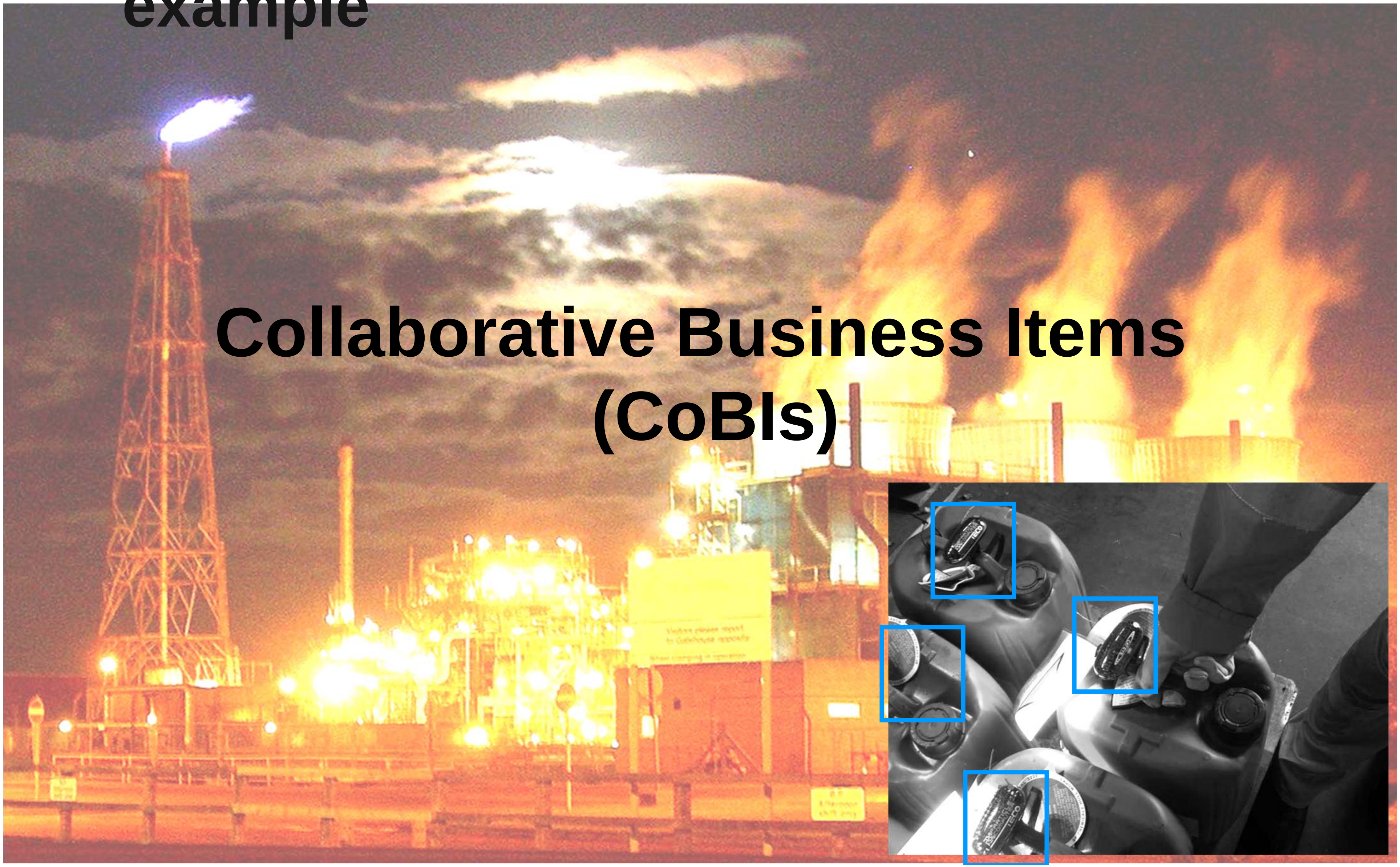
Applications for self-organisation in collaborative sensor networks

Organic Computing Workshop

ARCS Conference, Hanover February, 23 2010

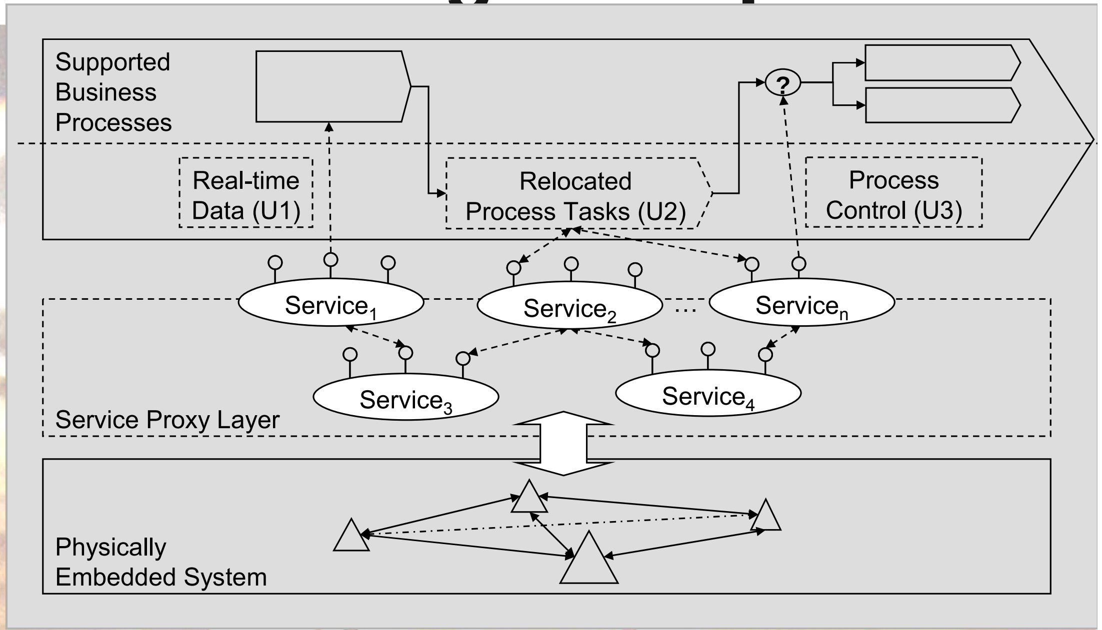
Michael Beigl
TU Braunschweig
Institute of Operating Systems
and Computer Networks
www.ibr.cs.tu-bs.de/dus

How do collaborative sensor network oApps look like I: A motivating example



Collaborative Business Items (CoBIs)

How do collaborative sensor network oApps look like I: Motivating example



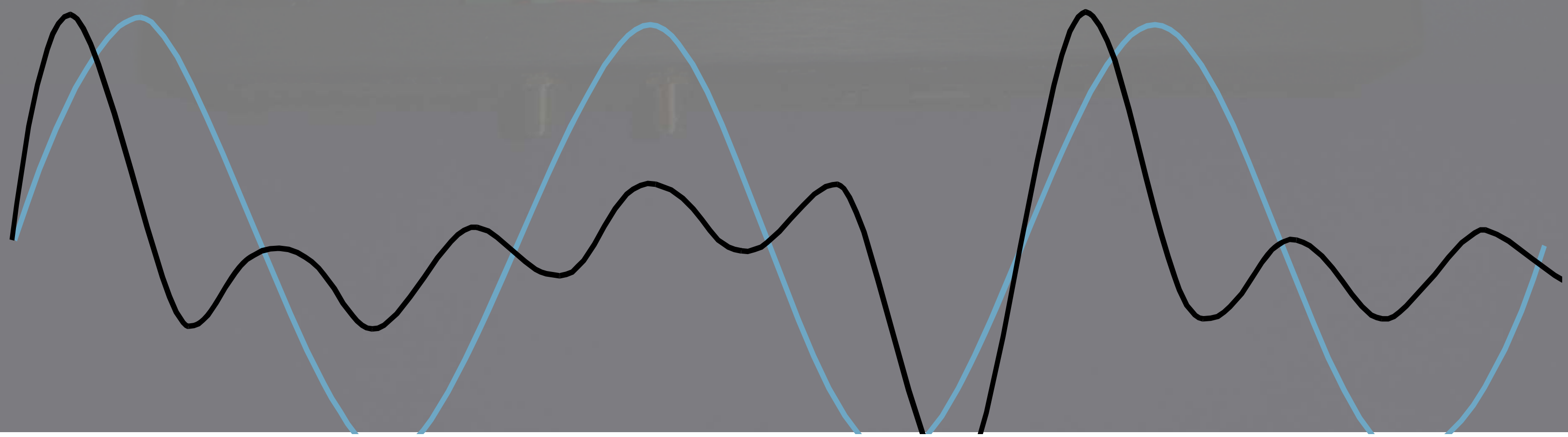
- Chemical-Containers at BP equipped with sensor nodes
- MANY types of self-organization in a **system**
  - Real-time channel access
  - Organizing the collaboration of sensor nodes, heterogonous collaboration
  - Reasoning about faults, failures, errors
    - Backend reasons about critical conditions, provides new rules for middleware and sensor nodes

How do collaborative sensor network oApps look like II: Tools

Software Defined Radio

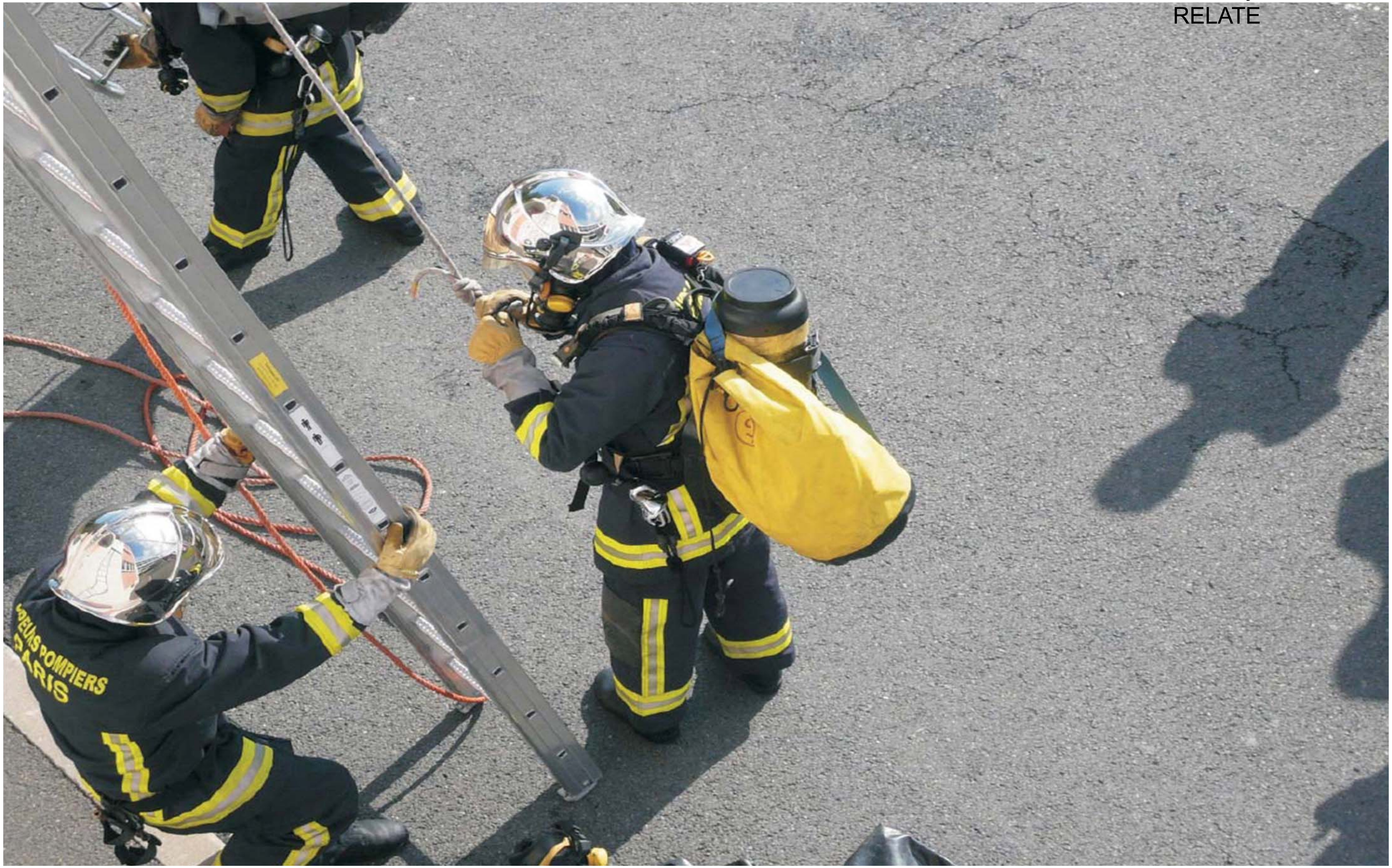
How do collaborative sensor network oApps look like II: Tools & Apps

- Coherent transmission: collaboration to self-organize a set of nodes that sing together like a chorus
  - Application: Field deployed wireless sensor networks
- Non-Coherent transmission: collaboration to self-organize a set of nodes e.g. to process data on the channel
  - Application RELATE



How do collaborative sensor network oApps look like III: The RELATE example

EU Projekt RELATE

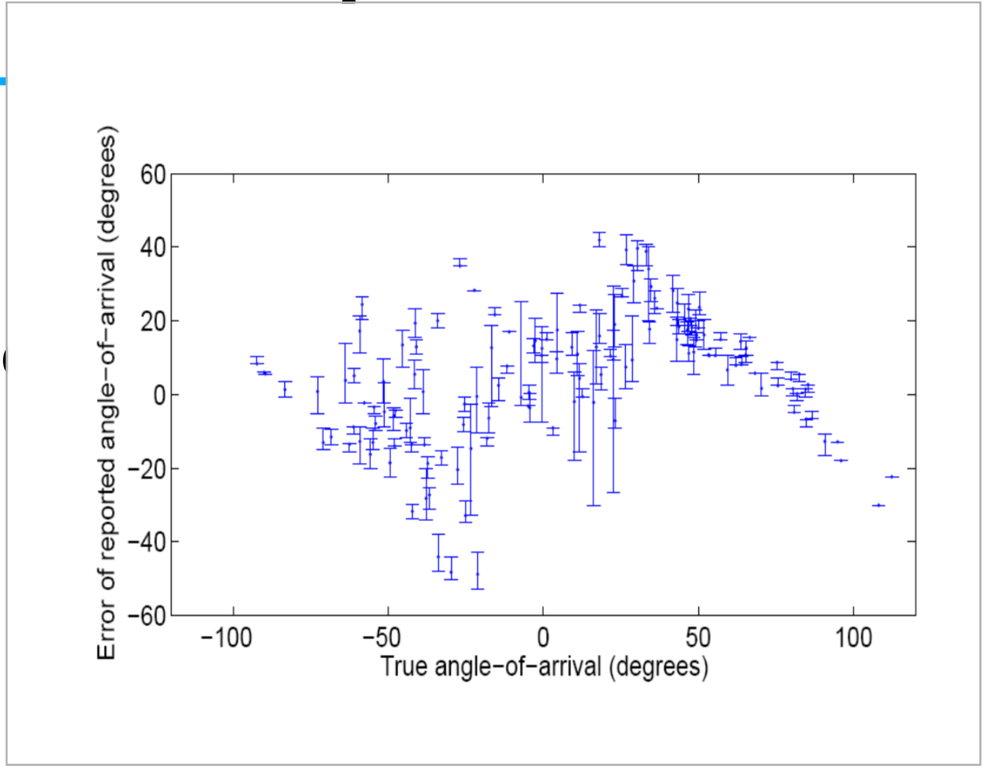


Motivation: Distributed Map for Fireman

EU Projekt
RELATE

- **Goal**
  - Replacement of „Lifeline“ for fireman
  - System: Determine position of fireman with best possible accuracy
- **Method**
  - Automatically drop sensor nodes in a building
  - Sensor nodes measures and communicate distance peer to peer
- **Dynamic**
  - Several fireman work in parallel in one building
    - High node density, area coverage
  - Sensor nodes operate in harsh environment
    - Disturbance, destruction of nodes

RELATE: Distributed Map creation



Problem

- High measurement error (systematic, statistical)
- Communication errors, nois
- Highly dynamic setting, no stable set of nodes

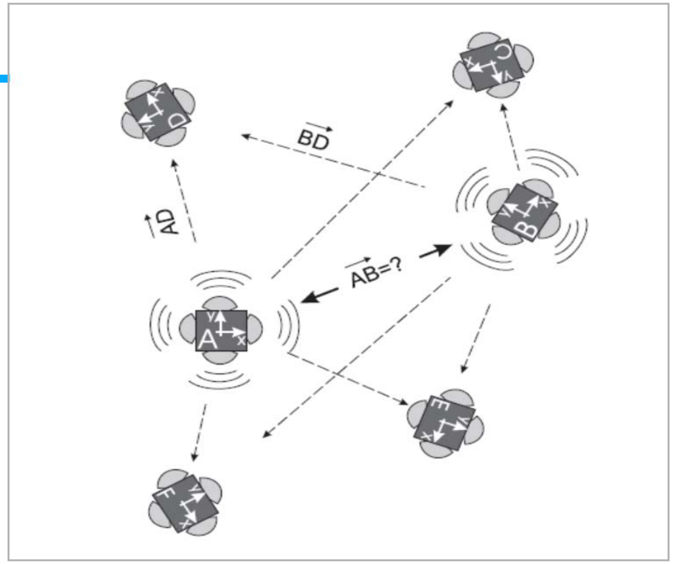
Sensor node tasks

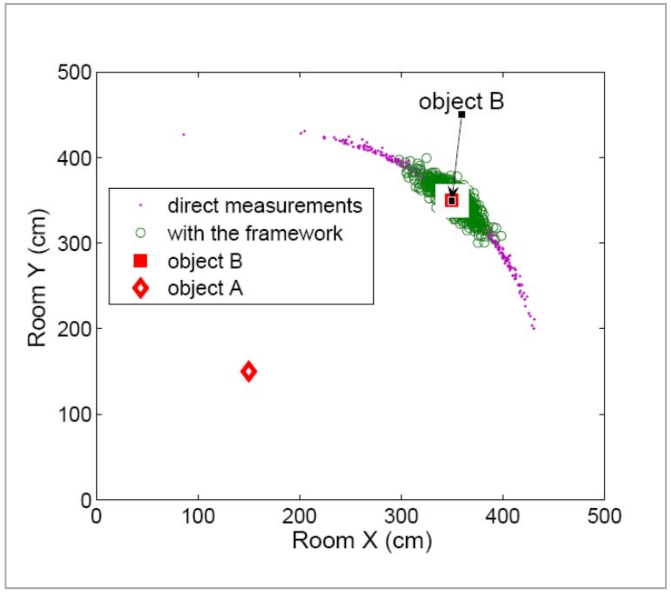
- Measure Distance to other nodes
- Result: Estimation of Distance
- Receive estimations from other nodes
- Distribute Distances
- Calculate new distances (Average)
- Show result to end user

Properties RELATE sensor networks

Properties

- Self-optimizing the local view: **ask neighbors**, build collective view
- Converts systematic to a statistical error with Gaussian distribution
- Degree of self-optimization depends on time, energy, conditions and # of nodes
- **Problem**: **Don’t trust anybody**: Quality of distances differs

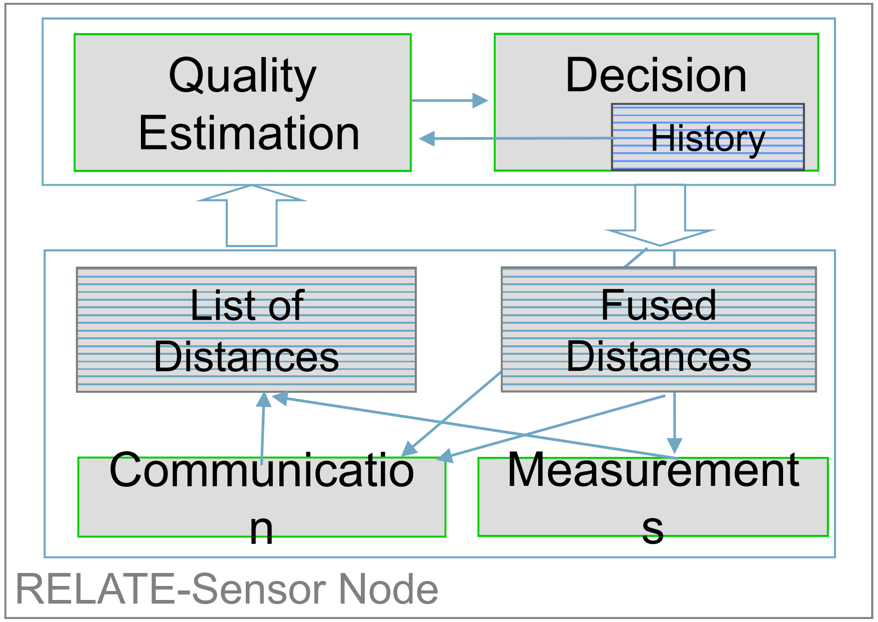




RELATE Distance Measurement

- **All measurement values are error prone**
  - Resulting fusion problem: Instead of improvement we might worsen the result
  - But errors follow a certain pattern, e.g. correlate to type of sensor, sensor node, context etc.
- **Solution: Selforga + Context-awareness**
  - Additional contextual values while measurement
  - Annotate distance and context/value pairs
  - Rate quality of measurement according to actual context, history
  - Fusion/calculation of distances uses quality measurement

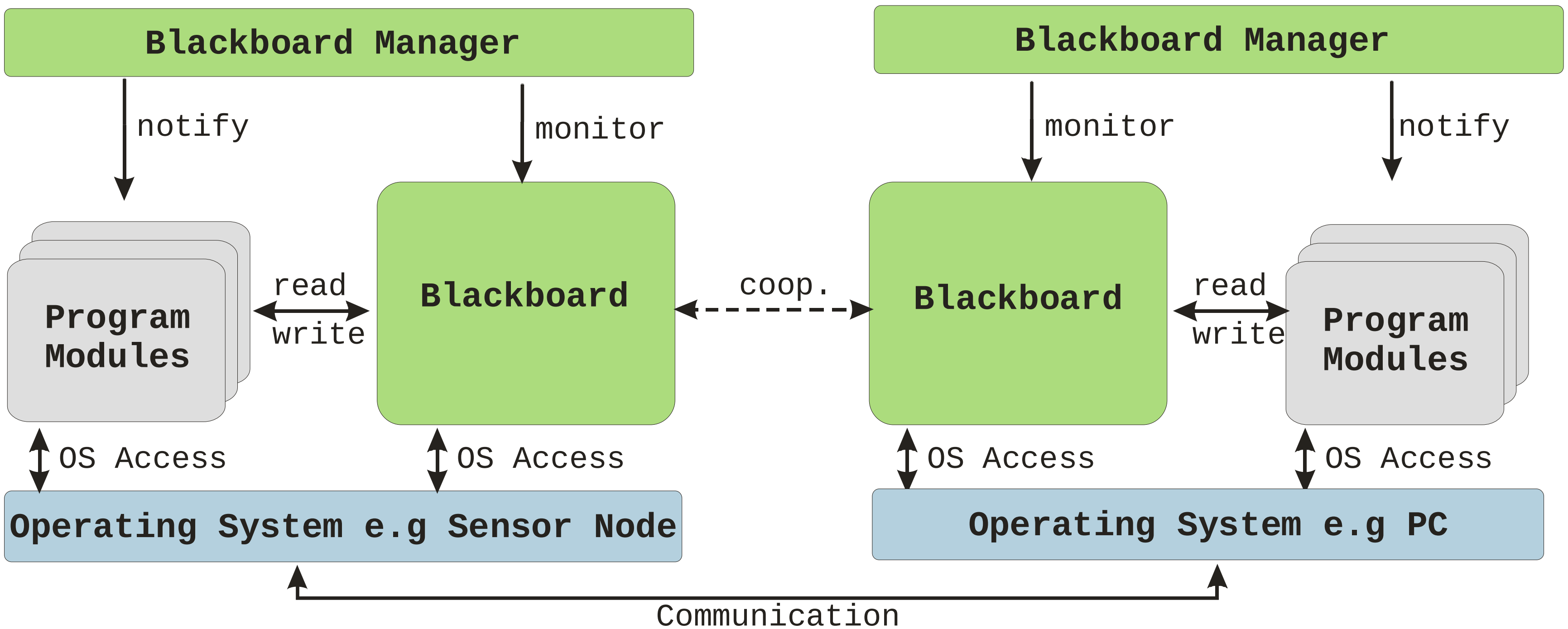
Model of RELATE Sensor Node



- **Communication**
  - Distance and quality
- **Sensory**
  - Measure distance
- **Quality Estimation**
  - Estimation based on context, self-aware
- **Decision**
  - Fuse distances considering quality estimation
- **List of Distances**
  - Quality values and distances
- **Problem**
  - Very complex system in one node
  - Even more complex when looking at several nodes

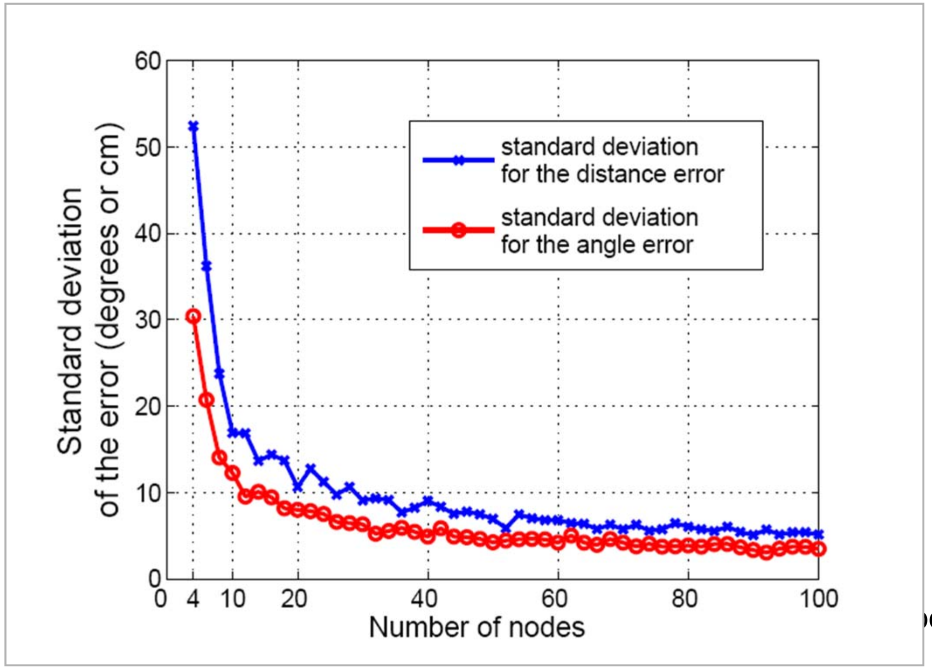
Tool: Blackboard Implementation

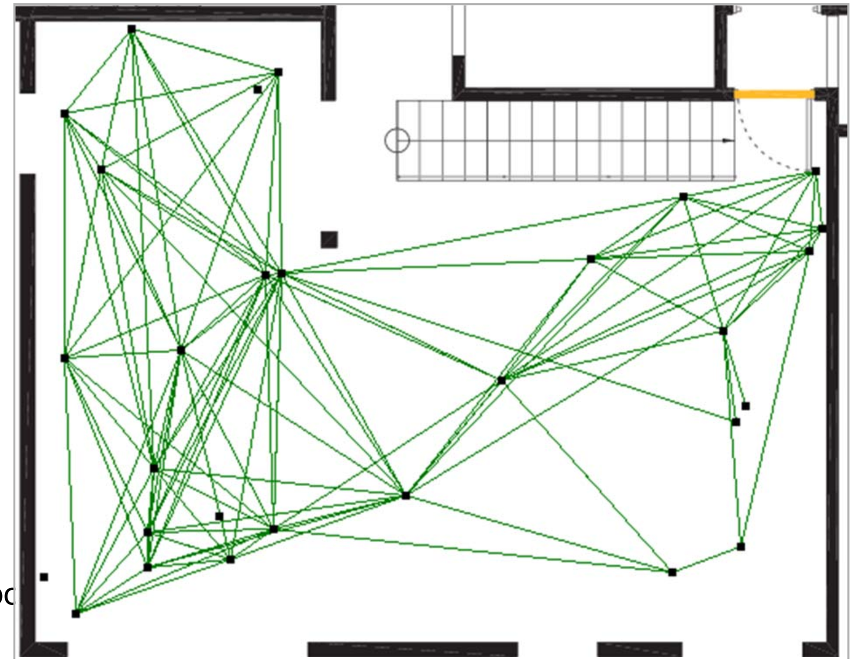
- All modules communicate via Blackboard
- Modules are local or remote (simulated)
- Time is real or virtual, allows to follow progress
- Blackboard console as debugging tool



Superimposing Signals: Collaborative Communication

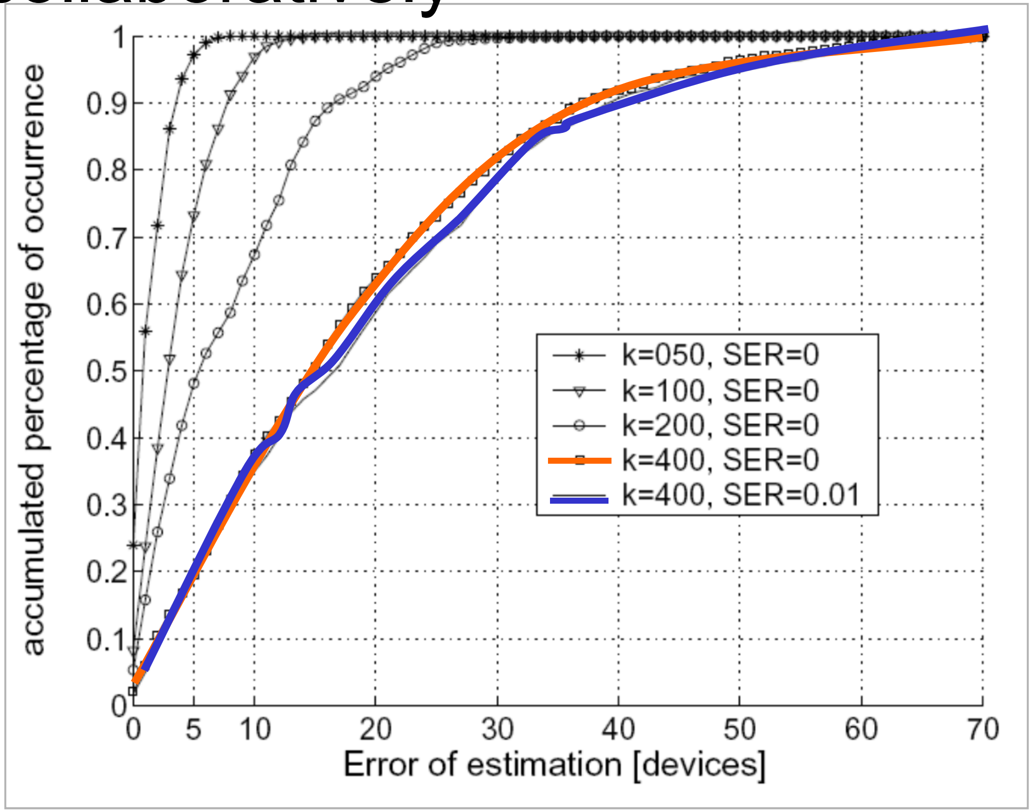
- Problem: We need to send n² packets to compute weighted sums
- Solution: Use channel to compute weighted sum
- O(n²)->O(n)
- **Collaborative Signaling**





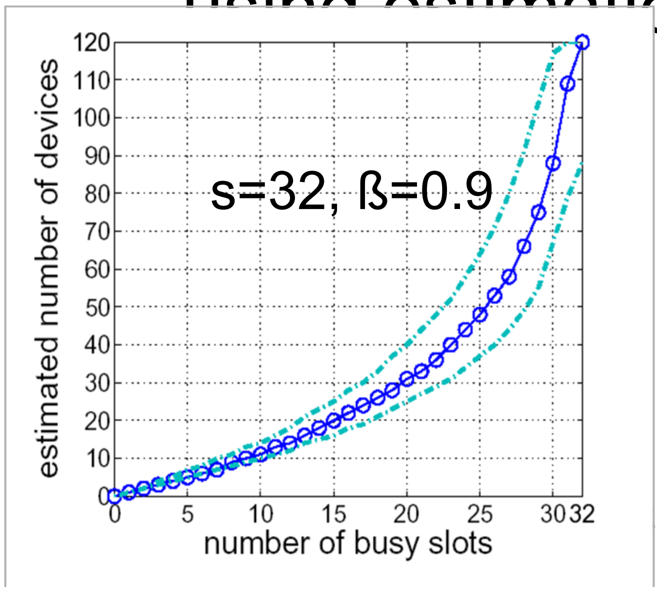
1: Analog Network Coding

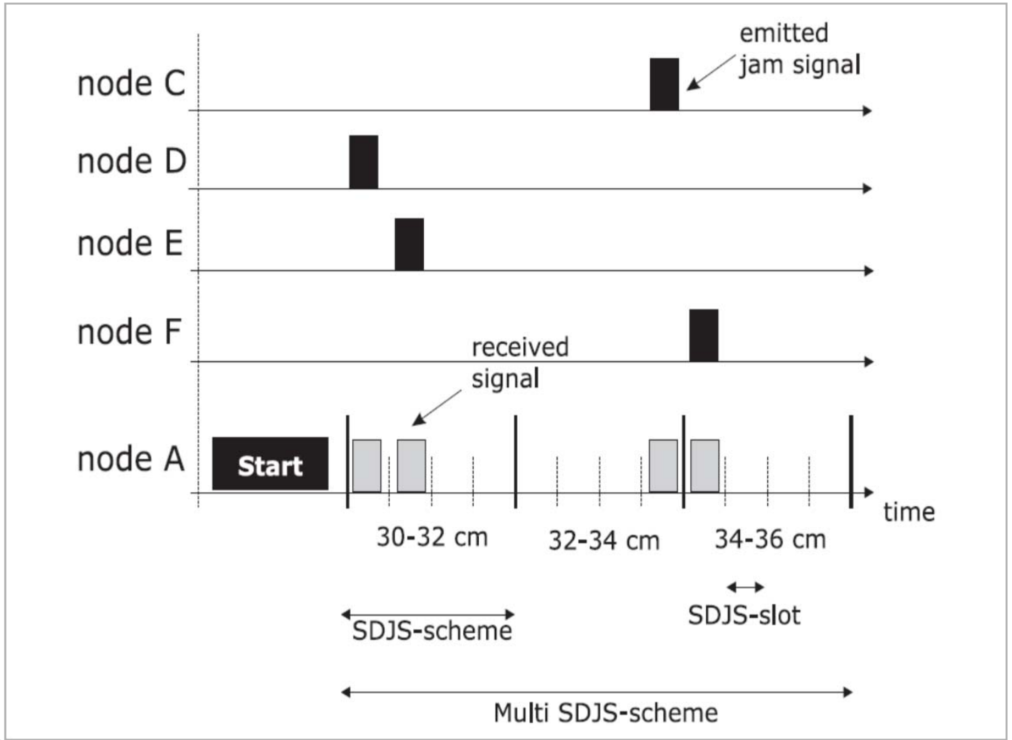
- Principle: Use (analog) coding on the channel to transfer information collaboratively
- Robust against errors
- Reduction of Energy consump. Up to 1000x
- Real-time wireless communcation



2: processing on the channel

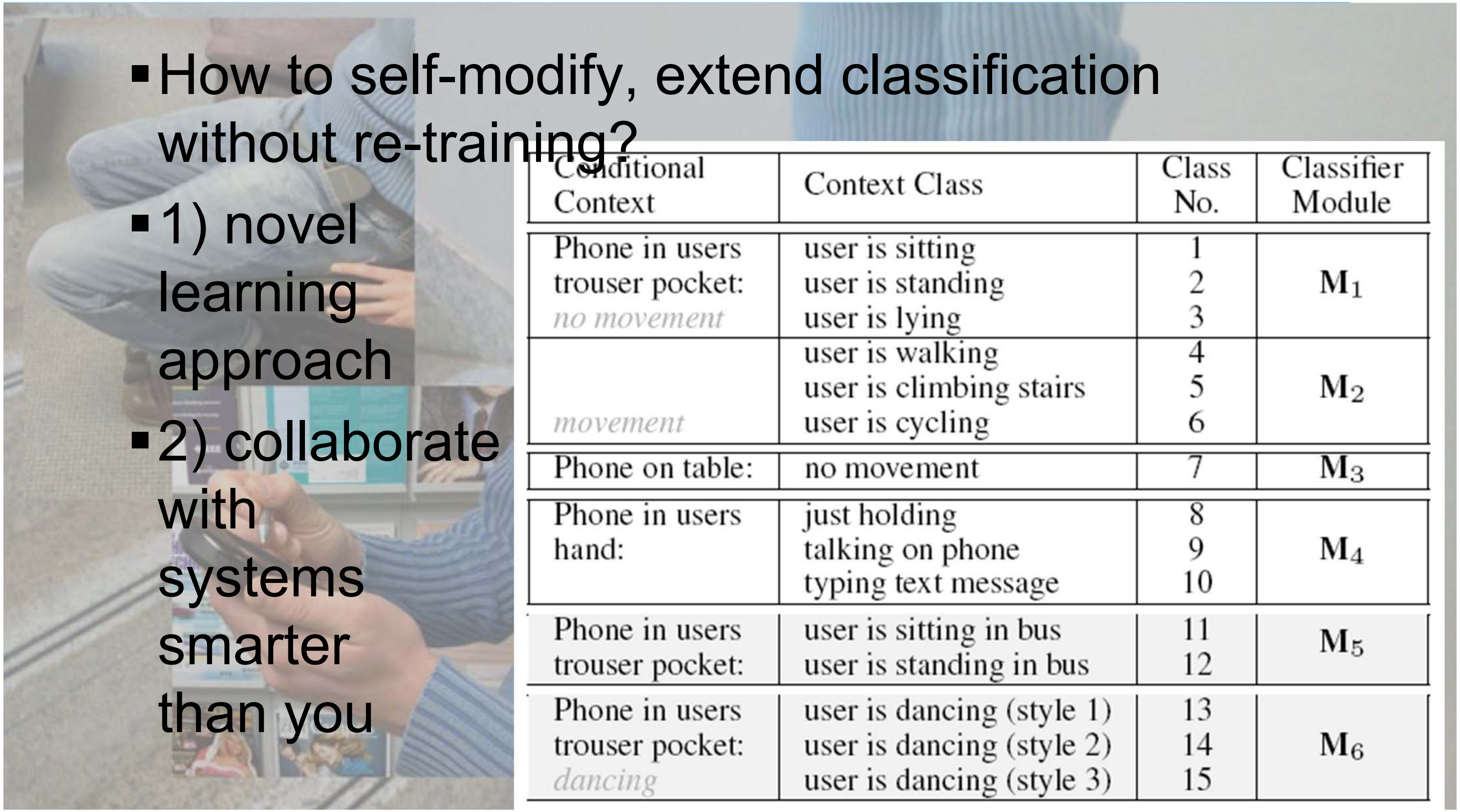
- E.g. operations „Or“, „Average“, „Weighted Avg“
- Priciple: Transmission of extremly short, overlaying signals
- Interpretation using estimation





More oApp‘s: Context Phone
Detecting situations and activities

More oApp's: Context Phone Detecting situations and activities



- How to self-modify, extend classification without re-training?
- 1) novel learning approach
- 2) collaborate with systems smarter than you

Conditional Context	Context Class	Class No.	Classifier Module
Phone in users trouser pocket: *no movement*	user is sitting user is standing user is lying	1 2 3	$\mathbf{M}_1$
movement	user is walking user is climbing stairs user is cycling	4 5 6	$\mathbf{M}_2$
Phone on table:	no movement	7	$\mathbf{M}_3$
Phone in users hand:	just holding talking on phone typing text message	8 9 10	$\mathbf{M}_4$
Phone in users trouser pocket:	user is sitting in bus user is standing in bus	11 12	$\mathbf{M}_5$
Phone in users trouser pocket: *dancing*	user is dancing (style 1) user is dancing (style 2) user is dancing (style 3)	13 14 15	$\mathbf{M}_6$

Conclusion

- **Organic Computing methods help to efficiently implement features for computing systems**
  - Avoids specification of too many possible conditions
  - Provides robustness in case of errors, failures, faults
  - Allows heterogeneous integration of knowledge & functionality
- **For improved robustness in real-world settings, context and self-awareness is helpful**
- **Tools are required to efficiently run complex projects**
  - But tools are often specific to project
  - Although re-use is thinkable and would be helpful

Applications for self-organisation in collaborative sensor networks

Thank you for your attention

Michael Beigl
TU Braunschweig
Institute of Operating Systems
and Computer Networks
www.ibr.cs.tu-bs.de/dus