Using Ad-Hoc Services for Mobile Augmented Reality Systems

Asa MacWilliams

Summary

- Concepts and technologies for self-assembling mobile AR systems
- Design of the Middleware for DWARF
- First implementation of the Middleware, validating the design

Outline

- *Self-Assembling AR Systems*
- *Requirements*
- *System Design*
- *Results*
- *Future Work*

Self-Assembling AR Systems

- Context: a user is roaming through an intelligent environment with a mobile AR system.
- Goal: his mobile system should automatically take advantage of devices in the environment, such as external trackers.

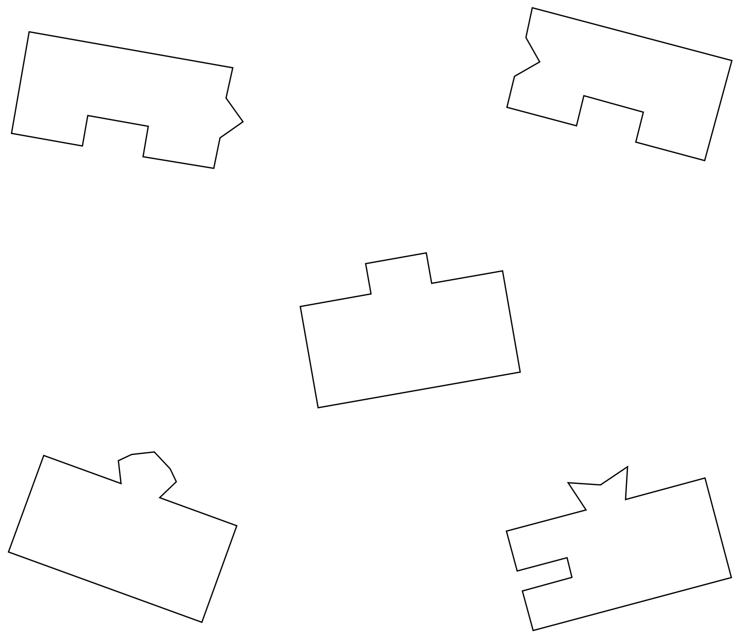
Self-Assembling AR Systems

- Context: a user is roaming through an intelligent environment with a mobile AR system.
- Goal: his mobile system should automatically take advantage of devices in the environment, such as external trackers.
- Idea:
  - **Divide the system into different *Services* for tracking, display, etc.**
  - **Deploy the Services on different mobile and stationary computers**
  - **As they come within range of one another, the Services assemble into a complete AR system**
- This requires intelligent *Middleware*.

Requirements (1)

Functional Requirements

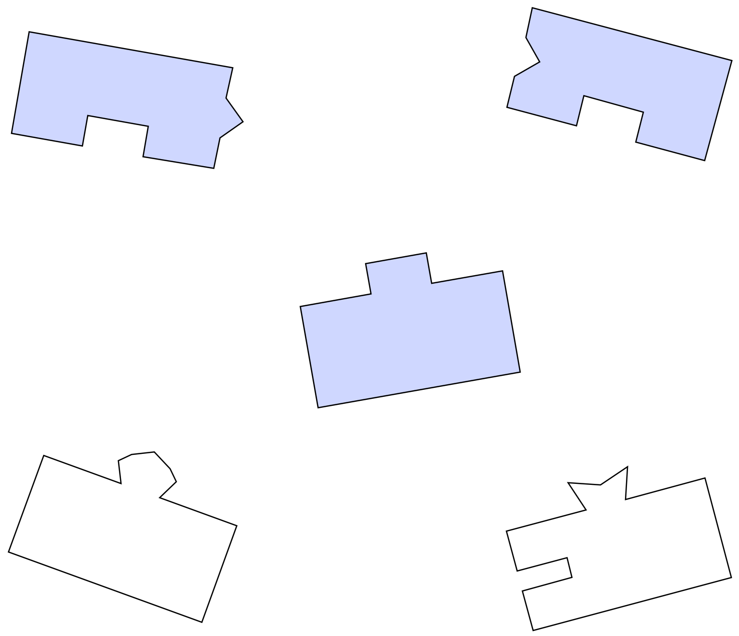
- DWARF consists of self-assembling Services.



Requirements (1)

Functional Requirements

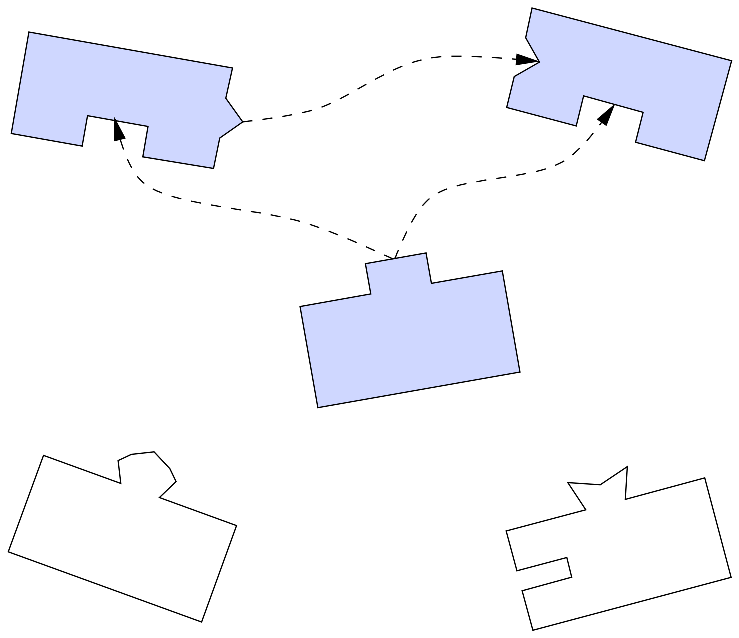
- DWARF consists of self-assembling Services.
- These Services must be able to *find* each other...



Requirements (1)

Functional Requirements

- DWARF consists of self-assembling Services.
- These Services must be able to *find* each other...
- ...and *communicate* with one another.



Requirements (2)

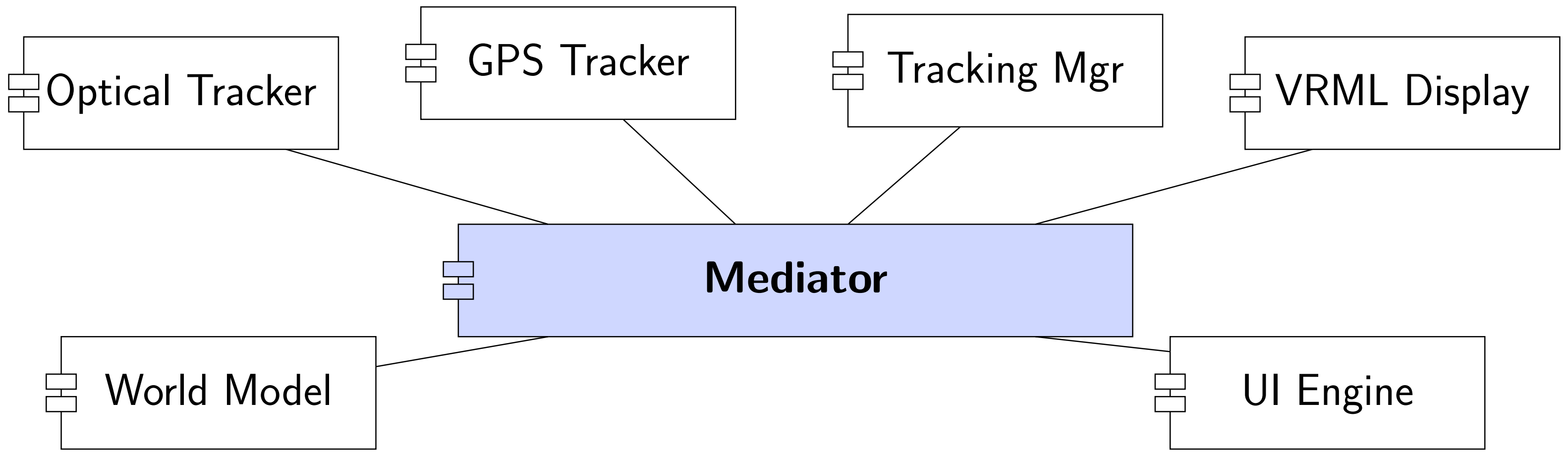
Nonfunctional Requirements

- For convincing AR, we need fast communication with *low latency*.
- To use ad-hoc Services as they are found, however, we need to choose the communication partners *flexibly*.
- This is a conflict in design goals.
- The design of the middleware must balance flexibiliy against speed.

Requirements (3)

Mediator Role of the Middleware

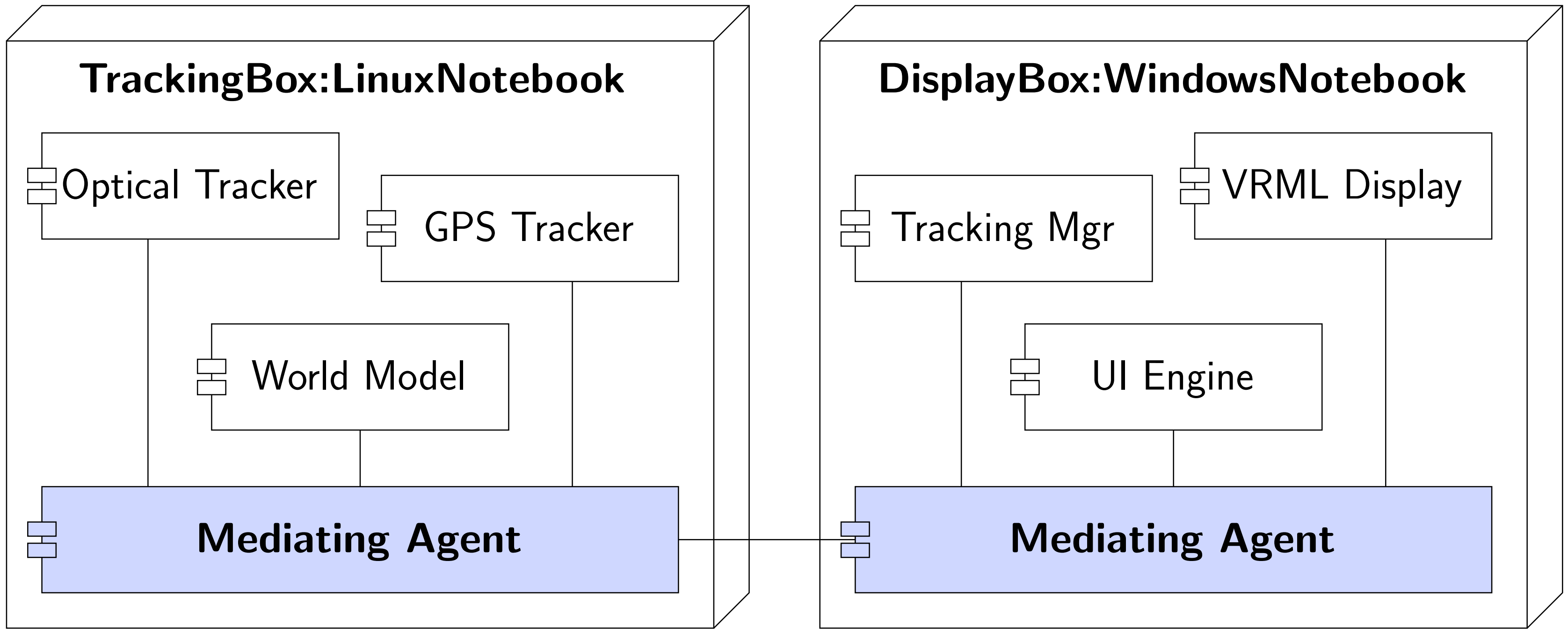
- The DWARF Services are designed as independently of one another as possible, so they can be combined into different applications.
- For them to cooperate, we used the *Mediator* pattern.



System Design (1)

Distributed Mediating Agents

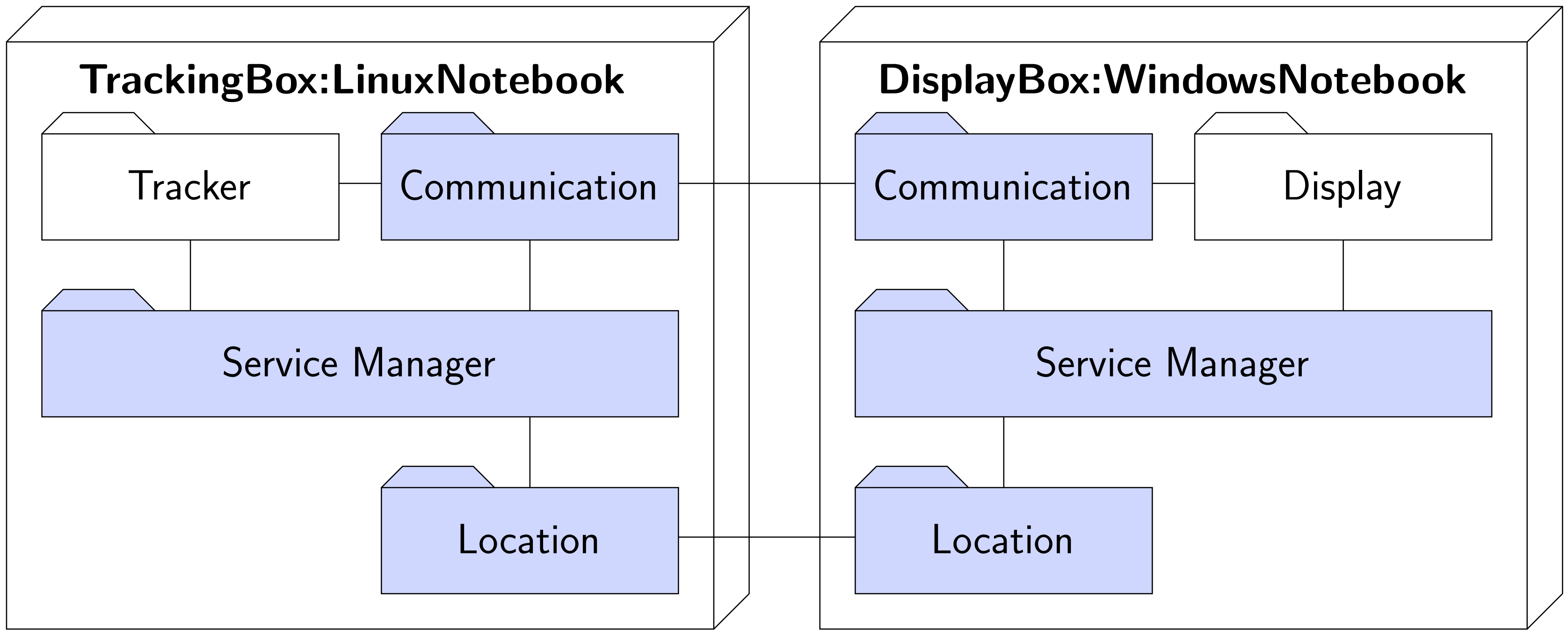
- If the middleware becomes a *central component*, it reduces fault tolerance and flexibility.
- Instead, I used *Distributed Mediating Agents*.



System Design (2)

Subsystem Decomposition

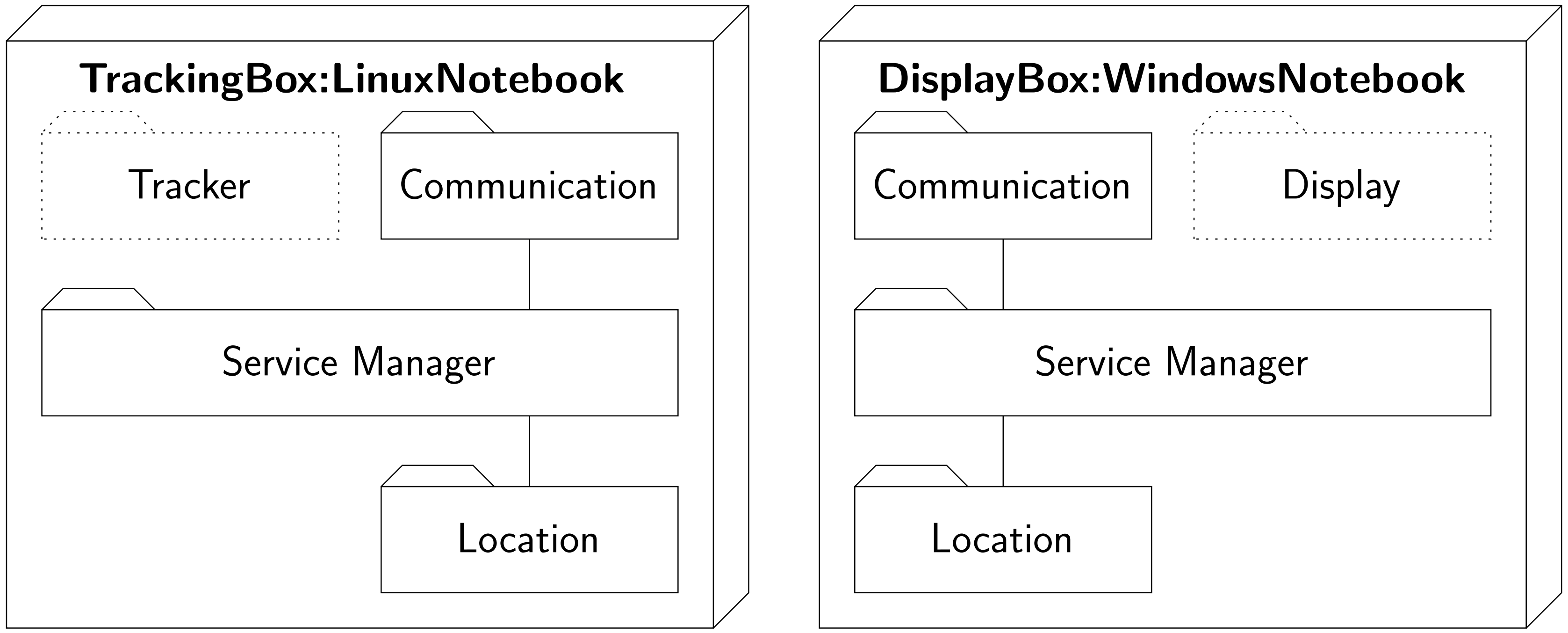
- Communication is fast
- Location is flexible
- Service Manager provides high-level interface



System Design (3)

Subsystem Interaction

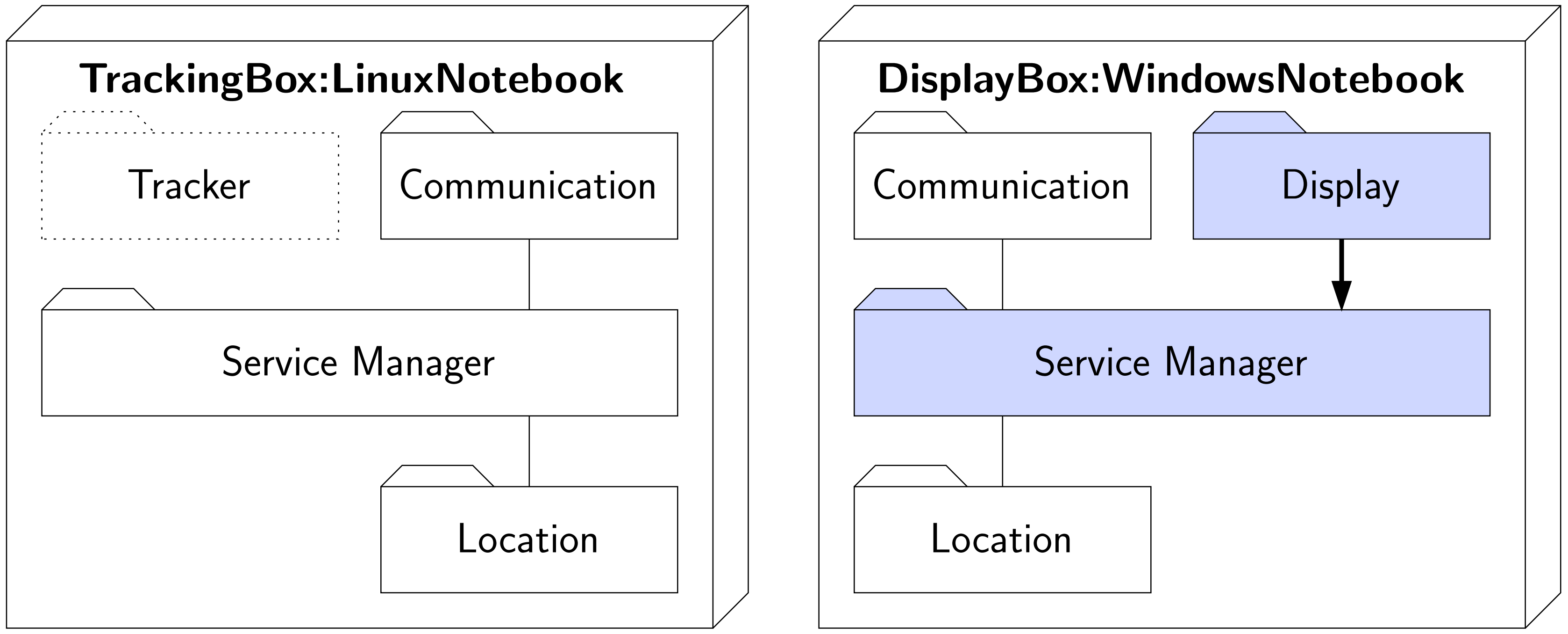
- Everything off



System Design (3)

Subsystem Interaction

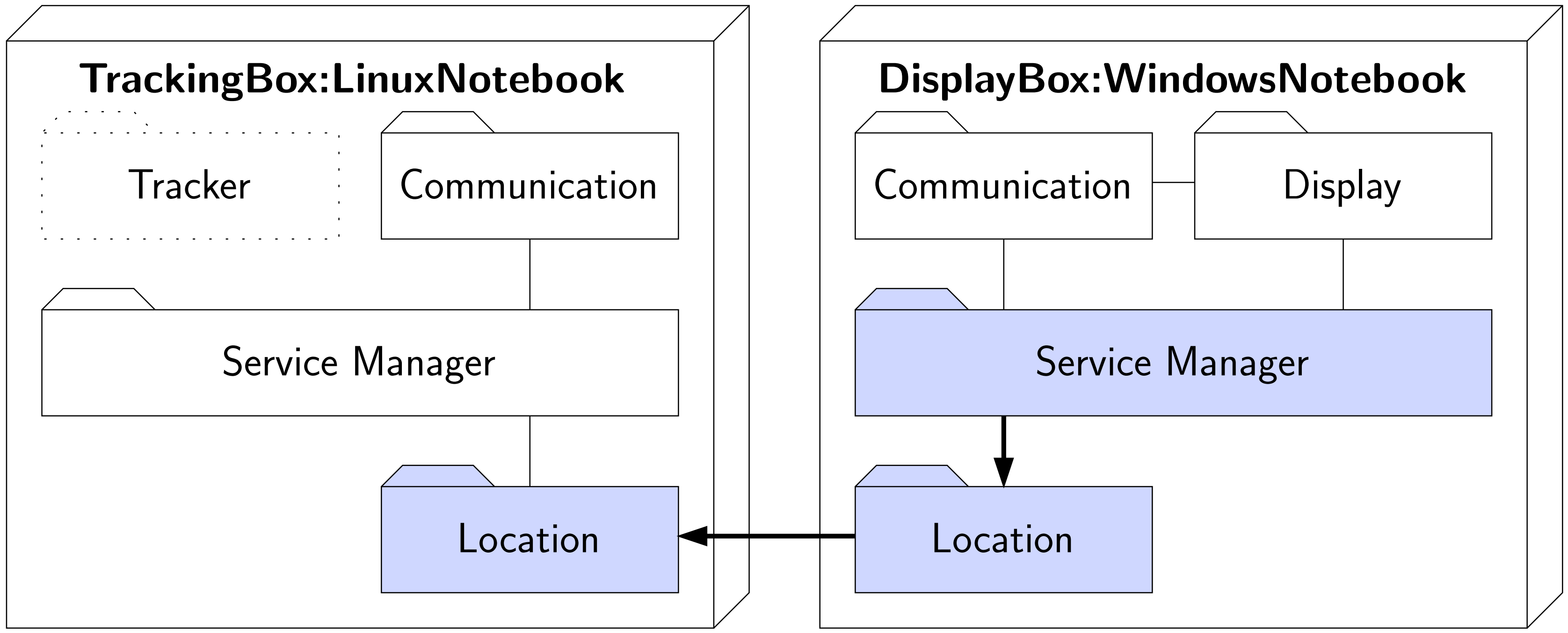
- Everything off
- Display starts
-
-
-
-



System Design (3)

Subsystem Interaction

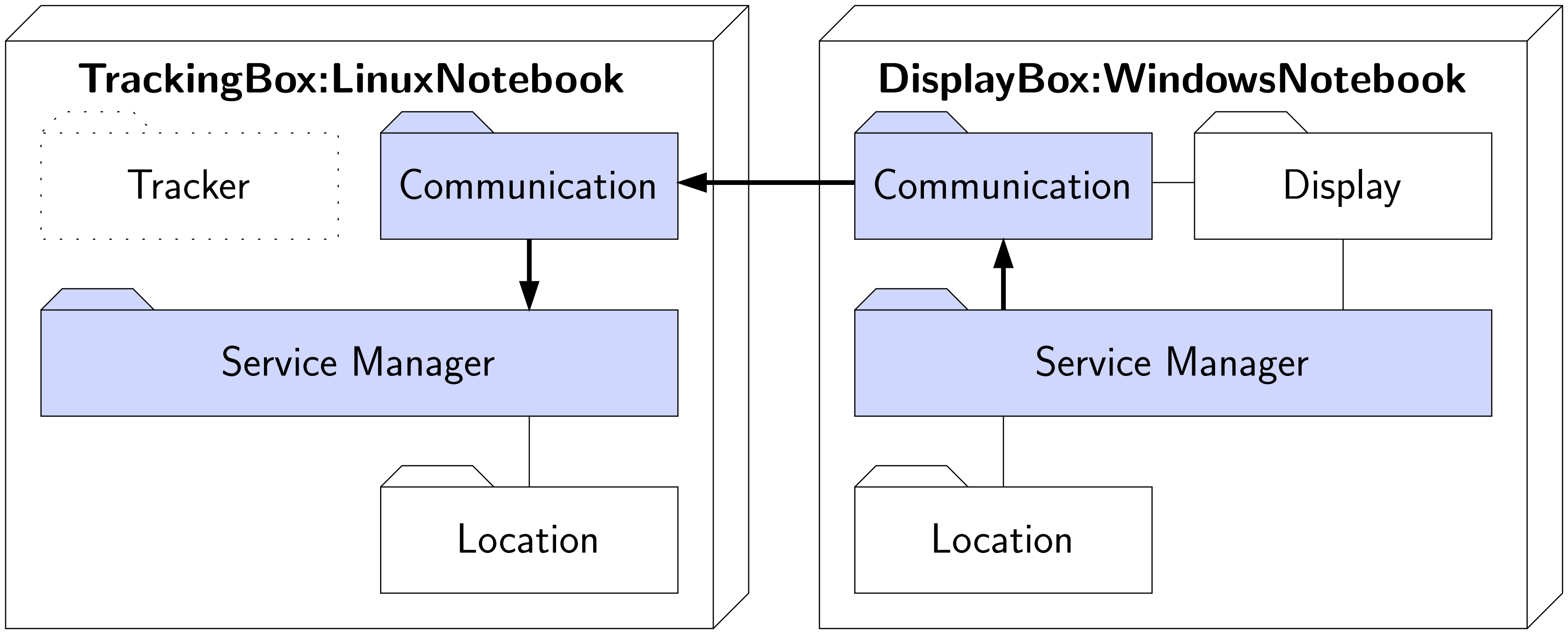
- ☐ Everything off
- ☐ Display starts
- ☐ Location
- ☐
- ☐
- ☐



System Design (3)

Subsystem Interaction

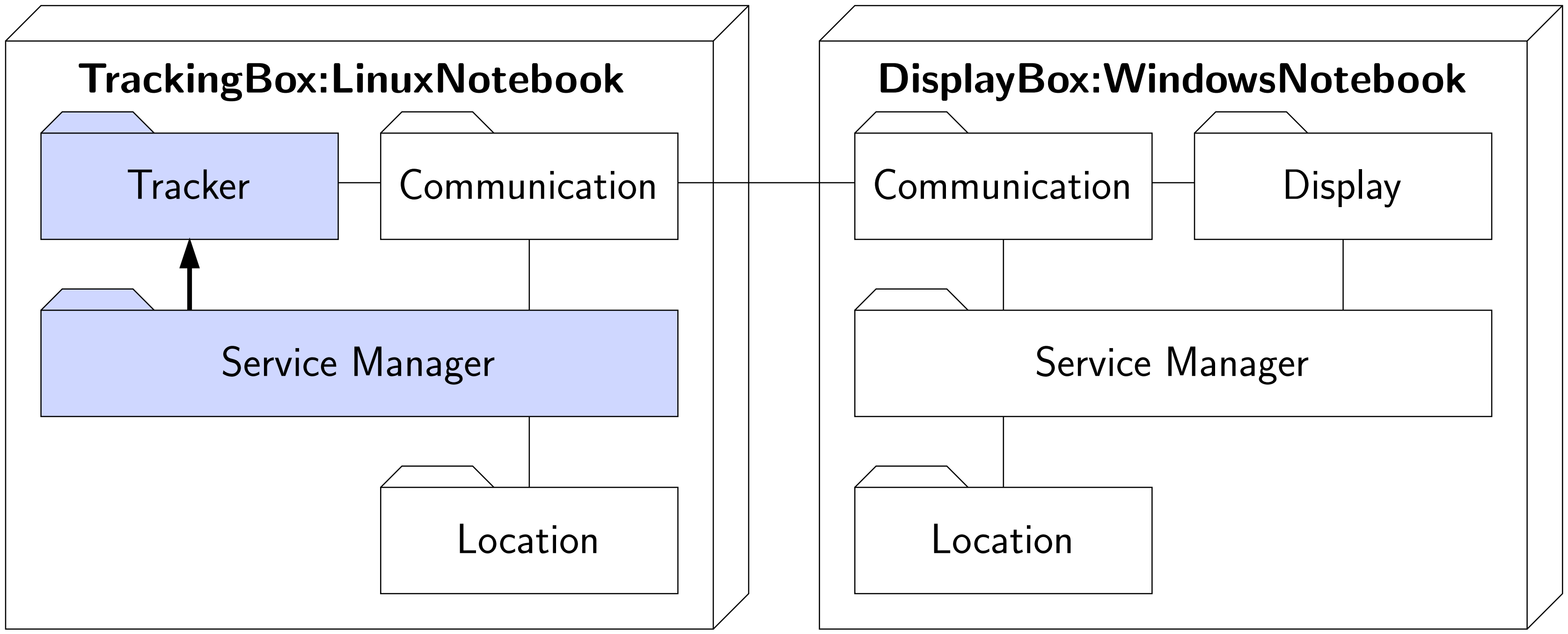
- Everything off
- Display starts
- Location
- Establish communication
-
-



System Design (3)

Subsystem Interaction

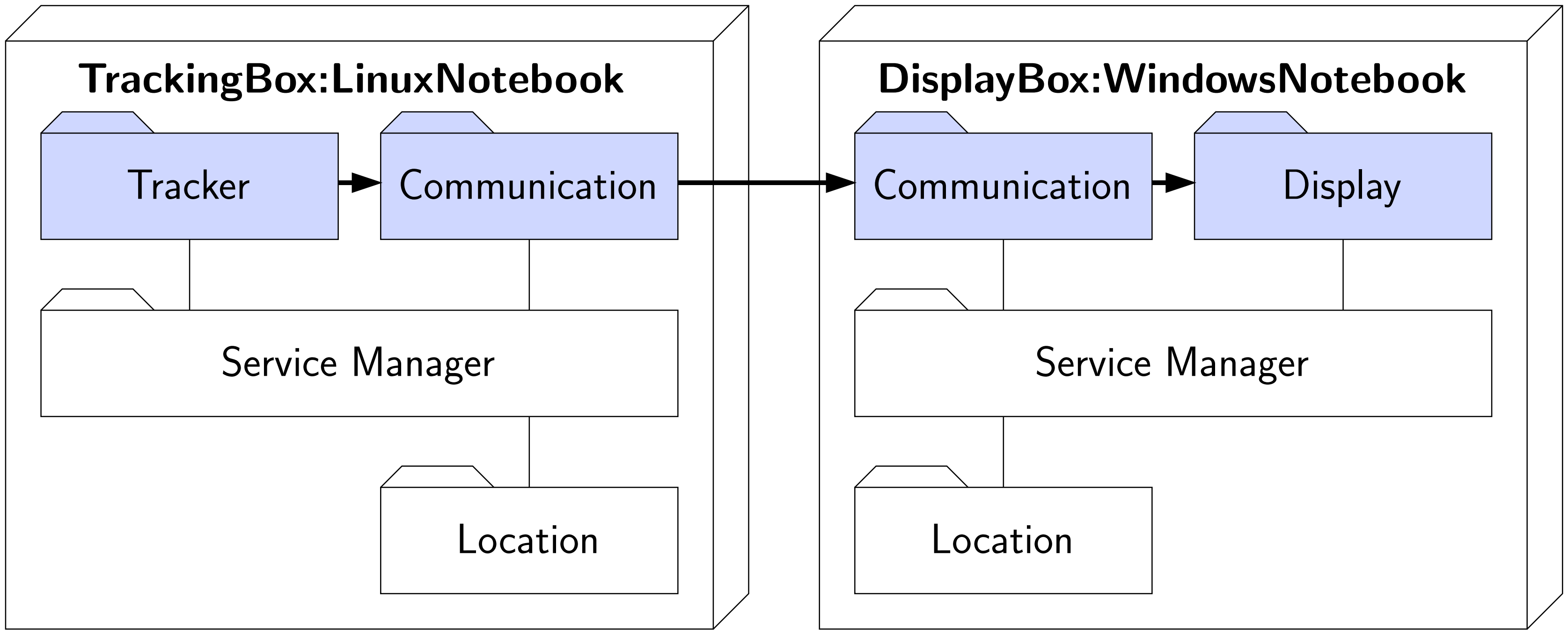
- ☐ Everything off
- ☐ Display starts
- ☐ Location
- ☐ Establish communication
- ☐ Activate Tracker
- ☐



System Design (3)

Subsystem Interaction

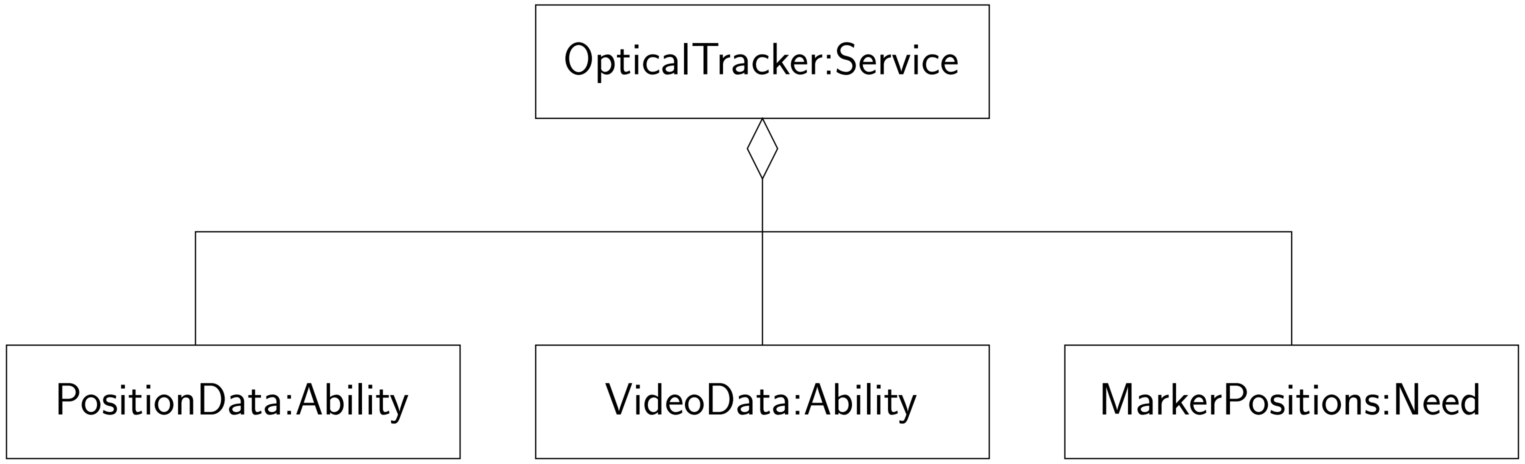
- Everything off
- Display starts
- Location
- Establish communication
- Activate Tracker
- Communicate



System Design (4)

Service Manager: Service Descriptions

- Each Service has a description that can be written in XML
- These describe Services' *Needs* and *Abilities*
- They also specify communication protocols



System Design (5)

Location Subsystem

- Various mechanisms are available to locate ad-hoc services: SLP, Jini, UPnP, SDP, etc.
- None address AR, many are for home networking
- The Location Subsystem defines a strategy pattern to use these different protocols
- The current version is designed to use the Service Location Protocol (SLP), a simple and open standard

System Design (6)

Communication Subsystem

- The DWARF components have different communication needs
- The Communication Subsystem can encapsulate many different protocols
- It currently supports:
  - **CORBA remote method calls**
  - **Event-based communication with the CORBA Notification Service**

System Design (7)

Summary—Design Challenges

- The Middleware needs to be fast, yet flexible
  - **Decomposition into Communication and Location sub-systems**
- The Middleware should not have to be in the middle
  - **Distributed Mediating Agents**
- Services that do not know each other have to cooperate
  - **Service Descriptions with Needs and Abilities**

Results

- Complete Design for a flexible, yet fast middleware system
- Supports roaming users in intelligent environments
- Systems built with DWARF can *spontaneously self-assemble*
- First implementation was successfully demonstrated on Linux and Windows, Intel and PowerPC

Future Work

- Further implementation:
  - Full SLP support
  - Full XML support
  - Optimizing of communication resources
  - Graceful handling of network errors
- Visualization tools
- Testing with different types of Services in different application domains

Thank You

Questions?