Infrastructures to engineer open environments as electronic institutions

Dave de Jonge, Juan A. Rodriguez-Aguilar, Bruno Rosell, Carles Sierra

Artificial Intelligence Research Institute, IIIA
Spanish National Research Council, CSIC
08193 Bellaterra, Spain
{davedejonge,jar,rosell,sierra}@iiia.csic.es

Abstract. Electronic institutions provide a computational analogue of human institutions to engineer open environments in which agents can interact in an autonomous way while complying with the norms of an institution. We survey the currently available infrastructures to engineer open environments as electronic institutions: (i) AMELI, the coordination infrastructure which is core to run EIs; and (ii) the conversion of the AMELI infrastructure to run over a peer to peer network. We also discuss the type of applications that both infrastructures target at.

1 Introduction

As the complexity of actual-world applications increases, particularly with the advent of the Internet, there is a need to incorporate organisational abstractions into computing systems that ease their design, development, and maintenance. Electronic Institutions (EIs) are at the heart of this approach[1]. EIs provide a computational analogue of human organisations in which intelligent agents play different organisational roles and interact to accomplish individual and organisational goals. EIs appear as the glue that puts together self-interested parties, coordinating, regulating, and auditing their collaborations. Just like any human institution, an EI is a place where participants come together and interact according to some pre-defined protocol. It warrants that the norms of the institution are enforced upon its participants, and thus prevents them from misbehaving. An EI therefore provides the environment in which agents can interact in an autonomous way within the norms of the institution.

EIs have been under development for more than 15 years, which has resulted in a large framework consisting of tools for implementing, testing, running and visualizing them [1]. The purpose of this paper is to survey the currently available infrastructures ot engineer open environments as electronic institutions. First, we briefly outline the features of AMELI, the coordination infrastructure which is core to run EIs. AMELI was conceived to support the development of open environments for agents as EIs. However, AMELI requires that the rules of an EI are designed off-line prior to its execution. Therefore, at run-time the rules of an EI are set and do not change. However, some open environments may require

that communities of participants design and enact at run-time their own rules of interaction. For instance, consider a social network that allows its users to form their own communities and choose and enact their own rules. With this aim, here we describe the conversion of the AMELI infrastructure to run over a peer to peer network. The resulting infrastructure enables users in an open environment to to design their own communities with their own norms in the form of electronic institutions. The decentralised implementation of AMELI allows to exploit the benefits inherent to P2P systems (e.g. self-organisation, resilience to faults and attacks, low barrier to deployment, privacy management, etc.).

The paper is organised as follows. Section 2 outlines the features of the AMELI infrastructure, whereas section 3 outlines the infrastructure supporting the exectution of an EI over a peer to peer network. Finally, section 4 draws some conclusions.

2 AMELI: a core infrastructure for electronic institutions

The infrastructure that enables the execution of EIs is called AMELI [4]. AMELI enables agents to act in an electronic institution and controls their behaviour. Its main functionalities are:

- to provide a way for different agents with different architectures to communicate with one another without any assumption about their respective internal architectures; and
- to enforce a protocol of behaviour as specified in an institution specification upon the agents. This means that AMELI makes sure that the agents can only do those actions that the protocol allows them to do.

AMELI was conceived as a general-purpose platform in the sense that the very same infrastructure can be used to deploy different institutions. With this purpose, agents composing AMELI load institution specifications as XML documents generated by ISLANDER. Thus, the implementation impact of introducing institutional changes amounts to the loading of a new (XML encoded) specification. Therefore, it must be regarded as domain independent, and it can be used in the deployment of any specified institution without any extra coding. During an EI execution, the agents composing AMELI keep the execution state and they use it, along with the institutional rules encoded in the specification, to validate agents' actions and to asses their consequences.

AMELI is composed of three layers: a communication layer, which enables agents to exchange messages, a layer composed of the agents that participate in an EI, and in between a social layer, which controls the behaviour of the participating agents. The social layer is implemented as a multi-agent system whose agents are responsible for guaranteeing the correct execution of an EI according to the specification of its rules.

The participation of each agent in an EI through AMELI is handled by a special type of mediator, the so-called governor. An agent must be able to communicate with its governor, but this only requires that the agent is capable

of opening a communication channel. Since no further architectural constraints are imposed on external agents, we can regard AMELI as agent-architecture neutral.

The current implementation of AMELI can either use JADE (Bellifemine et al., 2001) or a publish-subscribe event model as communication layer. When employing JADE, the execution of AMELI can be readily distributed among different machines, permitting the scalability of the infrastructure. From the point of view of the participating agents in an EI, AMELI is communication neutral, since they are not affected by changes in the communication layer.

3 Decentralising electronic institutions

Running AMELI as a centralized application on a server has several disadvantages. First of all it requires a server. Normal users may not have access to such a server, or may not be willing to pay for server space. Moreover, you may not want to keep total control over the institution and without having to rely on a third-party. Secondly, the centralized implementation of AMELI may fail due to network traffic if the number of users increases. Thirdly, people may not want to upload their EI-specifications and other resources to a server, because of copyright or privacy issues.

For these reasons we have implemented a Peer-to-Peer version of AMELI. We have built it on top of the *Freepastry*[1] library. A free and open-source Java library that implements Peer-to-Peer network [3, 2, 5]. Freepastry provides a number of useful features such as the routing of messages, or the possibility to create broadcast messages.

A running institution is managed by a number of agents called *scene managers* and *governors*. On a Peer-to-Peer system one needs to decide on which node in the network each of these agents will be running. For this reason we have added another type of agent to the framework called the *device manager*. Each node in the network runs exactly one device manager. Whenever a new agent needs to be launched, the Device Managers determine where that agent is going to be launched. In the current implementation this is decided randomly, but in future implementations the device managers will apply negotiation to make such decisions, taking into account the capacity of each node (e.g. bandwith and cpu power).

Note that the agent participating in the EI are not directly inside the P2P network. Instead, they are connected to a governor through a direct socket connection, which is inside the P2P network. We have chosen this model for security reasons, because messages in a P2P do not always go straight from sender to receiver, but may make several 'hops' between nodes of the network before arriving at the receiver. This means that agents participating in the institution would be able to intercept those messages and manipulate them.

The P2P version of AMELI also provides a distributed database where users can publish their EI-specifications, search for existing specifications, and search

[1] http://www.freepastry.org/FreePastry/

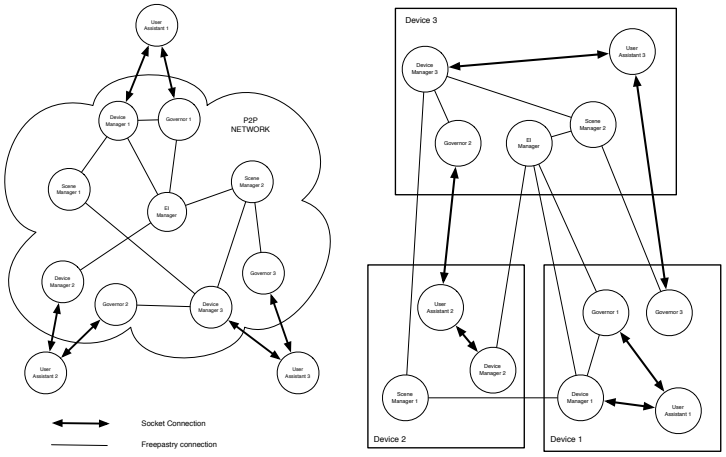

Fig. 1: Diagram of the Peer-to-Peer Electronic Institutions topology

for running instances of electronic institutions. This is implemented using the indexer/search library *Apache Lucene*.[2] Each node in the network has its own repository which is maintained by the device manager. When a query to the database is made, this query is sent to all the device managers in the network, and each of them sends back a reply, if possible.

Programming an agent that interacts in a P2P EI can be done in two ways. One way consists of extending an existing Java agent that abstracts away all the underlying communication protocols. The other way is to make use of a *rest governor*. The intended actions of the agent are then sent as http requests to the rest governor. The advantage of the second method is that one can use any kind of programming language or technology that allows making web requests, but has the disadvantage that one has to deal with the http protocol on a lower level.

4 Conclusions

We have surveyed the currently available infrastructures to engineer open environments as electronic institutions: (i) AMELI, the coordination infrastructure which is core to run EIs; and (ii) the conversion of the AMELI infrastructure to run over a peer to peer network. We argue that the decentralised implementation of AMELI eases the engineering of open environments that require that communities of participants design and enact at run-time their own rules of interaction. Furthermore, it allows to exploit the benefits inherent to P2P systems.

[2] http://lucene.apache.org/

References

1. Josep Lluís Arcos, Marc Esteva, Pablo Noriega, Juan A. Rodríguez-Aguilar, and Carles Sierra. Engineering open environments with electronic institutions. *Eng. Appl. of AI*, 18(2):191–204, 2005.
2. Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. One ring to rule them all: service discovery and binding in structured peer-to-peer overlay networks. In *Proceedings of the 10th workshop on ACM SIGOPS European workshop*, pages 140–145. ACM, 2002.
3. P Draschel and A Rowstron. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, pages 329–350, 2001.
4. Marc Esteva, Bruno Rosell, Juan A. Rodríguez-Aguilar, and Josep Lluís Arcos. Ameli: An agent-based middleware for electronic institutions. In *Proceedings of AAMAS*, pages 236–243, 2004.
5. Antony Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. Scribe: The design of a large-scale event notification infrastructure. In *Networked group communication*, pages 30–43. Springer, 2001.