Book Title
Book Editors
IOS Press, 2003

Learning Agents and Enhanced Presence for Generation of Services on the Grid

Clement Jonquet [a], Marc Eisenstadt [b] Stefano A. Cerri [a]
[a] *LIRMM & Montpellier II University, France*
{jonquet,cerri}@lirmm.fr
[b] *Knowledge Media Institute, The Open University, Milton Keynes, UK*
m.eisenstadt@open.ac.uk

Abstract. Human learning on the Grid will be based on the synergies between advanced artificial 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 presence 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.

Keywords. Learning, Grid, e-learning, Social Informatics, Dynamic Service Generation, Agents, Agent Communication Languages, Enhanced Presence, STROBE model.

1. Introduction[1]

The concept of learning agent is seducing as much as confusing. There is no clear definition of what an artificial agent is, and often, in the best AI tradition, one calls agents both Artificial Agents (AA) and Human Agents (HA) communicating in a network for jointly performing a task. The *learning* specification for an agent may refer to "learning" (or "e-learning") as it occurs for HA within a scenario such as the one of the ELeGI project[2], or else as "machine learning" as it occurs for AA. In this paper we wish to address the issue of learning both for AA and HA on the Grid [2]. The aim is include in a global logic to define a generic architecture, OGHSA, as the successor to OGSA [3]: the *Open Grid Human Service Architecture*. In this architecture, the concept of agent is fundamental. We assume first that AA are just software programs – and their associated processes – that at least are autonomous, distributed and able to communicate asynchronously with their environment and others agents by means of a communication language that is indepen-

[1] This paper is a revised version of [1].
[2] European Learning Grid Infrastructure project – www.elegi.org.

dent from the content of the communication and from the internals of agents[3]. Objects, for instance, are not AA as they are neither autonomous nor properly communicating, as the communication commands 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. Moreover, the choice of dealing with agents is also explained by considering that Grid and agent need each other [5]. In the ELeGI ambitions [6], perhaps the most challenging one is the personalization of learning services for HAs, that will not be achieved unless a minimal formal, computational model of HA 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 also serve the purpose of modelling HA in a deliberately well-circumscribed context.

Therefore, the global topic of this article is how to facilitate the ability of both AA and HA to dynamically generate services to each other. To generate a service s not product delivery. One of the fundamental differences between a product and a service is that when a user asks for a product, he exactly knows what he wants, he knows what the system can give to him and he knows how to formulate his request. A typical procedure call (with good parameters) is then realised as in RPC (Remote Procedure Call) as it is the case in the most widespread Web architecture, i.e. the client-server. In other words, the user asks a fixed algorithm (procedure) to be executed. At the opposite, when a user asks a service to be generated he does not exactly know what he wants and his needs and solutions appear progressively with the dialogue (conversational process) with the service provider until he obtains satisfaction. The service generation approach does not assume the user knows exactly what the service provider can offer him. The user finds out and constructs step by step what he wants as the result of the service provider's reactions. In fact, the service generation is a "nondeterministic" process depending on the conversation between two agents. As an example in everyday life, when somebody looks for clothes, buying ready-to-wear clothes is analogous to asking for a product, whereas having clothes made by a tailor is analogous to requiring a service to be generated. For another example of situation requiring a service generation see the scenario of Clancey [7]. We think that e-learning scenarios have to become stereotypical contexts for such "generated services" putting the learner at the center of the process instead of concentrating on information transfer as is the case in most e-learning environments. We show in [8] that the dynamic generation of service is a process creating new knowledge, particulary fit for e-learning scenarios. That means that making AA learn by interacting with HA today, will make HA learn by having service generated tomorrow.

The dynamic generation of services (See also [9]) can only be done by entities connected together in a network, member of a virtual community and sharing knowledge and resources, as the Grid [10]. Current Web Services or Grid Services [3] do not fulfill the requirements for dynamic service generation. Moreover, both types of agents can not be static, they have to change, learn each time they ask for or generate a service. That is why the paper focuses on learning. Therefore, the issue of learning for HA is put into correspondence with the same for AA. At the risk of oversimplifying, we will initially consider for HA as well as for AA just three types of learning:

[3]These languages are called Agent Communication Languages, ACLs, such as KQML or FIPA-ACL. See for example [4].

1. Learning by being told, the classical instructional effect.
2. Learning by abstracting and generalizing (or by classifying examples, extracting rules or forming theories).
3. Learning as a side effect of communication, what we like to call "serendipitous"[4] learning.

Type 2 is often addressed in a number AI papers, including for example [11]. Perhaps what they call induction and abduction in theory construction (or ontology negotiation) may be mapped 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 learning described above - for AA - did not really get much attention in the literature.

The role of conversation is of such great significance to education and learning (conversational processes are the key of arch of services), that we shall take a moment here to provide a theoretical underpinning of our approach. We have been very influenced by the Conversational Framework of Laurillard [12], who argues that learning can be viewed as a series of teacher-learner conversations taking place at multiple levels of abstraction. As summarized in [13]:

> At the most general level of description, the learning process is characterized as a 'conversation' between teacher and student (see Figure 1), operating on two levels, discursive and interactive, the two levels being linked by the twin processes of adaptation and reflection.

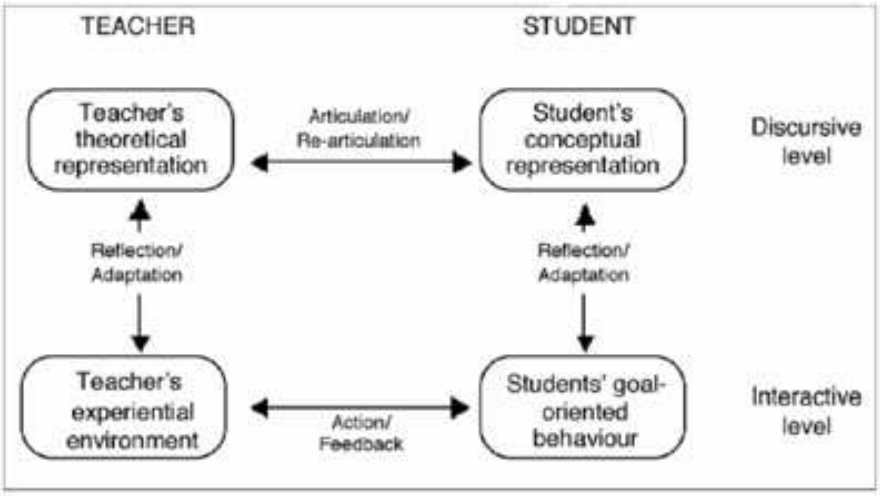

Figure 1. The conversational framework for the learning process (from [13]).

In fact, the framework is more powerful than the simple diagram shows, because multiple styles of interaction are possible, as well as iterations within any type of interaction. As Laurillard et al describe the framework:

> Students' work on an interactive resource will take place at the interactive level, where students' adapt their actions in the light of their current conceptual understanding, and discuss and reflect on their practical work in order to develop their conceptual understanding at the

[4]The faculty or phenomenon of finding valuable or agreeable things not sought for, by surprise.

discursive level. Similarly, the interactive resource, expressing the 'teacher's experiential environment' is an adaptive response by the teacher who chooses it in the light of discursive level conversations with students, and a sense of what they need to do in order to learn the topic. Again, the teacher's reflection on what the students do during the interaction will drive their further discussions at the discursive level.

In [14] we extended the model slightly (i) to highlight the iterative nature of every step (e.g. any action or thought process can involve multiple steps at different levels of abstraction); (ii) to add 'Peer-to-peer' (P2P) multiple boxes in the lower right hand corner, to highlight the fact that peer group interactions (not only student-student but also teacher-teacher) are in fact fundamental to many learning scenarios, and (iii) to emphasize that any instance of teacher or student may also be viewed as an agent, and indeed may be either an AA or a HA, in an arbitrary mix.

The rest of the paper is organized as follows: In Section 2 we introduce the STROBE model as a type 1 learning model for AA. In Section 3, we present the concept of Enhanced Presence for type 3 learning, based on practical tools. Section 4 explains why the Grid is a good environment for the integration of the two streams of research; integration unavoidable for developing dynamic service generation systems. Section 5 concludes the article.

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 comparable to *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 correspond to defining 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 (meta-level modification), in order to generate processes.

In order for conversational processes in ELeGI to be effective, they have to generate services that help HA to learn facts, rules and... languages. That will be possible insofar we model HA by means of AA able to learn dynamically facts, rules and languages. As a resulting side effect, we will have the opportunity to use AA that learn (by being told) during conversations with other AA, thus that show a dynamic behaviour that adapts to the context. The STROBE model (STReams, OBjects, Environments) [15,8] 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[5]. In this model, that inherits most of its features from classical lambda calculus, denotational semantics and the Scheme language [16], we have identified a few basic properties for agents:

1. First class[6] **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. As object can be represented by procedures and procedures are first class (in Scheme).
3. First class **STR**eams, modelling the evolving conversational process by using the delayed (lazy) evaluation of values associated to expressions.
4. First class continuations, in order to model non determinism and multiple conversational threads, i.e. a formal expression of the rest of the process consuming the results from the current one once it will be terminated or suspended.
5. First class interpreters, modelling how to generate processes from procedures. They are included in the above described environments.

For the representation of the interlocutor in a conversation, the STROBE agents uses the concept of Cognitive Environments [17] which give to the agent a "partner model" represented by an environment dedicated to this agent. In this environment 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 consists of modifying the dedicated environment; learning at the *Interpreter* level (meta-level) consists of modifying the dedicated interpreter. Therefore, if we consider a language as a pair consisting of: i) a language expression evaluation mechanism and ii) a memory to store this mechanism and abstractions constructed with the language. (i.e. Language = Interpreter + Environment) then we can say that the STROBE model allows agents to build a different language for each of their partner. For a formal description of the model see also [8]. Figure 2 illustrates an agent representation.

Within this model, it is not difficult to envison "learning-by-being-told" of variables (*Data*) or procedural abstractions (*Control*) insofar as 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 affect the interpreters. In the HA 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 dynamically to modify their interpreters.

[5]Notice that in STROBE, the REPL cycle itself is considered an exemple of control of processes occuring during a HA-AA conversation

[6]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.

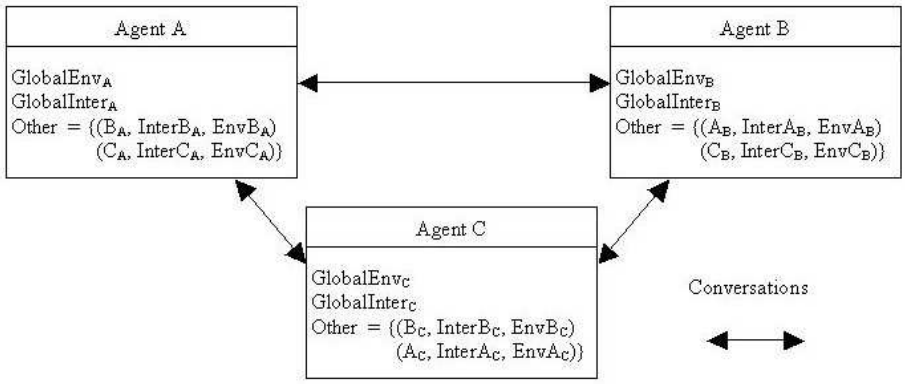

Figure 2. Representation of the others in the STROBE model.

2.2. The meta-level learning architecture

In programming languages, the classic REPL loop interpretation consists of waiting for a user expression, reading (*Read*) and interpreting (*Eval*) this expression, sending back the result (*Print*) and waiting for the next expression (*Listen*). 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 a meta-interpreter which itself calls the `evaluate` procedure (both stored in the dedicated Cognitive Environment) that interprets the user expressions. Thus, user expressions may modify the interpreter which evaluates them by changing the `evaluate` procedure as any procedure. The idea is to use this architecture with our agents. This feature is quite easily feasible in Scheme, which is an interpreted language such that the `evaluate` procedure is a feature of the langage itself.

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 3 shows that a STROBE agent 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. An agent teacher asks to another agent student to broadcast a message to all its correspondents. However, student does not initially know the performative[7] used by teacher. So, teacher transmits two messages (`assertion` and `order`) clarifying to the student the way of processing this performative by changing the function which interprets the messages (`evaluate-kqmlmsg`). Finally, teacher formulates again its request to student and obtains, this time, satisfaction. Figure 3 describes the exact dialogue occurring in the experimentation. After the last message process, the student function dedicated to the evaluation of message (`evaluate-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 sent by the teacher.

[7]The STROBE model is speech act oriented, such as KQML or FIPA-ACL 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 `evaluate-kqmlmsg` function:
`(kqmlmsg 'assertion teacher student learn-broadcast-code-msg)`
 	
Run this procedure:
`(kqmlmsg 'order teacher student '(set! evaluate-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 3. Learning of the performative `broadcast` in the *teacher-student* dialogue.

Notice that `learn-broadcast-code-msg` message indicates how the student agent generates 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 since it is the student, not the teacher who constructs the new knowledge.

This toy-example is very simple but interesting because it shows the potential of such a model. We consider that it is meta-level learning because a part of the agent interpreter is dynamically changed. Another paper [18] describes how to use a nondeterministic interpreter in the STROBE model to do dynamic specification of a problem, in order to fit with dynamic service generation scenarios. 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 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 HA and AA, then there are in fact no practical limits to scalability, because every one-to-one interaction that involves an AA can be replicated hundreds, thousands, or even millions of times.

For the time being, our research progresses by studying the nature of synchronous interactions that occur between two, three, or four HA. 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 ELeGI project, we provide a custom environment for learners, called 'BuddySpace' [19] 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 [20], 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.

Putting HA together to generate knowlegde in a serendipitous way in an artificial context (Buddy Space but more generally all computer science feature) is a first step for making them interact with AA. As we know that beyond each AA there is still an human brain, we can imagine that AA will learn by interacting with HA who have themselves learnt by interacting with other HA. The dynamic generation of service is still undone by AA but realised for years by HA, Thus, facilating HA-HA communication and learning through enhanced presence allow us to progressively transmit to AA the faculty of generate services, even if at the beginning theses services will be generated by HA, and just interfaced by AA.

4. Why The Grid ?

How does the concept of Grid fit into such a model? To answer this question, we draw a parallel between the essence of our approach and the initial reasons for conceiving and developing Grid technologies in the mid-1990's. The original observation was that

facing a major increase of computational needs, for instance in the scientific community (for weather modeling and nuclear physics), there was a parallel increase of computational resources (machines and networks), for which many resources were proprietary and therefore not accessible directly nor on-demand. The original question was how to build a new infrastructure (the Grid) that would satisfy the computational needs in an on-demand fashion by exploiting the computational resources in a seamless way, not requiring 'physical possession' of the resource by the potential clients. That simple observation was based upon an optimization approach: previously idle machines would be used by those who would have needs, satisfying both (in fact, a win-win solution). Our conversational processes and enhanced presence approach proposes, in exactly the same manner, to combine latent and explicit learning needs with potentially available teaching/learning resources able to satisfy those needs in a kind of "human grid" – in this case offer and demand are combined, driven by the demand and enabled by the infrastructure. This is a first approximation to our proposed OGHSA in the ELeGI project, where we study conversational processes among HA (mediated by AA) trying to offer human resources to learners in an on-demand (or 'as needed') manner.

To share services is the intrinsic notion of the Grid. That is why we deal with the Grid for realising the learning paradigm shift of ELeGI. The essence of the Grid concept is nicely reflected by its original metaphor: the delegation to the electricity network to offer somebody the service of providing him enough electric power as he needs it when he needs it even if he does not know where and how that power is generated. The resource firstly offer by the Grid were computational power and storage but today, recent research on Grid have extended it to the sharing of any kind of ressources and services [3,21]. The Grid has become the platorm for tomorrow's service oriented computing.

The recent evolution of Grid Services research coupled with Web Services produced the WSRF (Web Services Ressource Framework) specifications [21]. It specifies the association between a Web Service and one or more named typed state components. In fact, it separates stateless services, that act upon stateful resources from these resources. This view is typically what has been done by the STROBE model since years, which separates the control itself (done by the agent, represented by an object) from the structure keeping the data i.e. the different Cognitive Environments. That is one of the reasons we think that the STROBE model can be an important modelling and implementation element for future Grid Services, its analogy with HA facilitates future OGHSA [2]

5. Conclusion

In this paper we presented the current progress of our work in two domains: the STROBE model both as a model of communication and representation of agents and Enhanced Presence. These are, at a first sight, very different from 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 ELeGI. 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. HA and AA 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 HAs for what they are best in, and, at the same time, AAs for their optimal performance. The consequence is to adopt an approach in Distributed Artificial Intelligence where HA are privileged in their best roles (e.g. motivation, trust, depth of conceptual analysis ...) and AA in other roles (computing fast and reliably, instantaneous transmission of information through the net, storing and retrieving ...).

Acknowledgements

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

This work is partially supported by the European Community under the Innovation Society Technologies (IST) programme of the 6th Framework Programme for RTD - project ELeGI, contract IST-002205. This document does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of data appearing therein. For more information about *European Learning Grid Infrastructure* project: www.elegi.org.

References

[1] Stefano A. Cerri, Marc Eisenstadt, and Clement Jonquet. Dynamic Learning Agents and Enhanced Presence on the Grid. In *3rd International LeGE-WG Workshop: GRID Infrastructure to Support Future Technology Enhanced Learning*, Berlin, Germany, December 2003. Electronic Workshops in Computing (eWiC). http://ewic.bcs.org/conferences/2004/3rdlege/.

[2] Stefano A. Cerri. An integrated view of GRID services, Agents and Human Learning. In P. Ritrovato, C. Allison, S. A. Cerri, T. Dimitrakos, M. Gaeta, and S. Salerno, editors, *Towards the Learning GRID: advances in Human Learning Services*, Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, The Netherlands, September 2005.

[3] Ian Foster, Carl Kesselman, Jeff Nick, and Steve Tuecke. The physiology of the Grid: An Open Grid Services Architecture for distributed systems integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*. The Globus Alliance, June 2002. Extended version of Grid Services for Distributed System Integration.

[4] Marc-Philippe Huget, editor. *Communication in Multiagent Systems, Agent Communication Languages and Conversation Polocies*, volume 2650 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Heidelberg New York, 2003.

[5] Ian Foster, Nicholas R. Jennings, and Carl Kesselman. Brain meets brawn: Why Grid and Agents need each other. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS'04*, volume 1, pages 8–15, New York City, New York, USA, July 2004.

[6] Matteo Gaeta, Pierluigi Ritrovato, and Saverio Salerno. ELeGI the European Learning Grid Infrastructure. In P. Ritrovato, C. Allison, S. A. Cerri, T. Dimitrakos, M. Gaeta, and S. Salerno, editors, *Towards the Learning GRID: advances in Human Learning Services*, Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, The Netherlands, September 2005.

[7] William J. Clancey. Towards on-line services based on a holistic analysis of human activities. In P. Ritrovato, C. Allison, S. A. Cerri, T. Dimitrakos, M. Gaeta, and S. Salerno, editors, *Towards the Learning GRID: advances in Human Learning Services*, Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, The Netherlands, September 2005.

[8] Clement Jonquet and Stefano A. Cerri. Agents communicating for dynamic service generation. In *1^{st} International Workshop on GRID Learning Services, GLS'04*, pages 39–53, Maceio, Brazil, September 2004.

[9] Abdelkader Gouaich and Stefano A. Cerri. Movement and interaction in semantic grids: dynamic service generation for agents in the MIC* deployment environment. In *4^{th} LeGE-WG International Workshop: Towards a European Learning Grid Infrastructure: Progressing with a European Learning Grid*, Stuttgart, Germany, April 2004. Electronic Workshops in Computing (eWiC). http://ewic.bcs.org/conferences/2003/4thlege/.

[10] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[11] Philippe Lemoisson, Stefano A. Cerri, and Jean Sallantin. Conversational interactions among rational agents. In P. Ritrovato, C. Allison, S. A. Cerri, T. Dimitrakos, M. Gaeta, and S. Salerno, editors, *Towards the Learning GRID: advances in Human Learning Services*, Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, The Netherlands, September 2005.

[12] Diana Laurillard. *Rethinking University Teaching*. London: Routledge, 2^{nd} edition, 2002.

[13] Diana Laurillard, Matthew Stratfold, Rose Luckin, Lydia Plowman, and Josie Taylor. Affordances for learning in a non-linear narrative medium. *Interactive Media in Education*, 2, August 2000. http://www-jime.open.ac.uk/00/2/.

[14] Marc Eisenstadt, Jiri Komzak, and Stefano A. Cerri. Enhanced presence and messaging for large-scale e-learning. In *3^{rd} International Symposium on Tele-Education and Lifelong Learning, TelEduc'04*, Havana, Cuba, November 2004.

[15] Stefano A. Cerri. Shifting the focus from control to communication: the STReam OBjects Environments model of communicating agents. In Padget J.A., editor, *Collaboration between Human and Artificial Societies, Coordination and Agent-Based Distributed Computing*, volume 1624 of *Lecture Note in Artificial Intelligence*, pages 74–101. Springer-Verlag, Berlin Heidelberg New York, 1999.

[16] Harold Abelson, Geral Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, Massachusetts, USA, 2^{nd} edition, 1996.

[17] Stefano A. Cerri. Cognitive Environments in the STROBE model. In *European Conference in Artificial Intelligence and Education, EuroAIED'96*, Lisbon, Portugal, 1996.

[18] Clement Jonquet and Stefano A. Cerri. Agents as scheme interpreters: Enabling dynamic specification by communicating. In *14^{th} Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle, RFIA'04*, volume 2, pages 779–788, Toulouse, January 2004.

[19] M. Eisenstadt and Martin Dzbor. Buddyspace: Enhanced presence management for collaborative learning, working, gaming and beyond. In *JabberConf Europe*, Munich, Germany, June 2002.

[20] Howard Rheingold. *Smart Mobs: The Next Social Revolution*. Perseus, Cambridge, Massachusetts, USA, 2002.

[21] Ian Foster, Jeffrey Frey, Steve Graham, Steve Tuecke, Karl Czajkowski, Don Ferguson, Frank Leymann, Martin Nally, Igor Sedukhin, David Snelling, Tony Storey, William Vambenepe, and Sanjiva Weerawarana. Modeling stateful resources with web services. Whitepaper Ver. 1.1, Web Services Resource Framework, May 2004.