Dynamic Learning Agents and Enhanced Presence on the Grid

Stefano A. Cerri, Marc Eisenstadt, Clement Jonquet

Dynamic Learning Agents and Enhanced Presence on the Grid

Stefano A. Cerri*, Marc Eisenstadt**, Clement Jonquet*

* LIRMM, CNRS & University Montpellier II
161, Rue Ada 34392 Montpellier Cedex 5, France
{*cerri,jonquet*}*@lirmm.fr*

** Knowledge Media Institute, The Open University
Walton Hall, Milton Keynes, UK
m.eisenstadt@open.ac.uk

Abstract

Human Learning on the Grid will be based on the synergies between advanced software and Human agents. These synergies will be possible to the extent that conversational protocols among Agents, human and/or artificial ones, can be adapted to the ambitious goal of dynamically generating services for human learning. In the paper we highlight how conversations may procure learning both in human and in artificial Agents. The STROBE model for communicating Agents and its current evolutions shows how an artificial Agent may "learn" dynamically (at run time) at the *Data*, *Control* and *Interpreter* level, in particular exemplifying the "learning by being told" modality. The enhanced telepresence research, exemplified by Buddyspace, in parallel, puts human Agents in a rich communicative context where learning effects may occur also as a "serendipitous" side effect of communication. The integration of the two streams of research will be the result of a workpackage within the E-LeGI EU Integrated Project, currently under negotiation.

Keywords: e-Learning, Grid, Social Informatics, service generation, Agents, Agent Communication Languages, Enhanced Telepresence, STROBE model

1. INTRODUCTION

The concept of Learning Agent is seducing as much as confusing. There is no clear definition of what a software Agent is, and often, in the best AI tradition, one calls Agents both software and human Agents communicating in a network for performing jointly a task. The "learning" specification for an Agent may refer to learning as it occurs in human Agents within a scenario such as the one of the E-LeGI project, or else "machine learning" as it may be introduced in software. In this paper we wish to address the issue of Learning both in artificial and in human Agents [3]. In order to define our research directions, we need to identify a minimal lexicon, and its associated choices as a precondition for the comprehension of further statements. As we wish our model to be the simplest possible one, we will perhaps risk to oversimplify more complex situations; eventually extensions will be treated later as a result of critical remarks. We assume first that artificial Agents are just software programs - and their associated processes - that at least are autonomous, distributed and able to communicate asynchronously with the environment (consisting only of other Agents) by means of a communication language that is independent on the content of the communication. Objects, for instance, are neither autonomous nor properly communicating, as the communication language consists just of selectors for methods (interfaces) thus is not independent of the objects themselves. Looking at the literature, in spite of the high popularity of the Agent literature, one may seldom find Agents that really respect all the minimal requirements for agentship. In previous papers, in order to give a provocative definition for Agents, we proposed to consider them as "crazy" Operating Systems (the adjective denoting their autonomy in reacting to messages). In the E-LeGI ambitions, perhaps the most challenging one is the personalization of learning services for humans. That will not be achieved unless a minimal formal, computational model of human Agents will be exploited during the generation of the corresponding services. We take this challenge by trying to develop a computable model for Agents that can serve also the purpose of modelling humans in a deliberately well-circumscribed context. Therefore, the issue of learning for human Agents is put into correspondence with the same for artificial Agents. Our model for artificial Agents has to be capable of modelling artificial learning; its corresponding software Agents similarly have to show learning properties that are sufficient for modelling human learning and therefore personalizing the services.

Again with the risk of oversimplifying, we will initially consider for human as well as for artificial Agents just three types of learning:

1. Learning by being told, the classical ”instructional effect”.
2. Learning by abstracting and generalizing (or by classifying examples, extracting rules and forming ”theories”).
3. Learning as a side effect of communication, what we like to call ”serendipitous”[1] learning.

We know that in a parallel paper [7] type 2 learning is extensively addressed. Perhaps what they call induction and abduction in theory construction (or Ontology negotiation) may be reconducted to our abstraction and generalization. At any rate, most of the machine learning work has been performed in this direction, while type 1 and 3 of learning described above - for artificial Agents - did not really get much attention in the literature.

2. LEARNING BY BEING TOLD

2.1. The STROBE model and its evolution

Humans learn facts, rules (or procedures), and languages necessary to understand messages stating facts or procedures, as well as necessary for generating behaviour when applying a particular procedure to parameters. Although we are strong believers in the cognitive constructivist learning paradigm, we nevertheless focus on this highly restricted area of learning, which contains important elements that are so naturally inherited from the computational metaphor. Indeed facts, rules and languages are such as *Data*, *Control* and *Interpreter* levels in computing. These three abstractions levels may be found in all programming languages. One may distinguish *Data* (information) and *Control* (procedures) levels which corresponds to define new simple data and new procedures abstracting on the existing ones, from the *Interpreter* level which means to identify the way of evaluating an expression, or defining a special form which cannot be defined at the *Control* level. Currently, the two first levels could be reached during execution but the challenge is to allow *Interpreter* level modifications at run time, in order to generate processes.

In order for conversational processes in E-LeGI to be effective, they have to generate services that help humans to learn facts, rules and... languages. That will be possible insofar we model human Agents by means of artificial Agents able to learn dynamically facts, rules and languages. As a resulting side effect, we will have the opportunity to use artificial Agents that learn (by being told) during conversations with other artificial Agents, thus that show a dynamic behaviour that adapts to the context. The STROBE model proposes an architecture to support this Agent behaviour. The key initial idea is to give to Agents an environment as a representation of any type of knowledge and consider them as REPL (Read, Eval, Print, Listen) interpreters.

In the STROBE (STReams, OBjects, Environments) [2] model, that inherits most of its features from classical lambda calculus [8], denotational semantics [9] and the Scheme language we have identified a few basic properties for Agents:

1. First class[2] **E**nvironments to model memory: linking variable names to values, under the commitment that types are on the values (dynamic typing) and that procedural abstractions are first class.
2. First class **OB**jects to model the control: the classical loop (message, reaction with another message). In fact this option, very practical as a first approximation, is due to be abandoned for an Actor version, including a dynamic scheduler.
3. First class **STR**eams, modelling the evolving conversational processes by using the delayed (lazy) evaluation of values associated to expressions.
4. First class continuations (**K**), in order to model non determinism and multiple conversational threads, i.e: a formal model of the rest of the process consuming the results from the current one once it will be terminated or suspended.
5. First class interpreters (**I**), modelling how to generate processes from procedures. They are included in the above described environments.

For the representation of the interlocutor in a conversation STROBE uses the concept of Cognitive Environments [1] which give to the Agent a ”partner model” represented by an environment dedicated to this Agent. In this environment

[1]Webster dictionary: Main Entry: serendipity; Pronunciation: -'di-p&-tE; Function: noun; Etymology: from its possession by the heroes of the Persian fairy tale The Three Princes of Serendip; Date: 1754: the faculty or phenomenon of finding valuable or agreeable things not sought for.

[2]The notion of first-classness in a programming language was introduced by Christopher Strachey . First-class objects may be named by variables, may be passed as argument to procedures, may be returned as results and may be included in data structures. Therefore our environments, procedures, etc... profit from each of these properties.

a dedicated interpreter is stored and used to interpret messages (and their content) sent by this Agent. Actually, messages' interpretation is done in a given environment and with a given interpreter. Learning at the *Data* and *Control* level consist in modifying the dedicated environment; learning at the *Interpreter* level (meta-level) consist in modifying the dedicated interpreter. Figure 1 illustrates an Agent representation.

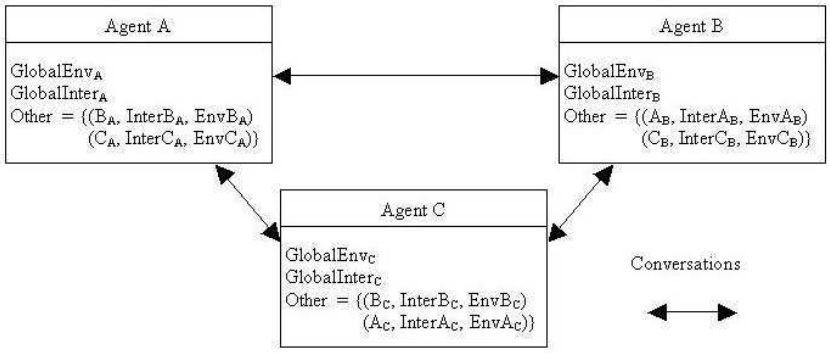

FIGURE 1: Agent attributes for representing others

Within this model, it is not difficult to envison "learning-by- being-told" of variables (*Data*) or procedural abstractions (*Control*) insofar both are acceptable when declared bound to names by an external Agent that has the right to "teach me" about facts or rules. That means that an Agent can learn from another one simple information (*Data*) and procedures (*Control*) using an interpreter to evaluate "assertion" typed messages (such as definition or assignment). What is not straightforward is how to learn at run time special forms or any other modification that affects the interpreters. In the human Agent scenario, it is not difficult to teach a fact or a rule; what is less simple is to teach a piece of "language", i.e. teach people how to modify dynamically their interpretation behaviour for new facts or rules. The STROBE model makes it possible by allowing Agents to modify dynamically their interpreters.

Agents, human and artificial ones, may be considered, in a first approximation, quite similar. Our challenge will be to show that both may be represented by combinations of primitives (environments, objects (procedures, schedulers), streams, continuations, interpreters) within the same model.

2.2. The Scheme architecture

Scheme classic REPL loop interpretation consists in waiting for a user expression, read and interpret this expression, send back the result and wait for the next expression. This represents eventually a typical *Data* and *Control* level modification. The higher level, *Interpreter* level, is not directly accessible. Our model uses another architecture. Instead of interpreting user expressions with the current interpreter, this one calls (via `apply-procedure`) another interpreter (stored in the dedicated cognitive environment), that interpret the user expressions. Thus, user expressions may modify the interpreter which evaluates them. Our idea is to use this architecture with our Agents. If one considers the three following interpreters each characterized by two procedures:

- Scheme: `eval` and `apply`
- Meta-eval: `evaluate` and `apply-procedure`
- Meta-ambeval: `ambevaluate` and `ambapply-procedure`

Then the classic REPL loop is equivalent to `(eval e r k)`. where `e` is the expression to evaluate, `r` the environment and `k` the continuation[3]. Our is: `(eval (apply-procedure 'ambevaluate e r ks kf))`. We need `eval` because Scheme is the programming language used; `apply-procedure` is used as a dedicated interpreter selector, and provides the first class environments. Finally, the last interpreter, the dedicated one (`ambevaluate`), is used to evaluate the message and its content[4].

2.3. Example of meta-level learning: "teacher-student" dialogue

We consider that the goal of education is to change the interlocutor's state. This change is done after evaluating new elements brought by the communication. The example in figure 2 (see also [5]) shows that a STROBE Agent

[3]These three elements constitute the execution context.
[4]This interpreter is a non deterministic one, useful to constraint programming and to allow multi-thread Agents.

can modify its way of seeing things (i.e. of evaluating messages) by ”changing” its dedicated interpreter while communicating. It is a standard ”teacher-student” dialogue, though of course highly simplified in light of our earlier comments. An Agent teacher asks to another Agent student to broadcast a message to all its correspondents. However, student does not initially know the performative[5] used by teacher. So, teacher transmits two messages (`assertion` and `order`) clarifying to the student the way of processing this performative. Finally, teacher formulates again its request to student and obtains, this time, satisfaction. Figure 2 describes the exact dialogue emerging from the experimentation. After the last message process, the student function dedicated to the evaluation of message (`ambevaluate-kqmlmsg`) is modified. Thus a part of its interpreter was dynamically changed. The corresponding code in its environment dedicated to this conversation is changed. Then student Agent can process `broadcast` messages.

TEACHER	STUDENT
Here is the definition of the `square` procedure: `(kqmlmsg 'assertion teacher student '(define (square x) (* x x)))`	Ok, I know now this procedure: `(kqmlmsg 'ack student teacher '(*.*))`
Broadcast to all your current correspondents: `(kqmlmsg 'broadcast teacher student '(order (square 3)))`	Sorry, I don't know this performative: `(kqmlmsg 'answer student teacher ''(no-such-performative broadcast))`

Ok, here is the method to add this performative to those you know:
 	
Here is the code you have to generate and add to your `ambevaluate-kqmlmsg` function:
`(kqmlmsg 'assertion teacher student learn-broadcast-code-msg)`
 	
Run this procedure:
`(kqmlmsg 'order teacher student '(set! ambevaluate-kqmlmsg learn-broadcast-code)))`
	Ok, I have added this code in a binding of my environment:
	`(kqmlmsg 'ack student teacher '(*.*))`
	
	Ok, I have just modified my interpreter:
	`(kqmlmsg 'executed student teacher '(*.*))`

Broadcast to all your current correspondents: `(kqmlmsg 'broadcast teacher student '(order (square 3)))`	Ok, I broadcast `(kqmlmsg 'order student ... '(square 3))`

FIGURE 2: `broadcast` learning *teacher-student* dialogue.

Notice that `learn-broadcast-code-msg` message indicates how to generate the new function code taking into account the previous student code definition, and to store it in the `learn-broadcast-code` variable. This is a constructivist method.

This toy-example is very simple but interesting because it shows the potentiality of such a model. We consider that it is a meta-level learning because a part of the Agent interpreter is dynamically changed. Another paper [6] describes also how using a nondeterministic interpreter in the model can enable the dynamic specification of a problem, in order to fit with dynamic service generation scenarios, such as necessary on Grids. The paper gives a typical e-commerce scenario example.

3. LEARNING AS A SIDE EFFECT OF COMMUNICATION

The examples provided in the previous section are of course characteristic of a very narrow spectrum of learning activities, namely those that occur during a particular kind of synchronous (i.e. real time communicative) interaction. Although the overwhelming majority of distance learning and e-learning literature emphasizes either the asynchronous space (particularly via discussion forums) or the one-to-many large-scale synchronous activities afforded by streaming media, there are nevertheless important and indeed profound opportunities that arise during very small-scale synchronous interactions (i.e. one-to-one or among very small study groups up to say, three or four participants). We note in particular the opportunistic learning dialogues that can occur in real time in such intensive tutorial situations, and which are precisely suited for the examples presented earlier. Although seemingly small and specialised in nature, it is nevertheless the case that if tutorial dialogues eventually occur between human and artificial agents, then there are in fact no practical limits to scalability, because every one-to-one interaction that involves an artificial agent can be replicated hundreds, thousands, or even millions of times.

[5] Our communication model protocol is speech act oriented, such as KQML or FIPA ACL messages.

For the time being, our research progresses by studying the nature of synchronous interactions that occur between two, three, or four human agents. The ideal paradigm for this is to investigate and facilitate the learning interactions that take place via the world's fastest-growing software phenomenon: Instant Messaging. In the context of the E-LeGI project, we provide a custom environment for learners, called 'BuddySpace' [4] which can be summed up as "Instant Messaging + Smart Maps = Enhanced Presence". It provides continual 'background presence awareness' of peers, by deploying significant extensions to the open-source XML technology from the Jabber Software Foundation. As argued in Smart Mobs [10], tools like BuddySpace leverage the overwhelming power of social cohesiveness that can be brought about by knowledge of the presence and location of others in both real and virtual spaces. We know also from the work of Reffell and Eklund (2001) that this kind of presence awareness is used by students to locate resources, for quick exchange of information and to organize meetings either online or face-to-face.

In reality, Enhanced Presence is much more than just 'messaging' and 'maps'. In particular, we aim to provide tools that enable us to express the entire situated context of the learner, which is clearly a lot more than just 'location X' and 'online' or 'offline'. The learner's current state of mind, including goals, plans and intentions, must be understood, as well as the way this connects with ongoing activities and devices accessible to the learner. When all these are modelled, plausible inferences can be drawn about what the leaner wants and needs to know, and this gives us an important 'foot in the door' for addressing the problem of delivering the right knowledge to the right people in the right place at the right time. So far, this notion of 'right knowledge' has been nothing more than a Knowledge Management 'slogan', but our belief is that Enhanced Presence capabilities, linked to the STROBE model, can make this dream a reality.

4. CONCLUSION

In this paper we presented the current progress of our work in two domains: Agent Communication Languages and Enhanced Presence. These are, at a first sight, very different one another nevertheless we have shown that they may become highly synergic and perhaps also mutually dependent within a rich experimental context such as the one of E-LeGI. The scenarios for integration of these two streams of research are currently under discussion. We perceive their properties as consisting of the identification of models and tools for generating learning services that enable and facilitate co-learning effects, i.e. humans AND machines construct their own representations - learn - by exploiting the representation of the partner through conversations.

This social constructivist approach extends to machines the full right membership of societies of learning Agents both as "destinations" of knowledge emerging from these societies, and as a continuous source of knowledge digitally represented and stored and potentially consumed by other members through conversations. In these societies, it will be important to partition the responsibilities among members according to their best capacities: exploiting therefore Human Agents for what they are best in, and, at the same time, Artificial Agents for their optimal performance. The consequence is to adopt an approach in Distributed Artificial Intelligence where Humans are privileged in their best roles (e.g. motivation, trust, depth of conceptual analysis ...) and Machines in other roles (computing fast and reliably, instantaneous transmission of information through the net, storing and retrieving ...).

5. ACKNOWLEDGEMENTS

The example cited in section 2 has been developed within the Ph.D research of one of the authors (CJ). Cited papers and implementations are available on www.lirmm.fr/?jonquet. The support of the EU project LeGE-WG (Learning Grid Excellence Working Group) is gratefully acknowledged.

REFERENCES

[1] Stefano A. Cerri. Cognitive environments in the strobe model. *Presented at EuroAIED: the European Conference in Artificial Intelligence and Education*, 1996.

[2] Stefano A. Cerri. Shifting the focus from control to communication: the STReams OBjects environments model of communicating agents. In *In Padget, J.A. (ed.) Collaboration between Human and Artificial Societies*, pages 74–101, Berlin, Heidelberg, 1999. New York: Springer-Verlag, Lecture Notes in Artificial Intelligence.

[3] Stefano A. Cerri. Human an artificial agents conversations on the grid. In *Electronic Workshops in Computing (eWiC), 1st LeGE-WG International Workshop on Educational Models for Grid Based Services*, Lausanne, Switzerland, September 2002.

[4] M. Eisenstadt and M. Dzbor. Buddyspace: Enhanced presence management for collaborative learning, working, gaming and beyond. *JabberConf2002 Europe*, June 2002.

[5] Clement Jonquet and Stefano A. Cerri. Cognitive agents learning by communicating. In *7ème Colloque Agents Logiciels, Coopération, Apprentissage et Activité Humaine, ALCAA'03*, pages 29–39, Bayonne, September 2003.
[6] Clement Jonquet and Stefano A. Cerri. Agents as scheme interpreters: Enabling dynamic specification by communicating. In *To be published in 14ème Congrès Reconnaissance des Formes et Intelligence Artificielle, RFIA'04*, Toulouse, January 2004.
[7] Philippe Lemoisson, Stefano A. Cerri, Jean Sallantin, and Serge-André Mahe. Constructive interactions. In *Summitted to Electronic Workshops in Computing (eWiC), 3rd LeGE-WG*, Berlin, December 2003.
[8] John McCarthy. Recursive functions of symbolic expressions and their computation by machine, part i. *Communications of the ACM*, 3(4):184–195, 1960.
[9] Christian Queinnec. *Les languages Lisp*. InterEditions, Paris, 1994.
[10] Howard Rheingold. *Smart Mobs: The Next Social Revolution*. Perseus, Cambridge, Mass., 2002.