Semiotic Oriented Autonomous Intelligent Systems Engineering

Rodrigo Gonçalves
rodrigo@dca.fee.unicamp.br

Ricardo Gudwin
gudwin@dca.fee.unicamp.br

Department of Computer Engineering and Industrial Automation - DCA - Faculty of Electrical and Computer Engineering - FEEC - State University of Campinas - UNICAMP – Brazil

ABSTRACT

This work introduces a first proposal on how to use semiotics in order to improve software engineering methods, when intelligent autonomous systems are targeted. First we investigate the current flaws in software engineering, concerning intelligent autonomous systems. Then we propose a knowledge taxonomy, based on semiotic ideas, aiming a tool to understand the information domain of intelligent autonomous systems. Further, we illustrate on how to use the notion of knowledge types during the development of a simple intelligent autonomous system, emphasizing the relationship between the described types of knowledge aiming at the understanding on how they are organized into an intelligent autonomous system. After that we review the idea of generalized subsistence machine (GSM) proposed by Meystel as a possible tool (not the unique nor the best one) to represent the information domain of intelligent autonomous systems. Finally, it is shown how both ideas might be used in the requirement analysis step of any software engineering method when an intelligent autonomous system is targeted. As a conclusion, we discuss the future works and trends of semiotic intelligent autonomous systems engineering.

KEYWORDS: *artificial intelligence, semiotics, artificial life, autonomous systems, software engineering, UML, knowledge unit, knowledge space, knowledge taxonomy*.

1. INTRODUCTION

During the early years of the computer era (1950 to around of 1965), software development was an art that was virtually unmanageable. There was very few systematic software development methods and they used to be very specific and with a limited distribution. Software design, by this time, was an implicit process of one's head, and documentation was often nonexistent [9]. As soon as general-purpose hardware became a commonplace and multiprogramming and multi-user systems were introduced, the second computer era had started (around mid-60's to the late 1970). Many other new and powerful concepts, like real-time computing and database, had been introduced. The complexity of the applications and the large scale in which they started to be sold introduced a dark cloud on the horizon: the software maintenance. This fact fired the software crisis that started the third computer era (late 70's to late 80's). This era introduced many software engineering paradigms and tools. Now, with the appearance of object-oriented paradigm, powerful desktop computers, Internet, multimedia and artificial intelligence (AI), we are living the fourth computer era.

Since the beginning of the computer science, artificial intelligence was an aspiration. In fact, the wish for intelligent machines came before the idea of computers, as they are today (an approximation to Turing machines). AI techniques and methods have been emerging and getting better since early 60's when the mathematical logic was first applied to achieve artificial intelligence and an AI sub-field has emerged: the autonomous intelligent systems. This field emphasized the use of AI techniques applied to autonomous systems, i.e., systems capable of taking decisions in an autonomous way.

Nowadays we have many powerful theories, methods and paradigms related to AI but, unfortunately, we still lack of systematic AI systems development methods able to provide a formal way to fully understand the knowledge processing and communication inside them. Additionally, AI researchers and engineers have another problem: current software representation models are often inadequate for autonomous systems. Due to the fact that, for most of autonomous systems, it is not possible to determine *a priori* all possible states, modeling tools based on finite state automata theory, e.g. state charts and Petri nets, become inadequate. This problem is especially important in the development of evolutive autonomous systems. In this case not only the representation models are inadequate. Analysis and design paradigms, e.g. objected oriented, lack concepts that would allow dynamic data structures definition as well as a way to formally deal with evolutive functions and methods (e.g. genetic programming).

This paper introduces some ideas based on semiotics in expectation to help overcoming some of those problems. But, why semiotics? Why should software/systems engineers concern about semiotics? The answer is that semiotics provides a new analysis domain for AI problems, in a way comparable to when frequency analysis was first introduced in the domain of sign processing problems : a new perspective for the analysis of the same problems. The ideas introduced in this paper are a preliminary study on this direction, aiming at achieving new insights for the artificial intelligence analysis domain. The approach introduced in this paper is a first step toward an autonomous system engineering method based on semiotics and software engineering.

2. AIMING AT A GOOD INFORMATION DOMAIN MODEL

Computational systems are dynamic systems that can be modeled by a Turing machine. But, for the pragmatic task of programming computers, we don’t use the artifacts given by Turing machine theory. Humans have developed high level languages and engineering methods to make the development of complex systems possible and affordable within computers. Software engineering methods can be divided into three parts: Analysis, Design and Construction. The analysis step intends the achievement of a good problem understanding for consistent following steps. The quality, maintenance and expandability of the software are grounded in this step. Anywise, this step can also be divided into two other ones: Requirement Analysis and Domain Analysis. The requirement analysis aims at understanding the problem scope and the domain analysis aims for the identification and partition of software roles and tasks. To help us understand, let’s consider a reference model [6]: a business organization. If our goal is to design a software for the management of a business organization, during the requirement analysis we should ask ourselves: “what are the business concepts (e.g. sale, employee, etc) and processes (e.g. making sales, paying employees, etc…)?” In the same way, during domain analysis we would ask: “what are the employee roles?” The domain analysis builds a conceptual model that is not a description of software components; it represents concepts in the real-world problem domain, focusing in data processing aspects (information flow, information structure, and information content). This problem domain with this focus is called “information domain”. In the current work, we introduce a semiotic based strategy to analyze the information domain of autonomous intelligent systems aiming for a consistent analysis task. In other words, a tool to help in a consistent understanding and partitioning of the system specification targetting a computer implementation of autonomous intelligent system.

An autonomous intelligent system (IS) can also be seen as a set of concepts and processes aiming for an adequate information treatment and behavior generation. But, although we may use current software engineering methods to focus on the information domain, in this case they do not provide a formal way to understand the knowledge transforming and communication needed by any autonomous IS. In other words, in this case, current software engineering methods still lack of an adequate methodology for knowledge representation. The appropriate understanding of information domain is important to achieve a good allocation of information transforming methods into each system module. The lack of such methodology may result in systems that will be more complicate, will need more time to be developed and also get the risk of becoming unmanageable as they grow in complexity. The use of software engineering techniques is prescribed here as a way of getting those systems manageable and affordable. The lack of an adequate information structure may lead to difficulties in the use of multiresolutional approaches (what is sometimes vital concerning intelligent autonomous systems) and in the system scalability, understandability and maintenance. All those reasons encouraged us to propose new techniques that seem to be more adequate when intelligent autonomous systems are concerned. These techniques explore the semiotic nature of knowledge and its potential use in the construction of intelligent autonomous systems. In next section we present a knowledge taxonomy based on C.S.Peirce’s and Morris’ semiotics and how it can be used in order to obtain a good information domain model.

3. A KNOWLEDGE TAXONOMY

Peirce’s semiotics introduced a signical taxonomy, where different kinds of signs (e.g. rhemes, dicents, arguments, icons, indexes, symbols, qualisigns, sinsigns, legisigns) were proposed, addressing different characteristics of its structure and signic function. From Peirce’s taxonomy, Gudwin [4] derived a taxonomy of types of knowledge, where each type of knowledge addresses a different way in which phenomena from the world can be modeled. This taxonomy can be summarized in Figure 1 and will be explained in the following sections. In this figure *R* means *Rhematic*; *D* means *Dicent*; *Ic* means *iconic*; *Ob* means *object*; *Sp* means *specific*; *G* means *generic*; *Sy* means *symbolic*; *In* means *indexical*; *Se* means *sensorial*; *Oc* means *occurrence*.

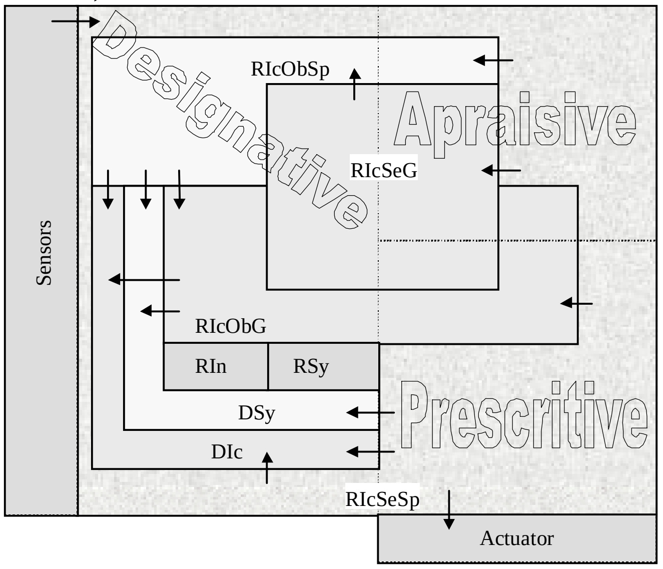

Figure 1 - Knowledge taxonomy

All arrows refer to argumentative knowledge (both analytic and synthetic). From this point forward, we will use the Figure 1 notation between brackets to specify the different kinds of knowledge. For example, {*RIcSeG*} means a rhematic iconic sensorial generic knowledge.

In this taxonomy a knowledge is first classified according to its functionality (designative, apraisive, prescriptive) and to its structure (rhematic, dicent). As we will see, argumentative knowledge is a special case and it is both a functional and a structural. Here we will present the structural classification before discussing about the functional one.

3.1 Rhematic knowledge

A rhematic ({*R*}) knowledge is the semantic that can be assigned to isolated words in a natural language. Usually, it is associated to a representation of environmental phenomena like sensorial experiences, objects and events. Sensorial experiences can be represented e.g. by adjectives, objects by substantives and events by verbs. In a last analysis, though, all of them are connected to perceptual data. The rhematic knowledge can be represented by an icon (rhematic iconic - {*RIc*}), a name (rhematic symbolic - {*RSy*}), or an index (rhematic indexical - {*RIn*}). A {*RIc*} knowledge is a direct representation of the phenomenon that it represents. A {*RSy*} knowledge is a name that refers the phenomenon. A {*RIn*} knowledge is an indirect reference to the phenomenon.

The {*RIc*} knowledge can be divided into three different classes: sensorial - {*RIcSe*}, usually modeled by adjectives; object - {*RIcOb*}, usually modeled by substantives; occurrence - {*RIcOc*}, usually modeled by verbs. Those three knowledges can be divided into two classes: specific (*Sp*) and generic (*G*). For example, a {*RIcSe*} knowledge is a sensorial information like an image or a temperature sensor output. A {*RIcSeSp*} knowledge is a particular instance of a sensorial pattern, e.g. a specific image, or the temperature in a given time. A {*RIcSeG*} knowledge is a generic knowledge of all time occurrences of some sensorial input. The knowledge that the outside temperature is 28 degrees Celsius is a specific sensorial knowledge but the knowledge of what is a high temperature is a generic sensorial knowledge.

The {*RIcO*}knowledge is the knowledge related to a real world object (existent or nonexistent). The {*RIcOSp*} knowledge is the knowledge of a specific occurrence of a specific object. It assumes an existence of an object model. This model is a {*RIcOG*} knowledge.

The {*RIcOc*} knowledge corresponds to the semantic of verbs. Usually, it associates an attribute to an object, or model the changes in one or more attributes in the object (or objects) as an effect of time. e.g. the property of an object of having a color, it's creation or destruction, etc. The {*RIcOcSp*} knowledge is related to a specific occurrence in time, e.g. the knowledge that a traffic light, in a specific time changed from red to green. The {*RIcOcG*} knowledge is related to a generic occurrence, e.g., the change of a traffic light from red to green, without specifying a particular instance.

3.2 Dicent knowledge

The dicent ({*D*}) knowledge is a proposition. The difference of a proposition and a term is that the proposition has a truth-value associated with it. Usually, this truth-value represents the belief of the proposition and it can vary from false to true using a multivalored logic or not (e.g. fuzzy logic). Compared to rhematic knowledge, we would say that if a rhematic knowledge represents the meaning of a single word in a natural language, dicent knowledge represents the semantic of phrases in a natural language.

A proposition is formed to the association of a term (or a set of terms) to a truth value. Propositions can also be formed by the association of more primitive propositions, linked by logical connectives. Examples: the knowledge that "A" is true; the knowledge that "$A \wedge B$" is false; the knowledge that "IF $A \wedge B$ THEN C" is true.

A dicent knowledge can be iconic ({*DIc*}) or symbolic ({*DSy*}). The {*DSy*} knowledge is a name for a whole phrase, and consequently, has attached a truth value, e.g., a label for a first order logic sentence. The {*DIc*} knowledge explicits its composing rhematic pieces of knowledge, associating to them a measure of the belief of this phenomenon occurrence.

3.3 Designative knowledge

Until now we presented a structural classification of knowledge. Now, we start discussing a functional one. The first functional category is the designative knowledge. Designative pieces of knowledge are employed to represent the world. An alive organism usually initiates its life with almost none designative knowledge.

A designative knowledge can be at the same time rhematic and designative as depicted in Figure 1.

3.4 Apraisive knowledge

Apraisive knowledge is used to make a judgment according to some objective. In an alive system this objective can be innate or learned. In this case the apraisive knowledge can be, for example, love, hate, pain, pleasure, desire, etc… An apraisive knowledge is usually classified as rhematic iconic.

3.5 Prescriptive knowledge

Prescriptive knowledge is used to plan, predict and actuate in the real world through actuators. In the same manner of apraisive knowledge the prescriptive knowledge is usually classified as rhematic iconic.

3.6 Argumentative knowledge

Before introducing the argumentative knowledge it is necessary to introduce the concepts of knowledge unit and knowledge space. A knowledge unit is an atomic structure of information carrying some meaning, i.e. modeling at a particular level of resolution, a phenomenon from the world. A knowledge space is a space that stores knowledge units. In this context, only the structural knowledge taxonomy is relevant because the concepts of knowledge unit and space are related only to knowledge's structure, not its functional purpose.

Argumentative knowledge is related to knowledge processing and transformation. It may be seen as an algorithm that creates or transforms knowledge units within a knowledge space. It is both a structural and functional classification. It is structural in the sense that it is a knowledge unit like any rhematic or dicent knowledge but its structure holds a program code instead of data. It is functional because it has an explicit functional role. A good metaphor for understanding what an argumentative knowledge is, is the representation of machine instructions within a computer memory. Those instructions can be seen both as data (a sequence of bytes) and code (processor instructions). In the same way, argumentative knowledge is both data (structure) and code (process). The argumentative knowledge ({*Ar*}) can be synthetic ({*ArSt*}) or analytic ({*ArAn*}). An {*ArAn*} knowledge unit is like a piece of code that creates new knowledge units that does not contain any new information - it only turns explicit an information that was implicit in the knowledge space. In other words, it only performs an "analysis" of existing knowledge units, making explicit something that was modeled into a more compact form. The most important {*ArAn*} is the "modus ponens". Different from the {*ArAn*}, the {*ArSt*} knowledge creates knowledge units that contain new information. In other words, it synthesizes new knowledge content. There are two kinds of {*ArSt*}: inductive ({*ArStId*}) and abductive ({*ArStAb*}). The {*ArStId*} knowledge makes small modifications in the premises knowledge units to produce a new one. In this sense, one can say that it is a constructive argument. One example of {*ArStId*} is the generalization. The {*ArStAb*} selects candidate units as true or false (at any degree) according to the pre-existent knowledge units in the knowledge space. The candidate units can be generated either by an {*ArStId*} knowledge or by any random method. Different of the {*ArStId*} knowledge, it is a destructive argument.

4. AN INFORMATION DOMAIN PARTITION

During the requirement analysis step of any software engineering method, our goal is to outline the scope of the system. We do this by finding its restrictions, objectives, inputs and outputs. After that, in the domain analysis, an information domain model is built focusing on "what" not on "how" [9]. In the semiotic point of view, the model should make clear the classification and distribution of all knowledge units of the system except the argumentative ones.

The knowledge taxonomy previously presented can be used to help us understand the information domain of the system. This approach may help us achieve better distributed and more representative models. Classifying the inputs, outputs, objectives and restrictions is the first step to achieve a good information domain model. As we said before, this classification leads to a better understanding of the information domain. At this step the focus should be on the information structure, not on the information functionality. Aiming a better understanding on how the information can be classified, lets consider the Terzopoulos' artificial fishes (Figure 2) as a case of study [10]. This example is very adequate because it is not too complex, it is ludic and it was not designed with the help of semiotics.

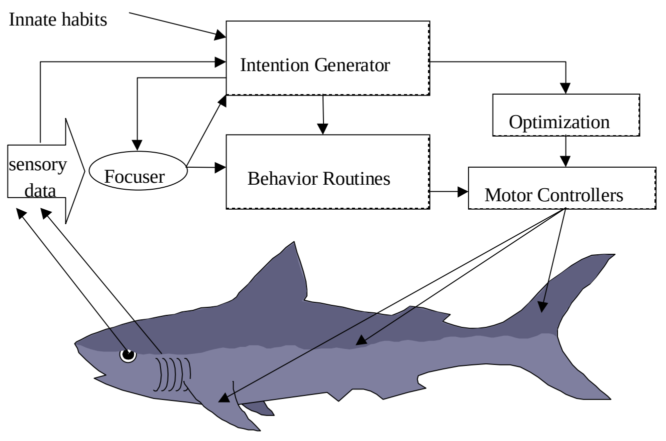

Figure 2 - Artificial fish model

The artificial fishes have innate knowledge units that determines whether or not it is male or female, predator or prey, if it likes schooling or not, if it likes brightness or darkness, cold or warmth, and so forth. Using the knowledge taxonomy we can say that these knowledge units are apraisive, more specifically {*RIc*}. Other apraisive knowledge units present in the artificial fishes model are the mental state variables: hunger (H), libido (L) and fear (F). Note that the mechanism used to calculate them is due to argumentative knowledge units. The behavior of an artificial fish is generated using intention generation and behavior generation modules based on predefined rules. Those rules correspond to prescriptive knowledge units ({*RIc*}). The artificial fishes learn how to coordinate their actuator to produce a coherent movement. This knowledge is prescriptive too.

Once understood the information structure (from the semiotics point of view), it is possible to understand the information functionality. At this point we will recall the general subsistence machine (GSM) concept. Meystel [7] has proposed the GSM architecture as "*a system which is unified by a goal to exist as an entity. In pursuit of this goal, GSM can perform tasks which has been developed internally or submitted externally*". The GSM architecture is very adequate to autonomous systems modeling but it would be naive to imagine that it is the only or the best architecture. In this work, the GSM structure is adopted due to the fact that it is generic and promising. Any other model can be used instead of the GSM, though. For that, it is only necessary to understand the functionality and information distribution of such model in the semiotic point of view. The GSM architecture can be seen in the Figure 3. The GSM modules are perception (P), world model (WM), value judgement (VJ), behavior generation (BG), "S" means sensors, "W" means world and "A" means actuators. Any of the GSM modules can be regarded as GSM too (embedded GSM).

In a first glance, this definition seems to be very adequate for autonomous agents and the like but not for others IS as, for example, medical diagnosis expert systems. It is not true and it is easy to see that GSM structure bears for all intelligent

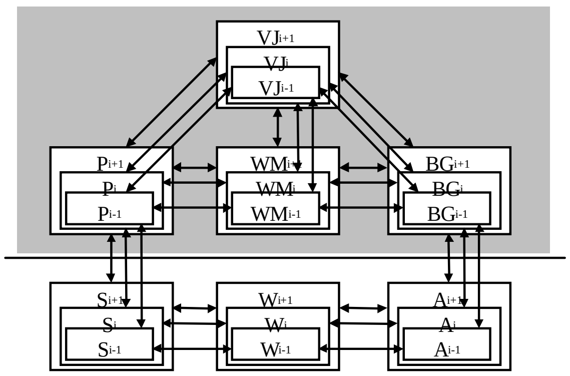

Figure 3 - Multiresolutional GSM structure

systems. In a medical diagnosis system for example, its “goal to exist as an entity” is related to its success in performing good diagnostics. Its actuators might be a GUI that make queries for the user and the sensors might be a GUI that receives his answers.

Based on the fact that the GSM maps intelligent systems, we can use its structure as a starting point for a domain analysis model. This approach leads to a multiresolutional representation of the IS that might be very interesting in most cases but it’s necessary to certificate that the IS mapping into the GSM will be adequately done. To make it possible it is necessary to know how to distribute our system elements into the GSM parts, otherwise the system model might turn to be confusing and unmanageable.

To ensure a properly autonomous IS information domain mapping into a GSM structure, two points should be observed: the function and objective of each GSM component and which kind of knowledge each of them supports. The first point should be carefully observed to avoid mistakes. When the engineer is working over a GSM-based IS information domain model he must remember that any GSM part can be a nested GSM. In a nested GSM the meaning of each GSM part changes according to its location. For example: a perception module is responsible for a sensory processing. So, a world model of a GSM nested in a perception module may contain a model of the sensorial space and the actuators may be something like a focus system. For a complete description of each GSM part please refer to [7].

The second point to be observed turns clear the importance of semiotics in the information domain analysis. GSM parts usually hold the same kind of knowledge, no matter if it is embedded or not. Thus, a prior knowledge of the knowledge distribution in the GSM structure helps us to build consistent information domain models. Table 1 shows the taxonomy of the knowledge contained by each GSM part. The notation "X $\rightarrow$ Y" means an argumentative knowledge that transforms a knowledge type X to a knowledge type Y.

Let’s recall the artificial fishes example again. In this case we can see that both the intention generator, behavior generator and optimization modules hold prescriptive and designative knowledge units. This suggests that they can be combined into one single behavior generation module in different hierarchical levels (nested GSM). The focuser acts into the sensorial space. It uses some prescriptive knowledge to change the designative knowledge generation. This fact suggests that the focuser is part of an embedded GSM in the perception module. The apraisive and some designative knowledge used by the intention generator to control the focuser can be grouped into a world model module, completing a GSM (Figure 4). The model obtained using the semiotics analysis structurally differs from the original model depicted in the Figure 2 but it is functionally equivalent. You might wonder why the second model is better if both of them are functionally equivalents. The main reason is that the first model may not exist, in other words, creating a model like the one depicted in the Figure 2 for autonomous systems is not a trivial task – that’s why people study architectures for autonomous systems. Using a generic architecture for autonomous intelligent systems as a start point is a good idea. The semiotic and more specifically the proposed knowledge taxonomy help us to find out how a specific system maps in a generic architecture like the GSM. Again it is important to note that the GSM is a possible architecture, not the architecture. Any other autonomous intelligent systems architectures can be used once fully understood the information exchange and processing in the semiotics point of view.

Knowledge	P	VJ	WM	BG
Apraisive		X		
Designative	X		X	
Prescriptive				X
{RIcSe}	X	X	X	X
{RIcO}	X		X	
{RIcOc}	X		X	
{D}			X	X
{RIcSeSp} $\rightarrow$ {RIcSeG}	X		X	
{RIcSeSp} $\rightarrow$ {RIcOSp}			X	
{RIcSeG} $\rightarrow$ {RIcOSp}			X	
{RIcSeG} $\rightarrow$ {RIcOG}			X	
{RIcOSp} $\rightarrow$ {RIcOG}			X	
{RIcSeSp} $\rightarrow$ {RIcOcSp}			X	
{RIcSeG} $\rightarrow$ {RIcOcG}			X	
{RicOcSp} $\rightarrow$ {RIcOcG}			X	
any {RIc} $\rightarrow$ {RIcSeSp}				X

Table 1

Another important point is that the resulted model should be built in conformance with the Analysis, Design and Project paradigm adopted. For example, if the analysis method is the *structured analysis* the GSM structure should be represented as a data flow diagram (DFD). In this case, many DFDs in different detail levels can represent the GSM. In an *object-oriented analysis* (OOA) the GSM structure can be easily

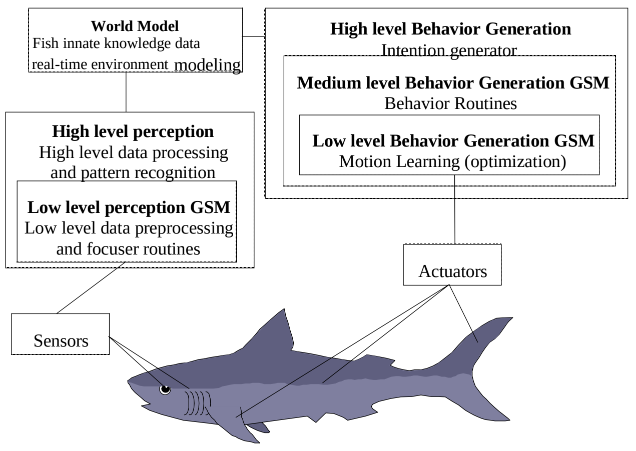
World Model
Fish innate knowledge data
real-time environment modeling
High level Behavior Generation
Intention generator
Medium level Behavior Generation GSM
Behavior Routines
Low level Behavior Generation GSM
Motion Learning (optimization)
High level perception
High level data processing
and pattern recognition
Low level perception GSM
Low level data preprocessing
and focuser routines
Actuators
Sensors

Figure 4 - GSM based artificial fishes

mapped into an object-oriented structure. We will introduce in future works how this method integrates with a specific software engineering method called UML (Unified Modeling Language) [6].

5. CONCLUSION

This work introduced important issues concerning the development of a semiotic oriented software engineering methodology, aimed for the analysis and design of autonomous intelligent systems. First of all, we emphasized the utility of discriminating among different types of knowledge, when considering the domain of intelligent autonomous systems as a development platform. The hierarchy developed by Gudwin is a good starting point, in this case, that allows us to discriminate on the different functions performed within an autonomous intelligent system and also point out concepts of different nature that need to be treated in these systems. Differently from common software systems, intelligent autonomous systems concepts and processes are directly related to knowledge and knowledge processing. At this point, the introduction of the concepts of knowledge unit, in its structural and functional forms proves to be very useful during the targeted analysis development phase. In future works we will address the relevance of this concept in others development phases.

The definition of knowledge space allows the determination of an architecture suitable to handle the particularities of such class of software systems. The semiotic nature of different types of knowledge is the guideline on the whole process. Understanding the subtleties among the different knowledge types, and the knowledge types required in order to build an intelligent autonomous system, allows a better comprehension of its behavior, and also makes more easy its development because it allows an easier application of known autonomous intelligent systems architectures.

Most of the work in autonomous intelligent systems architectures are top-down in the sense that they first propose a model, showing their capabilities and characteristics and then try to apply these models in real problems. At this point they have problems because the low level, in other words, the real problems specifications, often have excess of information and it is difficult to understand the mapping from the low-level to the high-level (from the real problem specification to the autonomous intelligent systems architecture). The work presented in this paper is a first step toward a methodology to allow the usage of autonomous intelligent systems architectures in more reliable way.

This is a work in progress. Next publications should include a more in depth discussion concerning the concept of knowledge space and knowledge unit and address others software engineering phases as development and construction. Some work regarding a development the development phase is in progress. As a preliminary results we can point out the development of the Object Networks (ON) [4][5] and the Fielded Object Networks (FON) [3]. They are both frameworks to implement intelligent systems and will be used in a further development phase development.

As we said before, this work is the first step towards a semiotic engineering method and does not adopt any specific software engineering methodology. Future works should introduce how these concepts apply in a specific method such UML.

6. REFERENCES

[1] Albus, J.S., *Outline for a Theory of Intelligence*, IEEE Transactions on System Man and Cybernetics, Vol. 21, No. 3, May/June 1991.

[2] Coleman D., *Object-oriented development: the fusion method*, Prentice-Hall Inc, New Jersey, 1996.

[3] Gonçalves, R., Gudwin, R., Gomide, F., *Fielded Object Networks as a Framework for Computer Intelligence*. Proceedings of the ISAS'98 – Inteligent Systems and Semiotics: A Learning Perspective –Gaithersburg, MD, USA – Semptember, 1998.

[4] Gudwin, R., *Contribuições ao Estudo Matemático de Sistemas Inteligentes – PhD Thesis – DCA-FEEC-UNICAMP, May 1996 (in portuguese).*

[5] Gudwin, R. R., Gomide, F., *An Approach to Computational Semiotics*. Proceedings of the ISAS'97 – Inteligent Systems and Semiotics: A Learning Perspective –Gaithersburg, MD, USA – 22-25 Semptember, 1997.

[6] Larman, C., *Applying UML and Patterns: an introduction to object-oriented analysis and design*, Prentice Hall, 1998

[7] Meystel, A. M., *Intelligent Systems: A Semiotic Perspective*, International Journal of Intelligent Control and Systems, Vol. 1, No. 1, pp. 31-57, 1996.

[8] Peirce, C. S., *Collected Papers of Charles Sanders Peirce*. Edited by Charles Hartshorne and Paul Weiss – Belknap Press of Harvard University Press – Cambridge, Massachussets, 2nd printing, 1960.

[9] Pressman, R. S., *Software Engineering a Practitioner's Approach – third edition*, McGraw Hill, 1992

[10] Terzopoulos D., *Artificial Fishes*, Artificial Life, Volume 1, Number 4, MIT Press, 1994