

# MACS

## A framework for developmental learning of Affordances

MACS 3<sup>rd</sup> Review

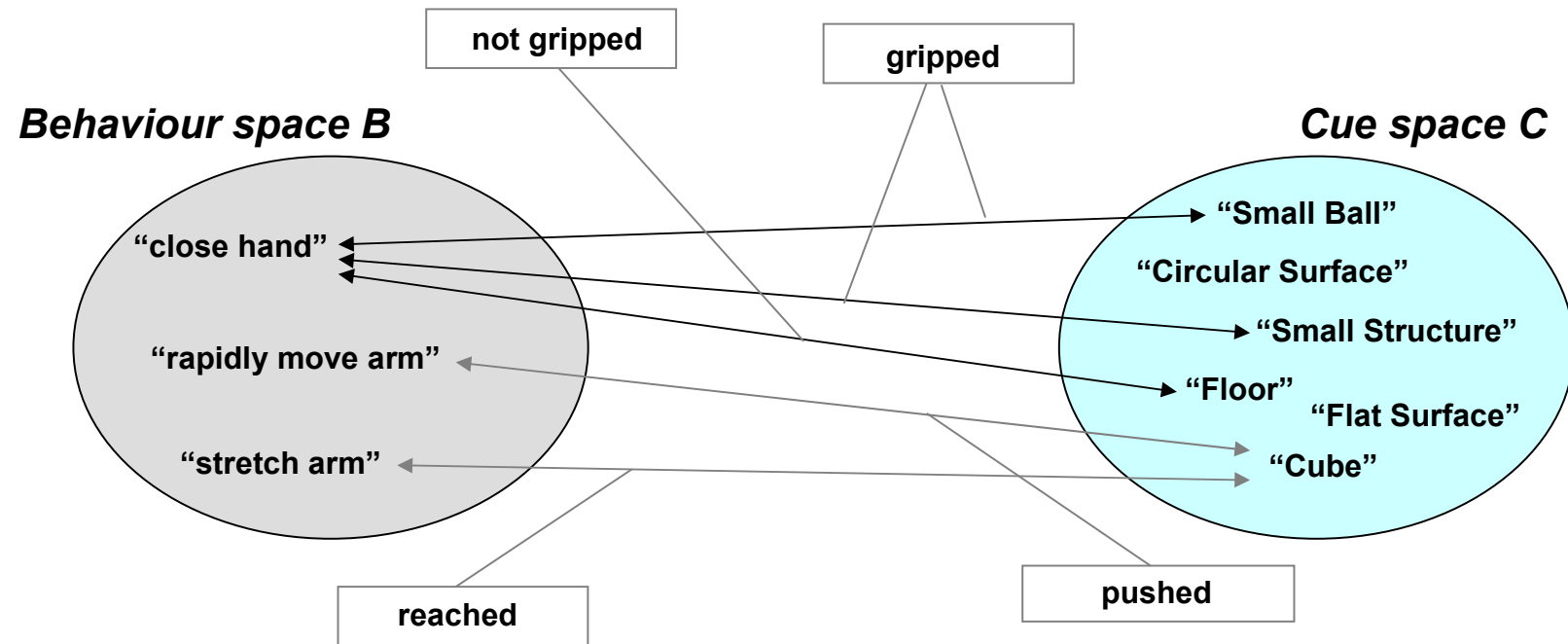
Sankt Augustin, 15<sup>th</sup> Feb 2008

Florian Kintzler, Jörg Irran, Georg Dorffner  
(OFAI)

- Learning of Affordances – Concept
- Behaviour-Monitoring/Self Observation (LM, ESGM and PM)
- Learning-Algorithm (LM)
  - Clustering of data from one channel
  - Finding correlating channels
  - Re-Learning
- Using the derived knowledge:
  - Generating Filters – (ESGM and ARR)
  - Using the generated filters (ESGM, ARR, DM and BM)
- Connection to Cue-Learning by JR\_DIB (LM, PM)
- Outlook

# Affordance

## Initial Idea – Learning Perspective

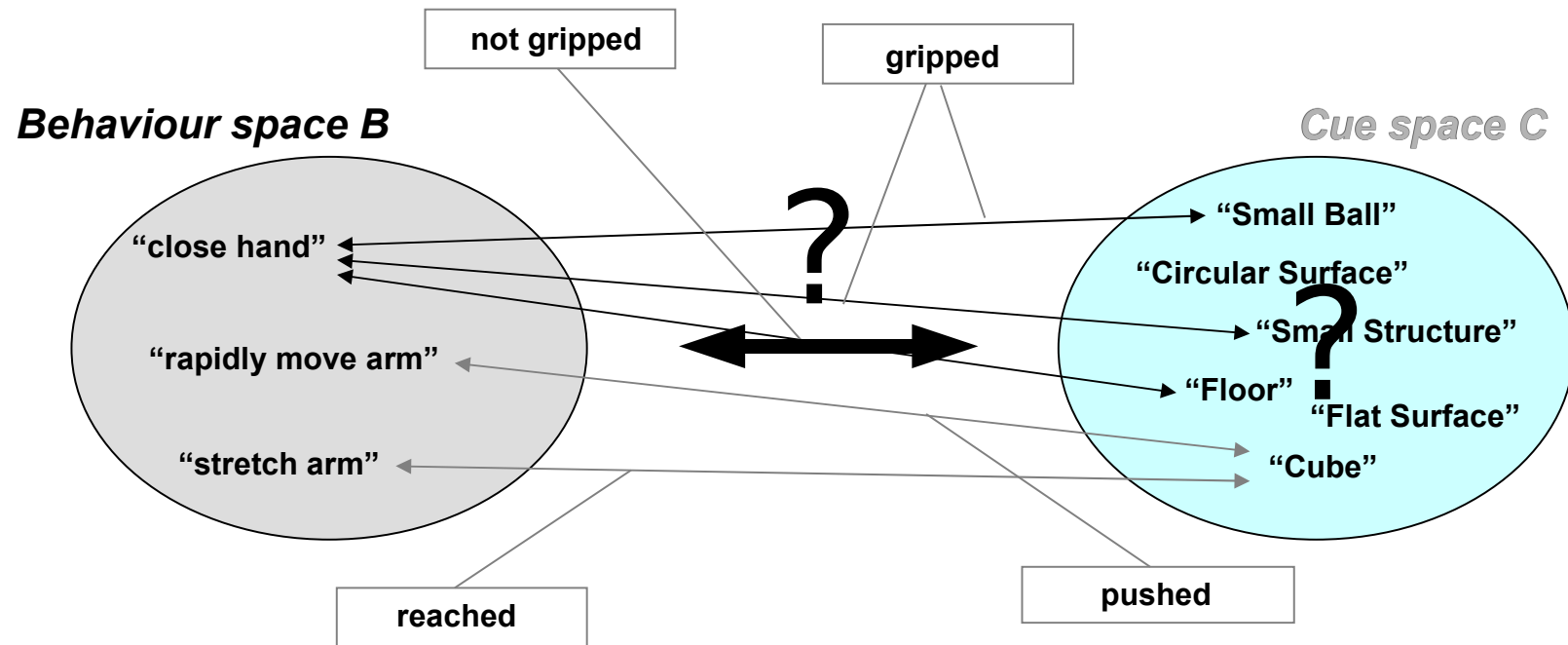


“The *affordances* of the environment are what it *offers* the animal, what it *provides* or *furnishes*, either for good or ill. [...]

I mean by it [affordances] something that **refers to both** the environment and the animal.” (Gibson, 1986)

# Affordance

## Agents View

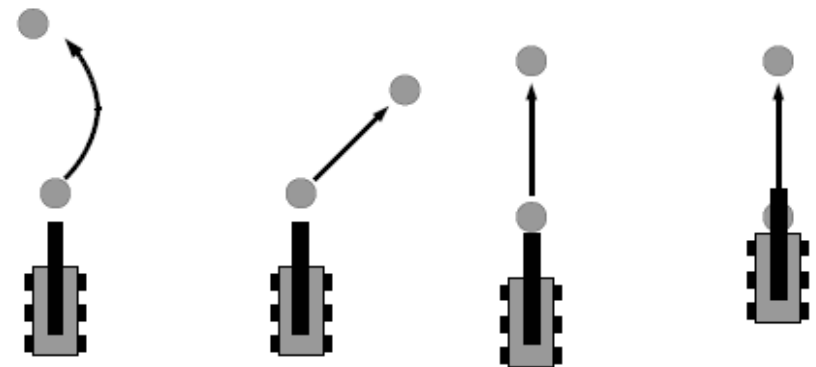
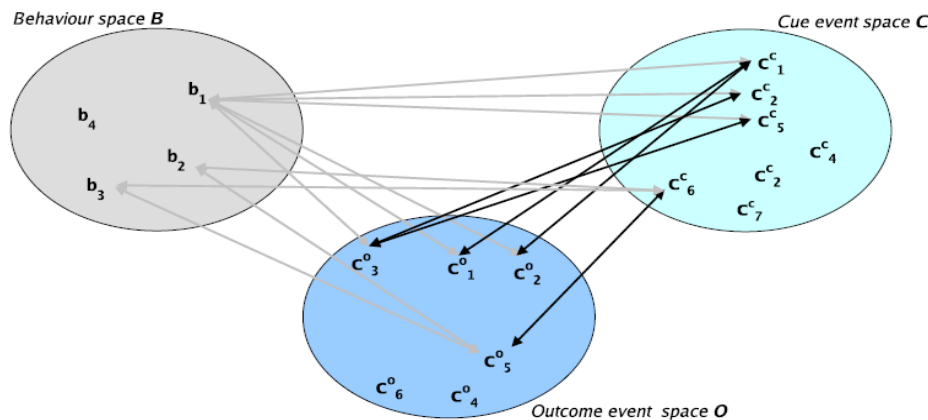


- Behaviours, Outcomes, Cues
- There is no meaning of one without the other

# Learning of Affordances

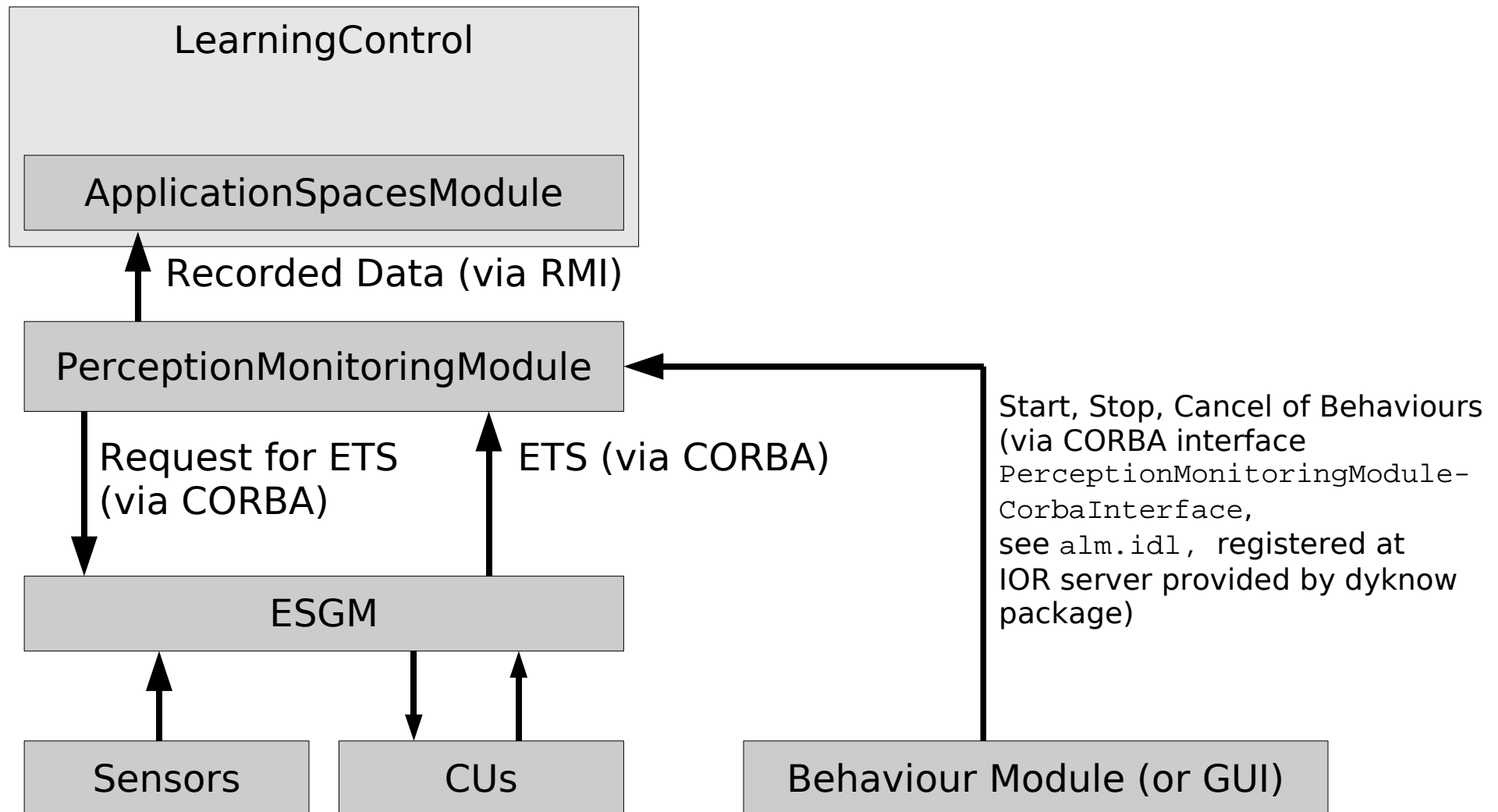
## Concept - Observing Own Behaviour

- One behaviour can cause multiple outcome events
- Multiple behaviours can cause similar outcome events
- One cue can indicate the possibility to apply multiple behaviours
- Multiple cues can indicate the possibility to apply one behaviour

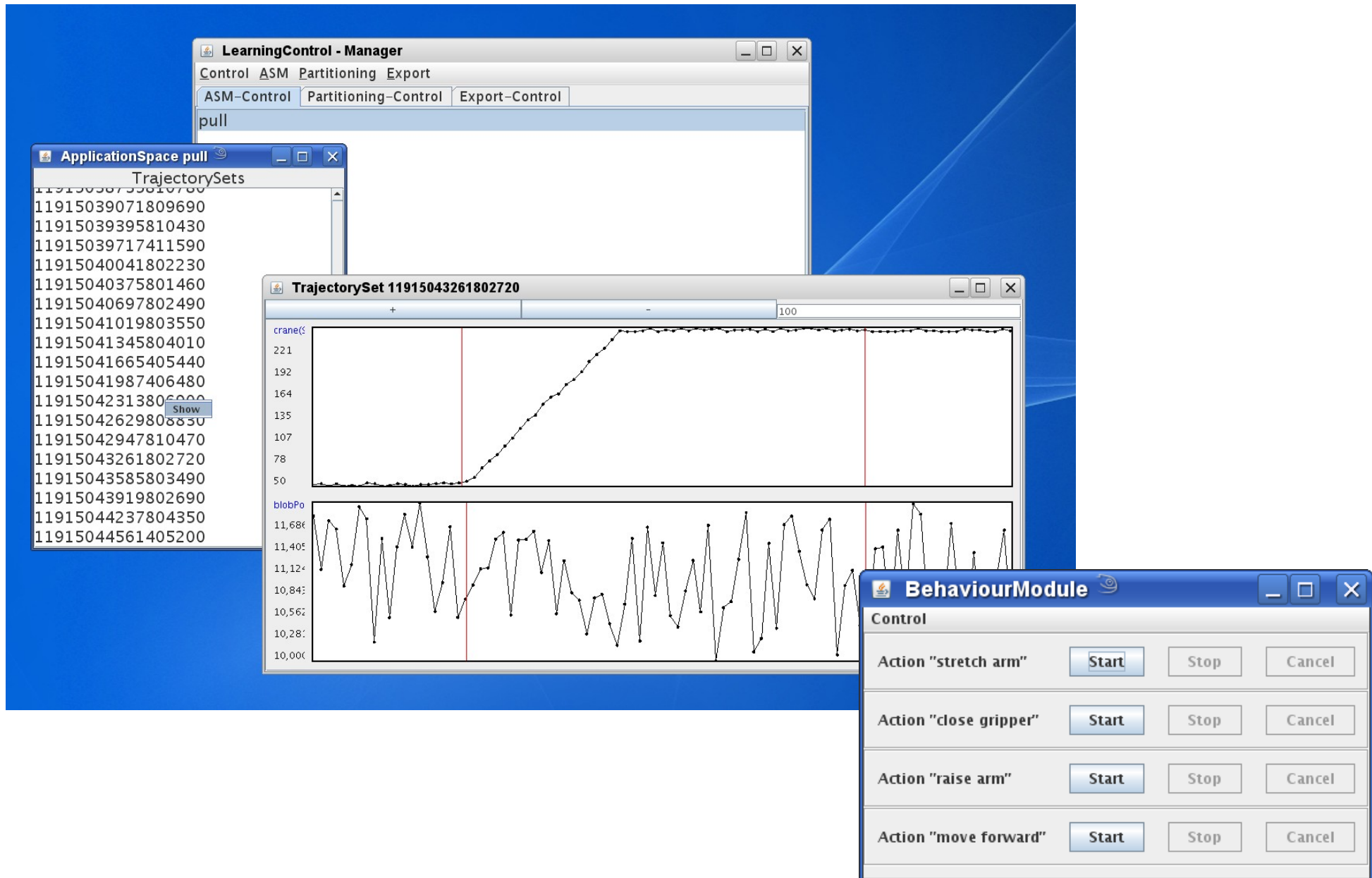


# Monitoring of Perception and Behaviour

## Application Flow



# Monitoring of Perception and Behaviour GUI



# Monitoring of Perception and Behaviour

## Implementation – Problems and Solution

- Sun JDK-ORB, IBM JDK-ORB, OpenORB are not compatible with ESGM
- JacORB compiles the idl files, but the class *AttributeValuePair* is empty.
- Added the following lines to class *AttributeValuePair*:
 

```
public String name;
public Value val;
```
- Added the following lines to method `read(...)` in class *AttributeValuePairHelper* (method `write` analogue):
 

```
result.name=in.read_string();
result.val=ValueHelper.read(in);
```
- Helper classes (like `SubscriptionProxy`) that are provided in native C/C++ were re-implemented



# Partition an Application-Space

## Clustering of Data from One Sensor Channel

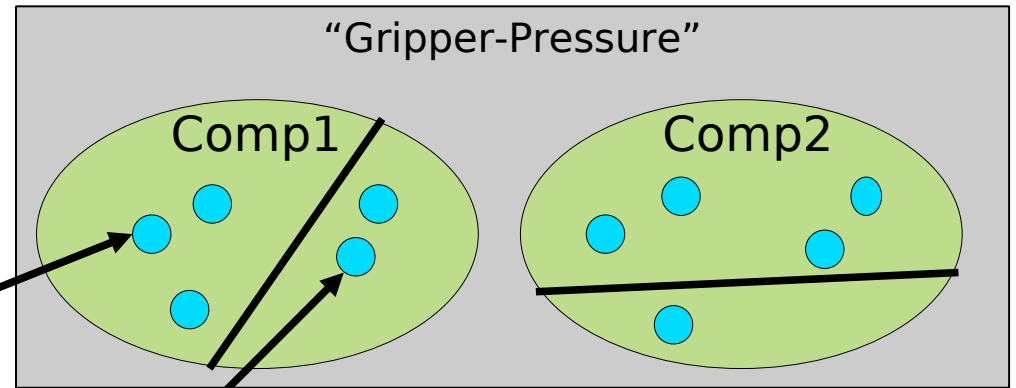
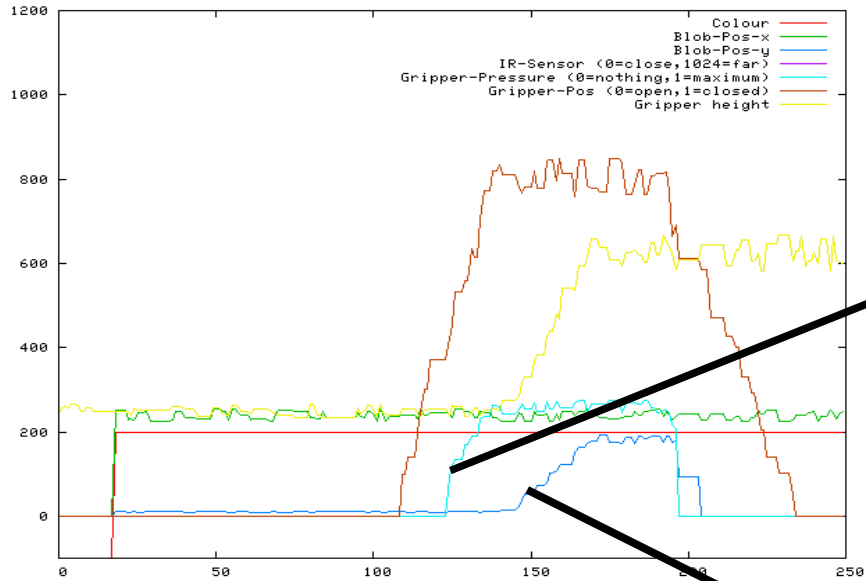
- Learning Approach described in D5.3.1

$$\begin{aligned}
 S_j^o(a_i) &= S(\{t[o](a_i) \mid t(a_i) \in P_j(a_i)\}) \\
 &= S(\{t[State_{t1}, State_{t2} + \delta_{t2}](a_i) \mid t(a_i) \in P_j(a_i)\}) \\
 &= \{s_1^o[P_j(a_i)], s_2^o[P_j(a_i)] \dots\}, \\
 &\quad \text{for each partition } j, 0 \leq i < N
 \end{aligned}$$

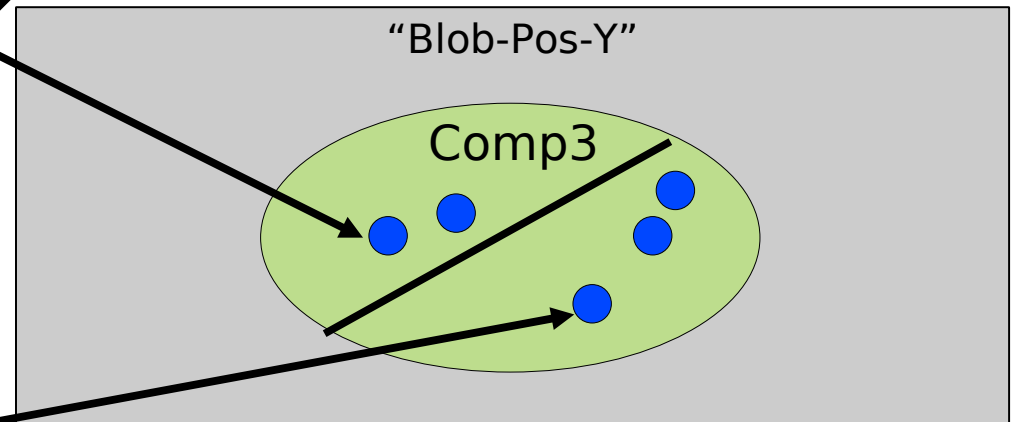
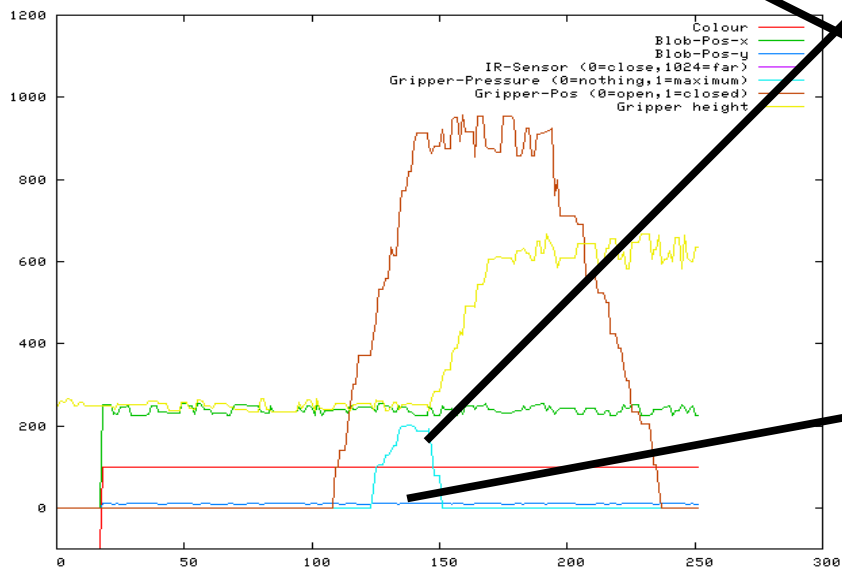
- Procedure is implemented data type independent
- New types of data can be handled by providing
  - the data type  $T$
  - Comparator(**s!**) to compare time series consisting of data of data type  $T$
  - GUI components for displaying (optional)

# Partition an Application-Space

## Clustering of Data from One Sensor Channel

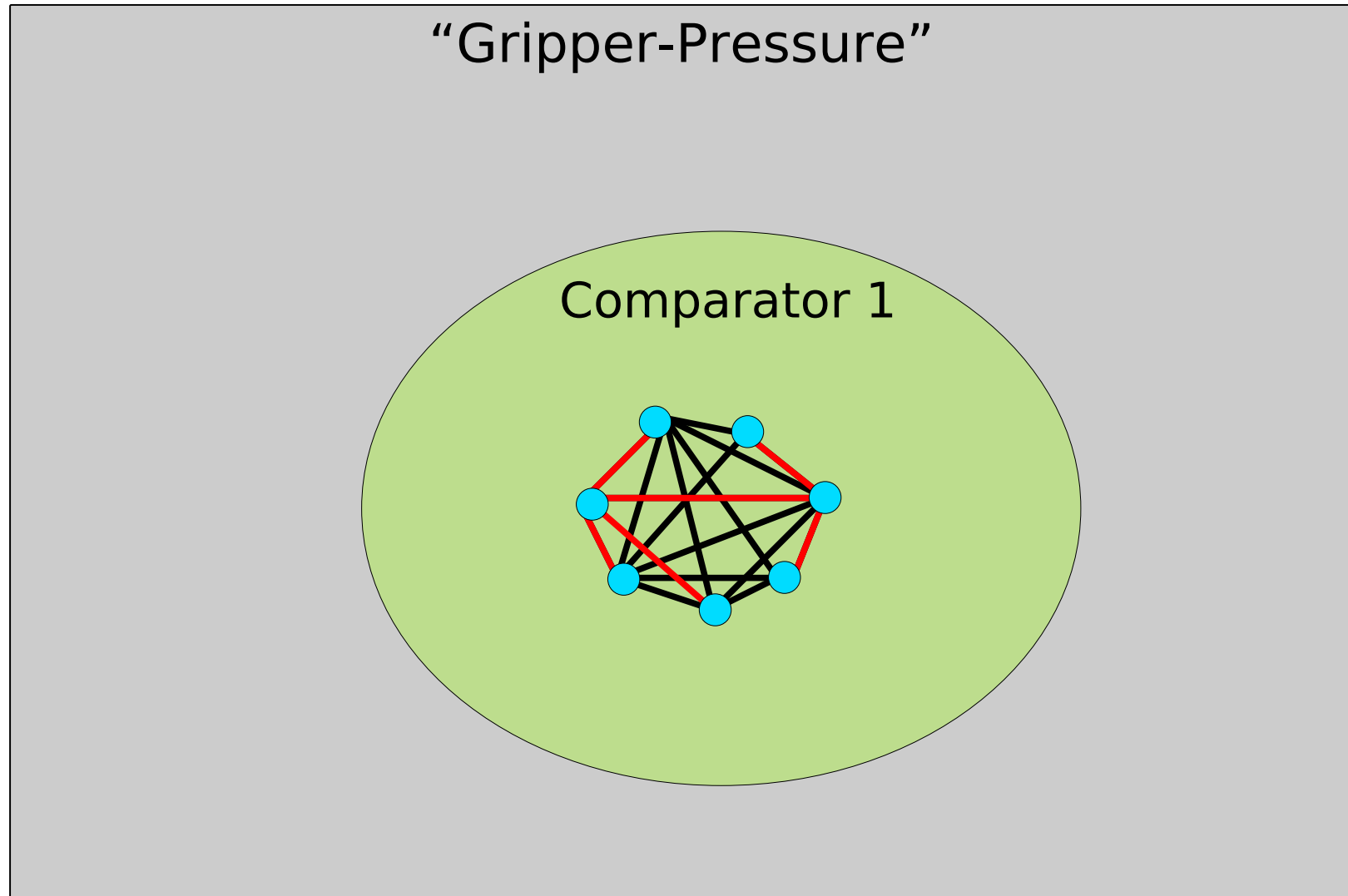


■ ■ ■



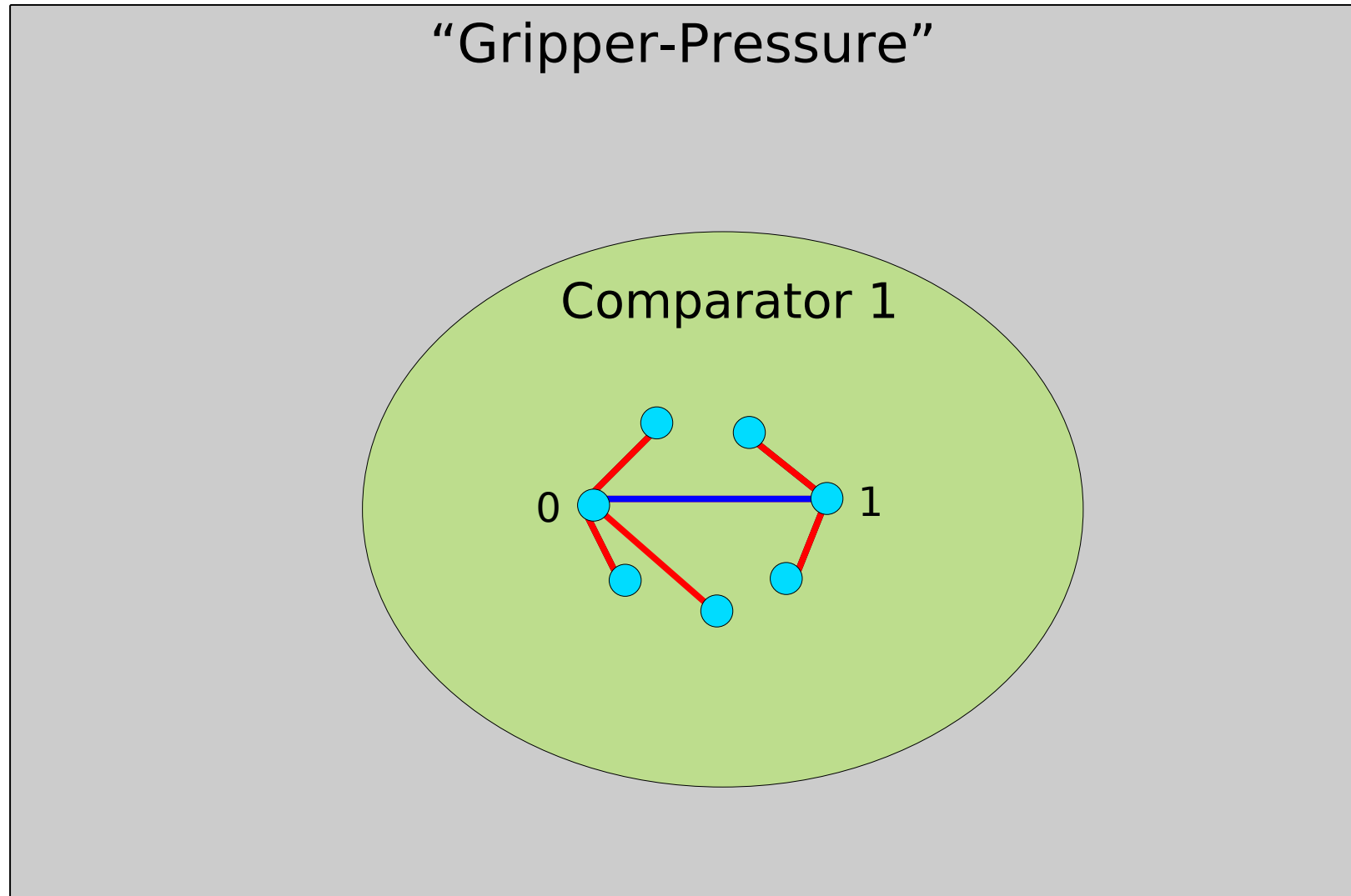
# Partition an Application-Space

## Clustering of Data from One Sensor Channel



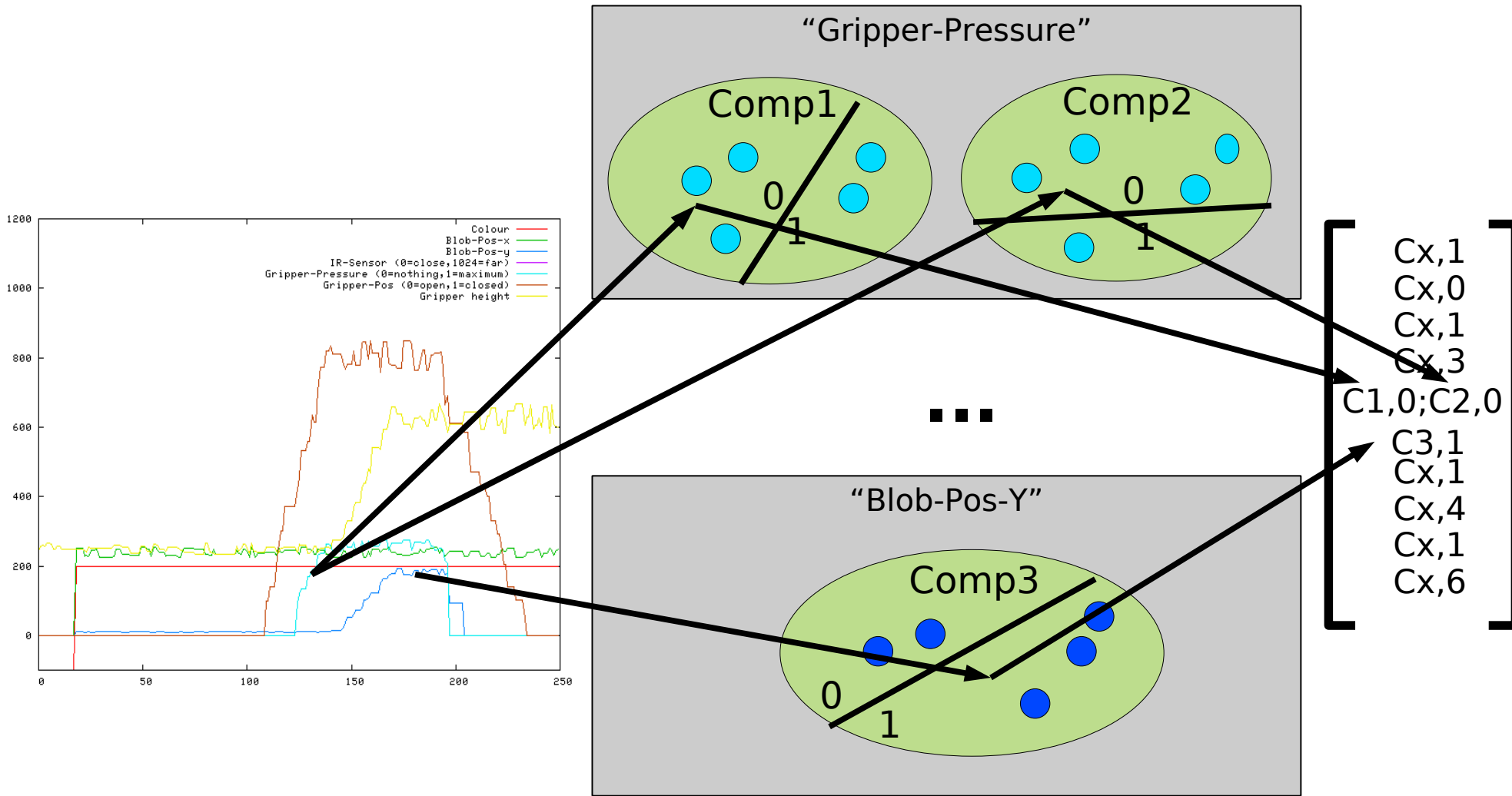
# Partition an Application-Space

## Clustering of Data from One Sensor Channel



# Partition an Application-Space

## Clustering of Data from One Sensor Channel



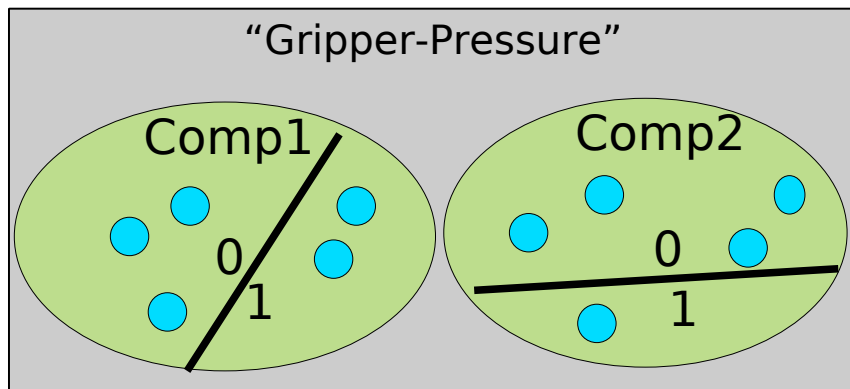
# Partition an Application-Space

## Finding Correlating Channel-Clusters

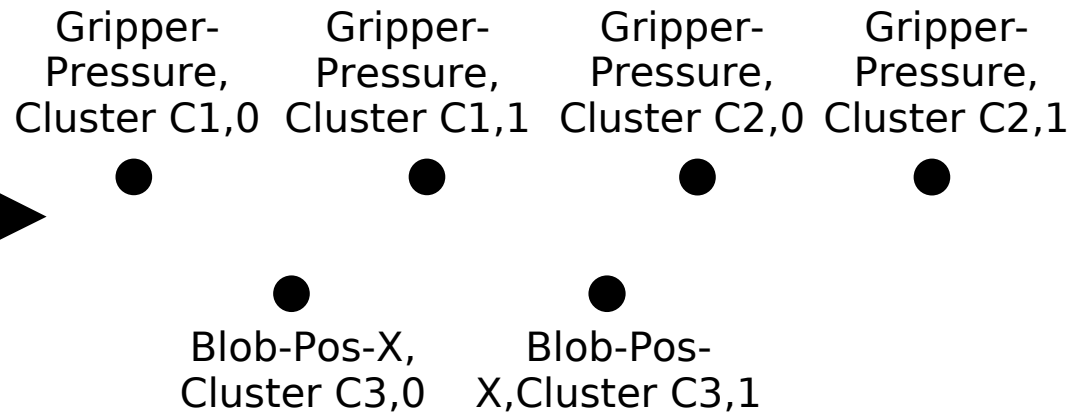
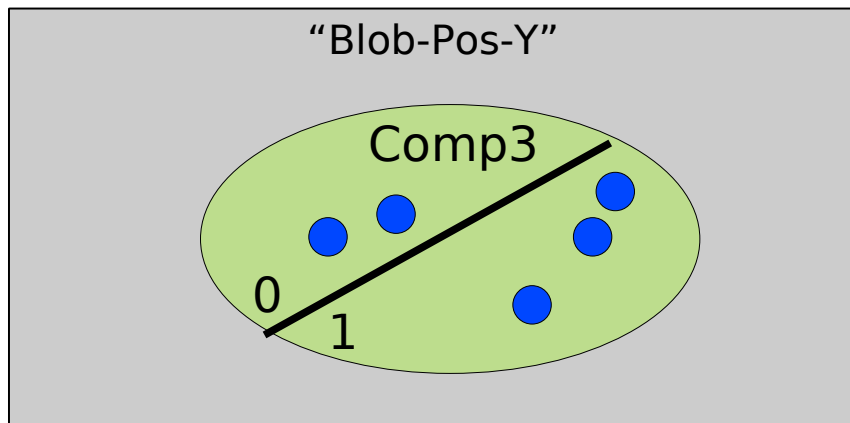
- Transfers all available time series sets into cluster-vectors
- Tries to find correlations between the clusters, i.e. matches like “whenever the data of channel A belongs to cluster  $C_{x,y}$  the the data of channel B belongs to cluster  $C_{z,r}$ ” (can also include or-relations)
- Implementation is thus data type independent
- Can be used independent of the concrete implementation of the clustering of data channel specific data

# Partition an Application-Space

## Finding Correlating Channel-Clusters

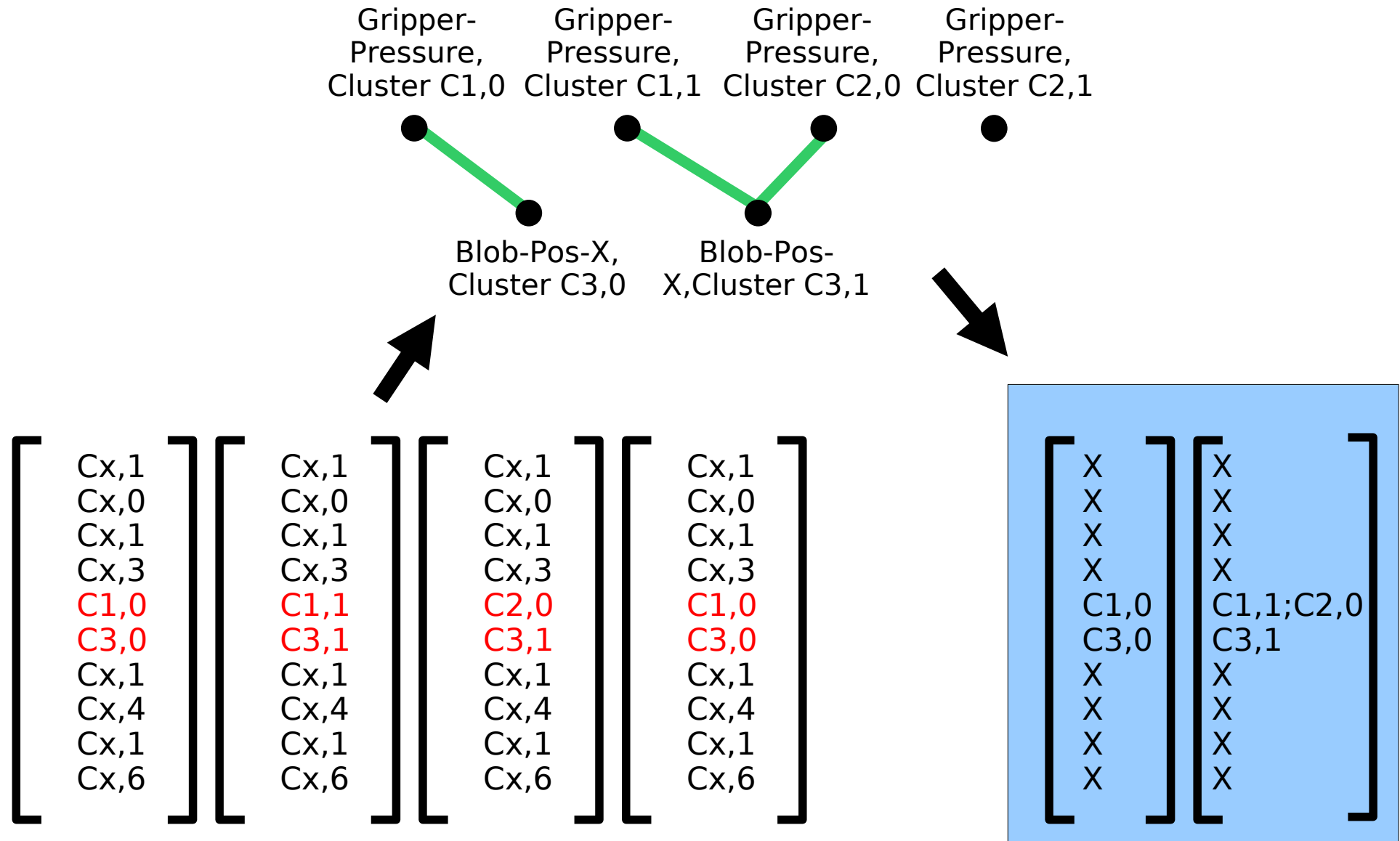


...



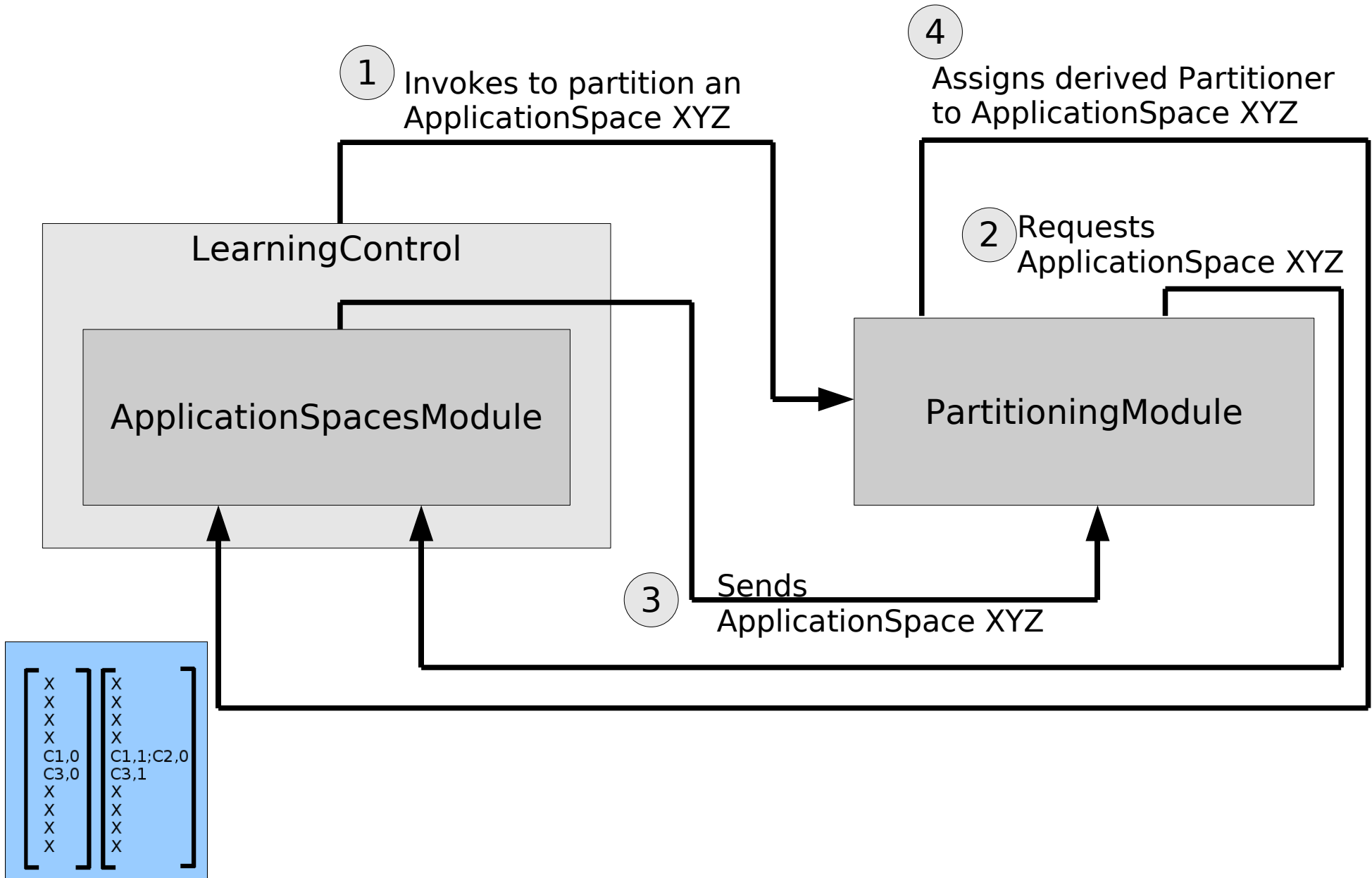
# Partition an Application-Space

## Finding Correlating Channel-Clusters



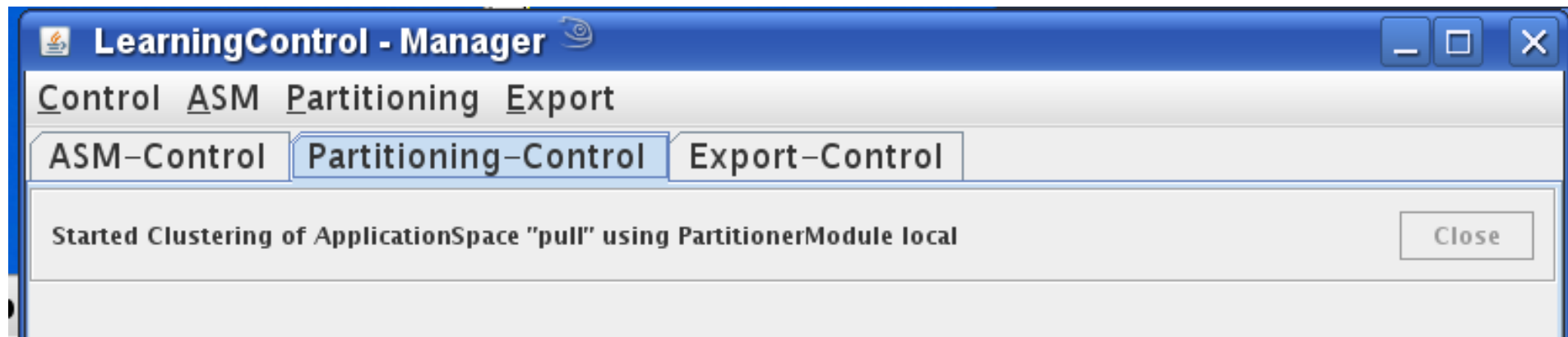
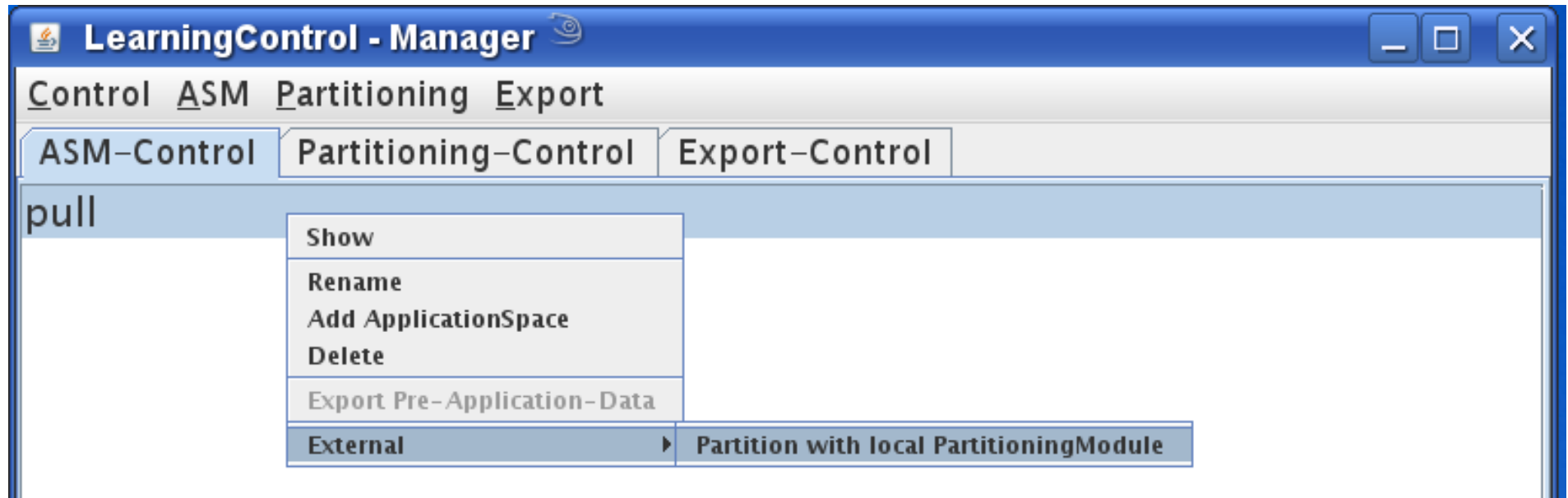


# Partitioning Application Flow

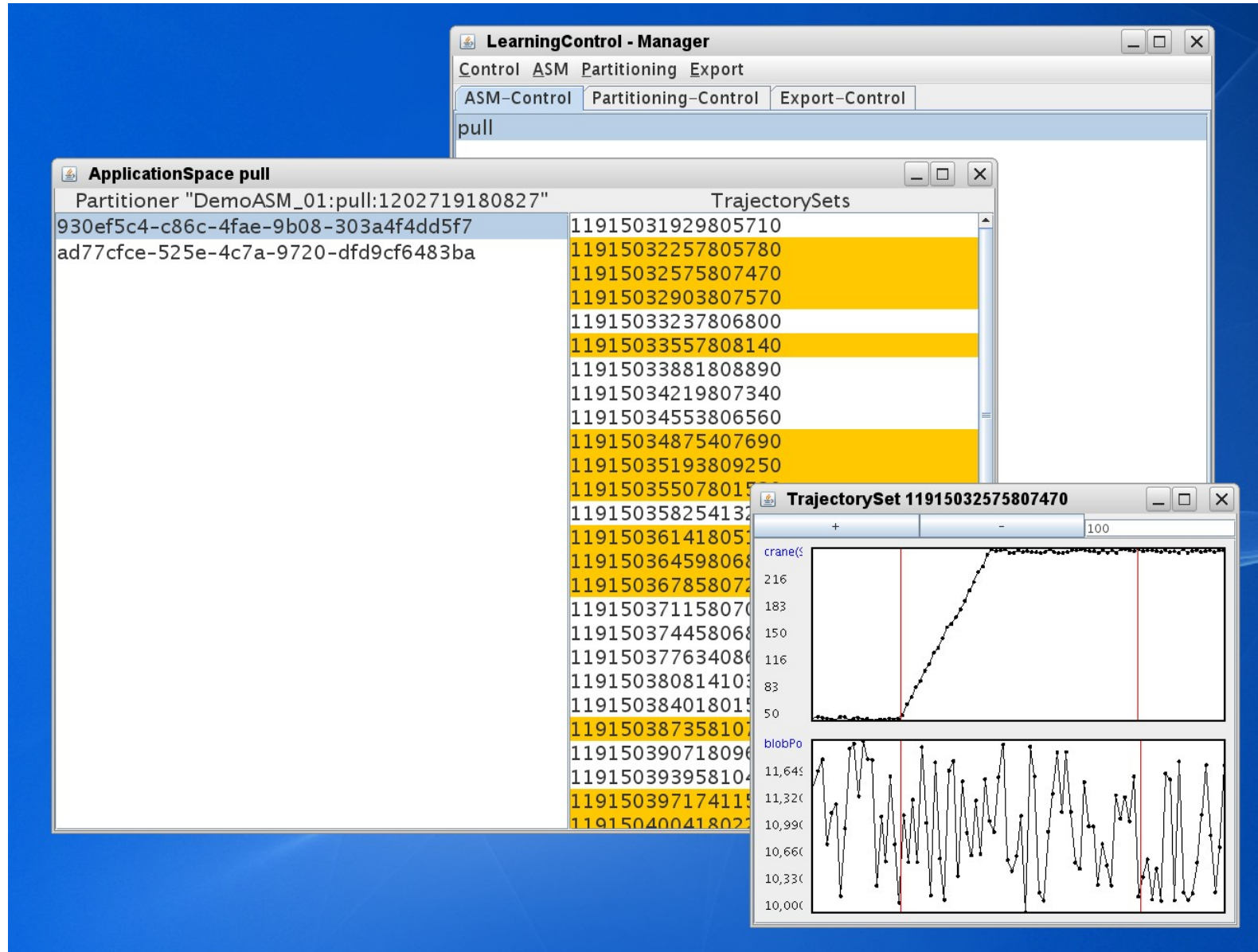


# Partitioning

## Clustering – Start via GUI

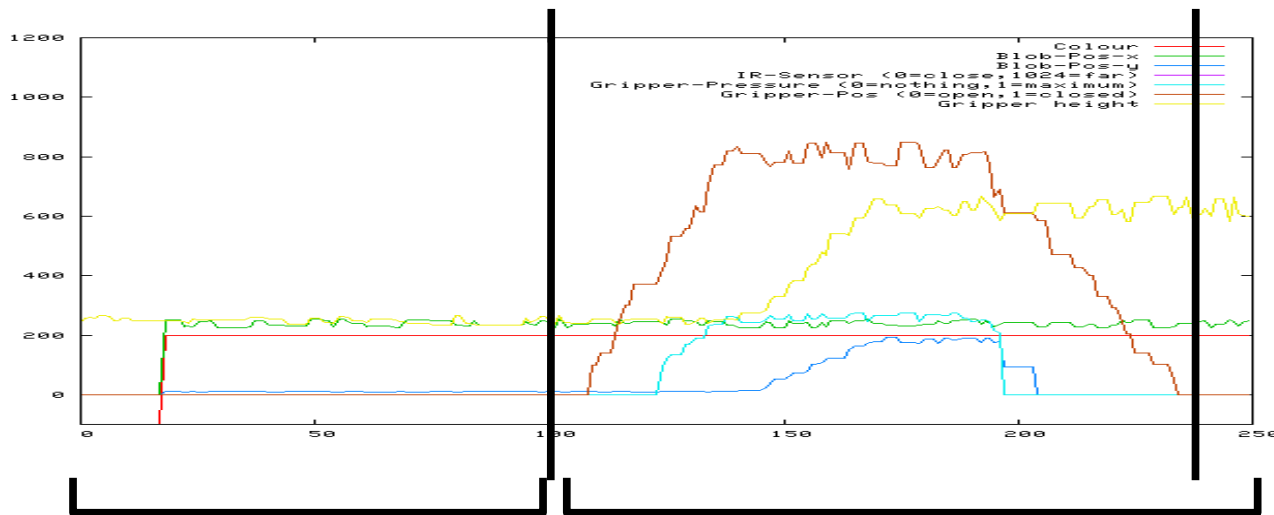
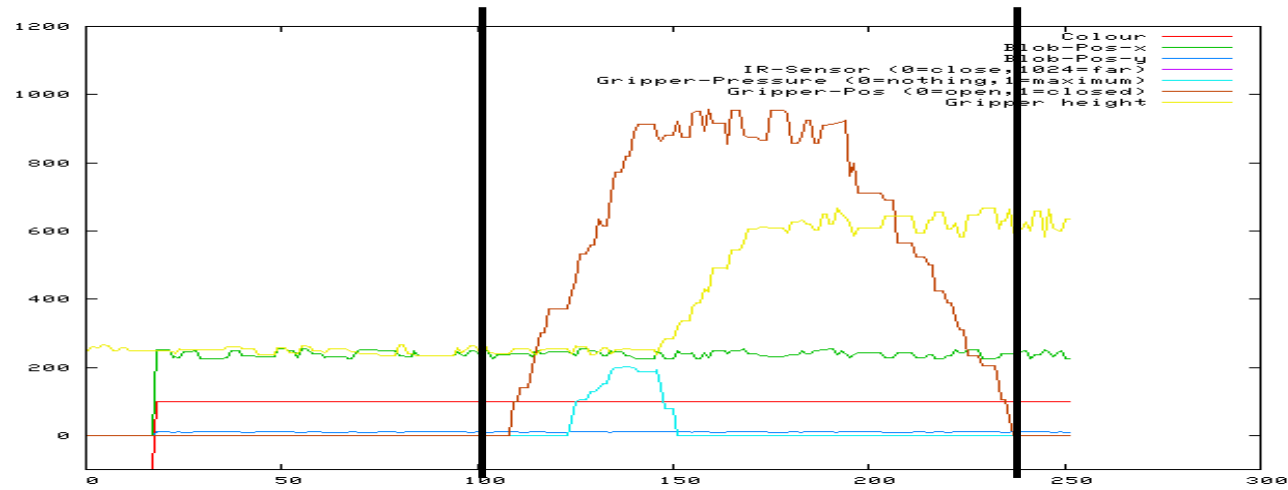


# Partitioning Clustering – Result: Partitioner



# Partitioning

## Clustering for deriving Outcomes and Cues

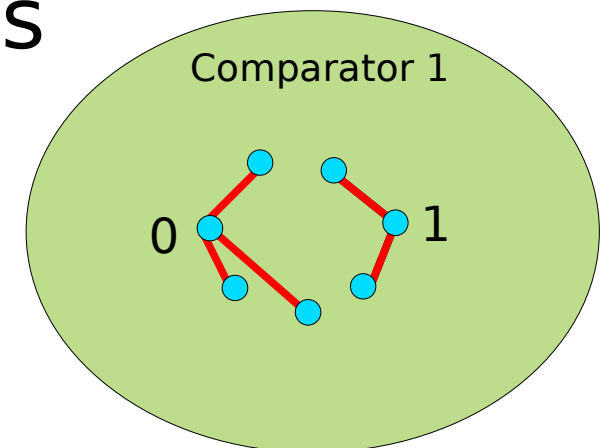


Pre-Application-Phase  
used for extraction  
of relevant cue channels (Step 2a)

Application- and Post-ApplicationPhase  
used for extraction  
of relevant outcome channels (Step 3a)

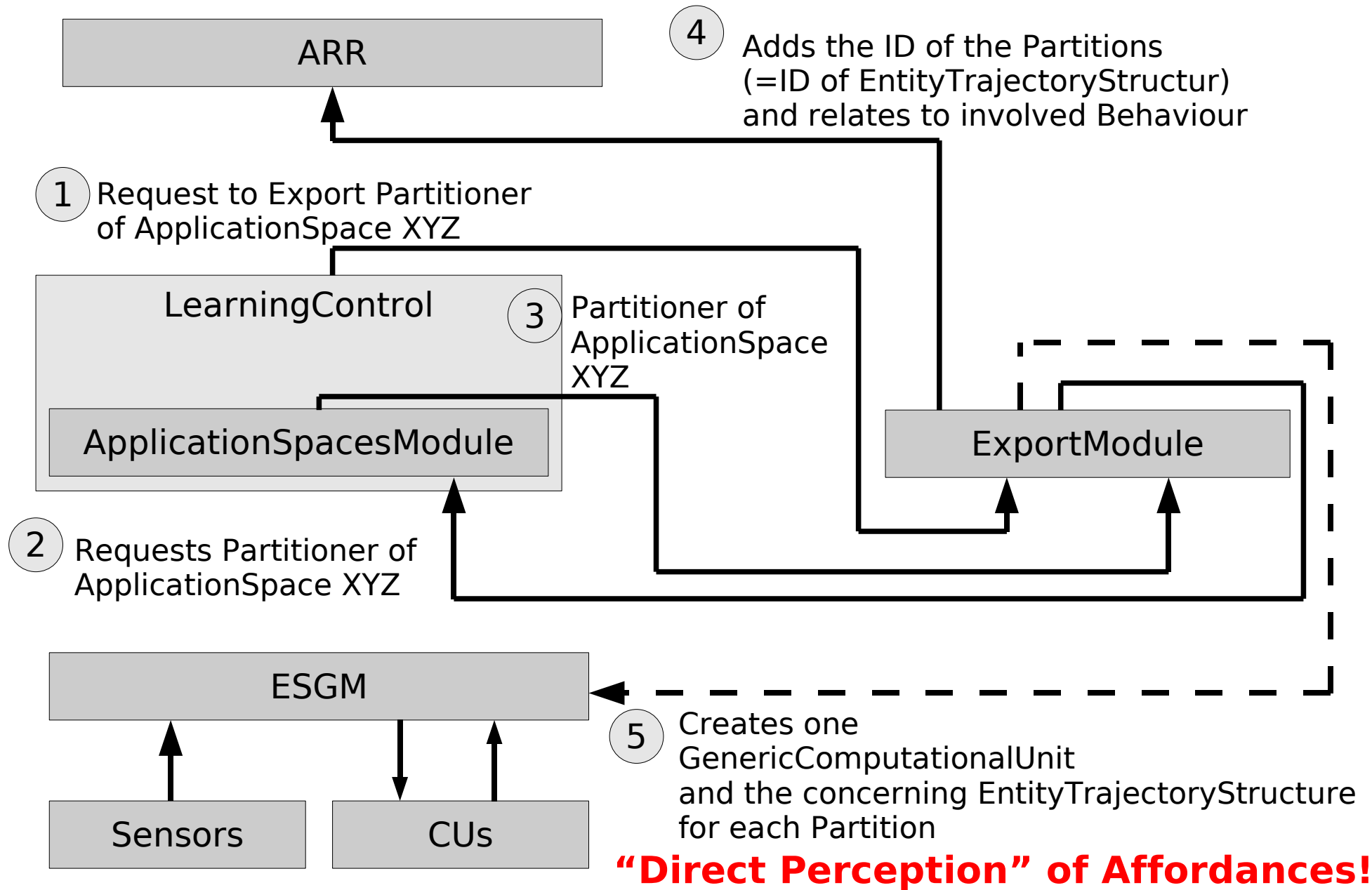
- Sub-Partitioning of the Partitioned ApplicationSpaces
- Using the methods developed by JR\_DIB Interfaces:
  - File System (storing data with respect to the derived outcome)
  - CORBA-Interface
  - Using the derived Filters

- Adaptation of the derived networks
- When a new cluster is found for one channel, the network of the Inter-Channel-Relations must be adapted too
- Implementation via *Listeners* : Data structures representing the inter cluster-relation-graphs are informed when the clusters are changed



# Providing the Derived Knowledge

## Application Flow – Invocation of ESGM and ARR



# Affordance Representation Repository

## ARR-Interface and ARR-GUI

- Stored IDs of the derived Computational Units (CUs) and their Interrelations
- Using these IDs
- Interface via CORBA and RMI
  - to get IDs
  - to query related Cues
- GUI for managing (storing/loading/removing)

CUEs	BEHAVIOURs	OUTCOMES
CamPos(low) and Blob(blue)	raise arm	lifted
Blob(green)		dropped
Sift(xyz)		

Partitioner "DemoASM_01:pull:1202719180827"	TrajectorySets
930ef5c4-c86c-4fae-9b08-303a4f4dd5f7	11915031929805710
ad77cfce-525e-4c7a-9720-dfd9cf6483ba	11915032257805780
	11915032575807470

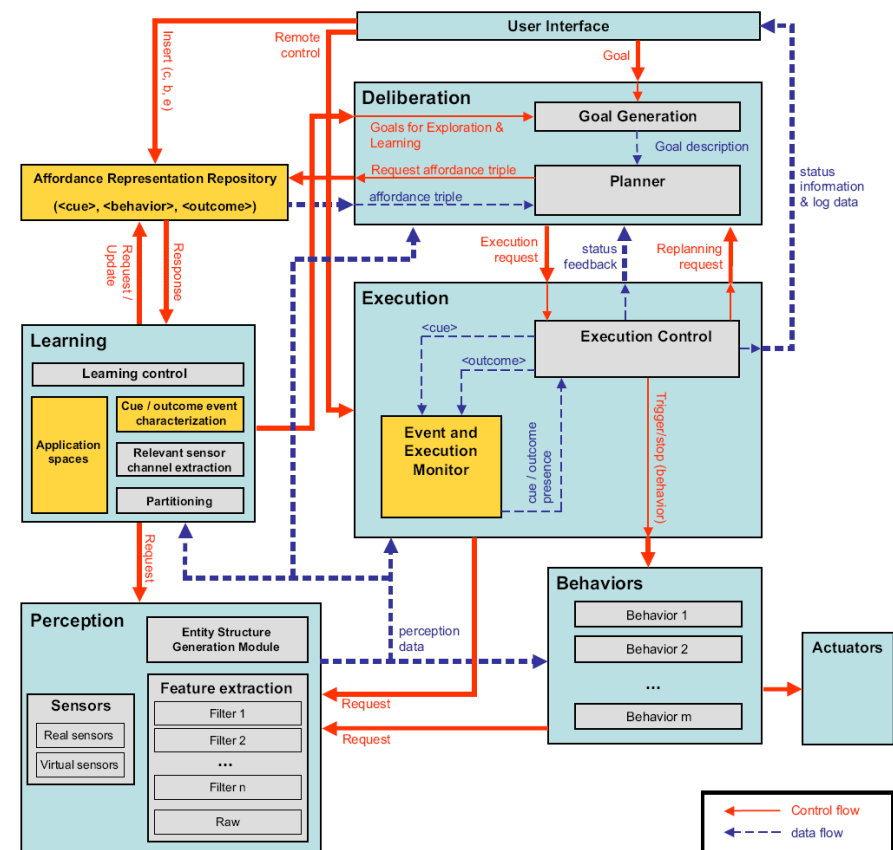


# Using the Derived Knowledge

## Execution and Planning

Behaviour of agent becomes robust and flexible

- reinforce actions if an Affordance is present
- suppress reflex like actions (don't bite a stone)
- fulfil needs (Hunger etc.)
- accomplish complex missions
- Using the derived knowledge as input for learning enables learning based on Affordances



# Demonstration Afternoon

The screenshot displays a software interface for robot learning and control, consisting of several overlapping windows:

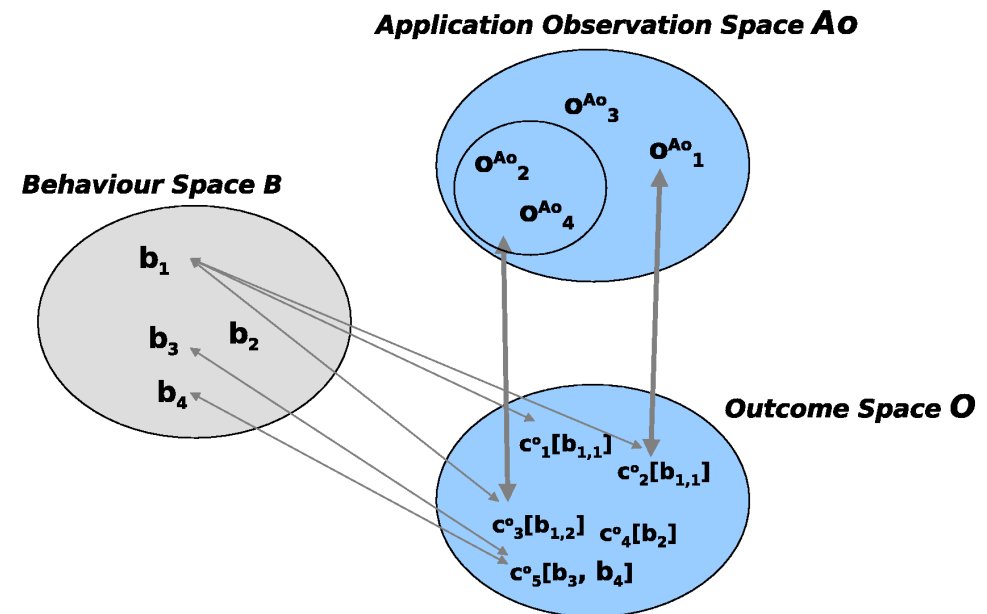
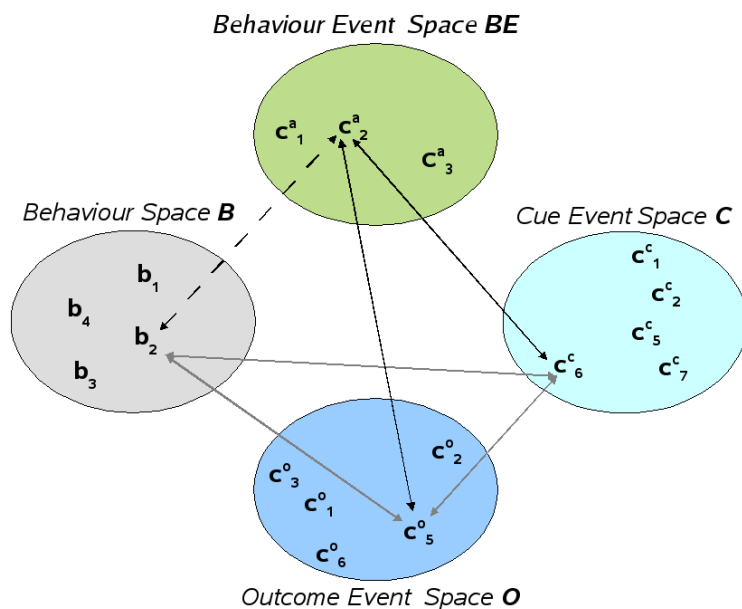
- LearningControl - Manager:** A window with tabs for 'Control', 'ASM', 'Partitioning', and 'Export'. The 'Control' tab is active, showing a 'pull' action.
- ApplicationSpace pull:** A window showing a list of trajectory sets. The 'Partitioner' is 'DemoASM\_01:pull:1202719180827'. The list includes various IDs, with several highlighted in yellow.
- ARR-Manager:** A window with a table showing the relationship between CUEs, BEHAVIOURs, and OUTCOMES.
 

CUEs	BEHAVIOURs	OUTCOMES
CamPos(low) and Blob(blue)	raise arm	lifted
Blob(green)		dropped
Sift(xyz)		
- TrajectorySet 11915032575807470:** A window showing two plots. The top plot, labeled 'craneC', shows a step function that rises from 50 to 216. The bottom plot, labeled 'blobPo', shows a highly oscillatory signal between 10,000 and 11,645.
- BehaviourModule:** A control window with four rows of actions, each with 'Start', 'Stop', and 'Cancel' buttons:
  - Action "stretch arm"
  - Action "close gripper"
  - Action "raise arm"
  - Action "move forward"

# D5.4.5 - Outlook

## Learning by Observation

- Learn to detect Behaviours by matching observations to already known Outcomes
- Imitation of Outcomes/Behaviours, not detailed action imitation
- New Cues can be learned without interactions
- Observing Cues and detecting Behaviours can be used to learn new Outcomes.
- Hypotheses must be verified by self-experience



- Hibernate (for a standardised data storage)
- Maven (for the building process also without Eclipse)
- Increase the Test-Coverage
- Divide Clustering into two independent Modules
- Add Alternatives to ESGM and more Comparators
- Add Software to support Learning by Observation
- GUI components to visualise the learning process
- ...

Thank you!