Use Case Scenario
Clear Up The Kitchen Table

- Indoor
- Rooms with furniture
- Mobility
- Task-relevant objects
- Object manipulation
- Spatial knowledge
- Failure-safe operation
- Fault tolerance

- Possibly also
  - Several rooms
  - Doors
  - Moving people
  - Moving objects
Use Case Hardware Platform

User Interaction Unit
- Database Storage
- GUI
- Windows
- TCP/IP
- USB 2.0
- CAN
- IEEE 1394
- analog
- RS422

Vision Unit
- Linux
- IEEE 1394
- Cameras
- PTU
- CAN

Manipulation Unit
- Vision
- IEEE 1394
- PTU
- CAN
- Cameras
- IEEE 1394

Use Case Hardware Options

User Interaction Unit
- Speech Out
- User Interaction Unit
- GUI
- Speech In
- Database Storage
- TCP/IP
- USB 2.0
- CAN
- IEEE 1394
- analog
- RS422

Vision Unit
- Linux
- IEEE 1394
- Cameras
- PTU
- CAN

Manipulation Unit
- Vision
- IEEE 1394
- PTU
- CAN
- Cameras
- IEEE 1394

1 KHz
## Characteristics of the Robotics Domain

- Extremely heterogeneous hardware
- Inherently concurrent
- Inherently distributed
- Device dependent
- Stochastic properties of physical world
- Real-time constrained
- Resource constrained

- Currently not adequately supported by available
  - robot software architectures
  - robot software development environments

- Inadequate evaluation and assessment
- Mere demonstration character

## Software for Autonomous Mobile Robots: Heterogeneity of Hardware

- Robots, robot teams, sensor networks are distributed system composed of very heterogeneous hardware
  - sensors:
    - bumpers, IRs, sonars, laser scanners, accelerometers, gyros, GPS, microphones, cameras, omnicams, stereoheads
  - actuators:
    - DC motors, steppers, servos, kickers, pan-tilts, arms, hands, legs, HDoF bodies, polymorphic systems
  - computational entities:
    - microcontrollers, embedded PCs, PDAs, notebooks, remote PCs
  - communication devices, mechanisms, and protocols:
    - I2C, serial, CAN, USB, UDP, TCP/IP, Firewire

- No plug and play!
- No configuration management!
- Heterogeneity grows over system lifetime!
- By-and-large, hardware and software maintenance for large robot teams and large embedded sensor networks must be considered unsolved
Software for Autonomous Mobile Robots: 
Distribution and Realtime Constraints

- Hardware and communication environment forces to deal with
  - Distributed programming concepts
    - Load balancing, multi-threading, concurrency, synchronization, signalling, event-driven activation, event ordering, ...
  - Communication protocols
    - Latency, timeouts, partial system failures, ...
  - GUI event loops

- Responsiveness to sensor- and actuator-initiated signals
  - Requires realtime or pseudo-realtime computing
  - Noisy sensors and actuators
  - Location dependency
  - Need for probabilistic models
  - Need for elaborate world models

Software for Autonomous Mobile Robots: 
Diversity of Software

- Roboticists use a wide diversity of often computationally intensive methods
  - Control theory
  - Computational geometry
  - Neural networks
  - Genetic algorithms and evolutionary methods
  - Reinforcement learning
  - Vision processing routines
  - AI planning techniques
  - Behavior systems
  - Probabilistic reasoning
  - Optimization techniques
  - Search techniques

- All these problems make software development for mobile robots very complex and error-prone
**Programming Mobile Robots**

- Responsiveness to sensor- and actuator-initiated signals requires multithreaded programming
- Realtime or pseudo-realtime computing
- Distribution, concurrency, reactivity, usability
- Communication, multi-threading, synchronization, event-driven activation, and GUI event loops
- Partial failures, latency, load balancing, signalling, event ordering, ...

- These problems
  - make software development in robotics complex and error-prone
  - hinder research
  - limit exchange of scientific results
  - jeopardize commercialization

**What Makes The Problem Hard?**

- No common architectures
- No common methods
- Hardware-dependency of developed code
- Missing abstractions
- No reusable components
A First Conclusion

- Any system, which takes away or limits the programmer’s freedom to implement her architectural or computational ideas, is bound to fail.

- Any restrictions or commitments imposed by a system must be significantly outweighed by advantages gained.

Use Case Open Questions

- Mobile manipulation: integration of mobility and robot manipulation

- Challenge is the integration of multiple functionalities from both areas and finding solutions to new problems

- Use of pre-developed components, like arm, hand, base, etc., poses possibly hard integration issues

- In particular:
  - Different operating systems
  - Different communication protocols
  - Different inherent internal cycle times in functional modules

- Another hard problem: Detecting and handling failure situations
What Does Miro Offer?

- **Miro Device Layer**
  - Clean, coherent object-oriented class interfaces
  - Available already for major parts
- **Miro Communication and Configuration Layer**
  - Various often-used communication patterns
  - Group communication via notify-multicast protocol
  - Extended XML-based configuration facilities
- **Miro Service Layer**
  - Unified network-transparent access to object services
  - Built-in facilities for data acquisition and logging
- **Miro Framework Layer**
  - Fine-grained control over complete visual processing via VIP
  - Flexible hierarchical reactive control via BAP
  - Particle filter-based self-localization

Example: Kinematics and Motion Interfaces

Different kinematics:
- Synchro drive
- Differential drive
- Ackermann steering

Different coverage of velocity space
Abstract Actuator APIs Example: Drive Motion Services

- Base abstract interface: target velocity, velocity bounds
- Specialized abstract interfaces: left/right wheel velocities, translation, rotation, wheel speed
- Customized interfaces: motor power, torques

Abstract Sensor APIs Example: Laser Range Finder Services

- Base abstract interface: activation/deactivation, setting resolution and scan range, getting range scans
- Vendor-specific abstract interfaces: setting scan range, clustering, getting intensity scans
- Product-specific abstract interfaces: setting scan range
Abstract Data APIs Example: Range Scan Services

- Generalization for laser range finder, infrared, sonar, bumper scans
- Reference to specification of sensor layout
- Multiple modes of data publishing
- Multiple modes of data updating
- Permits for generic obstacle avoidance services

AbstractRangeScan

- update()
- setScanResolution()
- getDistance()

AbstractBumperScan AbstractInfraredScan AbstractSonarScan AbstractLaserScan

Miro Middleware Layers

Database Storage

Speech In

Speech Out

GUI

RS422

RS232/422

GPIO

CAN

IEEE 1394

Cameras

PTU

CAN

Gripper

CAN

Arm

Manipulation Unit

VxWorks

Linux

User Interaction Unit

Windows

Vision Unit

Linux

RS232/422

USB 2.0

IEEE 1394

TCP/IP

Done

To Do
Use Case Scenario:
Possible Functional Architecture
B-IT Tutorial in December 2005

- Player/Stage/Gazebo
- MCA2
- Smartsoft
- Miro
- Marie
- ORCA2

Synthesis by Functionality

- application frameworks
  - domain knowledge
  - component libraries
  - CBSE
  - functional class libraries
    - methods
    - patterns
    - generic utilities
- services
  - network-transp. access
- classes
  - object-orientation
- file I/O
  - coherent file IF
- functions + protocols
  - plurality of vendor IFs
### Synthesis by Functionality (current)

| Service Robot Applications | - application frameworks  
|                           |  - domain knowledge  
|                           |  - component libraries  
|                           |  + CBSE  
|                           |  - functional class libraries  
|                           |  + methods  
|                           |  + patterns  
|                           |  + generic utilities  
| RCA Framework Layer       |  + methods  
| Robot Component Framework Layer |  + patterns  
| Robot Method Framework Layer |  + generic utilities  
| Network Service Layer     |  - services  
|                           |  + network-transp. access  
|                           |  - classes  
|                           |  + object-orientation  
|                           |  - file I/O  
|                           |  + coherent file IF  
|                           |  - functions + protocols  
|                           |  + plurality of vendor IFs  

### Synthesis by Functionality (needed)

| Service Robot Applications | - application frameworks  
|                           |  - domain knowledge  
|                           |  - component libraries  
|                           |  + CBSE  
|                           |  - functional class libraries  
|                           |  + methods  
|                           |  + patterns  
|                           |  + generic utilities  
| RCA Framework Layer       |  + methods  
| Robot Component Framework Layer |  + patterns  
| Robot Method Framework Layer |  + generic utilities  
| Network Service Layer     |  - services  
|                           |  + network-transp. access  
|                           |  - classes  
|                           |  + object-orientation  
|                           |  - file I/O  
|                           |  + coherent file IF  
|                           |  - functions + protocols  
|                           |  + plurality of vendor IFs  

| Class Layer | WebServices  
| File Interface Layer |  - functions + protocols  
| Device Driver Layer |  + plurality of vendor IFs  

| X Component |  + methods  
| RHI GUI |  + patterns  
| Miro LAP |  + generic utilities  
| WebServices |  - services  
|                           |  + network-transp. access  
|                           |  - classes  
|                           |  + object-orientation  
|                           |  - file I/O  
|                           |  + coherent file IF  
|                           |  - functions + protocols  
|                           |  + plurality of vendor IFs  

**ORCA Components**
- SmartSoft Builder
- Miro Logging
- Miro Mediator Patterns
- Miro VIP
- Miro MCL
- Miro BAP
- Miro VIP
- Miro MCL
- Miro BAP
- Miro VIP
- Miro MCL
- Miro BAP
- Miro VIP
- Miro MCL
- Miro BAP
- Miro VIP
- Miro MCL
- Miro BAP

**Robot Method Framework Layer**
- Vision
- Object Recognition
- Object Track
- SLAM
- Path Planning
- Task Planning
- Learning
- Logging
- RHI GUI
- Miro LAP

**Network Service Layer**
- BaseDriveFileIF
- LaserFileIF
- Base Drive Device
- Laser Device

**Class Layer**
- Base
- L
- ArmClass
- CommClass

**File Interface Layer**
- Base
- L
- ArmFileIF
- CommFileIF

**Device Driver Layer**
- Base
- L
- Arm Device
- Comm Device

**Service Robot Applications**
- VacuumBot
- NurseBot
- ShopBot
- NannyBot
The Next Generation of Robotics Software Development

new developments yet to be fully appreciated by robotics
- agile software development
- software libraries of best practice algorithms
- model-based software engineering

cross-sectional topics
- harmonization for interoperability and portability
- robust autonomy
- openness

BRICS
Best Practice in Robotics

Showcase Research
Showcase Industry
MDE-based Tool chain
Best Practice Architectures
Middleware Interfaces
Best Practice Algorithms
Robust Autonomy
Openness
Flexibility
Harmonisation
Showcase Education
Consortium

KUKA

WP2: Architecture, Interfaces, Middleware:
Key Ideas and Concepts

- Handling issues characteristic to robotics
  - Heterogeneous hardware (self-describing components etc.)
  - Distributed systems (communication frameworks, middleware)
  - Heterogeneous software (stratified interfaces, configuration, simulation)

- Making robots safe
  - Error handling (sw quality, monitoring, sw patterns)
  - Fault tolerance (plug-and-play, QoS, service level maintenance)

- Providing usable software engineering frameworks
  - Refactoring (... known solutions for quality: efficiency and robustness)
  - Software patterns (... apply known sw patterns and develop/identify new)

- Building architectures for robotic applications
  - Method frameworks (best practice of algorithms)
  - Component-based software construction (configuration)
BRICS Software Architecture Concept

Robot Control Architecture Workbench

Abstract APIs II: Network-Transparent Services

Abstract APIs I: OO Device Interface Layer

Component Device Interfaces

Component Hardware

Method Framework Layer

visual servoing

+ control structure
+ domain knowledge
+ applications

+ algorithm libraries
+ component technology
+ configurable components
+ reusable components

+ communication middleware
+ remote object access
+ distributed objects

+ object orientation
+ consistent abstract APIs
+ sensor classes hierarchy
+ actuator classes hierarchy

HW heterogeneity
+ vendor-dependent interfaces
+ vendor-dependent protocols

BRICS Interoperability

Simple case: all components connected to a single computer

Robot Control Architecture Workbench

Method Framework Layer

visual servoing

Interoperability at all levels above!

Abstract APIs II: Network-Transparent Services

← Interoperability at this level!

Abstract APIs I: OO Device Interface Layer

← This level not used in this case; optimized away

Component Device Interfaces

Component Hardware

← Interoperability at this level!
**Conclusions**

- software development for robotics is extremely difficult
- robotics is (partially) waking up to software engineering issues
- some technology is around; using it is much better than not using it
- still a lot of work ahead of us
- BRICS project will address the pending issues
- outreach activities such as research camps allow community to get involved

- Thank your for your attention!