DWARF Overview

Distributed Wearable Augmented Reality Framework

Asa MacWilliams

November 5, 2001

Goals of this Presentation

- Become familiar with the purpose of DWARF
- Understand the basic design using distributed components
- Recognize which framework components will be relevant to your team

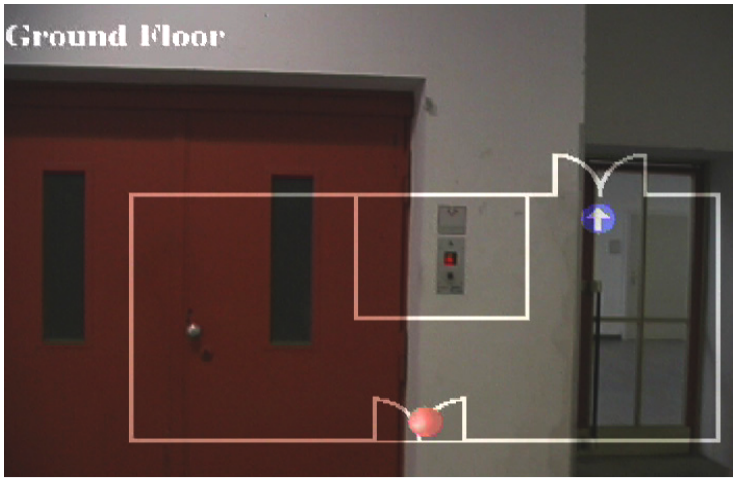
Outline

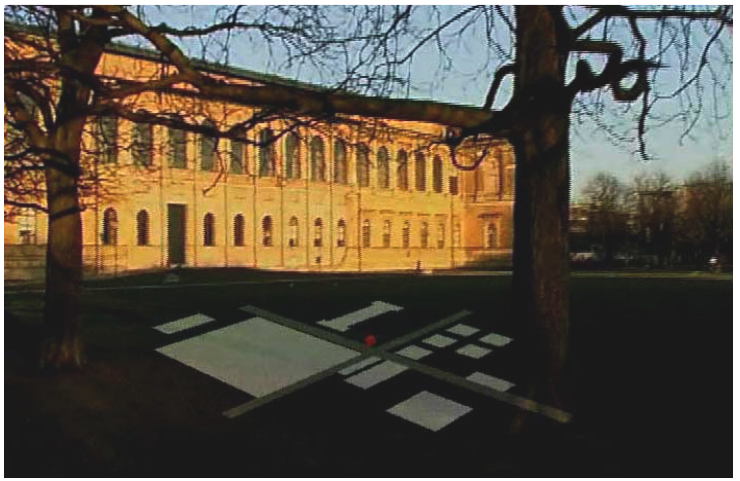
- A brief history of DWARF
- Basic design: distributed services
- Existing framework components
- Middleware

A brief history of DWARF

- The idea developed during a summer school on Augmented Reality in September 1999
- Goal: develop a platform for future Augmented Reality research at the TU München
- The DWARF Project started in spring 2000 and has resulted in six Diplomarbeiten
- A first demonstration system was implemented and shown successfully in December 2000
- Presented in a paper at ISAR 2001 in New York

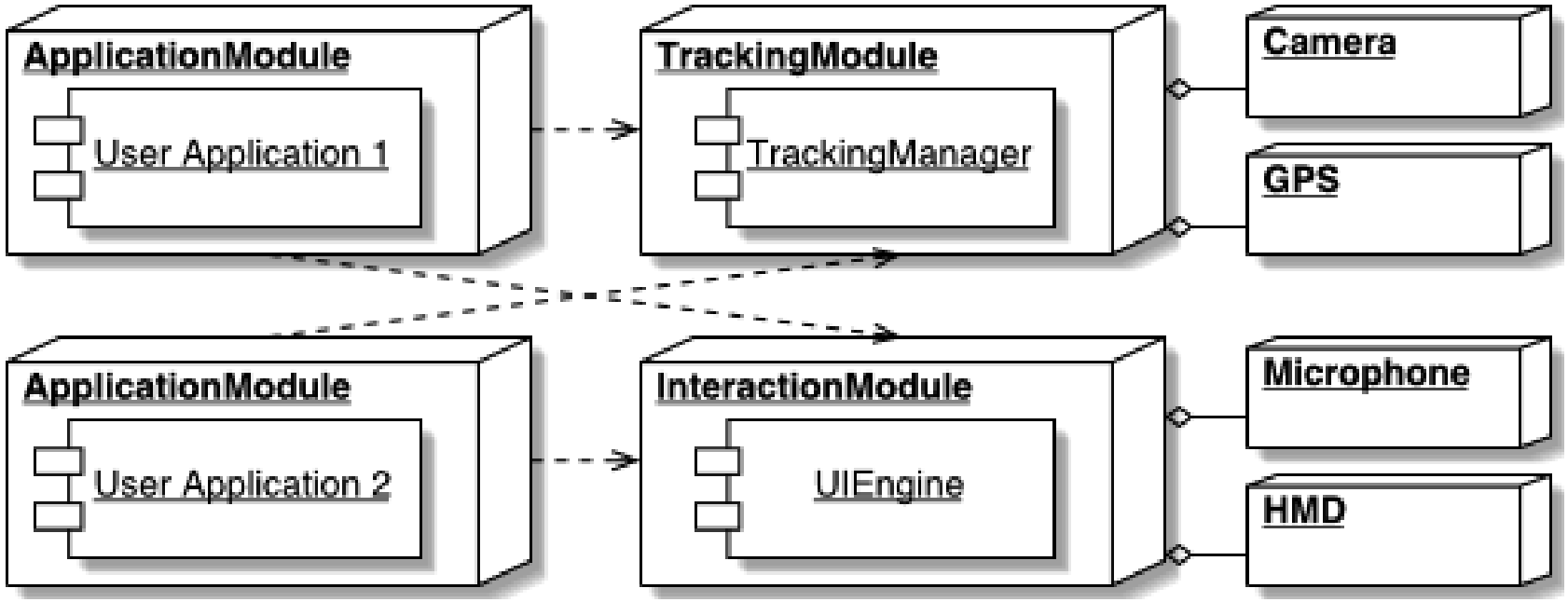
Demonstration system - campus navigation





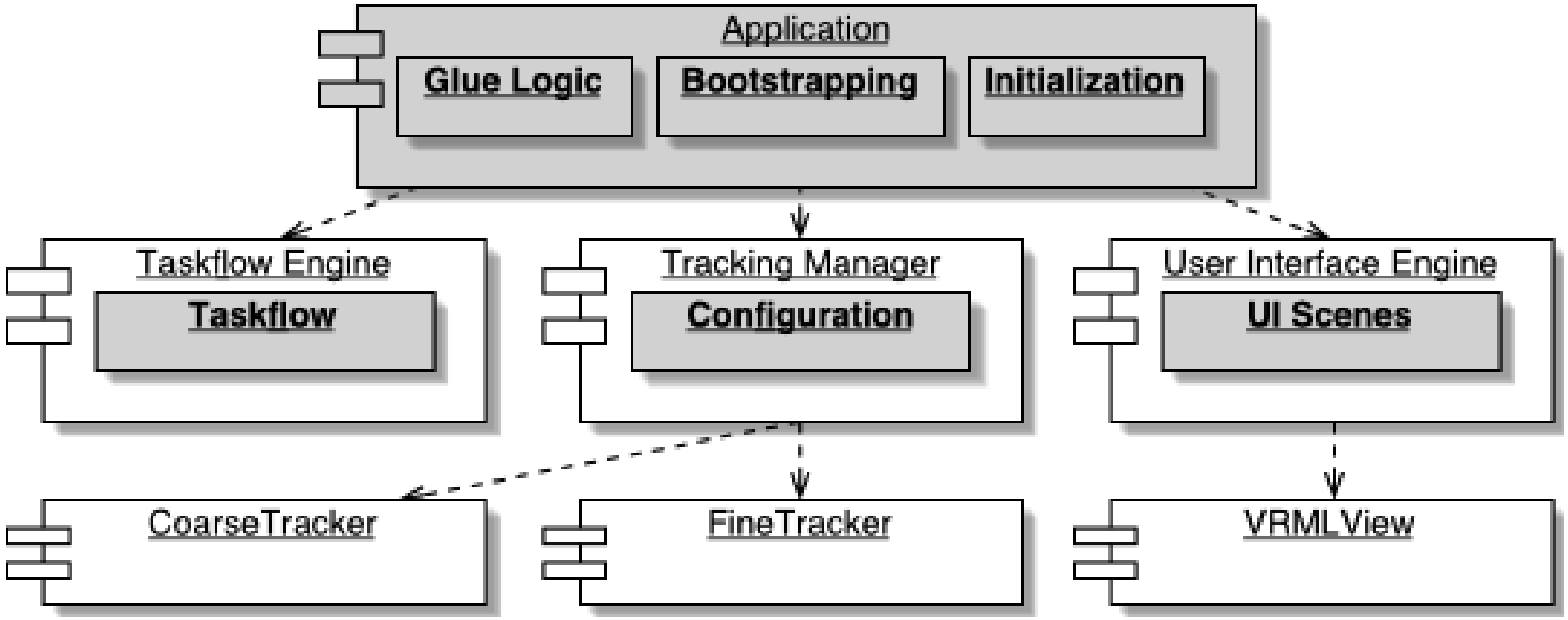
Distributed services: user's view

- Software is integrated with hardware in wearable or stationary *modules* with their own CPU
- System is reconfigured by re-plugging modules



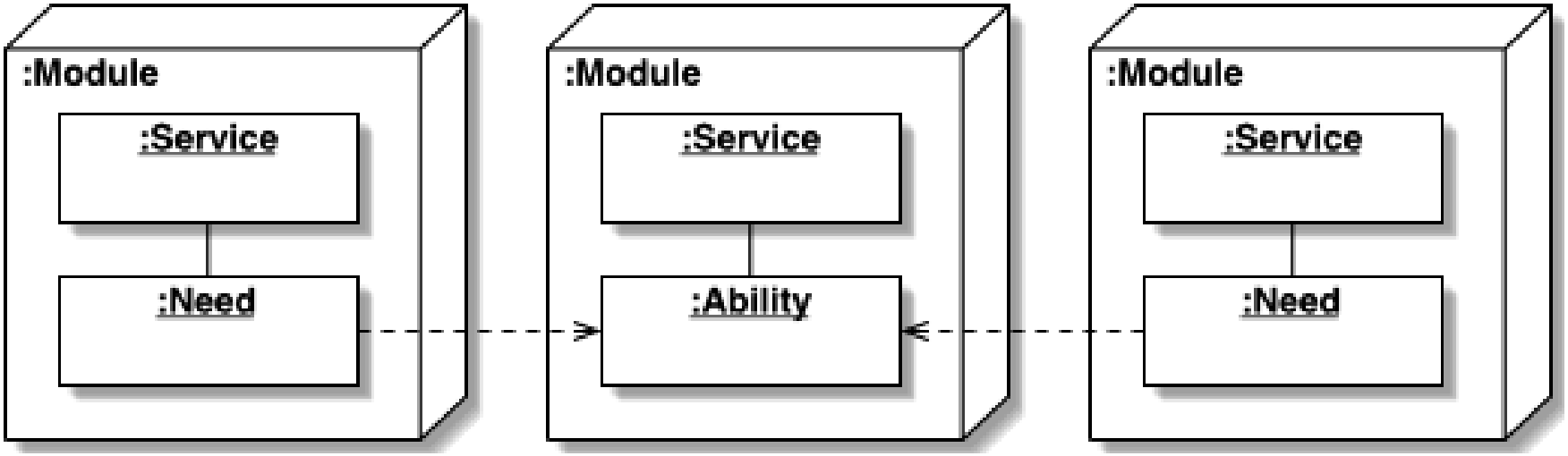
Application developer’s view

- System consists of layered *services*
- *Application* is modeled by configuring services

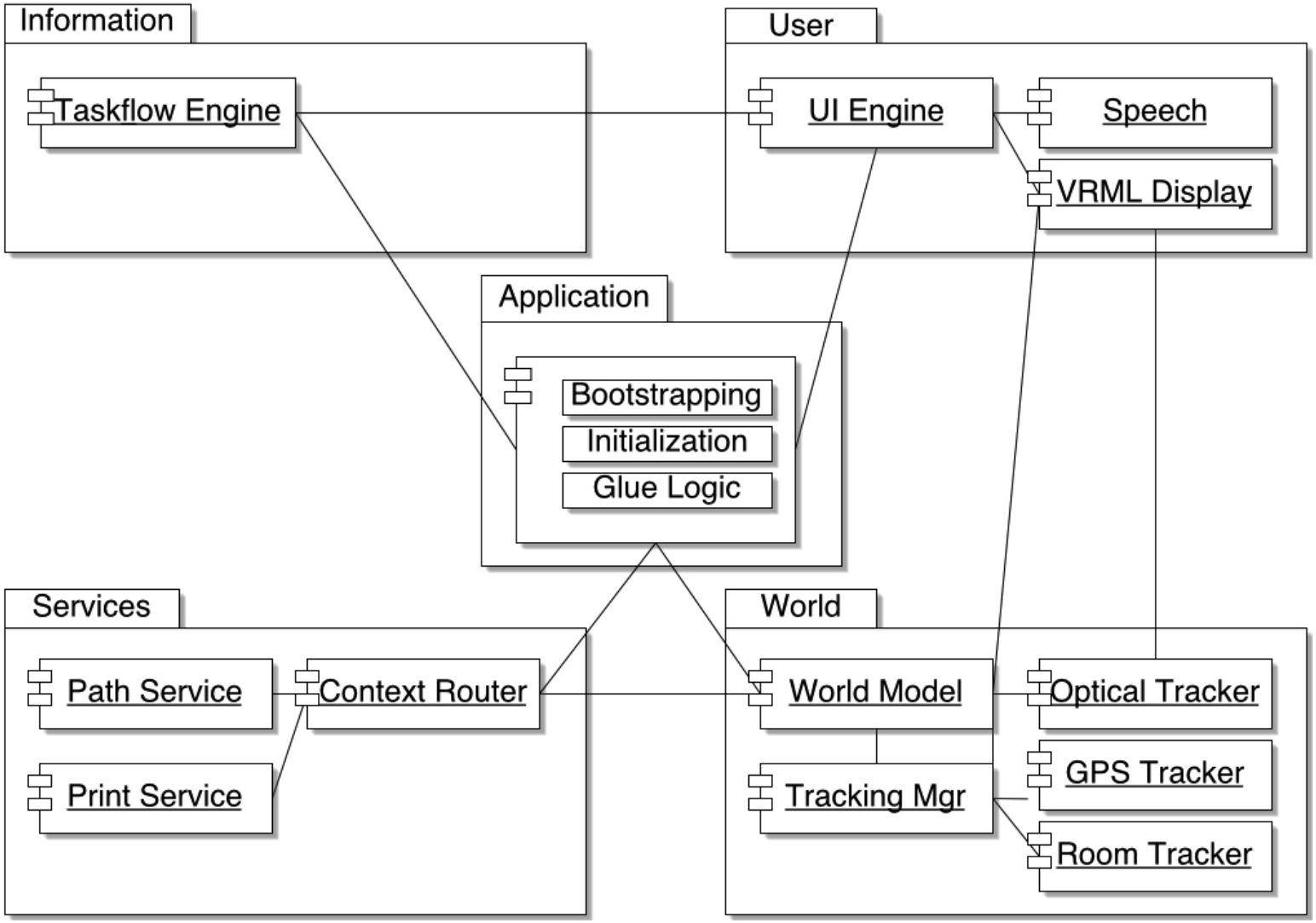


Module developer’s view

- Modules are hardware with software services
- Services provide basic functionality, e.g. tracking
- Services have *needs* and *abilities*

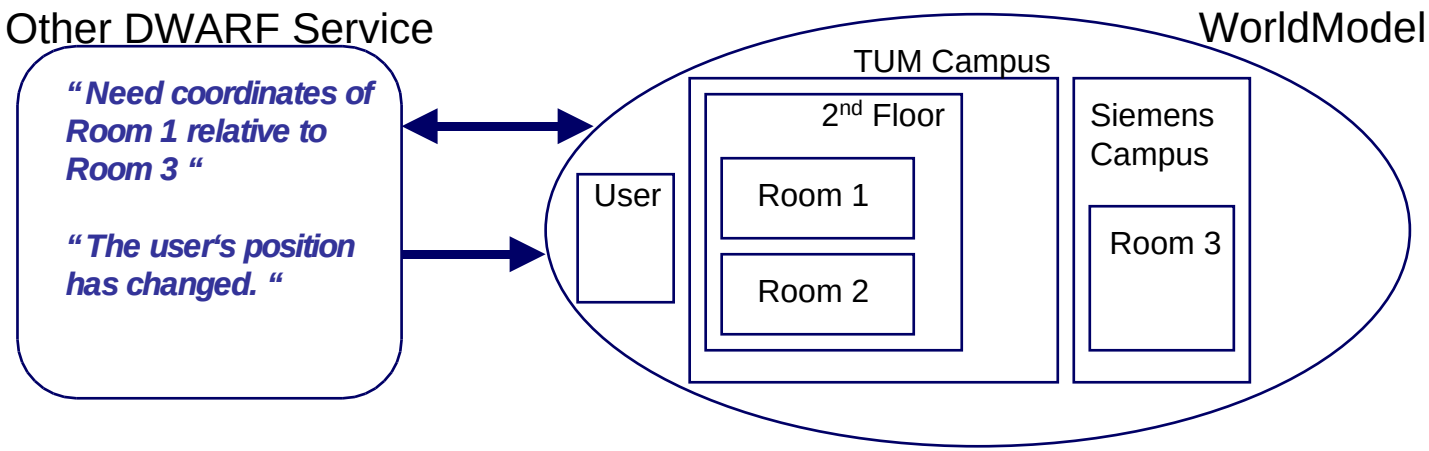


Generic system architecture



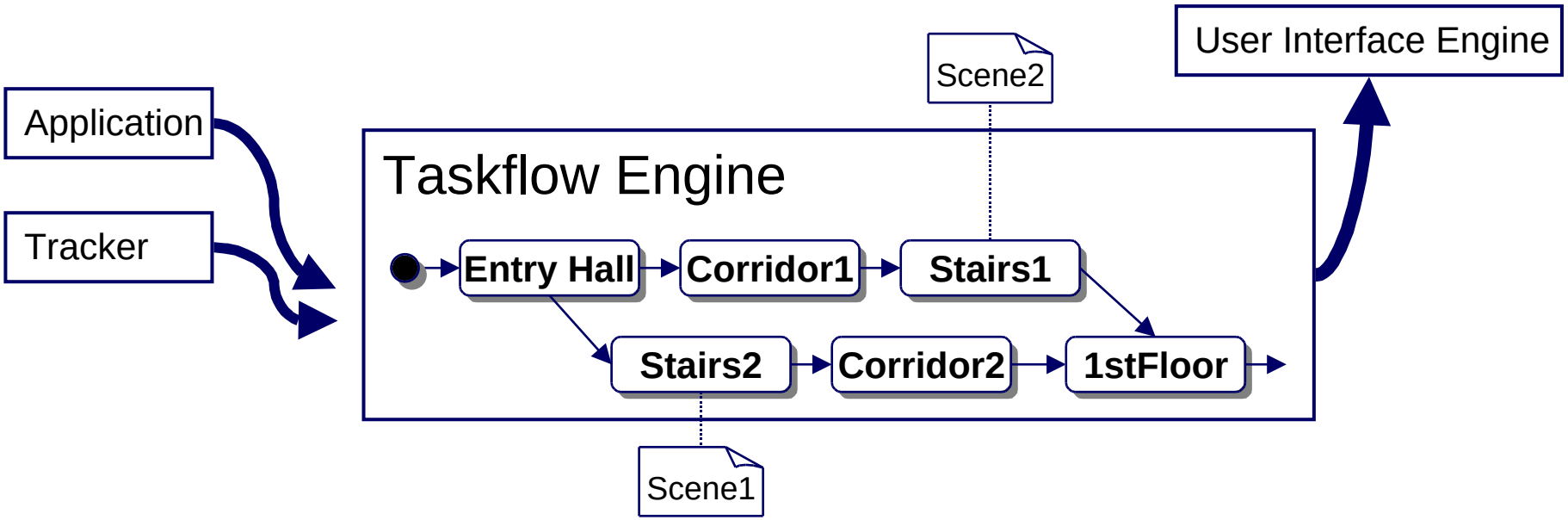
World Model

- Problem: Integration of various coordinate systems (UTM, 3D-coordinates, ...)
- Solution: Database containing all necessary information about the user's environment
- Implementation: C++, running on Windows NT



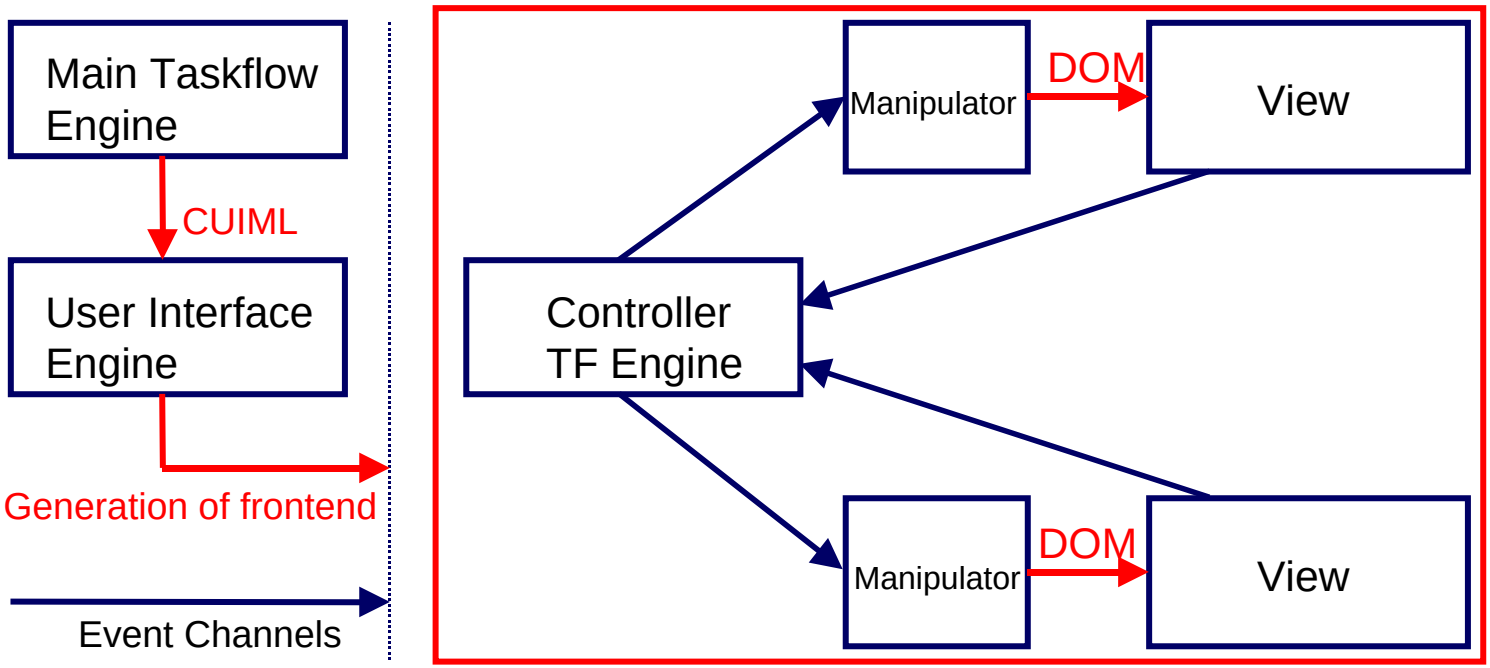
Taskflow Engine

- Problem: Description of the navigation information and keep track of the user‘s progress
- Solution: Use XML to model a taskflow as an extended finite state automaton



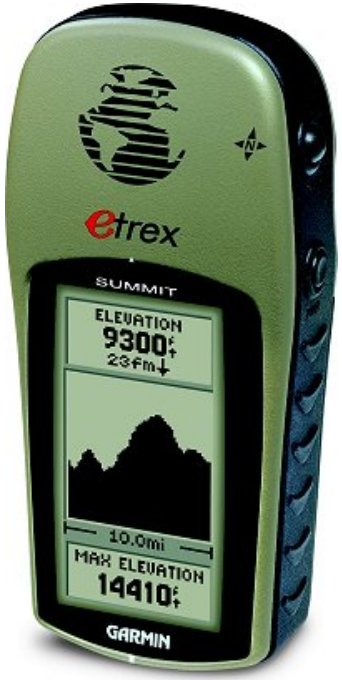
User Interface Engine

- Problem: I/O Hardware differs dramatically
- Solution: XML-based description language (CUIML) for device-independent user interfaces



Simple Trackers

- **Garmin eTrex Summit** GPS receiver
  - with built-in fluxgate compass
  - Communication via RS 232 using the NMEA-0183 protocol.
- **RFID Reader** (simulated)
  - Radio frequency (RF) identification of known positions from the World Model
  - Only simulated due to limited hardware availability

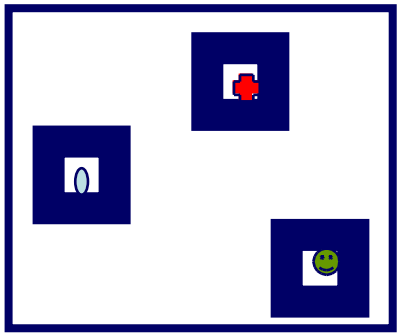


Optical Tracker

- Problem: Augmented Reality needs accurate information about the user's position and orientation ***(6D-Pose)*** in space
- Solution: Optical Tracker computes the pose by locating markers placed at known locations
- Implementation: C++, running on Windows 98

Image Capturing

Marker Detection

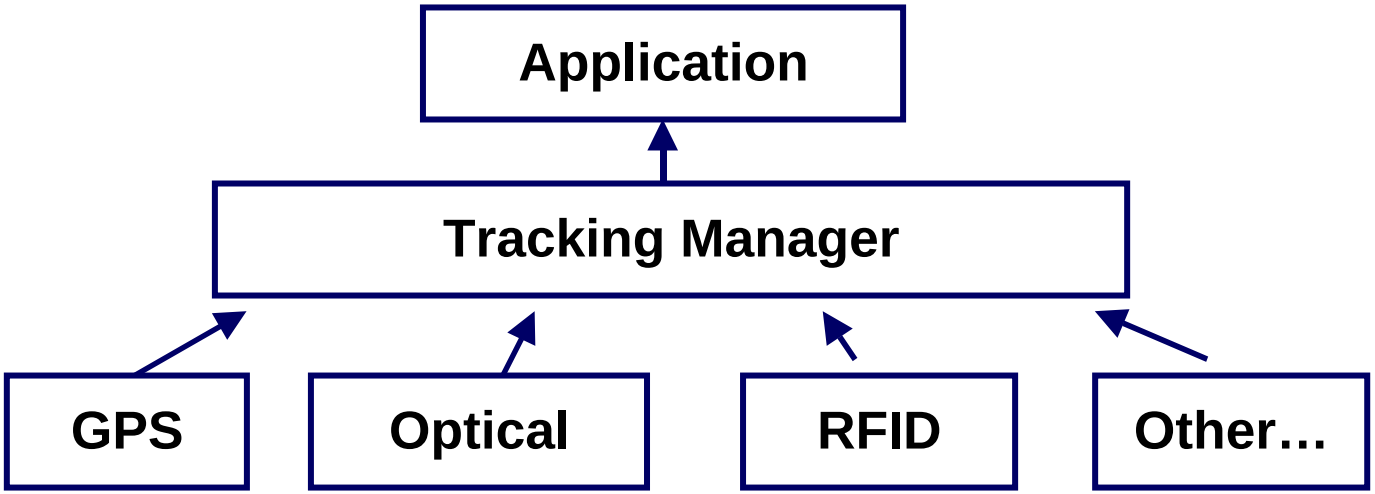


Pose Estimation

$$H = \begin{pmatrix} 1 & 0 & 0 & 3.7 \\ 0 & 1 & 0 & 2.1 \\ 0 & 0 & 1 & 1.54 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Sensor Fusion

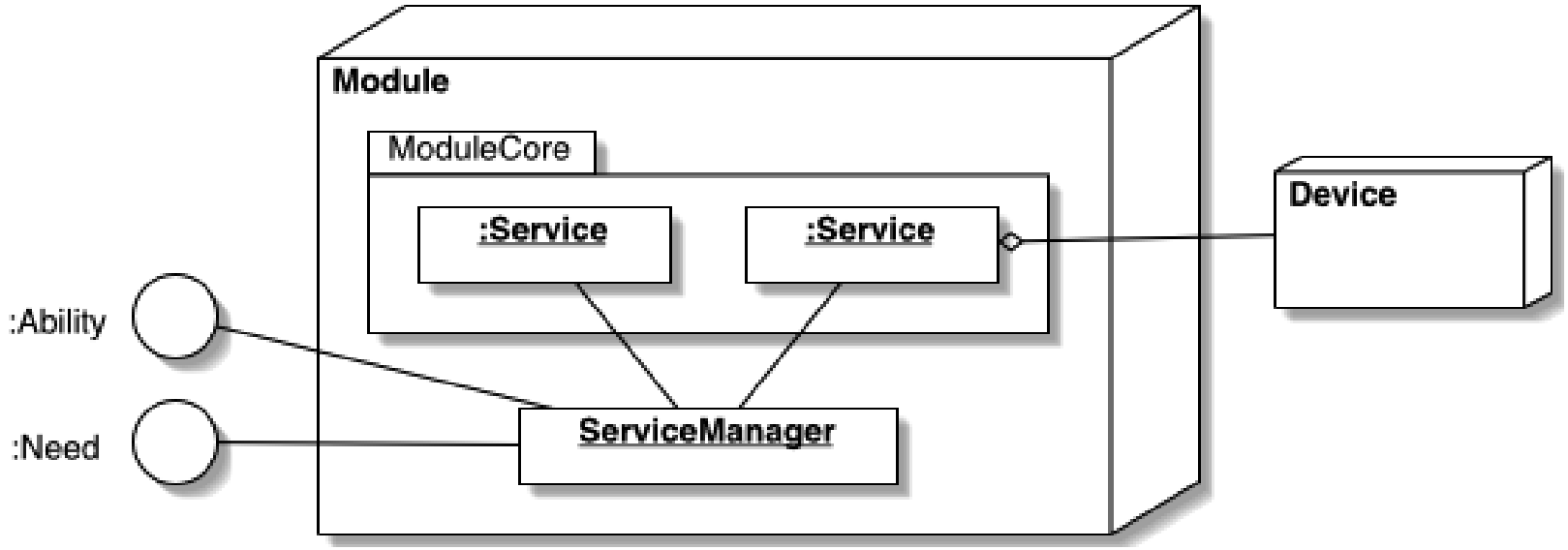
- Fusion of data from different trackers



- Quality of Service (QoS) parameters: accuracy, resolution, frequency, lag

Middleware

- To find each other spontaneously and communicate with one another, the DWARF services use CORBA-based Middleware
- This is distributed as local Service Managers on each hardware node to provide fault tolerance



Where to next?

- More details in the ISAR 2001 paper (to appear on the B-Boards)
- There will be a DWARF consultant
- …or ask Christian, Martin, or Asa