Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Intelligent System Architectures – Comparison by Translation

Benjamin Dittes, Christian Goerick

2011

Preprint:

This is an accepted article published in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. The final authenticated version is available online at: https://doi.org/[DOI not available]

Intelligent System Architectures – Comparison by Translation

Benjamin Dittes and Christian Goerick

***Abstract*— System architectures are a central element to enable construction of increasingly complex software systems for intelligent artifacts. Scientific discourse about such architectures requires means to compare approaches and identify common directions. In this contribution we put forth the hypothesis that a meaningful comparison is possible only by translation to a common language. We present SYSTEMATICA 2D as such a common language, suitable for translation and comparison of architectures because it is both formal in its description and flexible in the range of systems which can be expressed. Main result of this contribution is the translation and comparison of three recent popular system architectures: the CogX George system[1], the ALIS3 system running on the Honda humanoid robot[2] and the iCub Cognitive Architecture[3]. Common patterns in all three system become apparent from the two-dimensional ordering inherent in their SYSTEMATICA 2D translations and allow more founded discussions about architecture approaches and the exchange of concepts between real systems.**

I. INTRODUCTION

In the strive to create complex artifacts with artificial intelligence, an interplay of a great number of disciplines is required. In addition to software and hardware for interfacing with the physical world, cognitive architectures are receiving increasing attention as necessary means to integrate several (software) subsystems to an intelligent artifact system, see [4] for a survey. This growing attention calls for a scientific discourse about present and future system architectures, comparing approaches, identifying reoccurring patterns and providing best practices or posing scientific questions for the next generation of intelligent systems. However, while a scientific discussion about architecture *approaches* has been ongoing for the last 30 years (see Subsumption[5], 3-Tier[6] or CogAff[7], [8]), comparisons of architectures for real systems are missing. The central problem, as Christensen et al. precisely point out, is that

> "[...] there is no agreement on what the space of possible architectures is like, nor on the terminology for describing architectures or on criteria for evaluating and comparing them." [9]

This is not to say that comparing system architectures is completely impossible at the moment. Usually, system architectures are designed in the framework of an architecture approach (such as 3-Tier or Subsumption) and thus inherit some properties which allow comparison to systems built based on a different approach. What such a comparison cannot capture is a) that the reality of system construction may differ from the followed approach, b) that the construction process itself may reveal properties or necessities not considered in the approach and c) that as a result, systems built according to different approaches (e. g. CogAff vs. Subsumption) may be much closer in their design than the comparison of the approaches would suggest.

In this contribution we offer a more direct tool for comparing real, built system architectures to one another and to architecture approaches: translation to a common language. As such a common language we present SYSTEMATICA 2D which is both flexible enough to allow expression of many different systems (and system approaches) and formal enough to allow identifying similarities and differences as well as deriving construction patterns. Sec. III will give a concise overview of the concepts as well as the formal and visual notation of SYSTEMATICA 2D (for more details see [10]).

We will show that the language is able to express three state-of-the-art system architectures by providing and discussing SYSTEMATICA 2D translations in Sec. IV. The three discussed systems are: the CogX George system[1] ('George'), the ALIS3 system running on the Honda humanoid robot[2] ('ALIS3') and the iCub Cognitive Architecture[3] ('iCA').

The two-dimensional structure imposed by the SYSTEMATICA 2D language in turn allows to easily identify certain reoccurring pattern in the represented systems. Although all three discussed systems were designed according to different theoretical frameworks (George: CAS[11], ALIS3: SYSTEMATICA [12], iCA: Enaction[3]), Sec. V will show by means of comparing these patterns that all three system have strong similarities in their architecture approaches. Based on these similarities, an exchange of concepts and sub-architectures for future systems is discussed.

II. RELATED WORK

Two areas of related work are relevant for this contribution: comparative reviews of intelligent systems and formal notations or languages to express these systems.

As for the first, we see two comparisons of integration approaches in intelligent systems: Vernon et al.[4] give a survey of recent developments in cognitive architectures by analyzing a wide range of approaches and sorting them into three 'paradigms' of cognition (Cognitivist, Emergent and Hybrid). The focus of this survey is on evaluating qualitative properties of existing systems, where the relation between system architecture and resulting property is not the main focus. A different approach is pursued by Goerick in [12], where a new framework for modeling hierarchical architectures ('SYSTEMATICA', encapsulating the subsumption architecture[5]) is used to express popular architecture approaches in the same language and compare them on this basis. Here, the concept of comparison by translation is

present but the chosen formalism imposes a very rough granularity on the formulated designs which makes it more suited for comparing architecture approaches than final systems.

Formal notations or languages for intelligent systems today come from three areas. First, there are mathematical formalizations of system component interaction[13], [14], but it is unclear whether they can express established cognitive architectures such as 3-Tier[6] or CogAff[7]. Second, architecture description languages are a popular tool in the software engineering domain to describe large software systems, such as Rapide[15] or the more generic xADL[16]. These languages contain all relevant elements for describing an architecture but since, to the best of our knowledge, no application of such an ADL to the intelligent system domain has been attempted, it is unclear what can be said about an intelligent system architecture by translating it to e. g. xADL.

Finally, there are specific notations used in intelligent systems[8], [17], [2], [1], [3]. Among these notations, we see two groups: on the one hand there are systems described in ad-hoc formalisms used only once in the paper describing the system — this usually implies that these notations were not developed to allow generalization. On the other hand there are systems described in independently introduced notations such as 3-Tier[6], CogAff[7] and SYSTEMATICA[12]. These notations are developed to express a certain architecture approach and therefore provide specific constraints or structural bias towards this approach, which also makes them less applicable as a common language — for a detailed review of such formalisms see [10].

To conclude, we can say that there is a multitude of notations used to describe systems and a very small number of attempts to compare built system architectures or to establish a generic language so serve as a basis for comparison. In the following, we will therefore start by presenting such a common language. This language will then be used to translate and compare the George, ALIS3 and iCA architectures.

III. SYSTEMATICA 2D

The common language used for translation will be called 'SYSTEMATICA 2D' (short: 'SYS2D'). It describes systems on two levels: the functional and the descriptive. In set notation, a system $\mathbb{S} = (\boldsymbol{U}, \boldsymbol{A})$ is defined in SYS2D by a set of functional units $\boldsymbol{U}$, including interfaces, connections and dependencies, and a set of sub-architectures $\boldsymbol{A}$, including the description of their sensory & behavior spaces. Fig. 1 shows an example design, all relevant elements will be detailed in the following.

A. Functional System Design

On the functional level, a SYS2D system is composed of $\boldsymbol{N > 2}$ processing units $\boldsymbol{u_n \in U, n = 1..N}$. There is always one unit $\boldsymbol{u_1}$ responsible for emitting sensor events from exteroception $\boldsymbol{S_e}$ and proprioception $\boldsymbol{S_p}$ as the full sensor space $\boldsymbol{S = S_e \times S_p}$. A second predefined unit $\boldsymbol{u_2}$ is responsible for receiving and executing motor commands from the motor space $\boldsymbol{M}$.

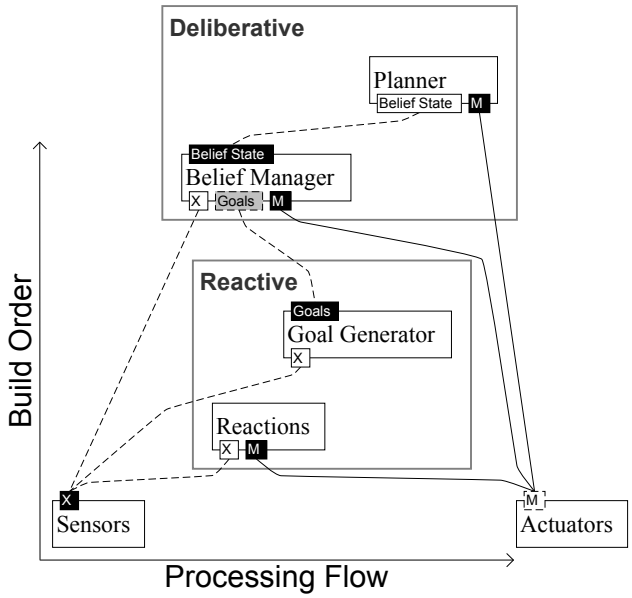

Fig. 1. Example of a SYSTEMATICA 2D system architecture, translated from a CogAff design[8]. The system is composed of connected units (large boxes) with attached ports (small boxes) which are arranged along the two axes according to processing flow and build order. Port coloring is according to Fig. 2, dashed lines between ports are pull connections, solid lines are push connections. Surrounding gray boxes denote sub-architectures.

1) Formal Notation: Every unit $\boldsymbol{u_n = \{(D_n, I_n, O_n, \textbf{Pull}_n, \textbf{Push}_n)\}}$, $\boldsymbol{n = 1..N}$ is described by the following features (see Fig. 1):

- it has an internal dynamics $\boldsymbol{D_n}$ running independently and asynchronously from all other units;
- it has an interface defined by a set of input ports $\boldsymbol{I_n}$, where each element is defined by its name, type and input role, thus $\boldsymbol{I_n \subset \{(\textbf{name}, \textbf{type}, \textbf{role})\}}$ and a set of output ports $\boldsymbol{O_n}$, where each element is defined by its name and type, thus $\boldsymbol{O_n \subset \{(\textbf{name}, \textbf{type})\}}$;
- it specifies the properties of each input port by assigning one of three 'roles', which we will call **Driving**, **DrivingOptional** or **Modulatory** — these roles define dependencies between units, see Sec. III-A.2;
- it may pull data from another units output port $\boldsymbol{o'}$ to one of its input ports $\boldsymbol{i}$, specified by a set of pull operations $\boldsymbol{\textbf{Pull}_n \subset \{(u_{\text{source}}, o', i)\}}$ (visualized as dashed lines);
- it may push data from one of its output ports $\boldsymbol{o}$ to another units input port $\boldsymbol{i'}$, specified by a set of push operations $\boldsymbol{\textbf{Push}_n \subset \{(u_{\text{target}}, i', o)\}}$ (visualized as solid lines).

A description of a system as a set of units (with arbitrary granularity), communicating over arbitrary connections is intuitively very flexible. In order to allow a meaningful comparison between SYS2D systems however, some level of common constraints is required. SYSTEMATICA 2D uses inputs roles, push/pull connections and a constraint on two-dimensional sortability to enforce modeling of unit dependencies. For brevity, Sec. III-A.2 will give only a concise review of these concepts, please see [10] for a full discussion of the theoretical foundations and implications.

2) Input Roles & Dependencies: Two formal elements allow formulating dependencies: push/pull connections and input roles. These two mechanisms are independent and can therefore be used to specify dependencies along two independent dimensions: the difference between push/pull defines the 'build order' dimension (i. e. build first, build later), the roles of input ports define the 'processing flow' dimension (i. e. from sensor to actuator).

Build Order: If unit $\boldsymbol{u_n}$ pulls data from or pushes data to unit $\boldsymbol{u_m}$ then $\boldsymbol{u_n}$ has to be built after unit $\boldsymbol{u_m}$ — in other words: only the newer unit needs to know about the *connections* it makes to older or preexisting units (although the older units must provide the *ports* to accept these connections). Since every connection between ports can be symmetrically formulated as either a push or a pull, this sorting by build order is completely in the hands of the designer, except for one constraint: The units must be sortable according to build order, i. e. there must not be loops of purely pull or purely push connections.

Processing Flow The concept of sorting units by their role or function in the processing chain is old: from the Sense-Plan-Act models, over the Controller-Sequencer-Deliberator sorting in 3-Tier to the Bottom-Up and Top-Down channels in SYSTEMATICA — not to mention the usage of these terms in neurological studies.

In the SYS2D functional model, we chose to model this quality locally, by specifying the 'role' of input ports as one of the following three (see Fig. 2):

- **Driving** inputs are mandatory and indicate input data from units prior to the recipient along the processing flow — this is typically used for sensor preprocessing results, representations, etc.
- **DrivingOptional** inputs are similar to Driving but optional, i. e. the recipient can function without receiving data on such ports — this is typically used for inputs to data fusion units, motor commands, etc.
- **Modulatory** inputs are optional and indicate input data from units further along the processing flow — this is typically used for modulation of parameters or operation modes

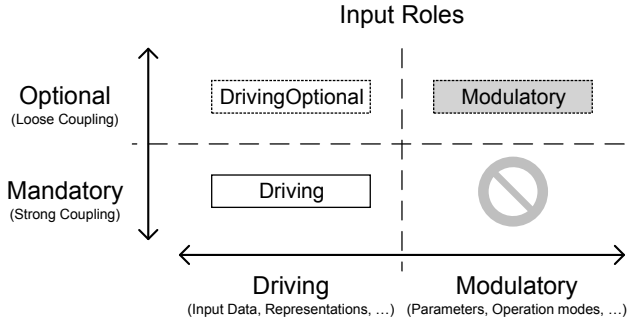

Fig. 2. Input roles in SYSTEMATICA 2D. Two criteria are interleaved: driving / modulatory inputs (sometimes referred to as bottom-up / top-down) and mandatory / optional inputs. Three of these combinations are supported by the designated roles, the combination 'mandatory and modulatory' is excluded by design. See text for details.

One combination is intentionally missing: mandatory inputs from modules further along the processing flow (i. e. mandatory modulation). This constraint does not prohibit processing loops in a system but only requires some links in a processing loop to be declared as loosely coupled, i. e. DrivingOptional or Modulatory (see [10]).

Sorting along the processing flow is now straightforward. If unit $\boldsymbol{u_n}$ receives (by push or pull) data to a driving (Driving or DrivingOptional) input from unit $\boldsymbol{u_m}$ then unit $\boldsymbol{u_n}$ is further along the processing flow than unit $\boldsymbol{u_m}$. Conversely, if unit $\boldsymbol{u_n}$ receives data to a modulatory input from unit $\boldsymbol{u_m}$ then unit $\boldsymbol{u_m}$ is further along the processing flow than unit $\boldsymbol{u_n}$. Like for the 'build order' dimension, SYS2D systems must be sortable along the 'processing flow' as well, i. e. there must not be loops of purely modulatory or purely driving connections.

B. Descriptive System Design

In addition to the set of units, SYSTEMATICA 2D allows definition of sub-architectures with groups of units and additional semantic information, following the definition of sensor and behavior spaces in [12].

This makes it possible to group a system into conceptual subsystems at an independent level of description whose granularity may not coincide with that of the units.

In a SYS2D design $\mathbb{S} = \{\boldsymbol{U}, \boldsymbol{A}\}$, a sub-architecture $\boldsymbol{a_k} \in \boldsymbol{A}$ is a tuple $\boldsymbol{a_k} = (\textbf{name}, \boldsymbol{U_k}, \boldsymbol{S_k}, \boldsymbol{B_k})$ with $\boldsymbol{U_k} \subset \boldsymbol{U}$ and $\forall(\boldsymbol{k}, \boldsymbol{l}) : \boldsymbol{U_k} \cap \boldsymbol{U_l} = \emptyset$ (a unit may not belong to more than one sub-architecture), where $\boldsymbol{S_k}$ describes the sensor space used by $\boldsymbol{a_k}$ and $\boldsymbol{B_k}$ describes the behavior spaces emitted by $\boldsymbol{a_k}$ (see Fig. 1 for a complete SYS2D design).

IV. SYSTEM ARCHITECTURE TRANSLATIONS

Based on the presented common language, we will now proceed to present translations of the George[1], ALIS3[2] and iCA[3] systems. The translations of these three systems are based on the more or less formal architecture descriptions in the respective publications.

While these descriptions usually explicitly describe units and connections, the connection types, unit interfaces and input roles have been inferred from the system descriptions or associated publications (ALIS3: [18], [19], iCA: [20]). The build order, and thus the distinction of push/pull connections, is usually implicit in the order of described system elements or in previously published subsystems. The input roles have been chosen based on the described role of a unit in the processing flow (sensor-near or actuator-near). In the following paragraphs, each system translation is presented and discussed by itself, before the comparison in Sec. V.

A. CogX George

Fig. 3 shows the SYS2D translation of the George system, together with a small version of the original system architecture taken from[1]. The original architecture decomposes the system into three sub-architectures, the SYS2D design splits the 'Communication SA' into 'Auditory SA' and 'Speech SA'

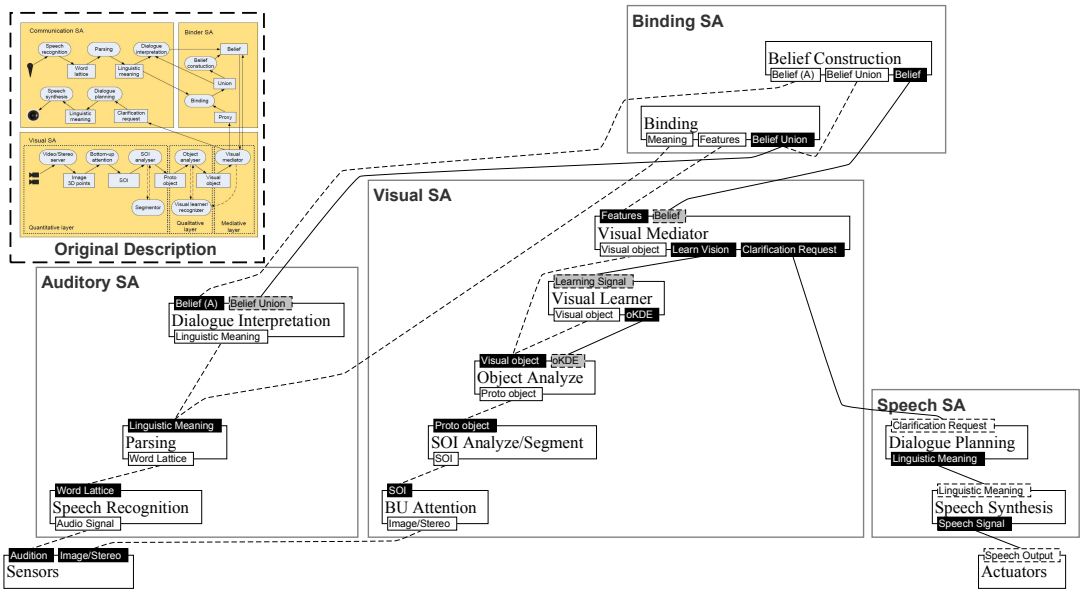

Fig. 3. Translation of the George system architecture to the SYSTEMATICA 2D language. Original system description taken from [1], see text for details.

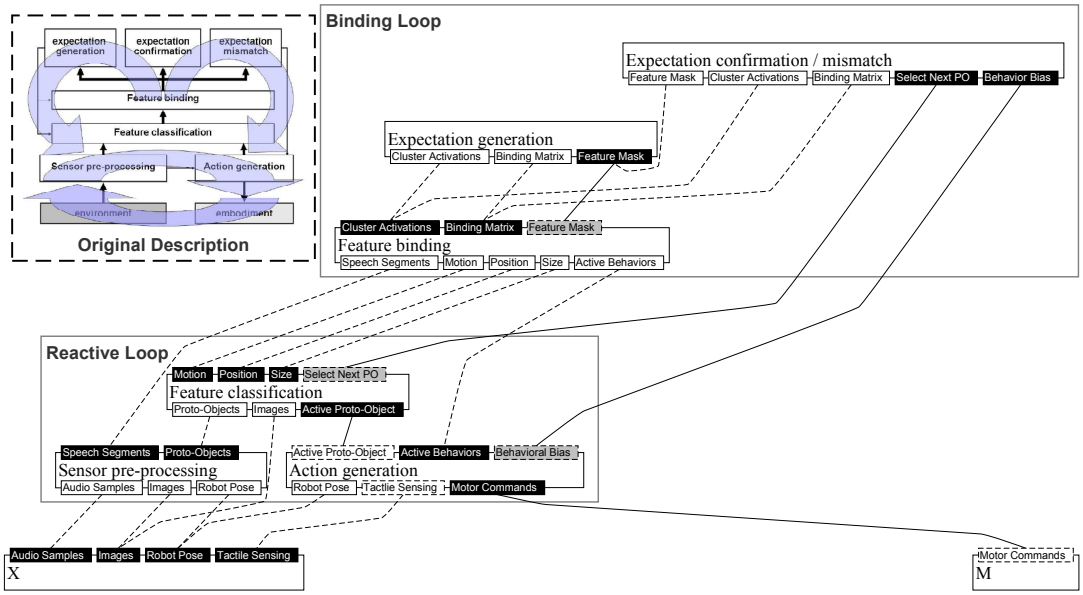

Fig. 4. Translation of the ALIS3 system architecture to the SYSTEMATICA 2D language. Original system description taken from [2], see text for details.

in order to separate sensor preprocessing from behavior generation along the processing flow. Descriptions of variables have been re-interpreted as unit ports but have kept the same function in the communication between units. Following the description of 'Visual SA' and 'Communication SA' as standalone entities, the input roles and push/pull connections have been modeled to place the 'Binding SA' on top of the system in terms of build order.

The resulting system fulfills all SYS2D constraints. At the same time, the resulting two-dimensional structure represents the idea of independent processing streams for audition, vision and speech, coupled together by a central binding subarchitecture.

B. ALIS3

Fig. 4 shows the SYS2D translation of the ALIS3 system and it's original architecture description from [2]. Although the original description implied two main systems layers and an explicit set of units, the lack of interfaces and the occasionally unclear meaning of connecting arrows required a strong consideration of related publications (see [18], [19]). Interfaces and most input roles could then be derived, based on described dependencies and bottom-up / top-down information flow.

The resulting design fulfills all SYS2D constraints and maintains the two-layer separation, highlighted by subarchitectures, of the original design.

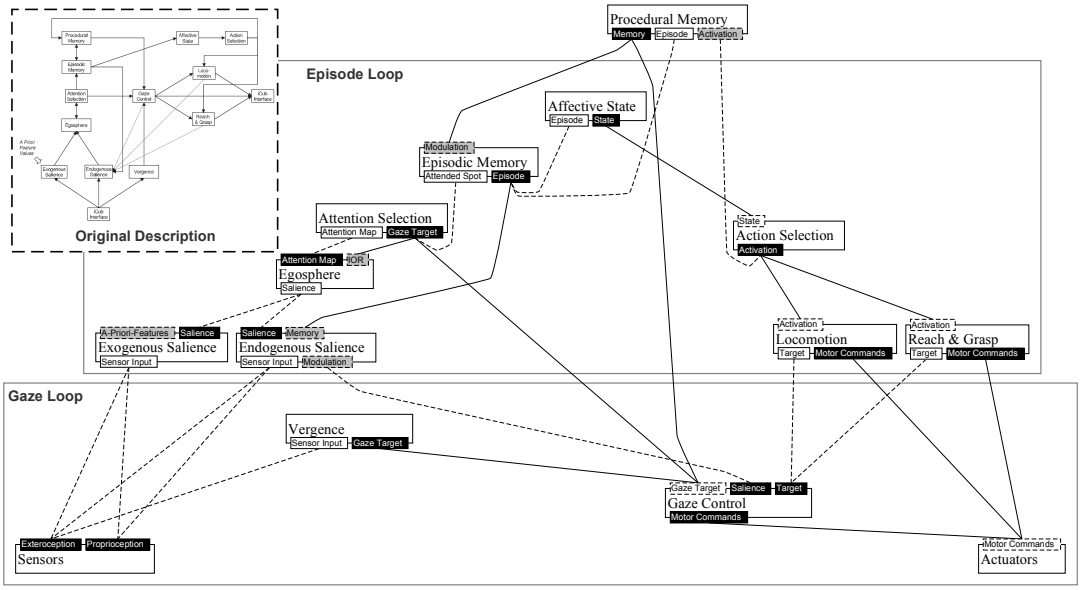

Fig. 5. Translation of the iCub Cognitive Architecture to the SYSTEMATICA 2D language. Original description taken from [3], see text for details.

C. iCub Cognitive Architecture

Fig. 5 shows the SYS2D translation of the iCub Cognitive Architecture, together with the original diagram from [3]. As with the other two architectures, the explicitly described units were adopted almost one-to-one, while deriving interfaces required consideration of the iCub project deliverables[20]. The sorting of units along the processing flow could be derived from the publication, the sorting along the build order was matched to subsystems presented in independent publications.

The resulting design fulfills all SYS2D constraints and allows a quicker visual understanding of the separate sensor-actuator loops inherent in the original design.

V. COMPARISON & DISCUSSION

The previous section has presented SYSTEMATICA 2D translations of the George, ALIS3 and iCA systems. Although these translations already allow defining some simple taxonomies (i. e. number of units, use of optional/mandatory inputs, minimal height/width according to sorting dimension, etc.) in order to perform a meaningful comparison we must define specific criteria about which the comparison should be performed. Since all considered systems are from the intelligent artifact domains, we will compare them according to the following, embodiment-related criteria:

- What sensor-actuator loops exist in in the system? Which behaviors do they emit? How do they interact?
- What is the role of units in the system which are *not* involved in these loops?

The answers to these questions can be derived from the two-dimensional ordering of SYS2D units: sensor-actuator loops are connected sets of units from lower left (sensor) to lower right (actuator). Separate loops will typically differ in build order and therefore be stacked on top of each other, with interactions and involved units clearly visible. Units not involved (directly) in these loops will be visible as input data or modulation source for the loops.

Fig. 6 shows low-resolution SYS2D designs with manually annotated arrows highlighting the main flows of information. From them, we can derive specific answers to the questions and attempt a comparison of the three systems.

A. Sensor-Actuator Loops

It is apparent from the designs that while George and ALIS3 only have one sensor-actuator loop[1], iCA has three: reactive vergence, attention-triggered gazing and specific behavior based on action selection. For the George system, however, it is not clear whether any behavior would be visible (or audible) without all four sub-architectures operational: the sensor-actuator loop of visual and speech SA is closed, but it will probably not issue clarification requests without the binding SA providing belief, which in turn depends on the auditory SA. The behavior spaces of ALIS3 and iCA are similar (gazing, pointing, locomotion, grasping on a humanoid robot) while George focuses on spoken communication with a tutor. The interaction of units is only relevant for iCA: while Vergence is operating on its own, the two higher loops are based on intermediate results of the same processing chain.

B. Additional Units

In the George system, the binding and auditory SA are not directly involved in a sensor-actuator loop. They provide belief as a modulation signal to the 'Visual Mediator', based on sensory preprocessing independent of the loop (the auditory SA). ALIS3 defines a binding sub-architecture in a similar way, but using the same preprocessing results as

[1]Actually, ALIS3 has two: the 'Action generation' unit is able to operate without input from 'Feature classification', but this is only used for returning the robot to starting position.

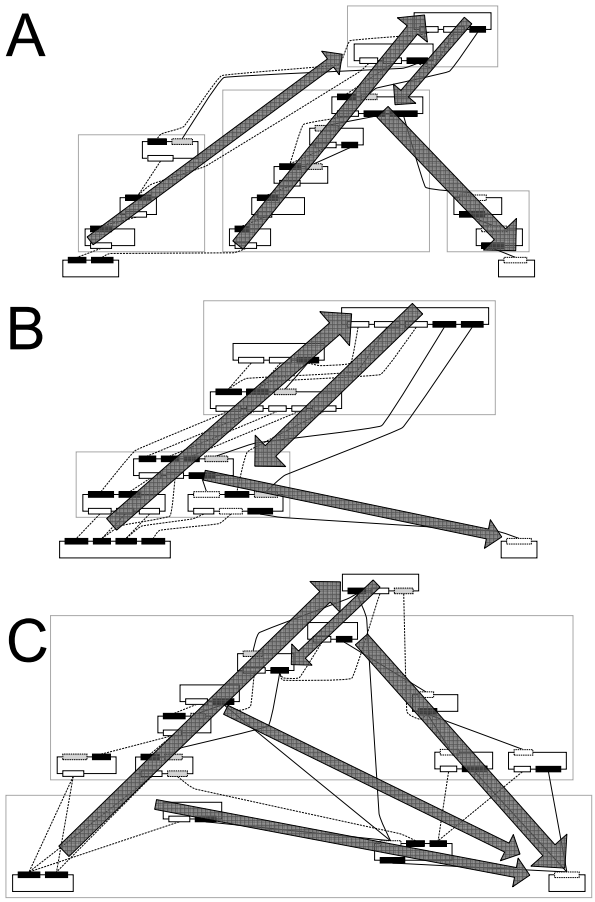

Fig. 6. Low-resolution SYS2D designs of the George (A), ALIS3 (B) and iCA (C) systems, with arrows highlighting the main information flows. See text for details.

necessary for the loop. The modulation produced in the binding SA affects both preprocessing and action generation. Finally, iCA has only one unit, the 'Procedural Memory', which is not involved in any of the three loops but is receiving episode and activation information and modulates the 'Episodic Memory'. In this respect it is very similar to the 'binding' parts of George and ALIS3.

C. Common architecture patterns

As a result of this comparison, we can say that all three systems share some common patterns in their architectures:

- All systems have at least one closed sensor-actuator loop, involving sensory preprocessing and behavior generation, and are thus able to operate on a physical artifact in the real world.
- All systems have a concept of 'proto-objects' (George, ALIS3) or 'gaze targets' (iCA) as a mechanism for visual scene decomposition in order to generate behaviors.
- All systems add a modulating or 'binding' subarchitecture (iCA: 'Procedural Memory') responsible for higher-level association or learning to modulate the sensor-actuator loops.

These qualities could be summarized as an architecture approach called 'modulated loops': one or more sensor-actuator loops are modulated by a high-level binding subarchitecture.

D. Implications of Differences

As a next step, we can isolate the architectural differences, try to understand their causes in the design considerations of each system and evaluate what implications for the cognitive abilities of the respective systems these differences have.

The first main difference is the number of sensor-actuator loops chosen. George and ALIS3 rely on only one loop with strong modulation, iCA focuses on three separate loops while the internal structure of the procedural memory is not fully specified in existing publications. As a main cause for the different number of loops we propose that the difference in targeted scenarios is most plausible: George and ALIS3 are always operating with a tutor, iCA is intended first for autonomous exploration and only later for interactive scenarios. The presence of a tutor from the start requires the former systems to emit behaviors on a relatively high level so as not to bore or confuse the tutor — this in turn implies the necessity for a more sophisticated preprocessing before it is reasonable to close the first sensor-actuator loop. While it would be beneficial for fault tolerance to have lower loops which can ensure basic robot function, this is not as essential in controlled scenarios as it is in autonomous exploration for iCA.

The second main difference is the range of behaviors emitted by the different systems. ALIS3 and iCA use an action selection mechanism to switch between gazing, pointing, grasping and locomotion while George only employs one mechanism for emitting speech[2]. Most probable cause for this difference is the kind of physical artifacts the systems are running on: while ALIS3 and iCA use a high-DOF robot, George uses a fixed-camera table-top scenario with microphone and speaker. In terms of cognitive abilities, the lack of different behavior modules in the George system makes is difficult to imagine other sensor-actuator loops without adding a new unit for behavior generation. On the other hand, the behavior spaces spanned by the action generation units in ALIS3 and iCA are sufficient for a variety of loops, as iCA demonstrates.

E. Practical Implications for Future Systems

We have now looked at similarities and differences in the three systems, based on their SYS2D translations. For the two main differences (number of loops and structure of behavior generation) we have identified their main causes in the respective system's design decisions (tutor-based/explorative scenario, shape of physical artifact) and their implications in term of cognitive abilities.

What makes this comparison relevant for future systems is that despite these fundamentally different design goals, the resulting SYS2D designs still share many architecture patterns. This indicates that it should be possible to combine

[2]This is not to say that the complexity of speech production is higher or lower than that of high-DOF motor control.

ideas or sub-architectures from different systems, maintaining the overall 'modulated loops' pattern, in order to overcome challenges in each systems. For example:

- To extend George to a mobile robot platform, lower sensor-actuator loops are required to couple proto-object detection to e. g. gazing behavior, as found in ALIS3 and iCA.
- To allow exploratory behaviors in ALIS3, at least one lower loop is required which, similar to the 'Episodic Memory' in iCA, switches proto-objects autonomously and learns bindings between different objects and possible actions.
- To enable voice communication in iCA, the George Auditory SA could be added as an input to the procedural memory and the Speech SA could be added as a possible behavior to the action selection.

We realize that constructing any of these extended systems is a major effort, mainly due to different software infrastructures, data formats, computing hardware etc. The ability to exchange ideas between systems described in a common language is only a first step towards easier exchange of software and hardware modules.

VI. Conclusion

In this contribution we have introduced translation to a common language as a feasible, direct and valuable technique to compare intelligent system architectures. As this common language we have presented SYSTEMATICA 2D which is suitable for this purpose because of its flexible but formal notation and resulting two-dimensional unit arrangement.

Main aim was then to translate and compare three state-of-the-art intelligent system architectures (George[1], ALIS3[2] and iCA[3]). It was shown that these systems could be expressed in SYSTEMATICA 2D, based on the original and supplementary publications, and that these translated designs allow evaluation of similarities and differences. Despite fundamental differences in the design goals of the three systems, regarding tutor-based/explorative scenarios and mobile/fixed platforms, we argued that the three designs share many common patterns. Based on these common patterns, the potential for exchange of concepts (on the design level) and software modules (on the system level) was explored.

Looking at the strong similarities concerning sensor-actuator loops in system architectures, it seems surprising that much more effort is spent arguing for the value of embodied intelligent systems than on discussing the next set of questions which embodied systems are facing today:

- How many sensor-actuator loops are best for a certain set of behaviors?
- Should modulation of these loops rely on the same pre-processing used in the loops or on separate information?
- Which architecture patterns can integrate high-level concepts like planning, goals, internal simulation etc. in order to combine symbolic and embodied approaches?

We believe that eventually the ability to share concepts among architectures expressed in a common language will allow defining commonly accepted architecture patterns necessary to achieve a new level of cognitive abilities.

References

[1] D. Skočaj, M. Janiček, M. Kristan, G.-J. M. Kruijff, A. Leonardis, P. Lison, A. Vrečko, and M. Zillich, "A basic cognitive system for interactive continuous learning of visual concepts," in *ICRA 2010 workshop ICAIR - Interactive Communication for Autonomous Intelligent Robots*, Anchorage, AK, USA, May 2010, pp. 30–36.

[2] C. Goerick, J. Schmuedderich, B. Bolder, H. Janssen, M. Gienger, A. Bendig, M. Heckmann, T. Rodemann, M. Dunn, H. Brandl, X. Domont, F. Joublin, and I. Mikhailova, "Interactive online multimodal association for internal concept building in humanoids," in *IEEE-RAS International Conference on Humanoids 2009*. IEEE, December 2009.

[3] D. Vernon, "Enaction as a conceptual framework for developmental cognitive robotics," *Paladyn*, vol. 1, no. 2, pp. 89–98, 2010.

[4] D. Vernon, G. Metta, and G. Sandini, "A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 151–180, 2007.

[5] R. Brooks, "A robust layered control system for a mobile robot," *IEEE journal of robotics and automation*, vol. 2, no. 1, pp. 14–23, 1986.

[6] E. Gat *et al.*, "On three-layer architectures," *Artificial Intelligence and Mobile Robots*, 1997.

[7] A. Sloman, "The cognition and affect project: Architectures, architecture-schemas, and the new science of mind," 2008.

[8] N. Hawes, "Architectures by design: The iterative development of an integrated intelligent agent," in *Proceedings of AI-2009, The Twenty-ninth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 2009.

[9] H. Christensen, A. Sloman, G.-J. Kruijff, and J. Wyatt, Eds., *Cognitive Systems*. Springer Verlag, 2009. [Online]. Available: http://www.cognitivesystems.org/cosybook/

[10] B. Dittes and C. Goerick, "A language for formal design of embedded intelligence research systems," *Robotics and Autonomous Systems*, vol. 59, no. 3-4, pp. 181 – 193, 2011.

[11] N. Hawes and J. Wyatt, "Engineering intelligent information-processing systems with cast," *Adv. Eng. Inform.*, vol. 24, no. 1, pp. 27–39, 2010.

[12] C. Goerick, "Towards an understanding of hierarchical architectures," *Autonomous Mental Development, IEEE Transactions on*, no. 99, p. 1, 2010.

[13] M. Scheutz and J. Kramer, "Radic: a generic component for the integration of existing reactive and deliberative layers," in *5th Intl. joint conf. on Autonomous agents and multiagent systems*. ACM New York, NY, USA, 2006, pp. 488–490.

[14] G. Gössler and J. Sifakis, "Composition for component-based modeling," in *Sci. Comput. Program.*, vol. 55, no. 1-3, 2005, pp. 161–183.

[15] D. Luckham, J. Kenney, L. Augustin, J. Vera, D. Bryan, and W. Mann, "Specification and analysis of system architecture using rapide," *IEEE Transactions on Software Engineering*, vol. 21, no. 4, pp. 336–354, 1995.

[16] R. Khare, M. Guntersdorfer, P. Oreizy, N. Medvidovic, and R. Taylor, "xadl: enabling architecture-centric tool integration with xml," in *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*. IEEE, 2002, p. 9.

[17] S. Wrede, J. Fritsch, C. Bauckhage, and G. Sagerer, "An xml based framework for cognitive vision architectures," in *17th Intl. Conf. on Pattern Recognition*, 2004.

[18] I. Mikhailova, M. Heracles, B. Bolder, H. Janssen, H. Brandl, J. Schmuedderich, and C. Goerick, "Coupling of mental concepts to a reactive layer: incremental approach in system design," in *Proceedings of the 8th International Workshop on Epigenetic Robotics, Brighton, England*. Lund University Cognitive Science Studies 117, 2008.

[19] J. Schmuedderich, H. Brandl, B. Bolder, M. Heracles, H. Janssen, I. Mikhailova, and C. Goerick, "Organizing multimodal perception for autonomous learning and interactive systems," in *8th IEEE-RAS International Conference on Humanoid Robots, 2008. Humanoids 2008*, 2008, pp. 312–319.

[20] C. von Hofsten, L. Fadiga, and D. Vernon. A roadmap for the development of cognitive capabilities in humanoid robots. [Online]. Available: http://www.robotcub.org/index.php/robotcub/content/download/696/2497/file/RC_DIST_DV_Deliverable_D2.1.pdf