Biol Res 40: 503-515, 2007

Programming Paradigms and Mind Metaphors: Convergence and Cross-fertilization in the Study of Cognition

DIEGO COSMELLI[1], JORGE SOTO-ANDRADE[2] and ERIC TANTER[3]

[1] Laboratorio de Neurociencias, Facultad de Ciencias Sociales, Escuela de Psicología, and Laboratorio de Ciencias Cognitivas, Departamento de Psiquiatría, Escuela de Medicina, Pontificia Universidad Católica de Chile, Santiago, Chile. E-Mail: dcosmelli@uc.cl
[2] Mathematics Department, Faculty of Sciences, Universidad de Chile, Santiago, Chile. Email: sotoandr@uchile.cl
[3] PLEIAD Lab, Computer Science Department (DCC), Universidad de Chile, Santiago, Chile. Email: etanter@dcc.uchile.cl

ABSTRACT

This paper describes a notable convergence between biological organization and programming language abstractions. Our aim is to explore possibilities of cross-fertilization, at both conceptual and empirical levels, towards the understanding of what cognition and cognitive systems might be.

Key terms: Programming Paradigms, Biologically Inspired Computing, Cognitive Systems, Software Agents, Regional Ontologies.

What does an essay on programming paradigms have to do with the study of the brain, nervous systems and cognition? One straightforward answer would be that through the development of ever more complex programs and sophisticated computational systems one could expect producing proper models of system such as neurons, neural networks, brains and ultimately nervous systems. While this is indeed an important part, it is obviously not the only case, as the computation-biology interface is an active research domain. Notable developments include, but are not limited to, modeling living beings (Varela et al., 1974) and their sub-processes (Fernandez-Ballester & Serrano,2006), genetic algorithms (Burtsev & Turchin, 2006), cellular automata (Wolfram, 1984), robotics (Steels & Brooks, 1995)], artificial intelligence and intelligent agent-design [McDermott, 2007; Sun & Franklin, 2007), neuronal-prosthetics (Fromherz, 2006), large-scale modeling of nervous systems (http: //bluebrain.epfl.ch/), machine consciousness (Aleksander, 2005), computational correlates of consciousness (Cleeremans, 2005), and so on.

The motivation of this paper is to navigate in the less-explored relation between biological theory and programming paradigms (OOPSLA 2006, 2006), with the aim of highlighting a series of converging ideas. We will argue that in the interphase of these two domains resides an important *metaphorical pool* to deal with some basic questions regarding understanding and modeling systems which can make sense of their current situation.

It is no news that software development faces a challenge with the explosion of highly extended, pervasive, and ultra-large scale computational systems (The Software Engineering Institute (SEI), 2006). The proliferation of mobile devices and highly interconnected systems in the face of multiple users in changing conditions represents a radically dynamic context.

Corresponding Author: Dr. Diego Cosmelli dcosmelli@uc.cl

Received: October 10, 2007. Accepted: March 3, 2008

Because programming paradigms and languages deal with the nature of and interaction among computational entities – and by extension with the interactions between computational systems and human beings–, research in these areas deals in a very concrete manner with the question of making sense out of a context of interaction or a certain environmental situation; a challenge that resembles the general issue of cognition.

The brain and the nervous system have historically been recognized as playing a fundamental role in supporting what we understand as cognitive capacities (Ashby,1952; Maturana, 1970b; Kandel & Jessell, 2000). As living beings incorporated nervous systems into their repertoire of biological tools, autonomous movement and perception made its full-fledged appearance, and with it, the enormous cognitive possibilities of sensorimotor animated beings. However, understanding *how* this role is played in the context of the living animal is far from clear. The question for the mechanisms of human and animal cognition, of how such systems appear to be able to establish meaningful points of view in an apparently objective world, is a challenge that has been called the "holy grail" of contemporary science (Dennett, 1991; Niedermeyer, 2005).

Drawing on an *analogy* between programming paradigms and biology could be considered far-fetched: meaning for a software agent has nothing to do with meaning for a rabbit. In other words, a mere as-if would have no use in explanatory attempts at a given problem. We wish to take stance against such argument. Throughout history, technological metaphors have been one of the main explanatory bases for the understanding of natural phenomena (Freeman, personal communication). Similarly, natural systems serve systematically as models for technological development. Even more than through deductive reasoning, it is through metaphors and analogies that insight often comes to be, usually for the benefit of many sides of the problem. Metaphors do not simply enable illustrative comparisons but can act as powerful cognitive tools (Soto-Andrade, 2007). While we do not pretend to answer the question of cognition or meaning we do contend that, on this rather thorny issue, the search for cross-fertilization is not only necessary and beneficial in theory, but already at work and worth taking into account seriously.

We will begin with the question of defining cognition as a way of setting a common conceptual background. To this end we will review three well-known approaches to this problem, namely cognitivism, connectionism and embodiment. We will follow with a brief survey of the history of programming paradigm research, with special attention to the underlying process of abstraction. We will then present some examples of convergence between the development of programming paradigms and some themes in biologically-oriented approaches to cognition. We will argue that these examples suggest relevant lines for *cross-fertilization* between disciplines. We end with a brief conclusion and round-up.

FROM COMPUTATIONAL COGNITIVISM TO EMBODIMENT: THE PROBLEM OF COGNITION

What is cognition? or to be more precise, what do we understand by cognition so that a cognitive system can be defined or distinguished in some operative way? It is obvious that dealing with this question is not trivial. Ramifications into almost any domain of human study can be found[1], so no simple answer will do. The following brief discussion in terms of different paradigms is therefore intended only as a working ground (see also Varela et al., 1997; Dupuy, 1999; Cisek, 1999; Thompson, 2007).

Cognitivism: into the inner workings

For classical cognitivism cognition consists essentially in the manipulation of discrete symbolic representational entities,

[1] Think of the number of disciplines that converge in the so-called Cognitive Sciences.

according to predefined rules. The quintessential theoretical account within this tradition is the Computational Theory of Mind. It is claimed that the mind is literally a digital computer, so that cognition is equivalent to computation[2]. From here numerous metaphors are available and play an important role in guiding further experiments and theorization: when we perceive and act we receive information from a world, process it and respond according to a sets of rules of evaluation, matching, decision making, etc; memory is a storage of something; attention is a matter of administering limited processing resources; consciousness, access to a widespread broadcast message (Baars, 2002). Accounting for *felt* meaning (Gendlin, 1997), as we human beings live it through, has always been a difficult task within this paradigm[3]. Nevertheless, it is noteworthy that a computational automated email response system built on the global workspace hypothesis (which implements basic computational cognitivist assumptions) could be capable of passing a (naive) Turing test (Sun & Franklin, 2007).

Delocalizing the process: connectionism

Through the study of complex natural and artificial systems, in particular model neural networks, the notion of distributed, non-localized, systemic processes has become well known (Prigogine & Nicolis, 1989). From self-organization in liquid media to social insects, the core idea is that through a set of local rules among simple components a global property obtains[4]. Such so-called *emergent* property pertains to the totality and is not localized in any subsystem of it (Shewmon, 2001).

In connectionist and related neo-connectionist approaches (Feldman & Ballard, 1982), properties of the system are delocalized, depending crucially on interactions (connections) among parts. The paradigmatic example is a neural network which through interconnected simple units (model neurons) is said to recognize complex patterns, discriminate, learn and so on. In addition to already adopting and profiting from cognitive, intentional terminology for the description of such systems, connectionism opens the place for novel metaphorical tools that work the other way. Symbolic entities can free themselves from discreteness, and cognition can be understood as a distributed, historical and interdependent process. Numerous approaches benefit some way or another from the conceptual tools that are available within this framework (Clark, 1997), including the one presented in section below. Cognition is now the dynamic emergence of coherent global states that are reached, upon a given environmental situation, through distributed cooperative interaction rather than through central, sequential processing. Importantly, in such systems, for global behavior to persist, the behavior of the parts is constrained reducing their degrees of freedom (i.e. 'enslavement' Haken,1983)[5].

In flesh and blood: the embodied mind

Embodied, enactive, and situated cognition approaches are in a way a mixture of heritages from cybernetics, biology and philosophy and take a more radical naturalistic perspective on cognition (Varela et al., 1997; Thompson, 2007; Noe, 2004; Clark, 1997; Cosmelli & Thompson, *In Press*). For embodiment[6], cognitive systems are systems that bring forth a contextually valid (ecological) world of meaning. This

[2] Under a given technical definition of what computation and a computer is ([Horst (Fall 2005 Edition); Putnam (1975); Fodor (1979)].

[3] Interested readers might wish to see for example John Searle's Chinese Room argument [Searle (1980)] and the discussion in [Thompson (1997)].

[4] The actual mechanism of this *obtaining* is not necessarily understood.

[5] In line with Cisek (1999), we would tend to see connectionism in this sense as a sophisticated variant of the more classical computational cognitivism: symbols are now simply dynamical and distributed representations, but which nevertheless are manipulated according to rules such as synaptic weights. However, we single it out because it opens the way to the notion of emergent distributed processes which have played an important role in both computer science [Bonabeau et al. (1999)] and neurobiology [Freeman (1999); Gelder (1998)]

[6] Used here as a synonym of all three terms above.

happens through a history of structural coupling between systems that actively maintain a self-asserted identity with an (only) partially predictable environment.

Living systems and living organization is the paradigmatic cognitive system under this perspective (Maturana, 1970a; Maturana & Varela, 1973; Varela, 1979)[7]. For the embodied approaches, the self-sustained body and the always present life-death tension is precedent to any symbol manipulation and foundational of any *value* whatsoever. Cognition is therefore endogenously motivated, exploratory and necessarily risky (Jonas, (2001(1979). In general, embodiment considers meaning as contextual, perspectival, and a matter of staying alive here and now: the world makes sense to me because whatever comes from it, I still have to stay alive. In other words, that which is not the system, i.e. the world, is now of the system's *vital* concern. While the intuitive appeal of this approach is considerable, operationalizations into actual experimental settings are still lacking.

A qualification: Hard and soft notions of cognition

The study of cognition is undoubtedly an open question. It appears that meaning, sense making and knowledge are at the heart of it, yet defining these three concepts is again a huge challenge we do not mean to attempt. One aspect does need to be pointed out, however. From an embodiment perspective, cognition is usually understood as dependent on physically being there, 'in flesh and blood', and any virtual model can only highlight structural determinants but not become or produce a cognitive agent. Alternatively, this phenomenon could be exhibited by synthetic virtual entities of some kind as long as certain conditions are fulfilled. Here we rejoin a long-standing debate in Artificial Intelligence and Artificial Life (Varela & Bourgine, 1992; McDermott, 2007; Steward & Mossio, (n.d.); Thompson, 2004). We will suspend for the moment the question of whether a synthetic virtual or physical model will be cognizant in the sense of being able to *experience* a shared world of meaning (Varela et al., 1997).

We invite the reader to keep in mind the different approaches to cognition that we have sketched above as we turn in the following to revise programming paradigm research development. This will set the stage for discussing the three lines of convergence we propose can be relevant to both the way further programming paradigms are developed and how mind and cognitive systems are understood.

PROGRAMMING ABSTRACTIONS: FROM PROCEDURES TO AMBIENT ACTORS

Abstraction in programming language has been the driving force behind research in programming *paradigms* (Hayes, 2003): the definition of a set of basic concepts and their rules of composition and interaction. For instance, in *functional programming*, a program is built up of functions calling other functions and returning values, pretty much like mathematical functions. On another trend, *logic programming* promotes the decomposition of a problem into a set of logic facts and rules, and uses inference as a means to compose these pieces together. In a *procedural* programming language, a program is made up of a set of data structures manipulated by procedures that can affect the state of the system as defined by the data structures. An *object-oriented* programming language adopts yet another means of structuring programs, as discussed later on.

The recurrent problems in the software industry have always pushed researchers to study new abstractions and paradigms for programming, with objectives such as conciseness of written code, understandability, extensibility, reusability, maintainability, etc. The ever-increasing and widespread use of information technology continuously brings new challenges and new degrees of complexity for software development. This pressure calls for new

[7] In this sense it is a rather strong framework as it equates cognition with a given way of being rather than an internal or external process of some kind, as was the case for the previous approaches.

paradigms addressing limitations of the established ones. Interestingly, as the level of abstraction rises, the *degrees of freedom* of the programmer get more and more restricted, because behind abstractions are a set of rules enforced by the language processor. But these very rules are in fact the enablers of abstractions, which make it possible for programs to address more and more complex tasks in a practical way. We now illustrate the interest of abstraction at the level of the programming language by giving a brief overview of the evolution of abstractions. We will start from procedural programming and work our way to the very recent paradigm of ambient-oriented programming, a paradigm that aims at addressing the challenging context of programming applications for a multitude of lightweight mobile devices carried by mobile users facing varying connectivity due to network limitations (Dedecker et al., 2006), resulting in a highly dynamic network topology.

The illustration of the power of programming language abstractions that follows is at best partial and incomplete: one cannot resume forty years of research in programming languages that briefly. Our aim is demonstrating in a concise manner that there are fundamental gains in tackling a given problem with a programming paradigm that integrates the appropriate abstractions, rather than building *ad-hoc* abstractions without adopting a paradigm shift.

Step 1: From Procedures to Objects

In traditional procedural programming languages, like C, Fortran, Pascal, and the like, a program is structured as a set of *data structures* on the one hand, which represent the state of the program, and *procedures* on the other. Procedures are composed by invoking each other and possibly returning values after performing some computation, which may depend on and affect the data structures.

History has shown that the separation of data structures and procedures is unfortunate because there is an inherent need for consistency between both (Hayes, (2003). For instance, the procedure that draws shapes on the screen is coupled with the types and representation of shape structures; updates to one imply updates to the other.

In the early 1970s, building upon his education as a biologist, Turing-award winner Alan Kay and colleagues came up with a fundamentally new abstraction for programming, using a clear inspiration from cells in living organisms: objects (Kay, 1993). In *object-oriented programming*, an executing program is made up of self-contained capsules, called objects, which provide a number of well-defined services available to the outside world, and *encapsulate* their own internal state and actual implementation of the services they provide (Fig. 1). Objects only communicate by sending messages to each other.

From a conservative viewpoint, this is nothing more than just packing data structures and procedures in the same units, and this model can be implemented using a procedural language. Yes, it is true that this is “just” such a packing, but precisely, it is nothing *less*. This paradigm shift then gave rise to a number of what could be called *accidental abstractions*, such as *classes* –describing the common structure and behavior of a set of similar objects–, *inheritance* –defining classes as incremental variations and extensions of more basic, abstract definitions–, *polymorphism* –the possibility to manipulate different types of objects uniformly–, etc.

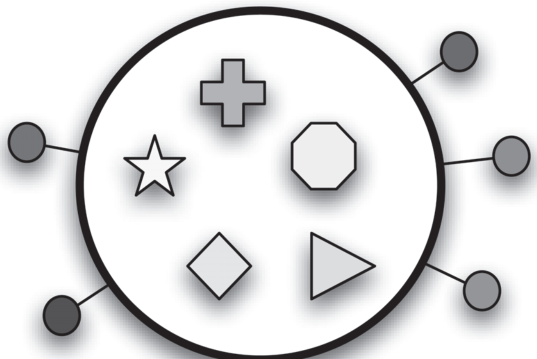

Figure 1: An object provides a set of services and encapsulates its internal state and implementation details.

All these accidental abstractions are, as a matter of fact, very hard to implement using a procedural language. The shift at the programming language level therefore brings new abstractions at a cost zero for the programmer; it is the implementor of the language processor that bears the increase of complexity. Programmers can then enjoy a convenient dedicated *syntax* to concisely define objects (Fig. 2), without necessarily knowing *how* an object-oriented program is reduced to a set of machine code instructions.

```
object Square {
  Position p;
  Size s;
  void scale(Integer factor){
    s.scale(factor);
  }
  void moveTo(Position p2){
    p = p2;
  }
  void draw(Screen s){
    s.drawLine(p.getX0(),p.getX1());
    ...
} }
```

Figure 2: An object-oriented programming language offers dedicated syntax for conveniently defining objects, their state (p, s) and behavior (scale, moveTo, draw).

Later on, as experience with these abstractions and mechanisms grows, higher-level programming idioms and patterns appear (Gamma et al., 1994). A typical industrial business application today easily has several hundreds of thousands objects dwelling within and makes heavy use of object-oriented idioms and patterns.

Step 2: From Objects to Actors

A further level of abstraction based on objects is the model of *actors*, also known as *active objects* (Agha, 1986; Briot et al., 1998). This model appeared as a response to the problem of managing *concurrent activities*, in an object-oriented program. Concurrency refers to the (at least simulated) simultaneous execution of different flows of computation within a program. In the traditional model of concurrency, objects are passive entities, and activity is driven by threads of control (Fig. 3) Birrell, 1989). A classic manner of explaining this model is that of bees (threads) flying around and visiting flowers (objects). This kind of concurrency model is easy to understand provided there are very few threads, or that the coordination between activities is kept to a strict minimum. However, as soon as some coordination between concurrent activities is needed, managing this kind of concurrency is a nightmare. Concurrent programming is notorious for its intricacies and it is very common that most computer science graduates do not get even a simple concurrency problem right. The thread-based concurrency model clearly diverges from the biological metaphor of cells: cells are not passive entities that wait for some thread of control to activate them and then stop meanwhile they exchange some chemicals with the outside world.

The actor model comes as an abstraction whereby threads and objects are not separate anymore. Instead, an actor is an active object with its *own activity*. The activity of an actor consists of processing the messages it receives from the outside world, in addition to its *own goals*, if any (much like we deal with our mail every day while pursuing some objectives of our own). Here an important shift has taken place: a first degree of functional autonomy, going beyond the previous passive encapsulation, is at work. The use of “actor” metaphor to refer to such constructs is illustrative of this fact. Indeed, one of the most important determinants of recognizing agenthood is the inference of inner purpose as a consequence of apparently self-initiated actions *on* the environment (Frith & Frith, 2007).

An actor language gives the linguistic means to define such active objects, the way they have to handle the messages they receive and emit their own, as well as their own pro-activity. Again, all the complexity entailed by the actor abstraction (i.e. synchronization matters) are handled by the language processor, not the programmer.

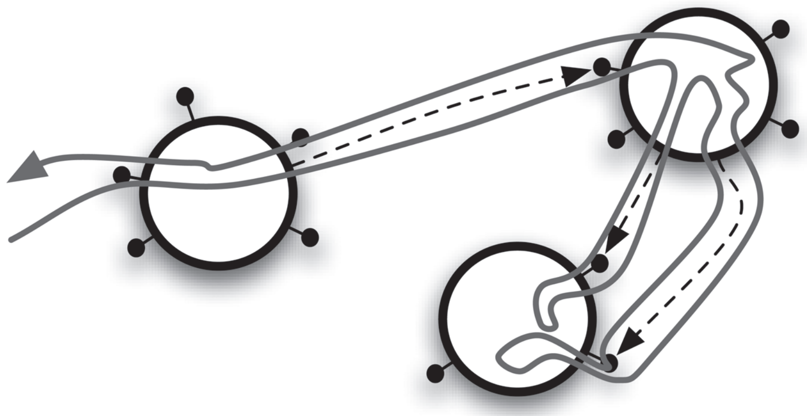

Figure 3: Thread-based concurrency with objects: each thread of activity traverses objects following the flow of computation.

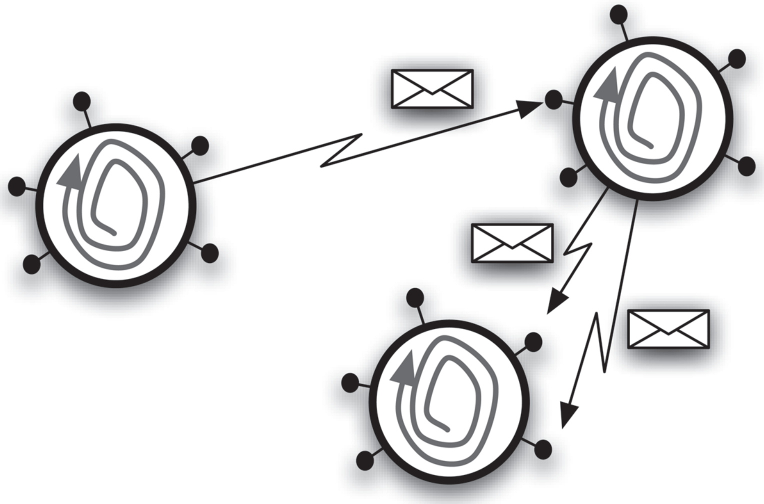

Figure 4: Actor-based concurrency: each actor has its own activity, communication between actors is asynchronous.

Step 3: From Actors to Ambient Actors

Considering the peculiarities of the modern computing infrastructure, which is made up of a variety of mobile devices held by mobile users, a number of strong challenges come up for software development: how to deal with a highly-dynamic topology, disconnections, failures, and volatility of resources? Here again, one can build up an ad-hoc solution whenever needed, but at which cost? *Ambient-oriented programming* (AmOP) (Dedecker et al., 2005) proposes to consider these issues as recurrent and complex enough to address them appropriately at the language level.

Another model has therefore been proposed, called the *ambient actor* model, which complements the above actor model with several characteristics. First and foremost, an ambient actor cannot be blind to its environment, which is dynamically changing: it needs to be able to *sense* and *adapt* to the ambient. To this end, an ambient actor explicitly states what services it *provides* to other actors in the ambient, and what services it *requires* from the ambient. It is also able to know which required services are bound in the current ambient, and when they are not bound anymore (*e.g.* because an actor was running on a machine that moved away). Similarly, in order to *rollback* some actions upon failure, an ambient actor has access to its *communication history* (Fig. 5).

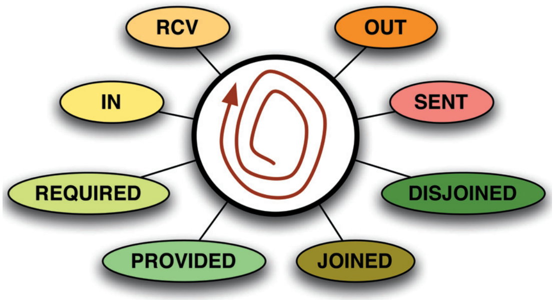

Figure 5: An ambient actor is an actor extended with a number of *mailboxes*: four are used to give access to the communication history (pending messages, processed messages, messages to send, sent messages); four are used to sense the environment (required services, provided services, services joined, and disjoined).

A very notable feature of an AmOP language like AmbientTalk (Dedecker et al., 2006), is the provision of a when construct (Fig. 6). This construct makes it possible to easily specify that *when* some condition is fulfilled (*e.g.* by the ambient), some action should be done. Programming languages typically do not have such a construct, rather they have an if construct. Although a when construct does not make it possible to *compute* something that is impossible to do without it, it does make it possible for a non-specialist to express such a statement. Manually implementing an equivalent of the when construct requires an in-depth knowledge of concurrency management, in order to avoid any subtle issue that would lead to program errors, and would be much less convenient to use.

CONVERGING METAPHORS

Apart from the evident inspiration on cells, actors and agents, there are a number of not so obvious lines of convergence between programming paradigm research and the study of mind and cognition. The following subsections present three examples.

On Computability, Abstractions and Reduction

From a *computability* point of view, most programming languages are equivalent in the sense that they are Turing-complete, that is to say, they can all express the same computations (Brainerd & Landweber, 1974). Therefore, *the abstractions embraced by a given programming language are, in the end, of no use in terms of computational power*. It is true that eventually, a program in a high-level language is interpreted (possibly after being compiled) by a device or program, called a *language processor*, that is at most equivalent in computational power to a Turing machine.

However, the theoretical argument of Turing equivalence does not take practical considerations into account (nor does it claim to), such as efficiency, difficulty, or time needed to address a particular problem with a given set of abstractions. Software programming is an engineering discipline, and as such must respond to criteria of feasibility. If it is true that even very huge software used today *could* have been programmed in assembly language or directly in machine code; however it is not so in practice.

```
actor {
  requires Printer
    onjoin: gui.showPrint();
    ondisjoin: gui.hidePrint() ;

  print(){
   printer = ambient Printer;
   answer = printer.print("hola");
   when(answer){ gui.done(); }
  }
}
```

Figure 6: Ambient actors in a programming language: specific syntax is there to declare a required service, what to do *when* it becomes available and not, to ask for an ambient reference, and to specify actions to perform *when* a result becomes available.

Now what happens in biological systems? From an orthodox physicalist perspective biology is a sub-ensemble of physics. This means that one should be able to reduce every biological phenomenon to physical laws (which do not need to be limited to the ones we know now) (Kim, 2000). One could be inclined, drawing from the previous discussion, to distinguish theoretical and practical reducibility: as you try to program (and therefore understand) a pervasive computing system using only assembler, imagine trying to study the nervous system of a relatively simple organism in terms of atomic interactions. In other words, while reduction might be possible *in theory*, this does not mean it is *practical* when trying to *understand* the workings of composite complex systems as these. Two important factors that might play a role in this being the case are, on the one hand, the nature of the relations established within such systems, and on the other, the casual efficacy of the systems' organization, whether it is abstraction in programming or organismic in biology (for a convergent idea see (Abbott, *submitted*). The latter is further discussed in the next subsection.

Keeping here to the nature of the relations established in complex systems, one could argue that the above question is close to the problem of statistical mechanics where a macroscopic coherent behavior emerges from microscopic random interactions. The only difference would reside in how much more numerous and diverse its 'parts' are so that our understanding of it is hampered only by resolving power. Indeed, mathematical models are quite successful in predicting the behavior of, say, a volume of a gas based on microscopic effects. However, what makes the difference is that both in nervous systems and pervasive computing systems, the degree of interaction and long-range interrelation is radically different. This is notoriously exemplified in the appearance of effective causal loops (what you see/request is dependent on how you move/provide and how you move/provide is dependent on what you see/request). These elements of circularity and self reference are absent in the gas or mechanical systems, but seem essential for the emergence of behaviors that are considered cognitive (Bell, 1999; Thompson & Varela, 2001).

Domains of Interaction and Regional Ontologies

Living systems sustain a (paradoxically) insubstantial identity in time. How they do so is an open question, but it appears that maintaining global identity depends to some extent on restricting the behavioral possibilities of the parts (Haken, 1983; Thompson, 2007). The degrees of freedom of the ensemble of macromolecules in the most humble bacteria is astronomical. Maintaining organization through the continuous change in components entails a reduction of degrees of freedom of the constituents (a paradigmatic phenomenon in highly interconnected complex systems, see section "Delocalizing the process: connectionism"). Importantly, as long as the system has to be kept organized by itself, a domain of behavioral interaction (i.e. of viability) for the ensuing whole will exist (Maturana & Varela, 1973). In this sense, biological organization is causally effective in the appearance of a world to be known.

Self-organization through enslavement is therefore at the basis of the appearance of what we will call regional ontologies. Regional ontologies, in a general sense, are *domains of meaning which depend on the particular organization of the making-sense system*[8]. In biology it is common to find such domains, or cognitive horizons, across multiple levels. The 'world' of a cell is not commensurable with that of the entire animal it is part of, however interdependent they might be. Nevertheless, both 'worlds' share the structural feature of *being relevant for* the cell or the animal. As stressed in the embodied perspectives both

[8] This notion of regional ontologies is comparable to the notion of cognitive domains [Maturana & Varela (1973); Varela et al. (1997)], and similar to the Gibsonian approach [Gibson (1979)]. By stressing both the "regional" and the "ontological" we wish to highlight its local validity and world-enactive structure.

units can be considered as establishing a domain of concern due to their particular organization and situation. They *make sense* of the environment they encounter. In a very concrete way, they constitute a *perspective* in the world[9].

As we have seen, a similar process happens in programming paradigms research through abstractions. Abstractions constrain the degrees of freedom of computational entities, upon which certain rules are imposed. Most importantly, successive abstractions through objects, actors, etc. *enable successive domains of practical competence* that were previously impossible, from a practitioner's perspective. Although such domains of interaction are still far from being meanningful in the strong sense described above, several very explicit primitives are recognizable: Object-oriented programming (Figure 1) enables a simple encapsulation of data and behavior that already facilitates collective functional interaction, but is radically passive; actors and active objects (Figure 4) incorporate an activity into the objects, and a certain level of *self-motivated* action in addition to processing requests and offering services; ambient actors (Figure 5) push further towards autonomous behavior by adapting to given environments through basic 'sensing'[10].

What this suggests is that abstractions in programming paradigms have similar consequences to those of different organizations in biological, natural systems; they bring forth quasi-cognitive domains that are structurally *coupled* to the implementation of the abstraction.

Evolutionary Dynamics

As we have seen, both in natural and computational systems, a certain organization is to be actively maintained in the face of dynamical environmental challenge. As both environments realize some level of reproduction (generations) of parent organizations, both systems are candidates for evolutionary dynamics.

We have seen how, in the rapid and unpredictable unfolding of the computational environment, abstractions accomplish efficient solutions. In many cases these are local, specific solutions to problems that arise precisely because of recently acquired possibilities of interaction. The latter are, in turn, dependent on previous abstractions, and so on. Moreover, abstractions usually have as a consequence novel possibilities that were not pursued when attempting to solve the initial problem. This is strongly reminiscent of an evolutionary drift (Maturana & Varela, 1984; Maturana & Mpodozis, 2000).

The evolutionary setting in programming paradigm research is obviously not that of organisms in natural ecological contexts. Yet it is notable that in the very short existence of the discipline of programming paradigms, already a trend towards increasing autonomy is clearly evident (section "Programming abstractions: from procedures to ambient actors"). Because of the nature of the problem posed to the programmer, this suggests, conversely, that autonomous organization might be an apt solution to the problem of dealing and making sense out of unpredictable, highly interactive contexts in effective ways. This necessarily falls back on our conceptualization of cognitive systems lending a 'plausibility' support to the main underlying assumption of embodied and enactive approaches to cognition (see section "In flesh and blood: the embodied mind"). Under this perspective the workings of the nervous system (or any

[9] The same goes for different animals. While we have a three-pigment visual system, the pigeon has a four-pigment based one [Thompson et al. (1992)]. It is not as if pigeons see more colors, it is rather that their color world probably has a different meaning for them than it has for us. This is, at least in part, because four dimensions are needed to specify their color spaces (while only three are needed for human beings, namely hue, saturation and brightness). Regional ontologies are not mappable unto one another, even if overlapping exists (which is obviously the case since actual experiments on the pigeon visual system can be done by human beings).

[10] Interestingly, the capacity to deal with the notion of 'when' in these software agents can also be seen as a primitive of a flexible, dynamical *temporality* going beyond the dichotomous 'if'. It is notable that the philosophical phenomenological tradition considers temporality to be fundamentally constitutive of meaningful experience (Cosmelli & Thompson, 2007).

subsystem) has to be understood first and foremost in terms of how it enables, or facilitates, autonomous behavior and not as how it captures a pre-given world, as several researchers have insistently pointed out (Maturana, 1970b; Steels & Brooks, 1995; Varela, 1979; Thompson, 2007).

CONCLUSIONS AND PERSPECTIVES

The guiding metaphorical relation we have developed links abstractions in programming paradigms and organization in natural, biological systems. The three examples we have put forth are intended to illustrate this convergence.

But what difference could this convergence make to our conceptualization of cognitive systems? Or in other words, in what sense could this *rapprochement* help us in advancing our understanding of what cognition might be? We believe there are both theoretical and empirical consequences of the exercise.

The first example (section "On computability, abstractions and reduction") supports the idea that theoretical reduction to bits and pieces is not the only way of understanding something, and that this might be particularly important for cognitive systems. As programming paradigms show, sometimes it might be necessary to go in the other direction so that system-level behavior and organization is brought into focus. If this is evident for modestly complex structures such as pervasive computing systems, one could contend that it will be all the more critical when the myriad of effective causal loops present in living beings is considered. Importantly, it is through causal sensorimotor loops that perception seems to be sustained in such systems (Noe, 2004).

On a more implementation-related level, the question of degrees of freedom reduction seems quite pertinent. Systems abound, from lasers to tribes, where the degrees of freedom of the components are restricted for the consistency of the whole. What is interesting is that in programming paradigms this takes place through abstractions which, as we have seen, enable novel possibilities of interaction. This is one of the main tenets of biological systems, whereby alternative organizations bring forth the notorious variety of cognitive horizons found in nature. This issue is strongly interrelated with the last example where we pointed out the apparent evolutionary dynamics in programming paradigms. It is obviously only the beginning of this discipline, so time will tell. Yet, given the type of situation programmers and programs have to cope with, this 'natural' tendency towards autonomy might be relevant to consider as a matter of fact.

Computational cognitivism has deeply permeated our way of thinking about what it is to perceive and act on-and in-a world; conceptualization in terms of input of information, processing, encoding, storing, retrieving and output of responses is commonplace when dealing with cognitive systems. But computation is much more than computational cognitivism as the development of programming paradigms shows. Fishing in the metaphorical pool, what seems to come out is how certain type of organizations make novel behavioral domains available. This suggests a plausible path to the emergence of meaningful worlds and therefore, cognition.

ACKNOWLEDGEMENTS

DC acknowledges support from the PBCT/ CONICYT Sensory Neuroscience Ring, ACT-45, and partial funding by FONDECYT Postdoctoral Project 3060094. JSA is partially supported by the Centro de Estudios Avanzados en Educación PBCT/ CONICYT and by FONDEF Project D06I1023. ET acknowledges partial funding from FONDECYT Project 11060493.

REFERENCES

ABBOTT R (*Submitted*). Reductionism, emergence and levels of abstractions. Communications of the ACM

AGHA G (1986) ACTORS: a model of concurrent computation in distributed systems. The MIT Press: Cambridge, MA

ALEKSANDER I (2005) Machine consciousness. Progress in Brain Research 150: 99-108

ASHBY WR (1952) Design for a brain: The origin of adaptive behavior. London: Chapman and Hall

BAARS B (2002) The conscious access hypothesis: origins and recent evidence. Trends Cogn Sci 6: 47-52

BELL A (1999) Levels and loops: The future of artificial intelligence and neuroscience. Philos Trans R Soc Lond B Biol Sci 354: 2013-20

BIRRELL A (1989) An introduction to programming with threads (Technical Report Nº 35). Digital Systems Research Center

BONABEAU E, DORIGO M, & THERAULAZ, G (1999) Swarm intelligence - from natural to artificial systems. Oxford University Press

BRAINERD WS & LANDWEBER LH (1974) Theory of computation. Wiley & Sons

BRIOT J-P, GUERRAOUI R, & LOHR K-P (1998) Concurrency and distribution in object-oriented programming. ACM Computing Surveys 30: 291-329

BURTSEV M & TURCHIN P (2006) Evolution of cooperative strategies from first principles. Nature 440: 1041-1044

CISEK P (1999) Beyond the computer metaphor: Behaviour as interaction. Journal of Consciousness Studies 6: 125-142

CLARK A (1997) Being there: Putting brain, body and world together again. MIT Press

CLEEREMANS A (2005) Computational correlates of consciousness. Progress in Brain Research 150: 81-98

COSMELLI D & THOMPSON E (2007) Mountains and valleys: Human experience and the flow of consciousness. Consciousness and Cognition, doi: 10.1016/j.concog.2007.06.013, In Press.

COSMELLI D & THOMPSON E (In Press) Enaction: Towards a new paradigm for cognitive science. In J. Stewart, O. GAPENNE, & E. DI PAOLO (Eds.), (chap. Embodiment or envatment? Reflections on the bodily basis of consciousness.). MIT Press

DEDECKER J, MOSTINCKX S, VAN CUTSEM T, DE MEUTER W, & D'HONDT T (2005) Ambient-oriented programming. In Oopsla 2005 onward! track.

DEDECKER J, VAN CUTSEM T, MOSTINCKX S, D'HONDT T & DE MEUTER, W (2006) Ambient-oriented programming in AmbientTalk. In D. Thomas (Ed.), Proceedings of the 20th European conference on object-oriented programming (ecoop 2006) (p. 230-254) Nantes, France: Springer-Verlag

DENNETT D (1991) Consciousness explained. Boston: Back Bay Books

DUPUY J (1999) Aux origines des sciences cognitives. Paris: La Découverte

FELDMAN J & BALLARD D (1982) Connectionist models and their properties. Cognitive Science 6: 205-254

FERNANDEZ-BALLESTER G & SERRANO L (2006) Prediction of protein-protein interaction based on structure. Methods Mol Biol 340: 207-34

FODOR J (1979) Representations; essays on the foundation of cognitive science. UK: Harvester Press

FREEMAN W (1999) Consciousness, intentionality, and causality. Journal of Consciousness Studies 6: 143-172

FRITH C & FRITH U (2007) Social cognition in humans. Current Biology 17: R724-R732

FROMHERZ P (2006) Three levels of neuroelectronic interfacing: silicon chips with ion channels, nerve cells, and brain tissue. Annals of the New York Academy of Sciences, 1093: 143-160

GABRIEL R P & GOLDMAN R (2006). Conscientious software. In Proceedings of the 21st ACM SIGPLAN conference on object-oriented programming systems, languages and applications (oopsla 2006) (pp. 433-450). Portland, Oregon, USA: ACM Press (ACM SIGPLAN Notices, 41(10))

GAMMA E, HELM R, JOHNSON R & VLISSIDES J (1994) Design patterns: Elements of reusable object-oriented software. Addison-Wesley

GELDER T VAN (1998) The dynamical hypothesis in cognitive science. Behavioral and Brain Sciences 21: 1-14

GENDLIN E (1997) Experiencing and the creation of meaning: A philosophical and psychological approach to the subjective. Boston: Northwestern University Press

GIBSON J (1979) The ecological approach to visual perception. Boston: Houghton Mifflin

HAKEN H (1983) Synergetics, an introduction: Nonequilibrium phase transitions and self-organization in physics, chemistry, and biology (3rd ed.). New York: Springer-Verlag

HAYES B (2003) The post-OOP paradigm. American Scientist 91: 106-110

HORST S (2005). The computational theory of mind. The Stanford Encyclopedia of Philosophy, Edward N. Zalta (ed.)

JONAS H (2001(1979)) The phenomenon of life: Towards a philosophical biology. Northwestern University Press

KANDEL E, SCHWARTZ J & JESSELL, T (2000). Principles of neural science (Fourth ed.). Mc Graw-Hill.

KAY AC (1993) The early history of Smalltalk. ACM SIGPLAN Notices 28: 69-95

KIM J (2000) Mind in a physical world: an essay on the mind-body problem and mental causation. Cambridge, MA: The MIT Press

MATURANA H (1970a) Biology of cognition (Tech. Rep.). Urbana IL: University of Illinois: Biological Computer Laboratory Research Report BCL 9.0

MATURANA H (1970b) Cognition: A multiple view. In P. Garvin (Ed.), (p. 3-23). New York/Washington

MATURANA H & MPODOZIS J (2000). The origin of species by means of natural drift. Revista Chilena de Historia Natural 73: 261-310

MATURANA H & VARELA F (1973) De máquinas y seres vivos: Una teoría sobre la organización biológica. Santiago de Chile: Editorial Universitaria

MATURANA H & VARELA F (1984) El árbol del conocimiento. Santiago de Chile: Editorial Universitaria

MCDERMOTT D (2007) The Cambridge Handbook of Consciousness. In M M P Zelazo & E Thompson (Eds.), (p. 117-150). Cambridge University Press

NIEDERMEYER E (2005) Electroencephalography: Basic principles, clinical applications, and related fields. In E Niedermeyer & F L da Silva (Eds.), (p. 1-15). Philadelphia, USA: Lippincott Williams and Wilkins

NOE A (2004) Action in perception. MIT Press

PRIGOGINE I & NICOLIS G (1989) Exploring complexity: An introduction. W.H. Freeman & Company

PUTNAM H (1975) Mind, language and reality. philosophical papers (Vol. 2). Cambridge University Press

SEARLE J (1980) Minds, brains, and programs. Behavioral and Brain Sciences, 3, 417-458

SHEWMON A (2001) The brain and somatic integration: insights into the standard biological rationale for equating "brain death" with death. The Journal of Medicine and Philosophy 26: 457-78

SOTO-ANDRADE J (2007) Metaphors and cognitive styles in the teaching-learning of mathematics. In Proc. CERME 5 Fifth conference of the European society for research in mathematics education), lrnaca, Cyprus

STEELS L & BROOKS R (1995) The artificial life route to artificial intelligence: Building embodied situated agents. (L. Steels & R. Brooks, Eds.). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc

STEWARD J & MOSSIO M (n.d.). Is life computable? (*Personal Communication*)

SUN R & FRANKLIN S (2007) The Cambridge Handbook of Consciousness. In M M P Zelazo & E Thompson (Eds.), (p. 151-174). Cambridge University Press

The Software Engineering Institute (SEI)(2006). The software challenge of the future: Ultra-large scale system*s*

THOMPSON E (1997) Symbol grounding: a bridge from artificial life, to artificial intelligence. Brain and Cognition 34: 48-71

THOMPSON E (2004) Life and mind: From autopoiesis to neurophenomenology. a tribute to Francisco Varela. Phenomenology and the Cognitive Sciences 3: 381-398

THOMPSON E (2007) Mind in life: Biology, phenomenology and the sciences of mind. Harvard University Press

THOMPSON E, PALACIOS A & VARELA F (1992) Ways of coloring: Comparative color vision as a case study in cognitive science. Behavioral Brain Sciences 15: 1-45

THOMPSON E & VARELA F (2001) Radical embodiment: Neural dynamics and consciousness. Trends in Cognitive Science 5: 418-425

VARELA F (1979) Principles of biological autonomy. New York: Elsevier North Holland

VARELA J F & BOURGINE P (Eds.) (1992) Towards a practice of autonomous systems. Cambridge: The MIT Press

VARELA F, MATURANA H & URIBE R (1974) Autopoiesis: the organization of living systems, its characterization and a model. Biosystems 5: 187-196

VARELA F, THOMPSON E & ROSCH E (1997) The embodied mind: Cognitive science and human experience. Cambridge, MA: The MIT Press

WOLFRAM S (1984) Computation theory of cellular automata. Communications in Mathematical Physics 96: 15-57