Self-awareness in Real-time Cognitive Control Architectures*

Ricardo Sanz, Ignacio López and **Carlos Hernández**

Autonomous Systems Laboratory
Universidad Politecnica de Madrid
José Gutierrez Abascal 2, 28006 Madrid, Spain

Abstract

In this paper we describe the approach taken in the UPM Autonomous Systems Laboratory for the development of technology of full, bounded autonomy. This effort is taking us into the study of the phenomenology of consciousness especially in relation with the construction of mechanisms for system self-awareness. The progress of this work can be tracked in our ASys and SOUL projects webpages. The ASys Project is a long-term project that deals with the development of an engineering process and a collection of assets for the implementation of autonomous systems. The SOUL Project develops a generic architecture for self-aware autonomous systems.

The Frontiers of Control Technology

The field of control technology addresses the different engineering aspects involved in the construction of control systems. Control systems are subsystems designed to improve operation of certain systems of interest constituting an integral part of them. There are plenty of control systems out there even when in most cases they may pass unnoticed to the point that Karl Åström, a major control guru, said of this technology that it is a *hidden* technology.

Control systems can be found in the heating of our homes, in the landing gear for planes, in our CD players, in heart pacemakers or insulin pumps, in keeping the electrical network stable, *etc.* In a sense, control systems do keep working in *sufficiently good* condition the technological infrastructure of our modern lives.

Research in control systems (see for example www.ifac-control.org) has been dealing with the different aspects of this technology and the science that supports it. From the theoretical underpinnings of control systems to the purely practical applications in several domains the field of control has always dealt with the issue of autonomy. In many cases limited to the system being able to address certain limited classes of perturbations, but there are many reasons for pursuing the objective of fully autonomous machines.

*We acknowledge the support of the Spanish Ministry of Education and Science through grant *C3: Control Consciente Cognitivo* and the European Commission thorough Grant *ICEA: Integrating Cognition, Emotion and Autonomy*.

Reducing operating costs and improving system performance —that is maximizing the benefit— were the main factors behind the drive for improved automation in the past. During the last years, however, a new force is gaining momentum: the need for augmented dependability of complex systems. This is a problem that classical control theory nor standard software engineering cannot deal with. This is the reason utimately grounding the very core of our research into the deeps of artificial consciousness.

Autonomy for cost reduction and performance

In many cases, automated system performance is much higher than manual system performance. This is due to many reasons: 1) Automated systems do not get tired; 2) automated systems are usually faster; 3) automated systems are more precise; etc.

There are even technical systems that cannot be manually operated at all due to the intrinsic limitations of humans —for example in high-speed machining— or practical or legal issues typically associated with worker health —for example in space exploration, foundries, chemical plants, *etc*.

In many practical cases the problem of building a controller for a well known production process in a well controlled environment is mostly solved; only minor details persist related with continuous optimisation.

The problem for control engineering appears with uncertainty. When the plant is not so well known, or when the operational and/or environmental conditions are of high uncertainty the traditional control system strategy fails. This is so because having a good knowledge of the plant is the first necessary step to building a good controller for it (Conant & Ashby 1970).

There are many textbooks on controller design in general terms, centered in particular kinds of controller designs or centered in concrete application domains. In the last case, the domain typically constrains the kind of implementation that a controller may have (e.g. table-driven controllers in resource constrained electronic control units in vehicles or software-less controllers in some safety-critical applications).

Industrial-level control system engineering is a well established professional practice that addresses the task of controller construction in six steps:

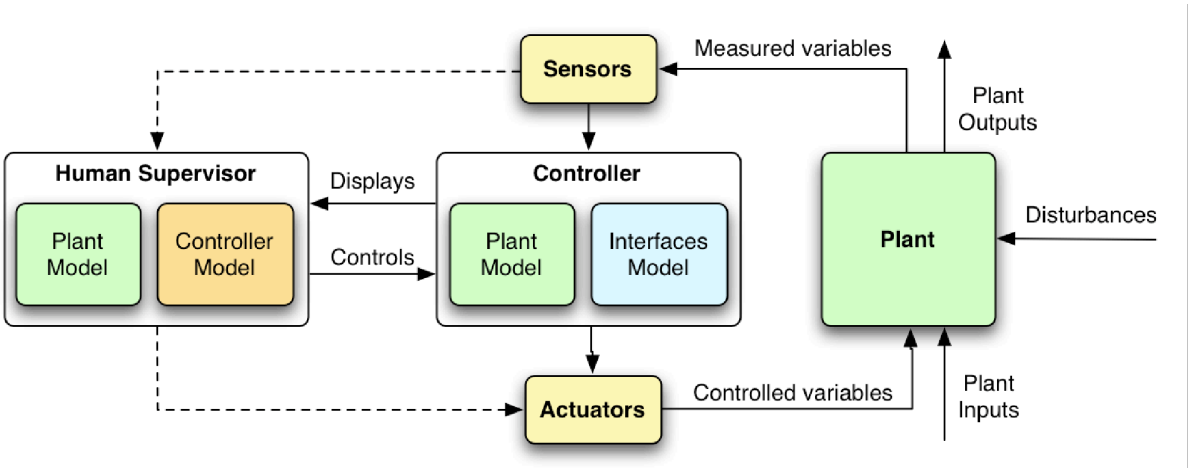

Figure 1: A standard hierarchical (supervisory) control scheme.

1. *Specify requirements* for the controlled system (i.e. the systems composed by the plant and the controller)
2. *Model the plant*, *e.g.* build a mathematical model of the plant suitable for the purposes expressed in the requirements.
3. *Design the controller* to fulfill the requirements of 1) for the model of plant obtained in 2).
4. *Build the controller*, usually using computer-based information technologies.
5. *Deploy the implementation* into the real setting.
6. *Test* and if it doesn't pass go back to 4, 3, 2 or 1 depending on test failure.

Steps 2 and 3 have been considered the traditional domain of control systems science. It assumes that we are able to model the plant to the precision enough for designing a controller for it. This focusing implies that the modeling & design tasks shape the whole engineering process. The vast majority of control technology restricts itself to simple modeling approaches due to the limitations of controller design methods (*e.g.* most control systems are based on linear models and controllers; models which are, in general, far from real plants).

Figure 1 shows a typical structure of a real-world supervisory control system for a large plant. In operational conditions, the *controller* is controlling the *plant* through the *actuators* by means of some calculation that is done using knowledge about the plant (a *plant model*) and information about the present plant state obtained through the *sensors*. In many cases, the knowledge about the plant captured in the plant model is not stored explicitly into the controller but embedded into its very implementation (then it being a controller tuned to a specific class of plants).

The reason why it is called *supervisory* is because, due to uncertainties (in plant knowledge and in plant disturbances during operation), the controller-plant system may depart from the specification. Then another, higher level control loop enters the scene modifying the controller behavior to adapt it to the new circumstances. In some cases this supervision is done automatically (this is usually called *adaptive control*) but in others this is done by humans (then called *supervisory control*).

We are then entering another world of control, one where knowledge is less than perfect and systems dependability becomes a critical aspect. This is, to our understanding, the main reason for pursuing research on machine consciousness. Not because we think that this is intelligence-on-steroids but because some of the properties perceivably associated to consciousness are of relevance to the dependability desiderata: self-awarenes, diachroneity, cognitive robustness, introspection, etc.

Qualia (Nagel 1974), however, remains untargeted mostly due to the lack of understanding of its evolutionary advantage; but it may well be the case that it will emerge as a side-effect or as an intrinsic co-property of some of the design patterns under development.

Autonomy for dependability

Dependability considerations have always been a matter of worries for real-world engineers. But today, in many complex technical systems of our environment —transportation, infrastructure, medical, etc.— dependability has evolved from a necessary issue just in a handful of safety-critical systems to become an urgent priority in many systems that constitute the very infrastructure of our technified world: utilities, telecoms, vetronics, distribution networks, etc.

These systems are complex, large-scale and usually networked structures built to improve the efficiency of human individuals and organizations through new levels of physical integration, cognitive integration, control and communication. However, the increased scale, distribution, integration and pervasiveness is accompanied by increased risks of malfunction, intrusion, compromise, and cascaded failures. Systems do not only fail due to their defects or their mismatches with reality but due to their integration with others

that fail. Improving autonomy into these systems can mitigate the effect of these risks in system dependability and even survivability.

Survivability (Ellison *et al.* 1997) emerges as a critical property of autonomous systems. It is the aspect of system dependability that focuses on preserving system core services, even when systems are faulty or compromised. As an emerging discipline, survivability builds on related fields of study (e.g. security, fault tolerance, safety, reliability, reuse, verification, and testing) and introduces new concepts and principles.

A key observation in survivability engineering —or in dependability in general— is that no amount of technology —clean process, replication, security, etc.— can guarantee that systems will survive (not fail, not be penetrated, not be compromised). These is so because the introduction of new assets into the system, while solving some problems, will add new failure modes both intrinsic to the new assets or emergent from the integration.

Of special relevance in the case of complex autonomous information-based systems is the issue of system-wide emerging disfunctions, where the root cause of lack of dependability is not a design or run-time fault, but the very behavior of the collection of interacting subsystems. In this case we can even wonder to what degree an engineering-phase approach can provide any amount of increased survivability or we should revert to the implementation of on-line survivability design patterns than could cope in operation time with the emerging disfunctions.

In the following sections, we shall try to explore how the concept of autonomy is understood in artificial systems and how the ASys and SOUL projects are addressing these issues and entering the deep waters of machine consciousness.

Product Lines for Autonomy

The ASys Project is a long term project of our research group that is focused in the development of science and technology for autonomous systems. The project has addressed so far the implementation of intelligent control technology and integration infrastructure.

Now our main involvement is the development of technology to increase the level of autonomy in complex systems by means of metacontrol loops. These metalevel control loops implement strategies similar to those of supervisory or adaptive control but the focus of the control is both:

- the already addressed rejection of disturbances in the plant and, more interestingly,
- the rejection of disturbances in the contoller itself, that is gaining in importance as the controllers grow in complexity (Sanz *et al.* 2000).

Engineering Product Lines and Families

The ASys development plans follow a product line/product family strategy (Bass, Cohen, & Northrop 1996). As the SEI software product line says[1]:

> *A software product line is a set of software-intensive systems that share a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.*

So, our approach can be described as a product line approach to real-time cognitive systems engineering. We plan to provide technological assets for engineers building real-time cognitive systems. This implies many different aspects, from core embeddable real-time controllers to high level *thinking* mechanisms or integration infrastructure. The core aspect under current research is a set of architectural assets to support system self-awareness and meta-control.

A product line is characterised for it addressing a specific application niche (e.g. intelligent control systems). This is a perspective from the side of the user. A product family is characterised for products being built from a common asset base. This is a perspective from the side of the developer. Obviously, what we try to do in ASys is to build product lines as product families; this is the most effective approach to product line engineering. If we are successful we will be able to provide resources to real-time intelligent systems developers to build concrete systems that would fit custom set of requirements.

An Autonomy Product Family

Autonomous systems are constituted by physical and logical subsystems —the body and the mind. The particular implementation of each one will depend on the characteristics of the activity, the environment and the resources available (Sanz, Matía, & Galan 2000). The autonomy product line addressed by ASys has minds as its central niche product. Bodies are typically application-bound but this seems to be not the case of minds, where a general purpose mind seems more plausible (obviously with the corresponding load of trade-offs).

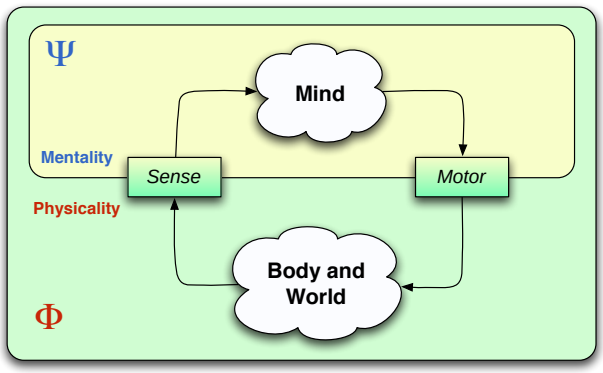

Figure 2: The Psy-Phi frontier between physical and logical subsystems of an autonomous system. The physical aspects of the body and surrounding world impose real-time constraints in the operation of the mental part.

The ASys project has as central requirement the scalability of the technology for autonomy across many dimensions:

[1] http://www.sei.cmu.edu/productlines/about_pl.html

- **Space** (localisation): from localised systems to complex wide-area plants.
- **Time** (multiple time-scale loops): from slow to fast and to hard real-time.
- **Rationality** (levels of thought): from minimal intelligence to human-level and beyond.
- **Size** (problem dimension) from embedded to mainframe hosts.
- **Precision** (uncertainty) from crisp to fuzzy processing.

These problems have been addressed —with more or less success— by diverse control system technologies but one of the remaining big challenges —in a sense the only remaining challenge— of any control design paradigm is being able to handle complex systems under unforeseen uncertainties. This is the old-age problem of dirty continuous process plants or the core problem of mobile robotics (or of any technical system that must perform in an uncontrolled environment). This problem is not restricted to embedded control systems but appears in many large system situations; e.g. web server scalability or security problems in open telecommand systems are examples of this.

The problem can be traced back to the difficulty of designing a strategy for survivability that can be scaled in the same dimensions. Extant solutions for resilient system technology do not scale well. This is the reason why we have added a new aspect into the ASys core requirements: the capability of the technology to handle itself. This has also been addressed in other fields —e.g. autonomic computing— but not considering the problem of domain-neutrality of the technology and its scalability. This adds a new dimension to the previous list:

- **Awareness** (meaning generation): from world to other minds, self and consciousness.

So the current technological objective of ASys is the elaboration of design patterns and reusable components for system self-awareness. Obviously, this objective is in strong relation with other researches in artificial consciousness (Chella & Manzotti 2007) but in some aspects it departs from some of the core ideas centered around phenomenology. Our main concern in this endeavour is not *qualia* but *self* —both perception and control.

Operational Aspects of System Autonomy

The general principle for autonomy in artificial systems is adaptivity. This enables systems to change their own configuration and way of operating in order to compensate for perturbances and the effects of the uncertainty of the environment, while preserving convergence to their objectives. A series of aspects are studied in artificial systems in order to enhance adaptivity: cognition, modularity, fault-tolerance, etc.

From an analytical point of view, the adaptation process depends on three system aspects:

1. the intrinsic adaptability of the system in the sense of minimisation of real structure (López 2007) — *cf* general systems theory
2. the observability of system state — *cf* control systems theory
3. the capability of structure modification by the system itself

Is in the point of observability where self-awareness play a critical role. The approach that we must take to provide the required generality implies the formulation of a general theory of how a system can observe its own functional state and derive actions from it. This implies not only the possibility of measuring certain system quantities but also the need of *understanding the functional implications* of the observed state. This understanding and the associated adaptation mechanism imply that the mind is able to separate the physical part of Figure 2 into body and world, *i.e.* the autonomous agent has a perception of its selfhood (see Figure 3).

This approach to system organisation and adaptivity is what we are pursuing in the development of the SOUL architecture; a novel, general architecture for real-time, self-aware cognitive controllers.

The nature of "self" in ASys

From the perspective of the ASys theory of autonomous systems, the perception of any entity is necessarily based on the existence of a mental referent for such an entity (López 2007). These mental referents are organised into integrated models that sustain mental operations (see next section).

This implies that the observation and understanding of the system by the system itself requires for it to have a mental referent for itself. This mental referent, this model of itself

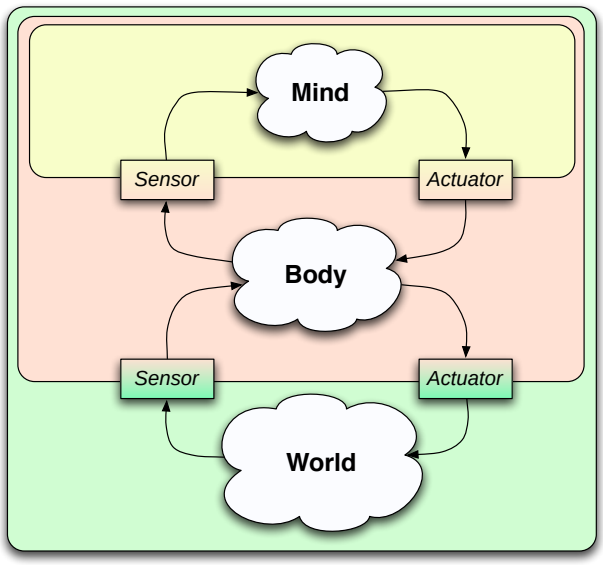

Figure 3: The Psy-Phi frontier between physical and logical subsystems of an autonomous system. The physical aspects of the body and surrounding world impose real-time constraints in the operation of the mental part.

that the system necessarily has to perceive its own state constitutes the very "self" in the ASys theory.

The perception of system self state follows a similar structural path as the perception of the state of the world (flows 6 and 1 in Figure 4). This implies that the mechanics of self-awareness must be the same as the mechanics of world-awareness. This is in strong opposition to other theories of consciousness that require very different mechanisms for both (Gallagher & Shear 2000).

This also implies a very interesting fact for artificial systems: the possibility of unifying perceptual patterns into general frameworks, reducing the development effort and maximising reusability.

Seven Principles for Consciousness

Instead of following the axiomatising approach proposed by Aleksander (Aleksander & Dunmall 2003) we propose seven guiding principles for the ASys research line on machine consciousness (Sanz *et al.* To appear). These principles are based on the core idea that minds are integrated, model-based controllers.

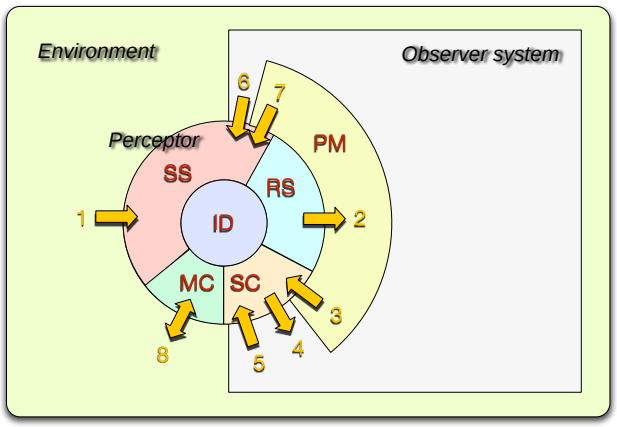

Figure 4: The autonomous system perceptor organisation following (López 2007). SS = sensory system, RS = representation system, PM = perceptual memory, SC = substratal coupling, MC = marginal coupling.

The principles are:

1. **Model-based cognition** — A system is said to be cognitive if it exploits models of other systems in their interaction with them.
2. **Model isomorphism** — An embodied, situated, cognitive system is as good as its internalised models are.
3. **Anticipatory behavior** — Except in degenerate cases, maximal timely performance can only be achieved using predictive models.
4. **Unified cognitive action generation** — Generating action based on an unified model of task, environment and self is the way for performance maximisation.
5. **Model-driven perception** — Perception is the continuous update of the integrated models used by the agent in a model-based cognitive control architecture by means of real-time sensorial information.
6. **System awareness** — A system is aware if it is continuously perceiving and generating meaning from the countinuously updated models.
7. **System self-awareness/consciousness** — A system is conscious if it is continuously generating meanings from continously updated self-models in a model-based cognitive control architecture.

Obviously, these principles translate the understanding and engineering of *consciousness* into the problem of *meaning*, that in the ASys approach is based on the application of value systems to potential trajectories in systems state space.

Conclusions

The development of the SOUL architecture is at the beginning phases. Current work is addressing the construction of a software ontology for autonomous systems and the development of system and software models using state of the art technology for model-driven development.

The hope in ASys is that we will be able to use automatic transformation of ASys/SOUL models to generate the very models that the system is using in real-time operation. This includes the referents used in perception —including *self*— and the inverse models used in action generation.

References

Albus, J., and Meystel, A. 2001. *Engineering of Mind: An Introduction to the Science of Intelligent Systems*. Wiley Series on Intelligent Systems. New York: Wiley.

Aleksander, I., and Dunmall, B. 2003. Axioms and tests for the presence of minimal consciousness in agents. *Journal of Consciousness Studies* 10(4-5):7–18.

Bass, L.; Cohen, S.; and Northrop, L. 1996. Product line architectures. In *Proceedings for the International Workshop on Development and Evolution of Software Architectures for Product Families*.

Chella, A., and Manzotti, R., eds. 2007. *Artificial Consciousness*. Exeter, UK: Imprint Academic.

Christensen, W. D., and Hooker, C. A. 2000. Autonomy and the emergence of intelligence: Organised interactive construction. *Communication and Cognition - Artificial Intelligence* 17(3-4):133–157.

Conant, R., and Ashby, W. 1970. Every good regulator of a system must be a model of that system. *International Journal of System Science* 1:89–97.

Ellison, R. J.; Fisher, D. A.; Linger, R. C.; Lipson, H. F.; Longstaff, T.; and Mead, N. R. 1997. Survivable network systems: An emerging discipline. Technical Report CMU/SEI-97-TR-013, Software Engineering Institute, Carnegie Mellon University.

Gallagher, S., and Shear, J., eds. 2000. *Models of the Self*. Exeter, UK: Imprint academic.

González-Baixauli, B., and Laguna, M. A. 2003. Software process specification for product line approach. Technical

Report DI-2003-01, Departamento de Informática. Universidad de Valladolid.

López, I. 2007. *A Foundation for Perception in Autonomous Systems*. Ph.D. Dissertation, Universidad Politécnica de Madrid.

Nagel, T. 1974. What is it like to be a bat? *The Philosophical Review*.

Sanz, R.; Schaufelberger, W.; Pfister, C.; and de Antonio, A. 2000. Software for complex control systems. In *Control of Complex Systems*. Springer. chapter 7.

Sanz, R.; López, I.; Rodríguez, M.; and Hernández, C. To appear. Principles for consciousness in integrated cognitive control. *Neural Networks*.

Sanz, R.; Matía, F.; and Galan, S. 2000. Fridges, elephants and the meaning of autonomy and intelligence. In *IEEE International Symposium on Intelligent Control, ISIC'2000*.