Information-Processing Architectures for Intelligent Robots: Designs, Tools, Examples and Experiments

Nick Hawes

Intelligent Robotics Lab, School of Computer Science, University of Birmingham

Computer Science Research Colloquium, University of Hertfordshire, 12/12/07

Motivation The CoSy Architecture Schema Illustrations Experiments Conclusion

Outline

Motivation

- 7-site 4-year EU project building robotic systems aiming to demonstrate both state-of-the-art components and systems.
- We are trying to advance of the *science* of *building* intelligent systems: integration is central.
- We see information-processing architectures as central to this problem.

Motivation The CoSy Architecture Schema Illustrations Experiments Conclusion

What Are Architectures?

- Information-processing structures that circumscribe the functionality of system.
- An understanding of *information-processing architectures* is central to the understanding of intelligent integrated systems.
- They are a useful abstraction for integration (more specific that communication frameworks, more general than particular representations).
- As a design and implementation tool they represent the battleground of science and engineering.

Levels of Description

- We use four different levels of description for architectures:
  - High-level principles and *requirements*.
  - A *schema*-level realisation of these.
  - *Instantiations* of a schema in a concrete design.
  - *Implementations* of a design in software and hardware.
- These relate to *niche space* (requirements) and *design space* (designs) as described by Aaron Sloman and the Birmingham CogAff group.

Contributions

Principled approaches for integrating functions (i.e. components and their representations) into a single intelligent robot.

- An architecture schema, combining insights from both robotics and AI/cognitive science, designed to support concurrent processing on shared information.
- An approach to binding information across multiple modalities into a single amodal representation.
- An investigation into filtering in various architecture instantiations.

Some (Selected) Key Problems

- Filtering: How does information flow between a subset of components in an architecture?
- Binding: How can information about the same thing from different components in an architecture be connected?
- Incrementality: How can architectures be easily extended with new capabilities?

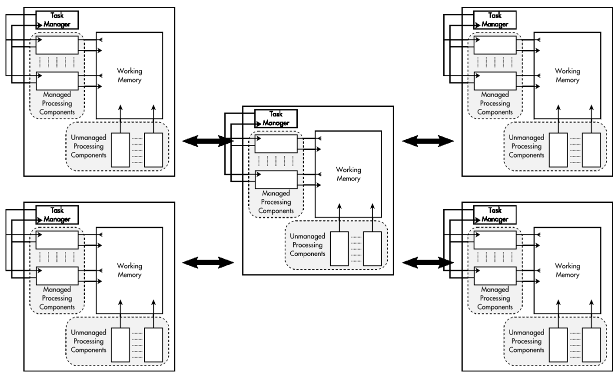
Outline

The CoSy Architecture Schema (CAS)

- A schema which defines a limited space of architectures and thus instantiations.
- Based on *requirements* drawn from an analysis of robotic scenarios, and common solutions in implemented systems.
- General enough for experimentation, specific enough to study design commitments.

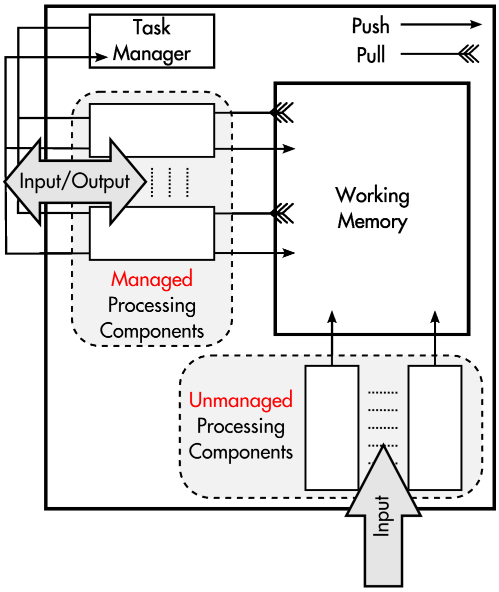
CAS Key Features

- Collection of loosely coupled subarchitectures.



CAS Key Features

- Collection of loosely coupled subarchitectures.
- Each subarchitecture contains processing components that update structures within a working memory (WM).
- Components can read all WMs but only write to the local WM (bar privileged components).
- Processing is controlled by a network of task managers.



CAS in Context

- CAS makes practical use of approaches from cognitive systems.
  - Shared working memories.
  - Management methods for components.
- . . . whilst attempting to formalise common practice in robot systems.
  - Multiple concurrent components.
  - Distributed design.
- We are motivated by cognition, although we are not aiming for human-like systems.

CAST: The CAS Toolkit

- 2-layer toolkit: BALT for communication, CAST implements CAS on top.
- Cross language, distributed design, open source, multi-OS. Supports incremental development.
- Biggest system about 30 components running on 5 machines.
- Key contribution: separation of architecture from content.

Outline

Architectures for Integration

- Over the last two years we have iteratively constructed systems for HRI in a table-top manipulation scenario.
- Each iteration has allowed us to further explore issues in integrated systems, architectures, binding, filtering etc.
- Iterations:
    1. Tutor-driven learning of visual properties.
    2. Language-driven manipulation.

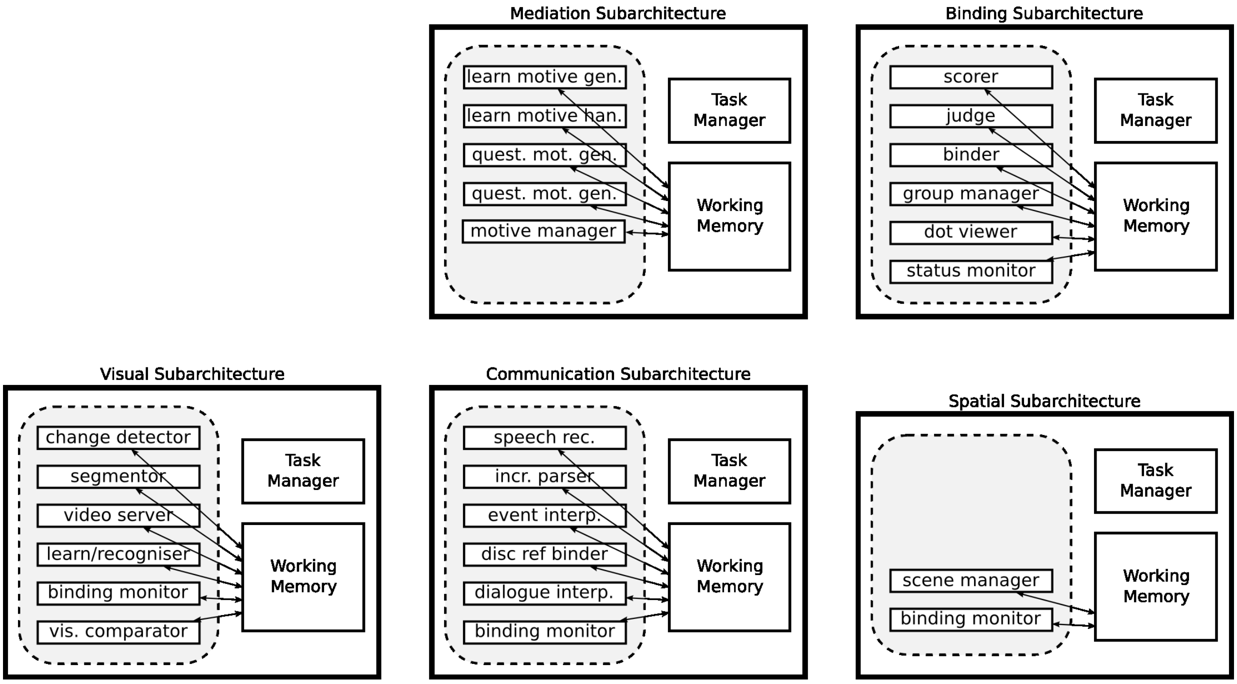
Tutor-Driven Learning of Visual Properties

Features

- 2 learning modes:
    - Tutor Driven: Learning task generated via language input.
    - Tutor Supervised: Learning task generated via visual input.
- Spatial WM: Stack of frames of objects in scene, quantitative to qualitative abstraction.
- Mediation: Raises learning goals, posts goals to visual and language SAs.
- Binding SA: Binding linguistic information to visual and spatio-temporal information to generate modality-neutral representations.

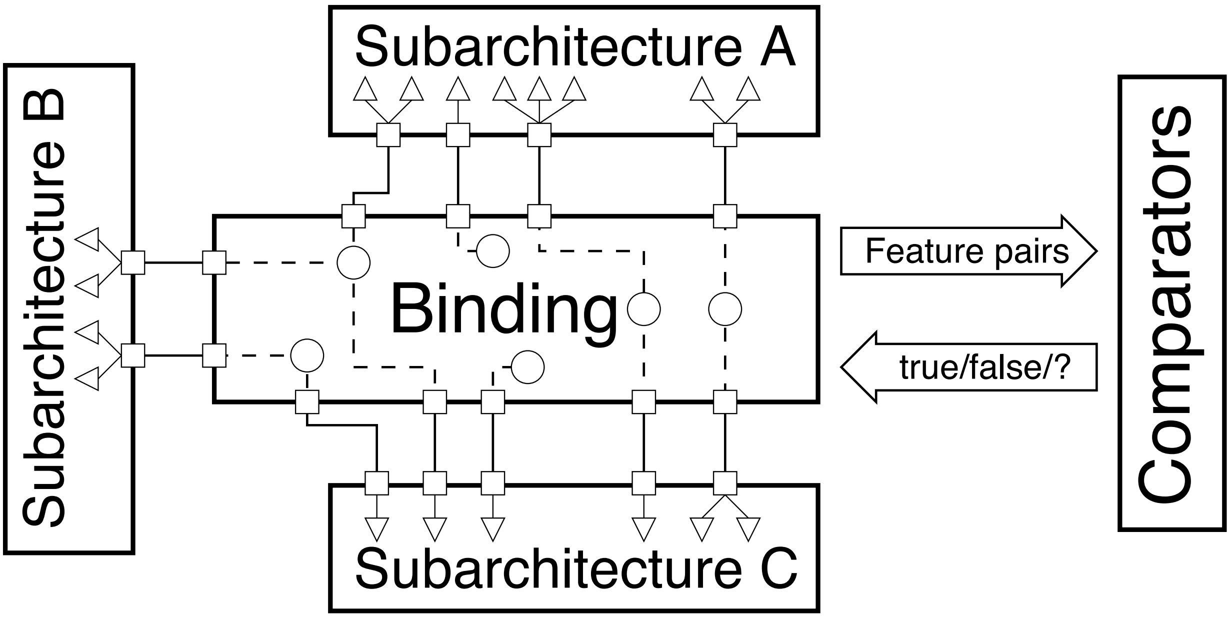
Tutor-Driven Learning of Visual Properties

Instantiation

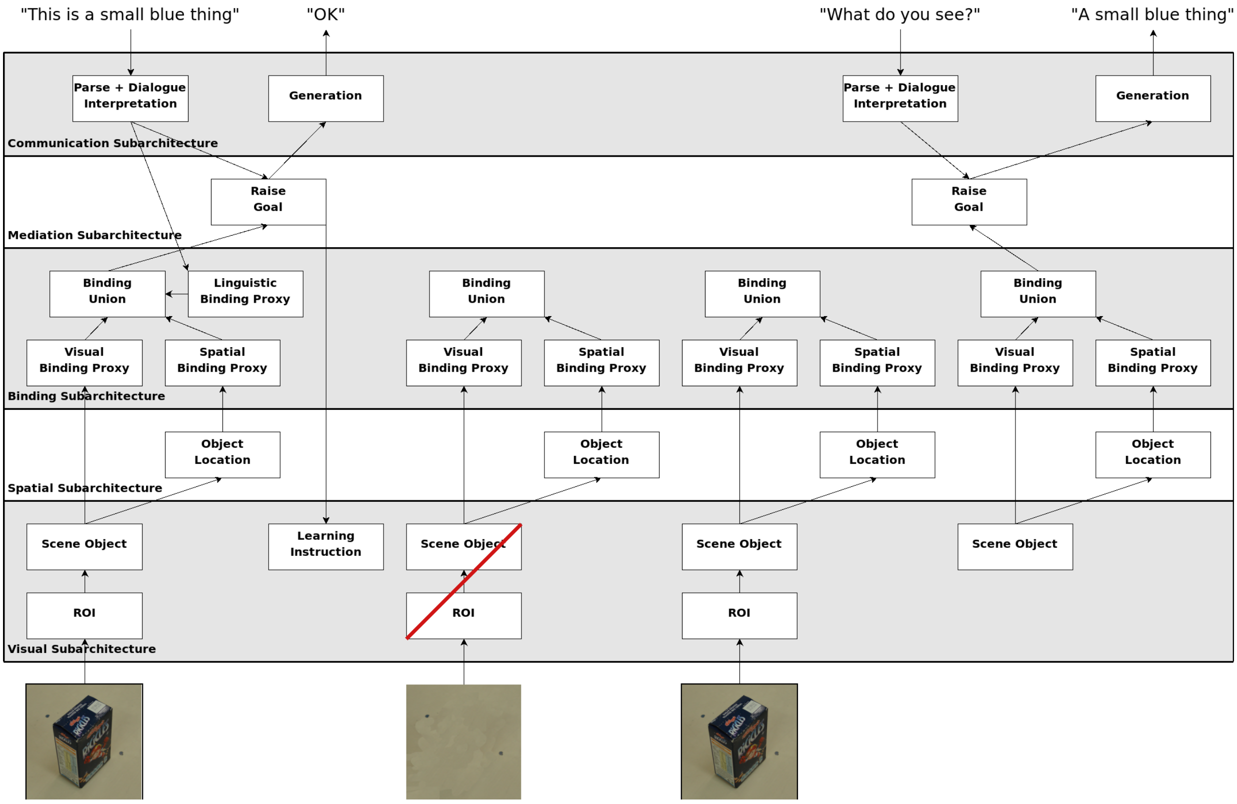


Tutor-Driven Learning of Visual Properties

Binding



Timeline



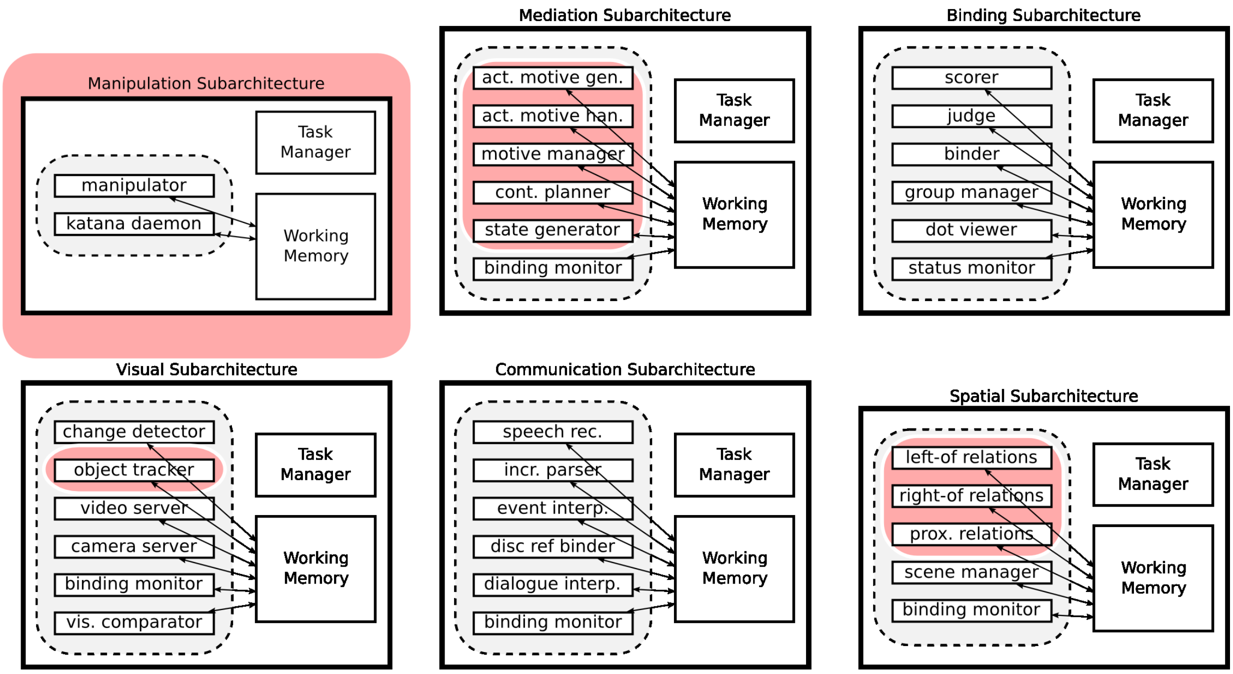
Language-Driven Manipulation

Features

- Goals are raised by language.
- References are made to objects using previously learned features.
- Robot plans intentional actions using a symbolic planner.
- Intention shifting is handled via execution monitoring and continual planning.
    - Symbolic state generated from binding features at regular intervals.
    - Current state checked against expectations during execution.
    - Feedback from manipulator checked during execution.

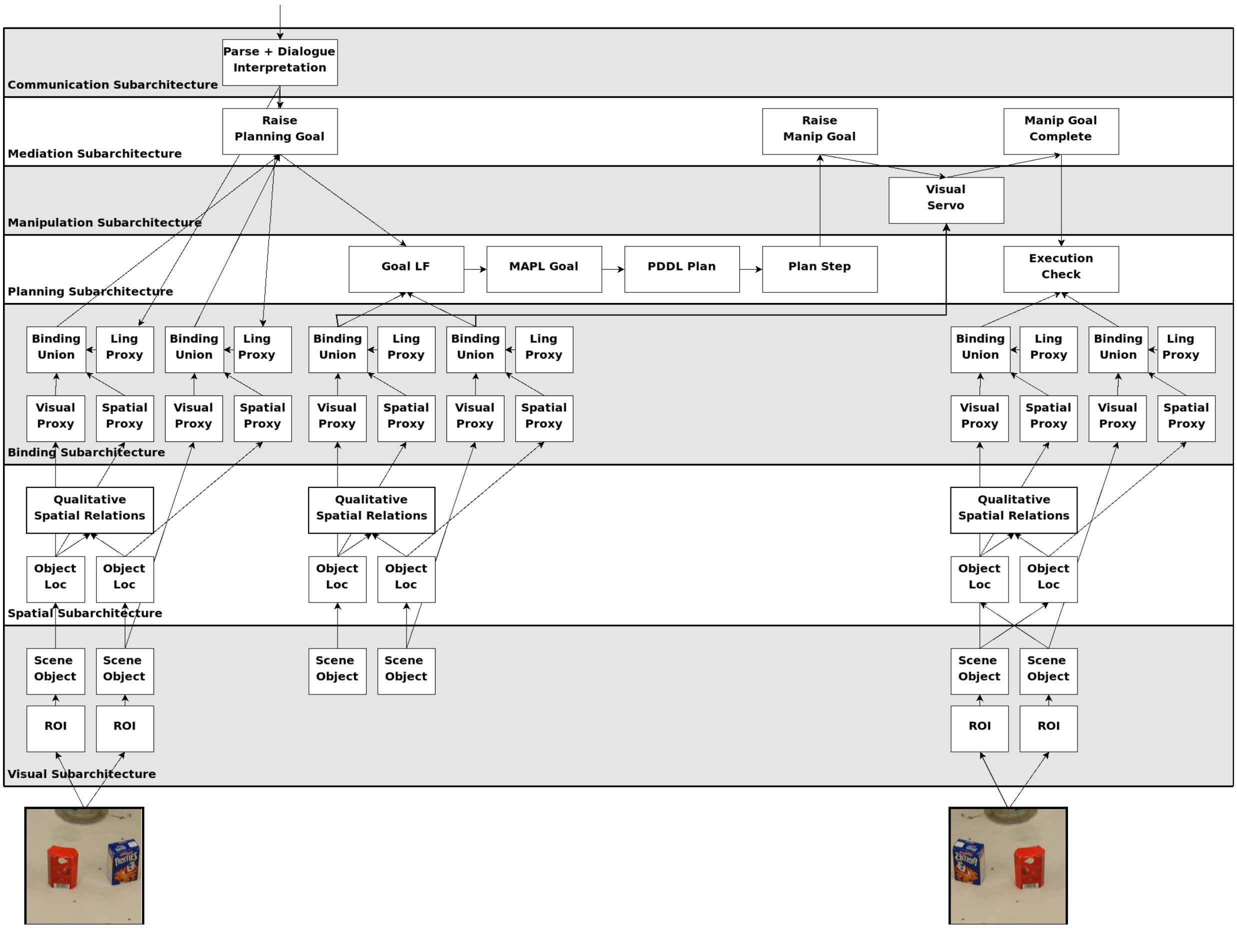
Language-Driven Manipulation

Instantiation



Timeline

"Put the blue thing to the left of the red thing"



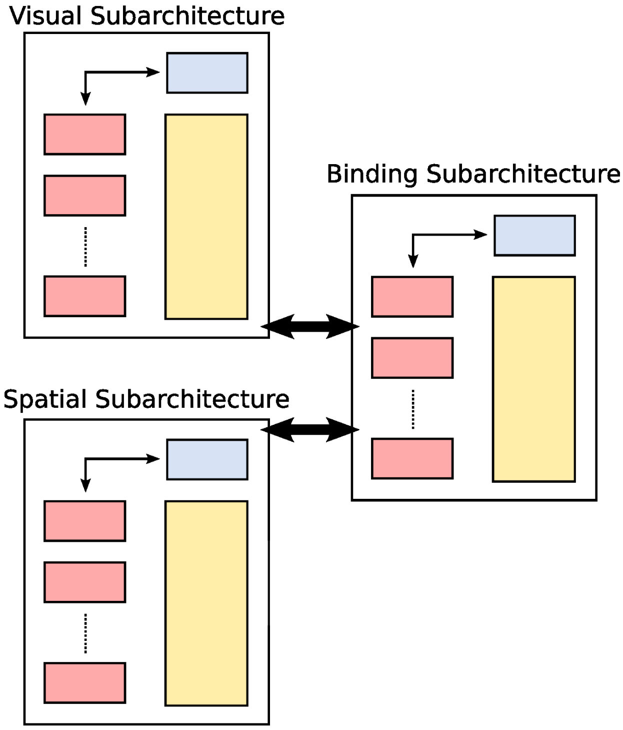
Outline

Exploring Design Space

- Given our stated aim of *understanding* systems, building them is not enough.
- Can we use CAST to explore trade-offs in architectural design space?
- Yes!
- Methodology: Build systems that represent different points in design space and measure various properties about them to characterise trade-offs.
- Investigate: Cost of communication and filtering at three points in design space.

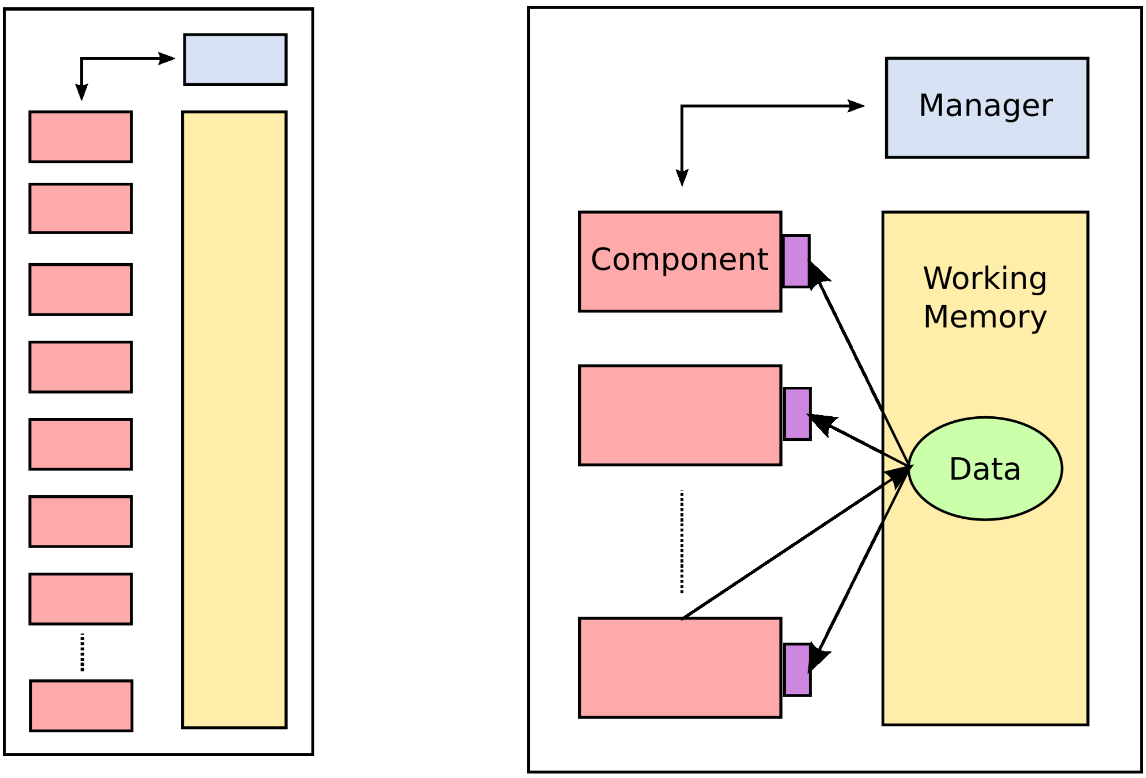
Three Schema Instantiations

N components : 1 subarchitecture



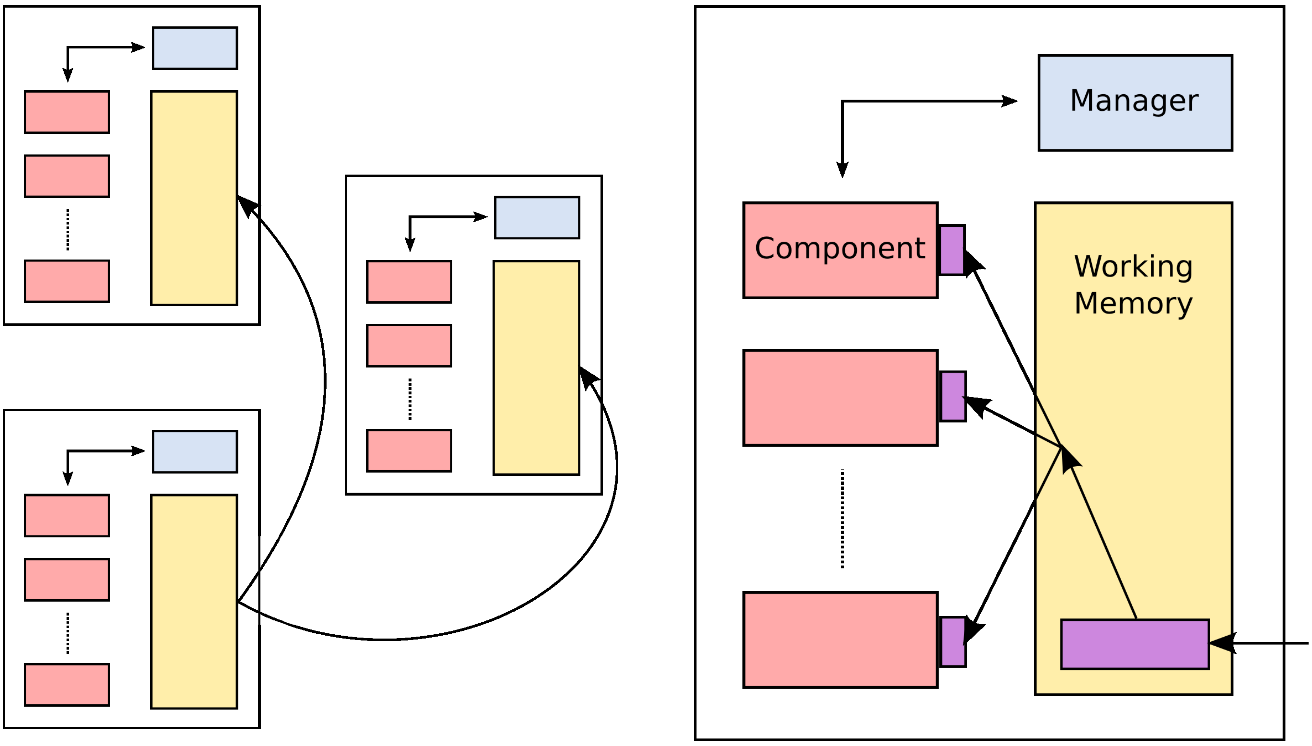
Three Schema Instantiations

N components : 1 subarchitecture



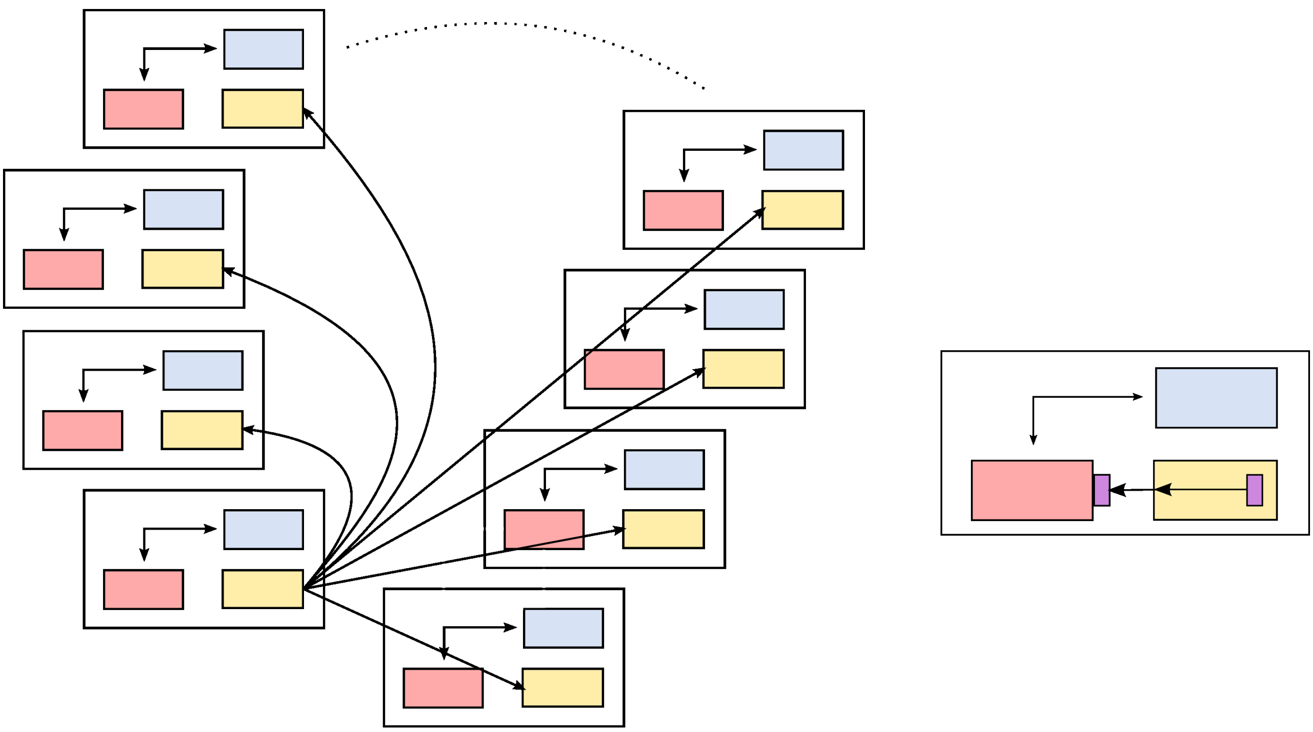
Three Schema Instantiations

N components : M subarchitectures (N > 1)



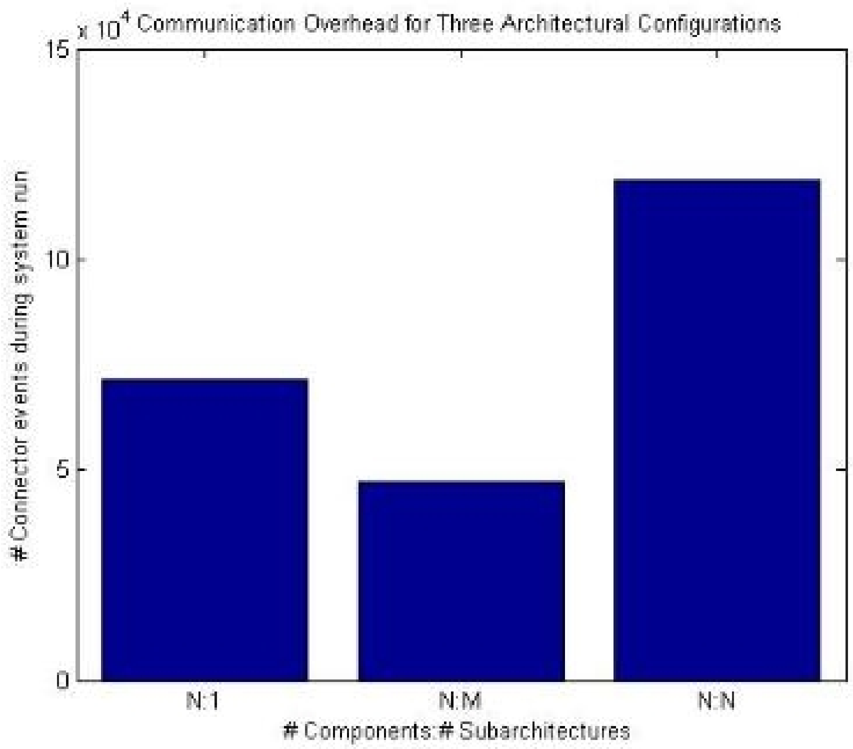
Three Schema Instantiations

N components : N subarchitectures



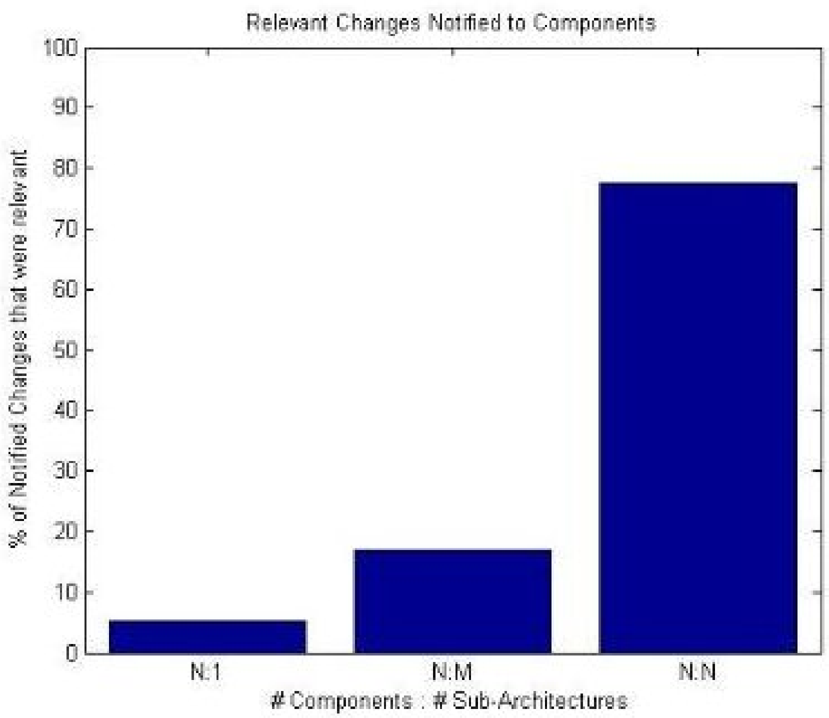
Results

Communication Overhead

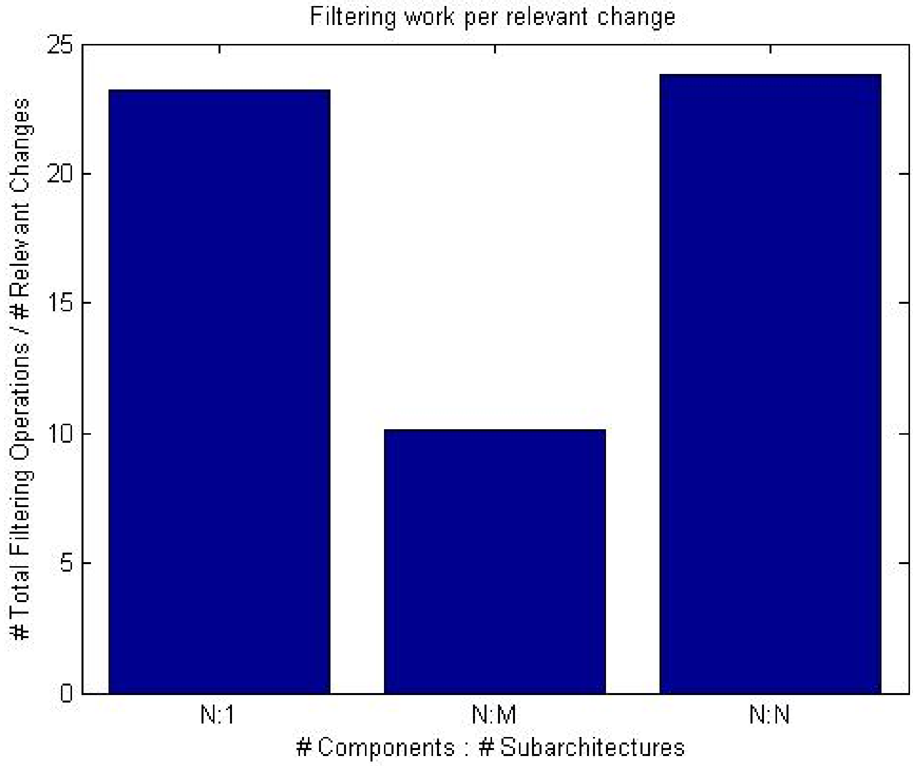


Results

Filtering Relevance



Results
Filtering Relevance



Results Summary

- N:M forms a sweet spot in the space of architectures we explored
- Better for:
  - Communication overhead.
  - Filtering work required to identify relevant information
- This is robust with changes in scene complexity and system complexity

Outline

Conclusion

- The CoSy Architecture Schema defines a limited space of possible architectures, allowing us to explore this space in a principled manner.
- A number of CAS instantiations have been implemented for HRI scenarios.
- These instantiations have allowed us to explore approaches to cross-modal binding and aspects of architectural design space.
- All implementations are based on our CAS toolkit. This is available as open source code: http://www.cs.bham.ac.uk/research/projects/cosy/cast/

Acknowledgements

- Birmingham: Jeremy Wyatt, Aaron Sloman, Michael Zillich, Marek Kopicki, Somboon Hongeng.
- DFKI, Saarbrucken: Henrik Jacobsson, Geert Jan Kruijff, John Kelleher (now at Dublin Inst. Tech).
- Albert-Ludwigs-Universität, Freiburg: Michael Brenner.
- University of Ljubljana: Danijel Skočaj, Gregor Berginc

Motivation The CoSy Architecture Schema Illustrations Experiments Conclusion

The End

Questions?