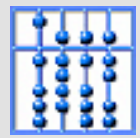
An Architecture for Distributed Spatial Configuration of Context Aware Applications

2nd International Conference on Mobile and Ubiquitous Multimedia

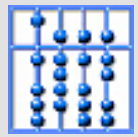
Martin Wagner and Gudrun Klinker
Augmented Reality Group
Institut für Informatik
Technische Universität München
martin.wagner@in.tum.de

December 11, 2003



Integrating Client Devices in UbiComp

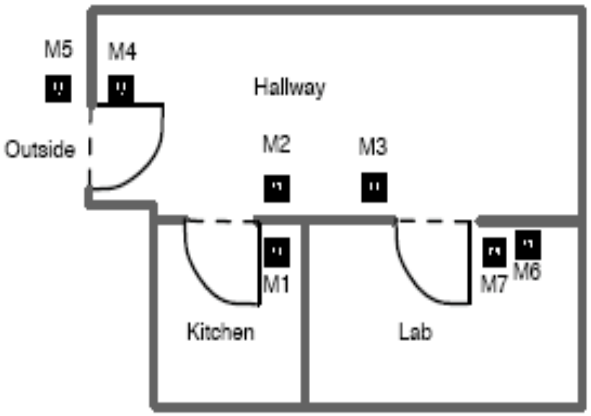
- In Ubiquitous Computing, devices in the environment interact with mobile client devices
- Applications adapt to these devices' capabilities and user's context
- Problem: context aware configuration of users' *mobile client* computers
  - Adapt configuration to current location, activity, time and identity of user
  - Use mobile client's capabilities within the UbiComp infrastructure
  - Allow integration of arbitrary mobile clients *without a priori knowledge*



Overview

- Example Scenario
- Approach: Distributed Configuration
- Software Basis: DWARF
- Automatic Context Aware Configuration
- Implementation Status
- Future Work

Example Scenario



- User walks around smart building, equipment:
  - Camera with optical tracker
  - 3D visual I/O system including HMD and Touchglove
- Mobile and stationary components collaborate in estimating user‘s context, mobile computers need to be configured dynamically
- Applications are composed of application logic in environment and user interface on mobile client
- Applications are chosen and configured based on current user context

Overview

- Example Scenario
- Approach: Distributed Configuration
- Software Basis: DWARF
- Automatic Context Aware Configuration
- Implementation Status
- Future Work

Approach: Distribute Configuration Data

- Configuration data:
  - Data needed for a generic hard- or software device to work correctly in an UbiComp environment
- Drawbacks of central configuration architecture:
  - Whole environment is single complex application
  - Unexpected side effects if configuration is adapted to new applications or users
  - Single point of failure
- Contextually distributed information storage
  - Simplifies partial reconfiguration
  - Allows users to store private configuration data on their mobile clients

Requirements for CA Architecture

- Context aware configuration data
  - Current configuration depends on n-tuple describing current context:
    {location, identity, activity, time, …}
- Transparent access to configuration data
  - Automatic partial or full reconfiguration of client and environment components
  - Transparency allows flexible organization of configuration databases
- Separate context estimation component
  - Facilitates processing of low-level sensor information of both environment and mobile client

Overview

- Example Scenario
- Approach: Distributed Configuration
- Software Basis: DWARF
- Automatic Context Aware Configuration
- Implementation Status
- Future Work

DWARF Overview

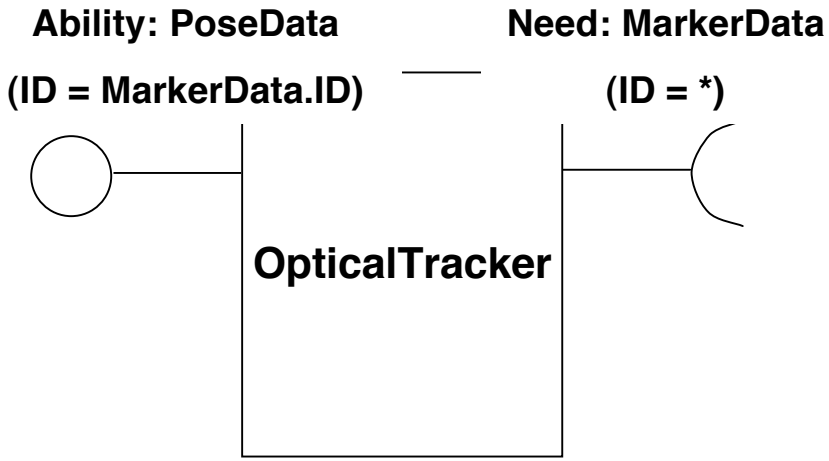
- Distributed Wearable Augmented Reality Framework
- CORBA-based middleware dynamically connects *Services* (DWARF components) based on description of their *Needs* and *Abilities*
- No central component, *Service Managers* running on each network node handle connection of services
- Ability descriptions may be enhanced using *Attributes* describing contextual information
- Need description may give *Predicates* for narrowing the search space of matching services
- Abilities may change at runtime depending on how needs are satisfied

DWARF overview (ctd.)

- Once a need and an ability match, DWARF sets up a *connector* that both services use to communicate
- Example: Optical tracker

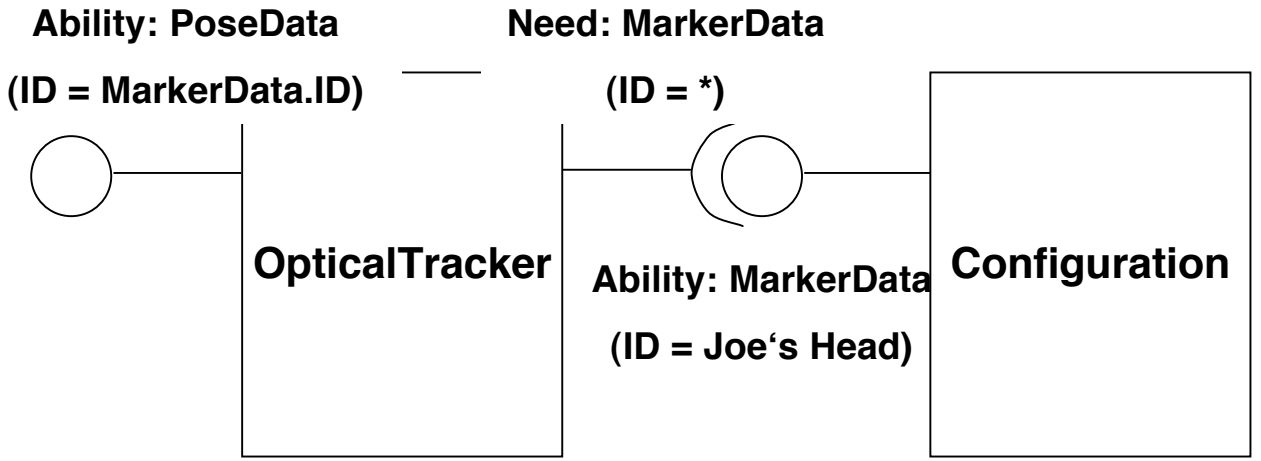
DWARF overview (ctd.)

- Once a need and an ability match, DWARF sets up a *connector* that both services use to communicate
- Example: Optical tracker



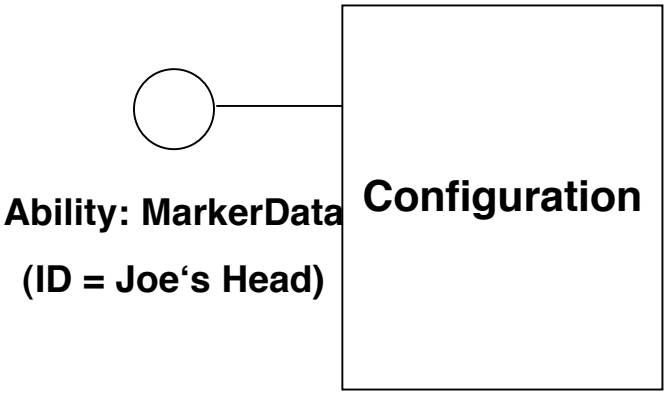
DWARF overview (ctd.)

- Once a need and an ability match, DWARF sets up a *connector* that both services use to communicate
- Example: Optical tracker



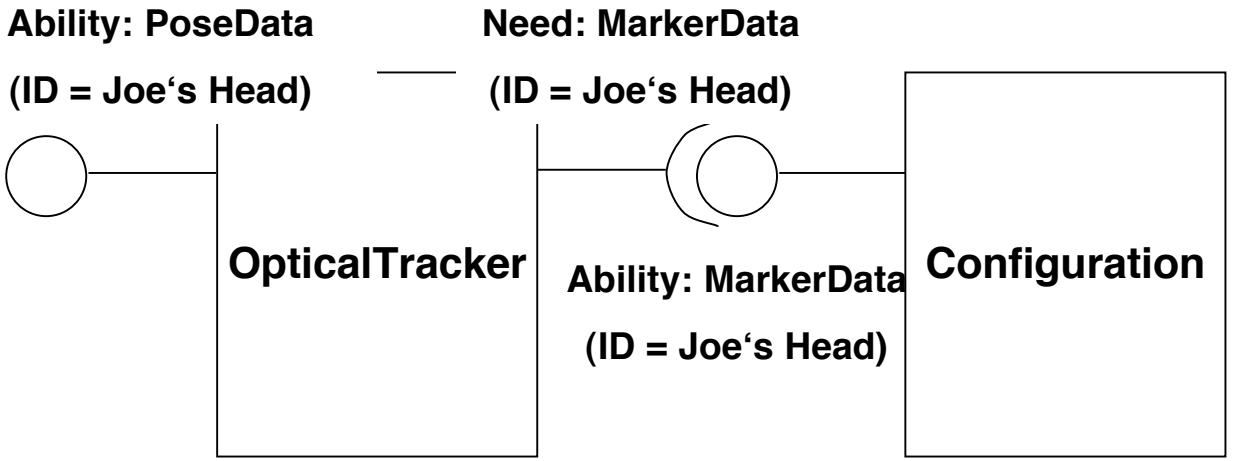
DWARF overview (ctd.)

- Once a need and an ability match, DWARF sets up a *connector* that both services use to communicate
- Example: Optical tracker



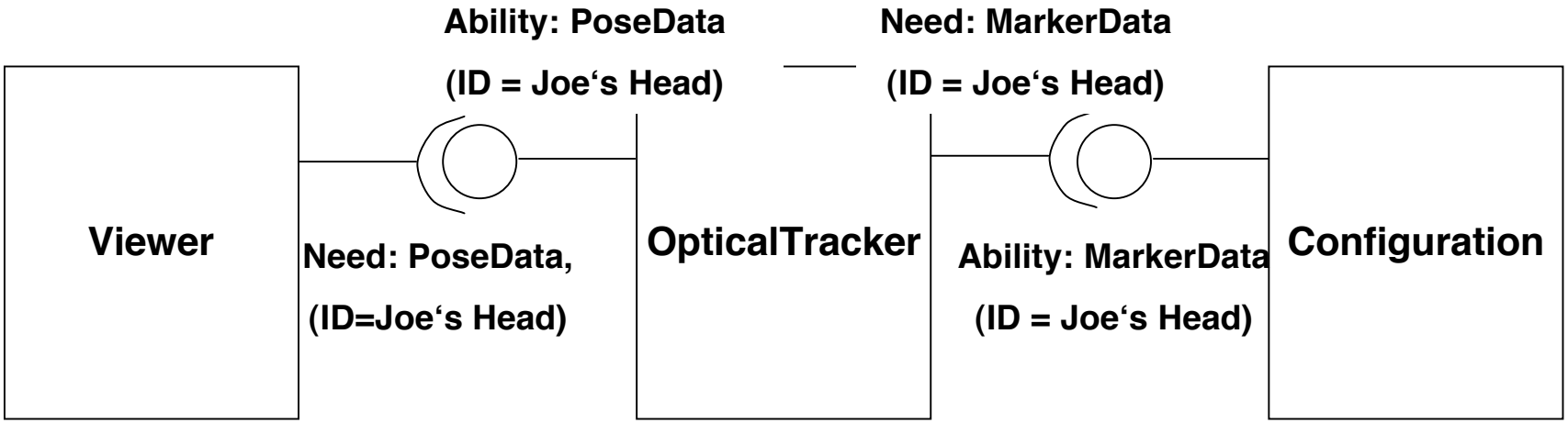
DWARF overview (ctd.)

- Once a need and an ability match, DWARF sets up a *connector* that both services use to communicate
- Example: Optical tracker



DWARF overview (ctd.)

- Once a need and an ability match, DWARF sets up a *connector* that both services use to communicate
- Example: Optical tracker



Overview

- Example Scenario
- Approach: Distributed Configuration
- Software Basis: DWARF
- Automatic Context Aware Configuration
- Implementation Status
- Future Work

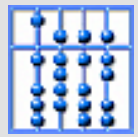
Configuration Architecture: Components

Sensors: Read low-level data influenced by the user's current state; may need to be configured; are both on user's mobile client and in the environment

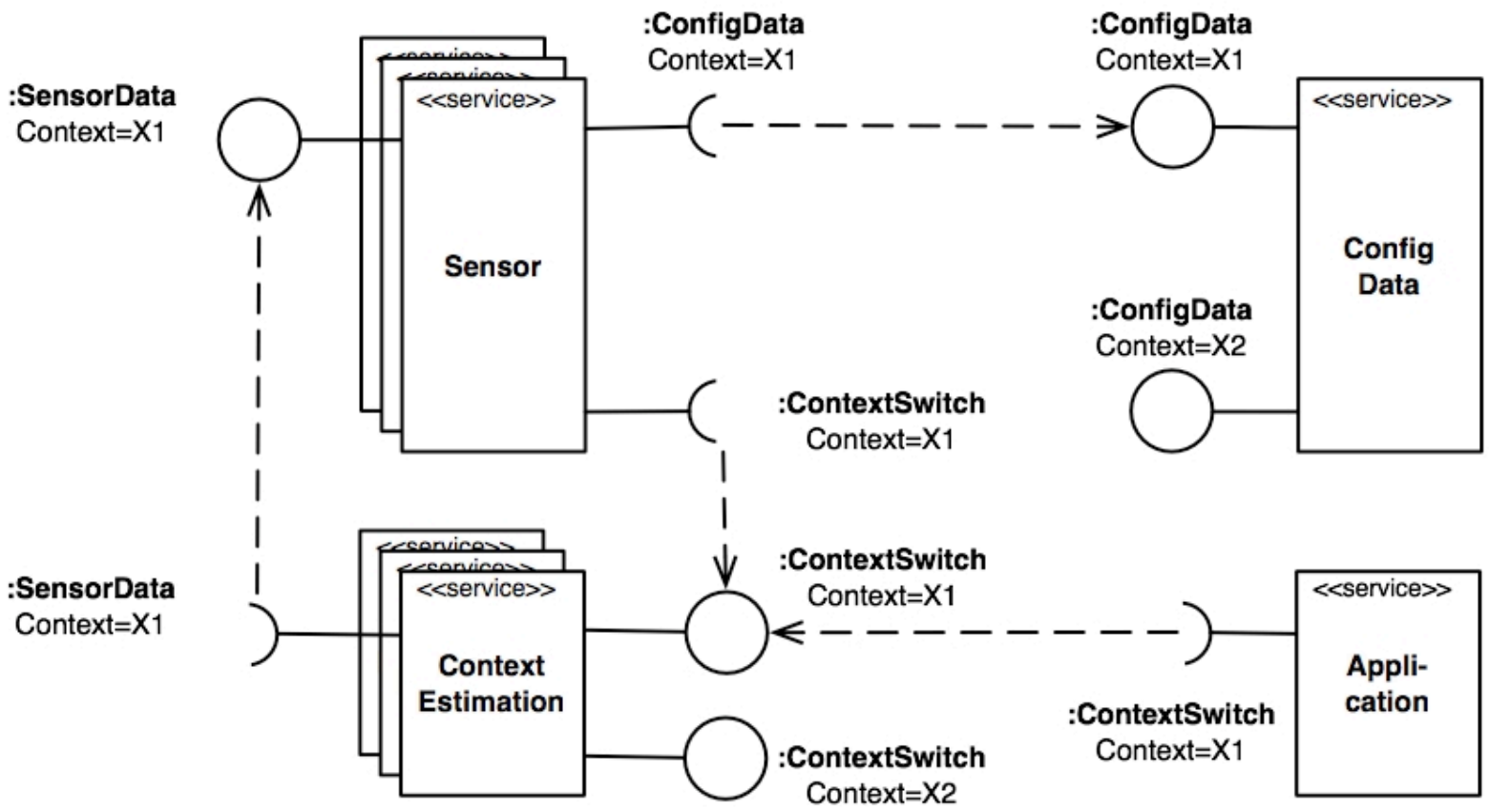
Context Estimation: Read sensor data and estimate high-level contextual information; may need to be configured; are both on user's mobile client and in the environment

Application: Performs certain task for the user; behavior influenced by current context

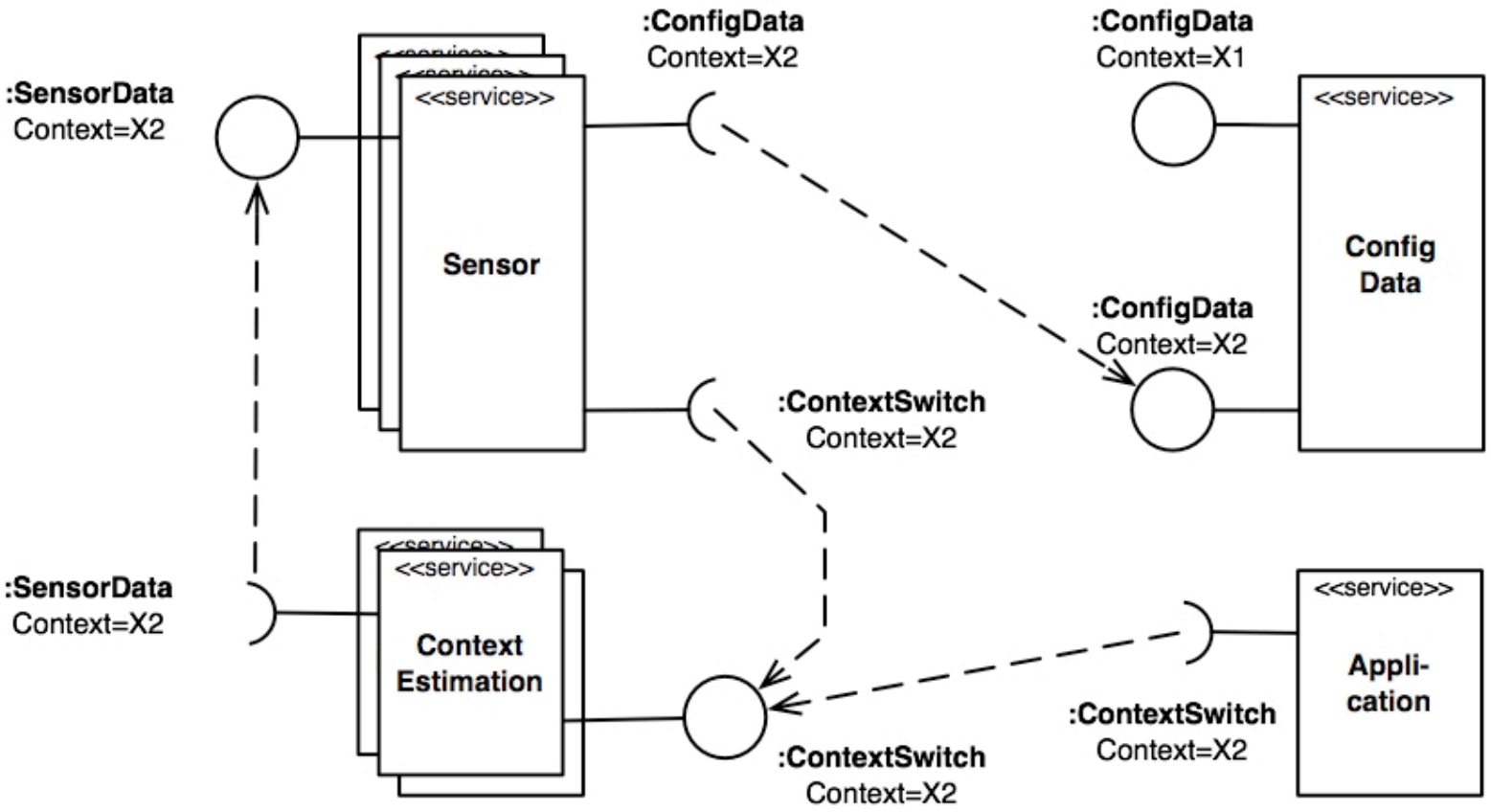
Config Data: Store configuration data for specific context tuples, reconfigure sensor and context estimation components accordingly



Configuration Architecture: Structure



Configuration Architecture: Structure



Configuration Architecture: Example

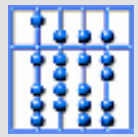
Video Grabber: gets video image and puts it in a shared memory segment

Optical Tracker: Detects fiducial markers in video image and reconstructs camera‘s (i.e. user‘s) position and orientation *(pose)*

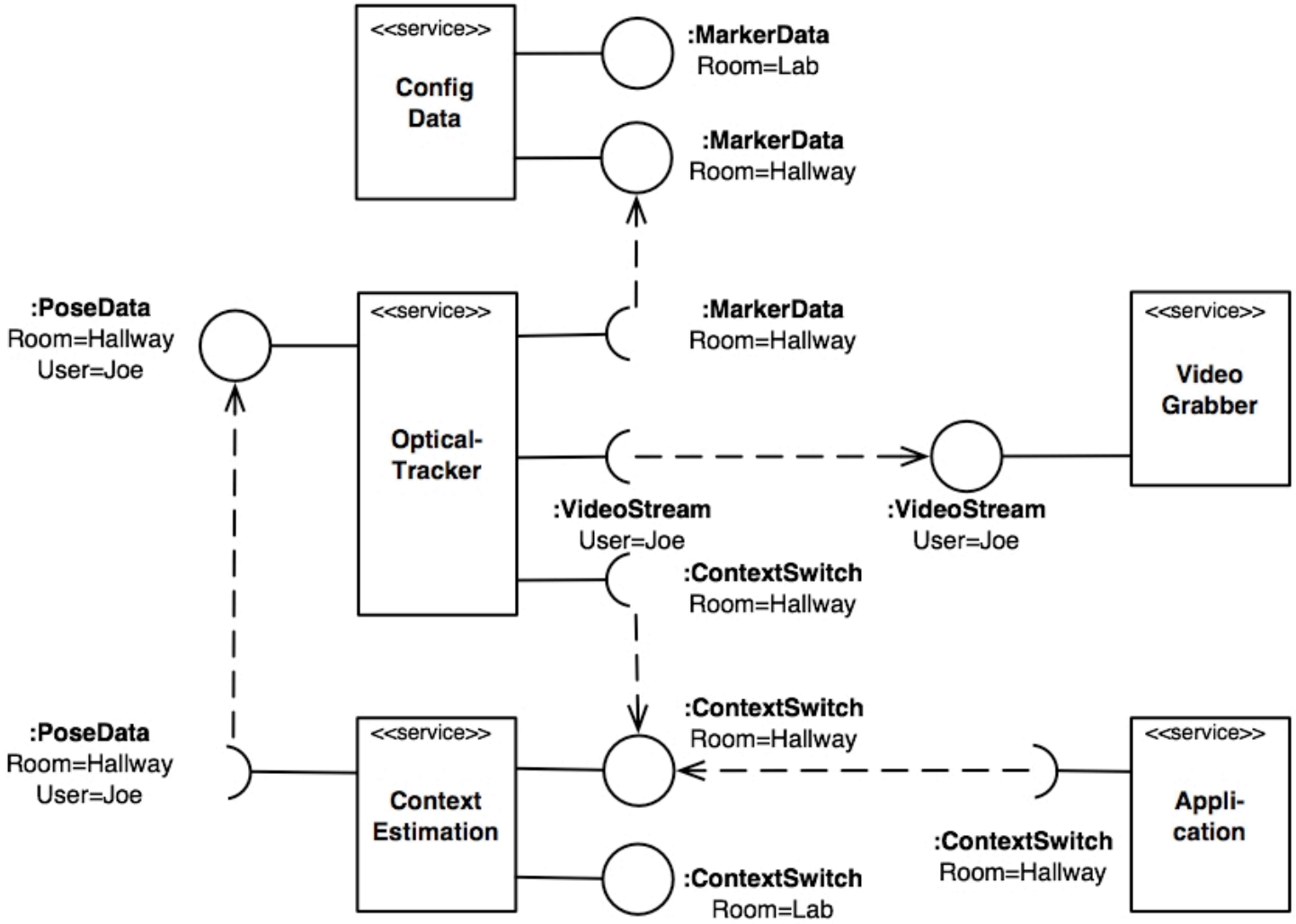
Application: Takes user‘s pose and superimposes augmentations over the user‘s view

Context Estimation: Reads camera‘s pose and estimates the room the user is currently in

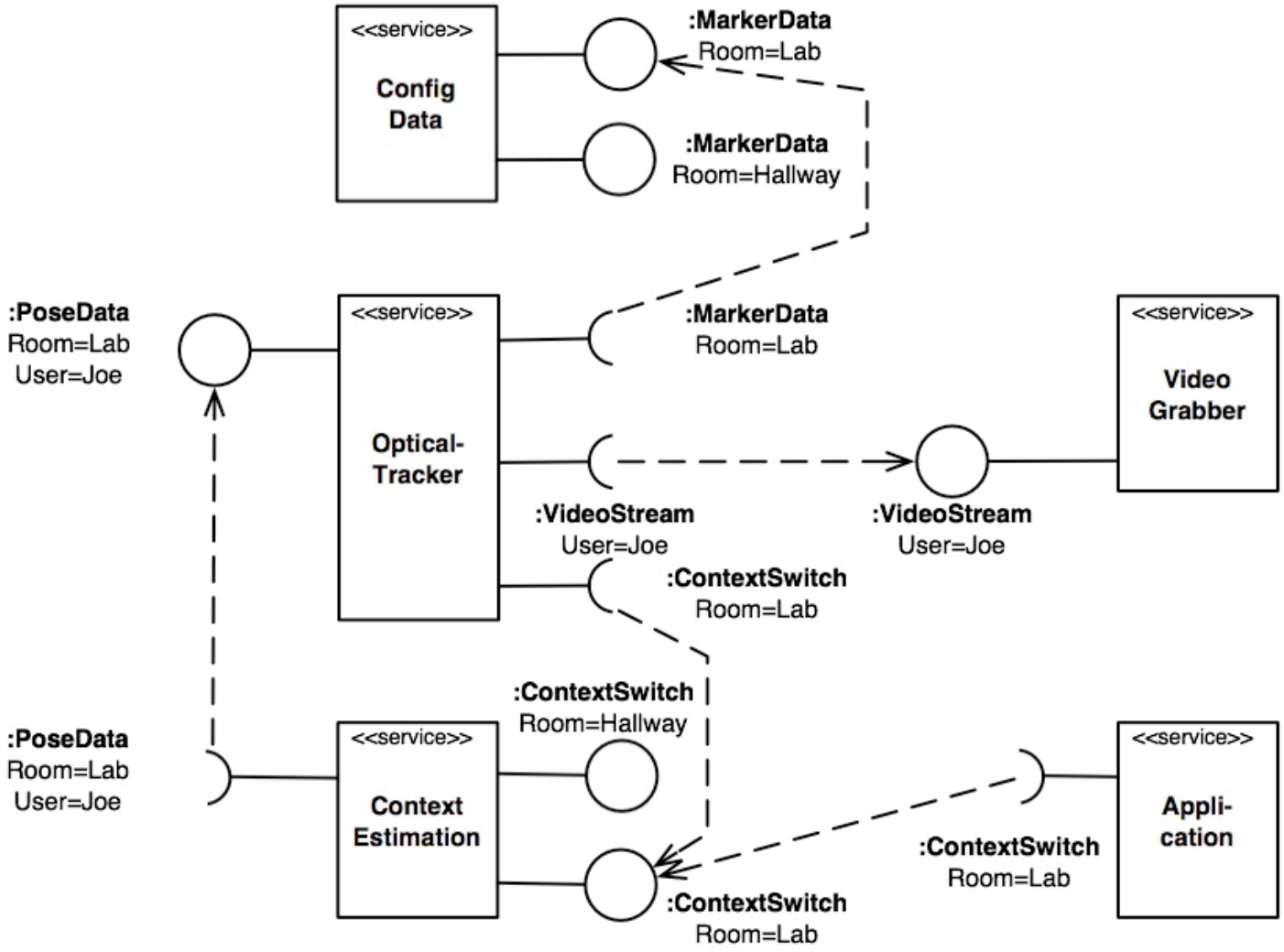
Config Data: Stores data organized along the different rooms, reconfigures other components accordingly



Configuration Arch.: Example Structure



Configuration Arch.: Example Structure



Configuration Example: Flow of Events

1. Building has several *Context Estimation*, *Config Data* and *Application* services running
2. User enters hallway, starts *Video Grabber*, *Application* and *Optical Tracker*
3. *Config Data* service in hallway gets connected to *Optical Tracker* and configures it
4. *Context Estimator* in hallway connects to *Optical Tracker*
5. User leaves hallway and enters lab
6. *Context Estimator* detects changed location context by reading events from *Optical Tracker*
7. *Context Estimator* changes mobile client‘s context attributes
8. *Config Data* service gets exchanged, thereby reconfiguring the *Optical Tracker*
9. *Context Estimator* and *Application* get exchanged as well, allowing user to interact with the lab‘s infrastructure

Overview

- Example Scenario
- Approach: Distributed Configuration
- Software Basis: DWARF
- Automatic Context Aware Configuration
- Implementation Status
- Future Work

Implementation Status

- Demonstration setup with location as only contextual information
- Location structured into 4 different rooms
- Optical tracker uses AR Toolkit
- Application uses speech to tell user information about the current room
- Configuration Data is kept in a MySQL database, a single DWARF ability is offered for every contextual state

Overview

- Example Scenario
- Approach: Distributed Configuration
- Software Basis: DWARF
- Automatic Context Aware Configuration
- Implementation Status
- Future Work

Future Work

- Build larger demonstration setup
  - Implement bootstrapping of contextual state
  - Evaluate scalability of approach
- Refine concept of contextual entities
  - Up to now, we use physical rooms
  - Structure context such that design of new applications becomes more intuitive
  - Learn context boundaries automatically
- Incorporate advanced configuration data access mechanisms
  - Distributed databases
  - Caching and prefetching

Thank You! Any Questions?

More Information:

Web page: http://www.augmentedreality.de

E-Mail: martin.wagner@in.tum.de

A DWARF project – Distributed Wearable Augmented Reality Framework