Electronic Institutions Development Environment

(Demo Paper)

M. Esteva, J. A. Rodriguez-Aguilar, J. LL. Arcos, C. Sierra, P. Noriega, B. Rosell, D. de la Cruz
Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
{marc,jar,arcos,sierra,pablo,rosell,davdela}@iiia.csic.es

ABSTRACT

In this paper we present the Electronic Institutions Development Environment (EIDE) to support the engineering of multiagent systems as Electronic Institutions. An electronic institution defines a set of rules that structure agent interactions, establishing what agents are permitted and forbidden to do, as well as the consequences of their actions. EIDE supports and facilitates all the stages of electronic institutions' engineering, namely from the specification of an institutional rules to its execution and monitoring.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms

Keywords

Electronic Institutions, Development Environments

1. INTRODUCTION

Multiagent systems (MAS) are complex systems populated by autonomous entities that interact to achieve some shared or individual goals. The design and development of MAS suffers from all the problems inherent to the development of distributed concurrent systems as well as the additional problems which arise from having flexible and complex interactions among autonomous entities [3]. Hence, the design of appropriate methodologies and software tools for assisting this process is one of the main areas of MAS research. One of the most prominent methodologies for engineering this kind of systems is the Electronic Institutions methodology [1]. An electronic institution (EI) defines a set of rules that establish what agents are permitted and forbidden to do, and the consequences of agent' actions. Hence, an EI can be regarded as a coordination artifact that mediates agent interactions. Since EIs do not impose restrictions on the type of agents that may participate, they can be used for building both closed and open MAS.

Cite as: Electronic Institutions Development Environment (Demo Paper), M. Esteva, J.A. Rodriguez-Aguilar, J.LL. Arcos, C. Sierra, P. Noriega, B. Rosell and D. de la Cruz, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

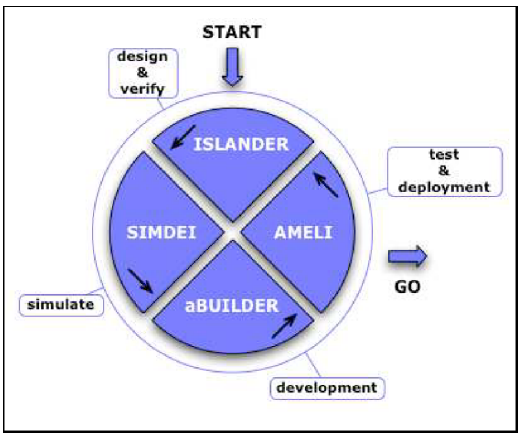

Figure 1: Electronic Institutions Development cycle.

The engineering of EIs is supported by the Electronic Institutions Development Environment (EIDE). EIDE is composed of a set of software tools that support all the stages of an EI engineering, from the specification of the institutional rules to its final deployment and execution. Notably, EIDE moves away from machine-oriented views of programming toward organisational-inspired concepts that more closely reflect the way in which we may understand distributed applications such as MAS. It supports a top-down engineering approach: firstly the organisation, secondly the individuals. In this paper we show the different stages engineers have to go through in order to build an EI and how EIDE support them. EIDE can be freely downloaded from http://e-institutor.iiia.csic.es/eide/pub/.

2. ENGINEERING ELECTRONIC INSTITUTIONS

Next we detail the steps to be followed when engineering and subsequently executing EIs. Figure 1 depicts the role of the EIDE tools in an EI engineering cycle. Notice that EIDE allows for engineering both the institutional rules and their participating software agents.

2.1 Specification and static verification

The engineering of EIs starts with a formal specification of the institutional rules according to its formalization presented in [1]. In order to specify an EI, designers have to

define the roles that agents may play and their relationships, a common ontology and communication language, a network of interaction protocols establishing the interactions agents can engage in and how they can move among them depending on their active role, and a set of normative rules defining the consequences of agent' actions within an EI.

The specification of EIs is supported by *ISLANDER*, which combines both graphical and textual specifications of EI components. The tool permits the graphical specification of the roles and their relationships, the interaction protocols, and the network of interaction protocols. We believe that graphical specifications facilitate the work of agent designers because they are easier to create and to understand. A new feature of the current version is that it incorporates a world model to specify the attributes or properties of the world in which the EI is situated (the World of Interest), that do not depend on the EI but that can be relevant for its participants.

ISLANDER also supports the static verification of specified EIs, which amounts to checking the structural correctness of specifications. For instance, to check that interaction protocols are correctly specified. The verification permits to detect and correct errors before moving to the next stage. The result is a sound and unambiguous definition of the institutional rules.

2.2 Dynamic verification

At run time EIs are open to varying populations of heterogeneous and self-interested agents. Hence, it is hard to know, at design time, the dynamics that may emerge caused by (possibly large populations of) agents. Eventually, unexpected chaotic behaviours that jeopardise the institution may come up. Thus, the static verification carried by ISLANDER is not enough and further verification tools are required in order to carry out dynamic verifications. In EIDE the dynamic verification is carried out by means of simulations supported by *SIMDEI* that allows to run EI simulations with different agent populations. EI designers should analyse the simulation results to decide whether to modify or not the institutional rules.

Since we have incorporated the notion of a world model to the EIs specification, we also have to simulate it to allow agents to sense and act on it. At this aim we have implemented a simulation bridge that: (i) synchronises both simulators; (ii) forwards the world model variables' values to *SIMDEI*; and (iii) translates actions within the simulated EI into actions in the world model. At present, we do offer implementations of the simulation bridge to connect *SIMDEI* to either Simulink [4] or EJS [2] simulations. In the current version simulations executed by *SIMDEI* can be visualised using a monitoring tool, which graphically shows all the events occurring during a simulation.

2.3 Agent development

An EI specification defines the possible behaviours agents may have, but it is a task of agent designers to incorporate agents with the decision making mechanisms that will determine the concrete agent behaviour. At this point we want to remark that we do not impose restrictions on the type of agents that can participate in an EI. Agent designers can choose the language and architecture that is best to fulfil their goals and they can use any software tools that facilitate their work. Therefore, it is not mandatory for them to use the *aBUILDER* tool.

Nonetheless, we believe that it is important to support this intricate development process via the aBUILDER tool. The tool supports the graphical specification of agent behaviours starting from an institution specification created with ISLANDER. aBUILDER supports the automatic generation of agent (code) skeletons based on graphical specifications of agent behaviours. The generated skeletons can be used on EI simulations supported by SIMDEI or in the real execution of the institution supported by AMELI.

2.4 Execution and Analysis

An EI defines a normative environment that shapes agent interactions. As an EI may be populated at execution time by heterogeneous self-interested agents, we cannot expect that these agents will behave according to the institutional rules encoded in the specification. Hence, unlike approaches that allow agents to openly interact with their peers via a communication layer, we advocate for the introduction of a social layer (*AMELI*) that mediates agent interactions at run time. On the one hand, *AMELI* provides participating agents with information about the current execution. For instance, information about the participating agents in an interaction protocol. On the other hand, it enforces whenever possible the institutional rules to the participating agents. At this aim, *AMELI* keeps track of the execution state, and uses it along with the institutional rules encoded in the specification to validate agents actions.

AMELI is a *domain-independent* platform as it can be used for the deployment of any specified EI without any extra coding. For this purpose, *AMELI* loads institution specifications as XML documents generated by *ISLANDER*. In order to take part in an institution, agents are only required to be capable of opening a communication channel with *AMELI*. Thus, *AMELI* allows the participation of agents with any internal architecture and developed using any available programming language.

An EI execution can be monitored thanks to the *monitoring tool* that depicts graphically all the events occurring during an EI execution. Fairness, trust and accountability are the main motivations for the development of a monitoring tool that registers all interactions in a given enactment of an EI.

Acknowledgements

This work was partially funded by projects AT (CONSOLIDER CSD2007-0022), IEA (TIN2006-15662-C02-01), EU-FEDER funds, and by the Generalitat de Catalunya under the grant 2005-SGR-00093. Marc Esteva enjoys a Ramon y Cajal contract from the Spanish Government.

3. REFERENCES

[1] J. L. Arcos, M. Esteva, P. Noriega, J. A. Rodríguez-Aguilar, and C. Sierra. Environment engineering for multiagent systems. *Engineering Applications of Artificial Intelligence*, 18(1):191–204, January 2005.

[2] Easy java simulations. http://www.um.es/fem/Ejs.

[3] N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-agent Systems*, 1:275–306, 1998.

[4] Simulink. http://www.mathworks.com/products/simulink/.