related Work (2)

- Nexus [2]
  - Generic plattform to support location aware applications
  - Environmental model
    - Spatial models of physical surrounding
    - Combination of several heterogeneous spatial models
    - Enrichment through virtual objects and services
  - Augmented world model
    - Enhancement of spatial model with virtual information spaces
  - Support context-aware applications and location-based services, among others
    - Location based communication
    - Spatial Events

Related Work (3)

- TRANSIT [3]
  - Proactive guidance of visitors of Olympics 2008 (Bejing) to their demanded destination
  - Personalized and situation-dependent real-time information supply
  - Components
    - Traffic Information Center
    - Routing Engine
    - Services
- Tourist Information Provider [4]
  - Event Notification in Location-based Services
  - Provide tourists with information of sights based on context
    - Location
    - Personal interests (profile)
    - Travel History
  - TIP Services
    - Recommendation service
    - Map service

Rule-Based Programming (1)

- General idea:

  A rule consist of a

  - Condition part – which circumstances lead to the execution of a rule
  - Action part – what actions are undertaken in case a certain state of the system is reached

“ **If** the device reaches state x
and information about y is available

then trigger the execution of z.”

Rule-based programming (2)

Motivation

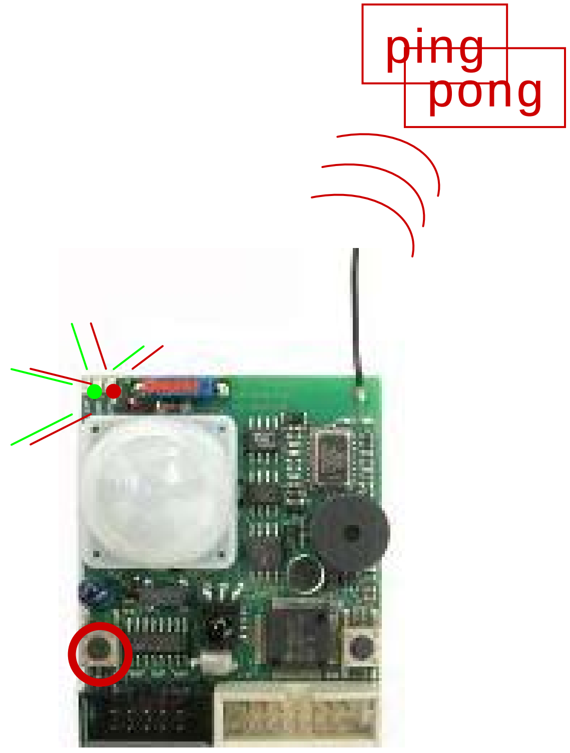
- Event semantics can be expressed naturally
- Energy-efficient approach for portable devices
  - Reactions to incoming data
  - Low-power mode of device in case no action needed
- Rules are stateless and modular in design
  - Upgrades/changes easily possible at runtime
- Language can be very concise
  - No dissipation of constraint resources
  - Well-suited for small devices

Example: PingPong ruleset

```
rule button 150
<- exists {button}
-> retract {button}
-> define ping
-> send 0 15 {ping}
-> retract {ping}
-> call toggleGreenLED

rule ping 100
<- exists {ping}
-> retract {ping}
-> define pong
-> send 0 15 {pong}
-> retract {pong}
-> call toggleYellowLED

rule pong 100
<- exists {pong}
-> retract {pong}
-> call toggleRedLed
```



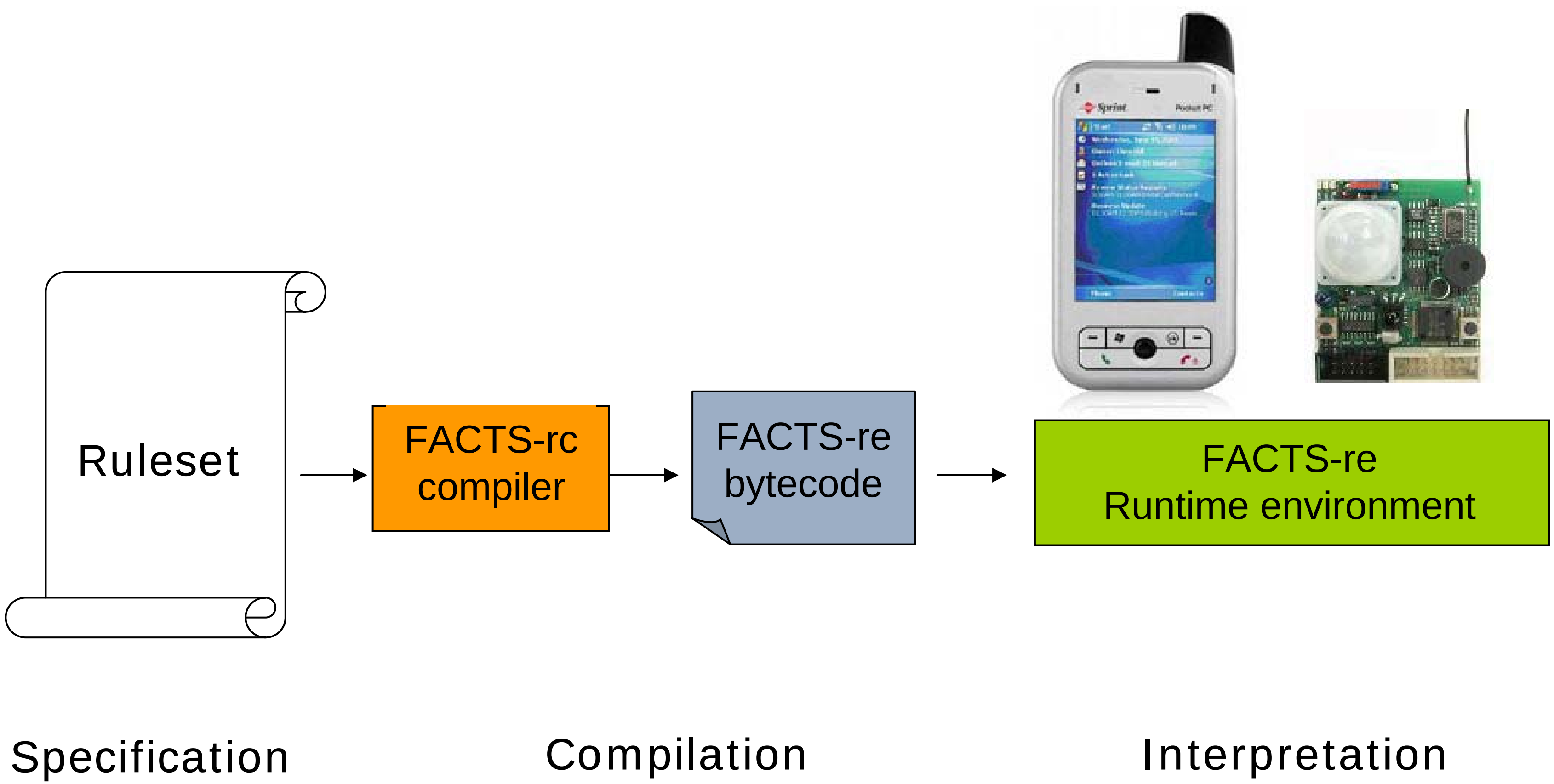
Condition: push / pong

Condition: ping

FACTS Middleware Architecture

- FACTS is a runtime environment for rules which are compiled to bytecode
- Building blocks:
  - Facts – Representation of any data in the system (state, sensor readings, global variables, etc.)
  - Rules – Specification of algorithms for data processing
  - Functions – Support for native code
- Advantages:
  - Inherent energy efficiency
  - Sandboxed execution environment prevents runtime failures due to managed memory access
  - Uniform data representation abstracts from underlying layers
    - Radio interface
    - Sensor hardware

FACTS Toolchain

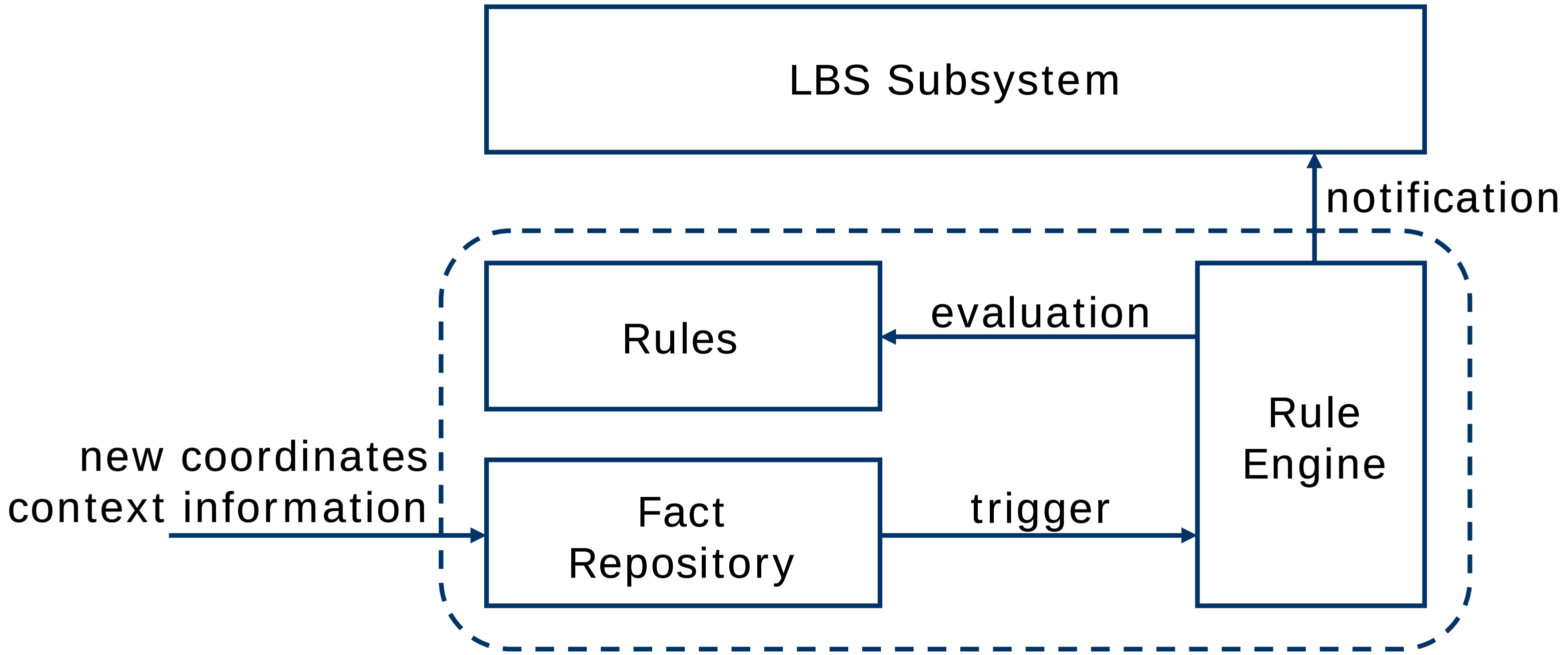


Rule Execution

1. Data (sensor data/radio packets/etc.) is received
   - ➔ Formating of this data into facts abstraction

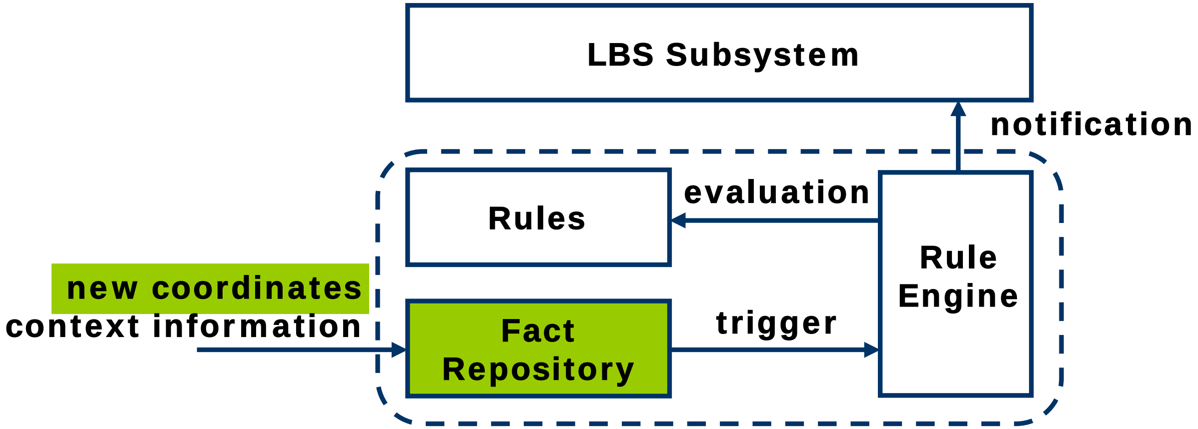
     (named data-value pairs)
   - ➔ Triggering of the rule engine
2. Rule engine checks conditions of available rules
   - ➔ Match of rule condition to available facts leads to rule execution
   - ➔ In case no match is found, the system stays quiet
3. Rules may produce new facts, thus trigger other rules
4. Rule execution stops when no modification to the fact repository has been performed in the last round.

LBS on FACTS


LBS Subsystem
notification
Rules
evaluation
Rule
Engine
new coordinates
context information
Fact
Repository
trigger

Rule-based EN on FACTS

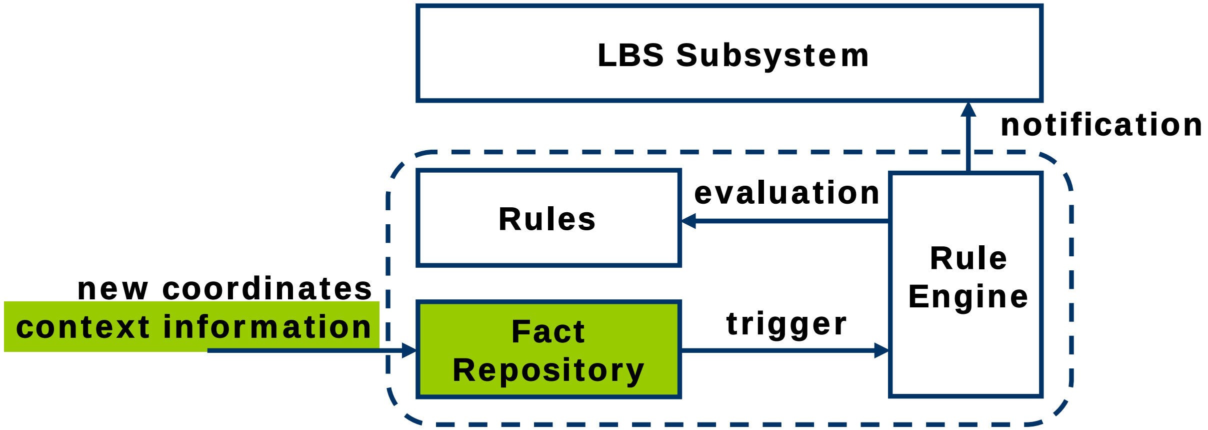
- Precise location matching
  - Sensor provides location information
  - Match location information to semantic location through previously defined fact
    - Fact location [N=52°30.62, E=13°24.97, place = „Goya“]



Rule-based EN on FACTS

- Abstract Location Matching
  - Inference of *Abstract Location* through
    - Precise location
    - Additional environmental sensors
  - Generalization of precise locations
    - Whenever precise location matching not possible

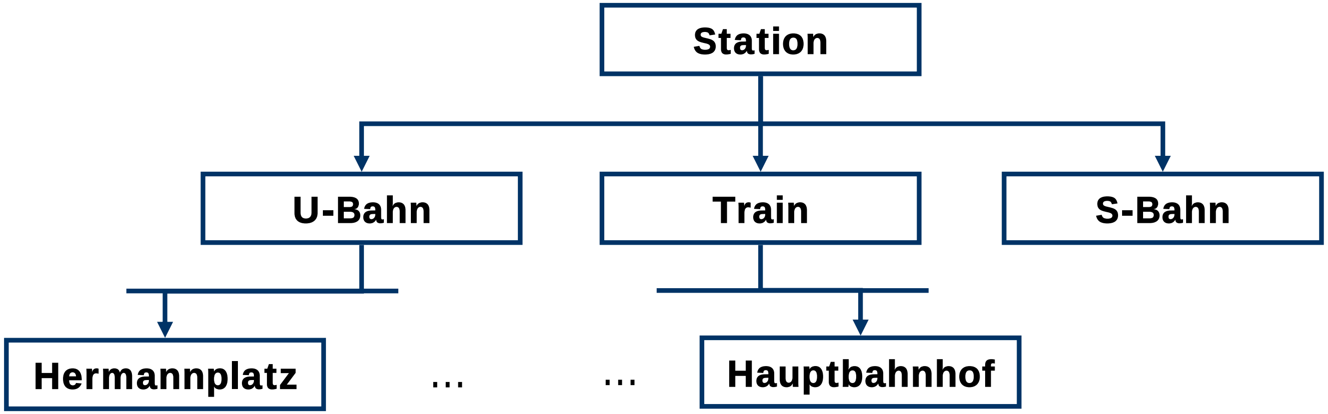
    - fact context [place = „Goya“ type = „night club“]



Rule-based EN on FACTS

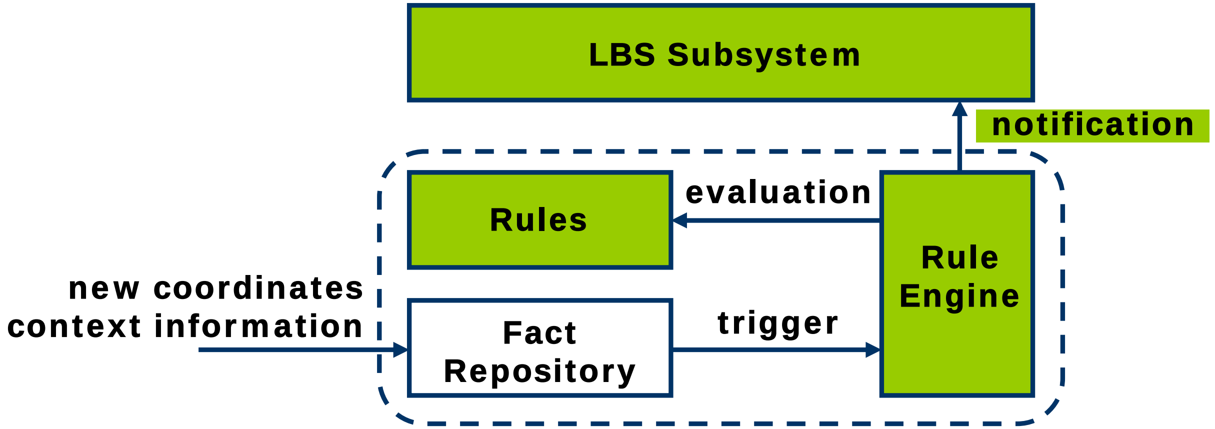
- Location Hierarchy
  - Define rule at super class
  - Sub-classes inherit rules of their abstract (super) classes
  - Prevention of multiple definition of one rule for several classes

  - Example services:
    - Station: ticket share
    - Hauptbahnhof: Display time table



Rule-based EN on FACTS

- Service Trigger
  - Services are triggered according to previously defined rules within the rule engine
  - Easy to add as rules to rule engine
  - System easy to expand for application programmer



Example: Dating Service (1)

```
fact location [
  long = "N52_29_91",
  lat = "E13_21_15",
  place = "Goya_Club"
]
```

```
fact context [
  place = "Goya_Club",
  type = "Night_Club"
]
```

```
fact coordinate [
  long = "N52_29_91",
  lat = "E13_21_15"
]
```

```
fact currentLocation [
  place = "Goya_Club"
]
```

```
fact currentContext [
  type = "Night_Club"
]
```

```
rule updateLocation 200
<- exists {coordinate}
-> retract {currentLocation}
-> define currentLocation [place = {location place
      <- eval ({this long} == newCoordinateLong)
      <- eval ({this lat} == newCoordinateLat)
}]
-> retract {coordinate}

rule updateContext 190
<- exists {currentLocation}
-> retract {currentContext}
-> define currentContext [type = {context type
      <- eval ({this place} == {currentLocation place})
}]
```

Example: Dating Service (2)

```
fact personalData [
  gender = "m",
  age = 26
]
```

```
fact datingProfile [
  gender = "f",
  minAge = 20,
  maxAge = 30
]
```

```
fact currentContext [
  type = "Night_Club"
]
```

```
fact datingServiceActive
```

```
fact periodicBroadcastTrigger
```

```
rule activateDatingService 100
<- eval ({currentContext type} == "Night_Club")
-> define datingServiceActive
-> call defineLater ({periodicBroadcastTrigger}, 60)

rule broadcastMyDatingProfile 90
<- exists {datingServiceActive}
<- exists {periodicBroadcastTrigger}
-> send systemBroadcast systemTxRange {datingProfile
      <- eval ({this owner} == systemID)
}
-> call defineLater ({periodicBroadcastTrigger}, 60)
```

Example: Dating Service (3)

```
fact personalData [
  gender = "f",
  age = 26
]
```

```
fact datingProfile [
  gender = "f",
  minAge = 20,
  maxAge = 30
]
```

```
fact datingProfile [
  gender = "m",
  minAge = 20,
  maxAge = 30
]
```

```
rule replyToDatingProfile 80
<- exists {datingServiceActive}
<- exists {datingProfile
      <- eval ({this owner} != systemID)
      <- eval ({this gender} == {personalData gender})
      <- eval ({this minAge} <= {personalData age})
      <- eval ({this maxAge} >= {personalData age})
}
-> send {datingProfile owner
      <- eval ({this owner} != systemID)
} systemTxRange {personalData}
-> retract {datingProfile
      <- eval ({this owner} != systemID)
}
```

Evaluation: Ruleset Metrics

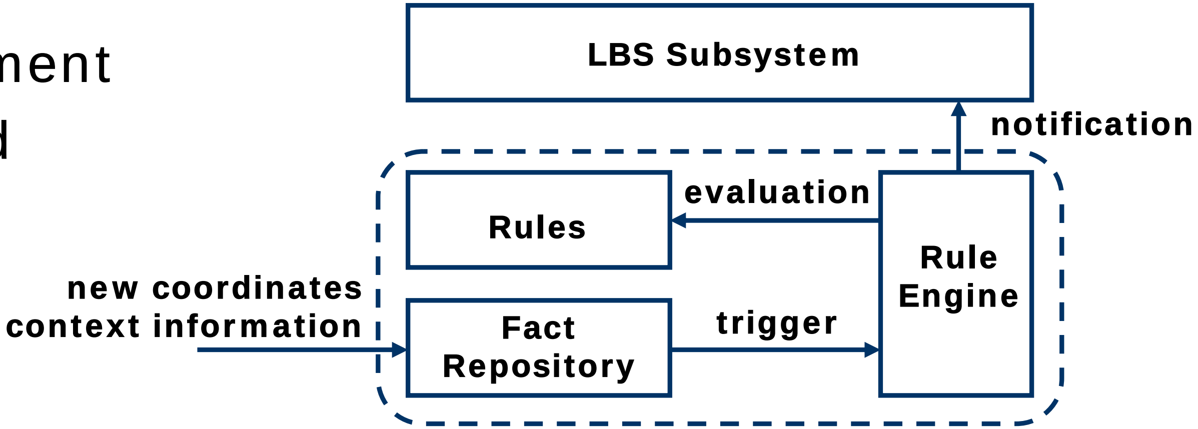
- Simple rule-based location inference algorithm in:

	unoptimized		optimized	
	#	**size (B)**	**#**	**size (B)**
Rules	5	60	5	60
Conditions	21	126	16	96
Statements	11	88	11	88
Slots	50	400	29	232
Expressions	40	240	24	144
Initializers	2	8	2	8
Variables	40	160	24	96
Facts	7	140	7	140
Properties	17	102	17	102
TOTAL		**1,324**		**966**

Conclusion

- Rule-based Event Notification on FACTS
- FACTS: rule-based middleware
  - Small size
  - Deployment on sensor nodes
- Interface of services and middleware
  - Through facts in fact repository
  - Rules in application-level ruleset

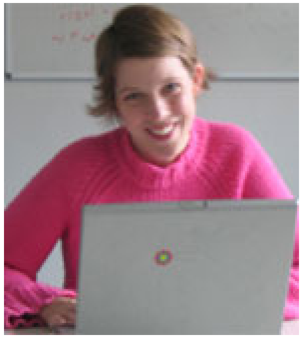
  - → easy to implement
  - → easy to extend



Future Work

- Porting of FACTS to end-user devices
- Automated configuration environment
  - Support for context fact definition
  - Support for rule specification

Thank you!



```
fact datingProfile [
  gender = "m",
  minAge = 25,
  maxAge = 35
]
```

```
fact datingProfile [
  gender = "m",
  minAge = 18,
  maxAge = 25
]
```

```
fact datingProfile [
  gender = "f",
  minAge = 20,
  maxAge = 30
]
```

References

[1] Nimbus

- Jörg Roth: A Decentralized Location Service Providing Semantic Locations. Habilitationsschrift, Fernuniversität Hagen. Januar 2005.
- http://wireless-earth.de/nimbus.html

[2] Nexus

- http://www.nexus.uni-stuttgart.de/

[3] Transit

- http://www.transit4events.org/index.html

[4] TIP

- A. Hinze, A. Voisard: Combining Event Notification Services and Location-based Services in Tourism. Technical Report TR-B-03-06, Freie Universität Berlin, 2003.
- http://isdb.cs.waikato.ac.nz/node/5