A framework for Agent Based Social Simulation

Candelaria Sansores and Juan Pavón 1

Dep. Sistemas Informáticos y Programación
Universidad Complutense Madrid
Facultad de Informática UCM, 28040 Madrid, Spain

csansores@fdi.ucm.es, jpavon@sip.ucm.es

Abstract. Our research centres in the application of computational modelling to social sciences, more concretely, computer simulation of social phenomena. When these programs are represented specifically like agents we refer to them as Agent Based Modelling (ABM). The executions of these agent-based computational models are known as Agent-Based Social Simulations (ABSS). There are various libraries for developing agent-based simulation systems but they require modellers to have a good working knowledge of the programming language that they are using. Our purpose is twofold: to ease agent-based simulation providing customizable tools for modellers with basic programming knowledge, and supporting the execution of scalable models.

1 Introduction

A current trend in social science research proceeds by building representations of social systems as societies of agents, executing and analysing their behaviour. Agents, in this context, are autonomous software systems that are intended to describe the behaviour of observed social entities (e.g. individuals, organizations). An advantage of this technique, known as Agent-Based Social Simulation (ABSS), is the ability to estimate the plausibility of the behaviour of agents, the way in which agents interact, and the consequences of that behaviour and interaction.

(Axtell 2000) reviews on the many motivations for employing agent-based computing in the social sciences, and argues that there exist three distinct uses of agent based simulation, depending on the complexity of the social model: agent models as classical simulation (when models can be formulated and completely solved), artificial agents as complementary to mathematical theorizing (for partially solvable models), and agent computing as a substitute for analysis (when models are intractable or probably insoluble). We focused this work on the last two, which are more common usage of computational agent models. Specially in the case that mathematical models can be written down but not completely solved, the agent-based model can shed significant light on the solution structure, illustrate dynamical

1 This work has been developed with the support of the Consejo Nacional de Ciencia y Tecnología (CONACYT) from México and the project TIC2002-04516-C03-03, funded by the Spanish Council for Science and Technology.
properties of the model, serve to test the dependence of results on parameters and assumptions, and be a source of counter examples. The other usage arises because there are important classes of problems for which, although it is possible to describe a model, there are some parameters and configurations which are not stable or can change in a variety of ways, and this makes extremely difficult to perform a full analysis. In those cases, agent-based simulation can provide further information on the behaviour of the system.

In ABSS the model is a multi-agent system (MAS), a set of autonomous agents that operate in parallel and that communicate with each other. The properties of individual agents describing their behaviour and interactions are known as elementary properties, and the properties emerging on the higher, collective level are known as emergent properties.

This emergence is an important issue for studying complex social systems. The global behaviour is said to emerge from the agents and their interactions when the simulation is run, the macro patterns and processes that emerge are then compared to the empirically observed patterns of the society, thus MAS allow the exploration of the micro-to-macro\(^2\) relation (Coleman 1990). Also, it is important to notice the subtle difference between MAS and ABSS framed by (Conte 1998): “if the MAS field can be characterised as the study of [or implementation of] societies of artificial autonomous agents, agent-based social simulation can be defined as the study of artificial societies of autonomous agents”.

As ABSS has gained in popularity as a simulation tool, software tools for modellers are emerging. There are various libraries for developing agent-based simulation systems. REpast (RePast 2004) is a set of Java libraries that allow programmers to build simulation environments, create agents in social networks, collect data from simulations automatically, and build user interfaces easily. Its features and design owe a lot to SWARM (Swarm 2004), one of the first agent-based modelling libraries. Another similar library is ASCAPE (Ascape 2000).

These libraries have great advantages for modellers over developing their own, but also have some limitations. They require modellers to have a good working knowledge of the programming language that they are aimed at: in the case of the tools mentioned above, Java. But usually, modellers are sociologists or economists, with basic programming knowledge, so we should support their activity with tools to specify a simulation model in a high-level declarative language instead of a low-level programming language.

This idea appears in SDML (SDML 1997), a modelling language based on Smalltalk. Unlike ASCAPE and REpast, it does not demand that users are fluent in the underlying programming language, but it does require users to learn a complex interface that can be as difficult to master as a full programming language.

Another requirement for ABSS is the ability to support scalability in the definition and execution of the agent-based simulation model. Compared with a traditional simulation system, like those based on mathematical equations, a simulation system based on MAS frequently requires a large amount of computational resources for its

\(^2\) Micro simulations are simulations based on the properties of lower level units such as individuals, in contrast to macro simulations of the system dynamics variety, which attempts to directly model emergent macro phenomena.
execution, besides its design and implementation are also often complex. Although several platforms have been implemented to develop this type of systems the models they support use to be limited to small scale. This is not a problem if only a single deliberative agent is tested in a dynamic environment or the coordination strategies of a moderate number of agents are tested. But this is not the case of systems that this work is oriented to. This proposal is oriented to social systems whose model size is important, since variation in the number of interacting agents affect the main results of agent-based social simulations (ABSS). Empirically, it is known that size is time-dependent in numerous social systems (e.g., urban centers) and that group or system size do really matter for systems and processes involving collective action and similarly significant social phenomena (Cioffi-Revilla 2002). Therefore, there are no theoretical foundations to state that a model with the same number of agents as individuals observed in a large sized social system will behave in the same way as a model with a few agents.

For this reason, in the case of these types of systems, we require means to scale up models and not limiting simulation size. Furthermore, agents that comprise these models should be able to make use of distributed computational resources, firstly because they often require huge computing resources and secondly because the data sets required by the simulation may also be geographically distributed. In this manner, we can argue that a parallel, distributed simulation is desirable, improving model validation. Grid computing (Foster and Kesselman 1998) plays an important role for this to happen, providing the technology for accessing a large scale of distributed resources and supplying the means for constructing and running complex systems like distributed social simulation applications.

Section 2 presents a global framework for ABSS covering those requirements. This framework is based on the use of the facilities of the INGENIAS Development Kit (IDK), a highly customizable tool for modelling and developing MAS, by adapting it for the social domain, and a set of support components on a grid computing infrastructure.

Section 3 discusses how we consider the adoption of a grid computing solution to the execution of large-scale agent systems.

Section 4 describes the activities in this framework that a modeller should perform to simulate a new social system model on the grid computing infrastructure.

Finally, in section 5 we summarize the benefits of the proposal, and describe the current status of this work.

2 Overview of the ABSS framework

Our goal is to provide a platform that addresses the modelling, implementation and deployment of ABSS systems on a Grid computing environment, removing the burden of programming from modellers. In Figure 1 we can see all the components of the

---

3 Validation is one of the steps of a generic research framework for all simulation studies (Gilbert and Troitzsch 1996), which concerns whether the simulation is a good model of the target. A model which can be relied on to reflect the behaviour of the target can be considered valid.
the ABSS framework, including the Globus Grid Infrastructure. Also, we consider the use of INGENIAS Development Kit (IDK) for presenting customizable editors and automatic code generation tools, which will facilitate the modelling of the systems.

Figure 1. ABSS framework components

The modeller basically has to define an agent-based social model with the customized IDK graphical editor, generate code for the target platform (Globus in this case), and run the simulation to gather results.

Considering the diversity of social problem domains, we think that instead of providing a general tool for ABSS implementation, a way to simplify the development of social models is to provide an easily customizable framework for modelling social phenomena at different domains. This implies that there could be specific model editors for each domain and different mappings into the execution platform. The IDK is built from the specification of MAS meta-models and provides APIs to manage them in order to generate the editors, verification tools, and code generation modules. Therefore, we just need to specify meta-models for each social simulation domain and the mappings for the execution infrastructure. The task of specification and generation of the domain-specific editors and code generators is not normally the responsibility of the modeller, but of some ABSS platform developer, who should know INGENIAS meta-models and Java programming skills for
customizing the editor and the code generation tools (this is supported by a plugin of IDK with Eclipse). This task has to be done once for a specific problem domain, and several modellers can benefit from the domain-specific tools.

The IDK (see Figure 2) is based on the principles of INGENIAS methodology, which considers the definition of a set of meta-models that describe the elements that form a MAS from several viewpoints, and that allow to define a specification language for MAS. The specification of a MAS in INGENIAS consists on the definition, control and management of each agent mental state, the agent interactions, the MAS organization, the environment, and the tasks and objectives assigned to each agent. MAS modelling is facilitated by a graphical editor and verification and validation tools. As complement to these tools, there is a generic process for parameterization and instantiation of MAS frameworks to concrete target platforms with the Code Generator component. Notice that meta-models could be adapted or new ones will be created as necessary depending of the social system to implement.

![Figure 2. Modelling components](image)

Also, we can see in Figure 1 the ABSS support, which is considered part of the platform and that is necessary to allow the execution and monitoring of agents on the Grid environment, in this case Globus Toolkit 3 (GT3). An essential component of the framework is the Agent-Grid enabler, an add-on to INGENIAS code generator. This add-on is a middleware that enable agents to act and run on GT3.
3 Agent simulation on a grid computing infrastructure

Computational Grids are wide-area distributed environments that focus on infrastructure, tools, and applications for reliable and secure large-scale resource sharing. The Grid provides a large-scale infrastructure environment for Multi-Agent Systems that need scalability, robustness, and security. Both technologies seem to have synergies between each other; however, this work centres its attention in exploiting the benefits that Grid technology can provide to large-scale MAS execution. This requires the definition of an efficient architecture of MAS on grid computing environments, which could become a complicated task as new mechanisms have to be designed and developed for distributing secure agents execution, service discovery, resource management, and load balancing on the Grid.

The Grid enables the creation of problem solving environments (PSEs). PSEs provide an environment through which scientists or end users run complex applications that usually require collaboration and interaction of a large amount of distributed heterogeneous resources. (Foster, Kesselman et al. 2001) stated that grid computing is concerned with coordinated resource sharing not subject to centralized control and problem solving in dynamic, multi-institutional, virtual organizations. Being a virtual organization a set of individuals and/or institutions defined by sharing rules.

Consequently, the interoperability in the Grid requires of protocols, interfaces and policies that are not only open and general-purpose but also standard. Standards allow us to establish resource-sharing arrangements dynamically with any interested party and thus to create something more than a plethora of incompatible or non-interoperable distributed systems.

On the other hand, the definition of these standard protocols is the single most critical problem facing the Grid community today. One of the accepted proposals is a widely used de facto standard, the open source Globus Toolkit (Globus). This software toolkit is a set of services and libraries that allow us to program Grid Services, making it easier to build Grid infrastructures and Grid applications; it also provides a set of Grid services for security, resource management, information access, and data management.

A Grid Service is a Web Service that conforms to a set of conventions that provide for the controlled, fault resilient and secure management of stateful services. OGSA (Open Grid Service Architecture) (Foster, Kesselman et al. 2002) addresses architectural issues related to the requirements and interrelationships of Grid Services. In this way, OGSA further refines the Web Services model to focus on the features required by Grid infrastructures and applications.

OGSA aims to define a new common and standard architecture for grid-based applications, defines what Grid Services are, what they should be capable of, what types of technologies they should be based on, but doesn't give a technical and detailed specification. OGSI (Open Grid Service Infrastructure) (Tuecke, Czajkowski et al. 2003) is a formal and technical specification of the concepts described in OGSA, including Grid Services.

Globus Toolkit 3 (GT3) is build around OGSA and is an open source implementation of OGSI. The implementation is intended to serve as a proof of concept for OGSI, and to be used as a reference for other implementations.
The toolkit components most relevant for this work are: (i) the Core infrastructure (Sandholm and Gawor 2003) that provides hosting environments for the services and a security infrastructure (GSI) as well as a number of system-level services, such as logging and management, (ii) the Index Service which provides information about the available resources within the grid and their status, and (iii) the GRAM (Globus Resource Allocation Manager) which simplifies the use of remote systems by providing a single standard interface for requesting and using remote system resources. This is typically used to support distributed computing applications.

Once the MAS code is generated, it will be deployed on top of the ABSS support as shown in Figure 3. This support consists of five components, which are classified as high-level services (a manager and monitors), low level services (schedulers), and services for agent execution and agent information. The primary goal of these components is the distribution, localization and management of agents belonging to a social simulation model for its distributed execution on the Grid.

To achieve its goal, the framework will offer (1) easy and convenient access to execute agents in the Grid, (2) security mechanisms, (3) ease of use, hiding the complexity of the Grid system to developers, and improved accessibility of Grid resources to end users, and (4) management and control of the simulation process.

The simulation manager allows submissions of agents to the Grid system, dispatches and delegates the management of them to schedulers, which in turn sends them to the Java execution services responsible indeed of the execution. The agents are implemented in Java because its portability in distributed, heterogeneous environments and for its current optimization that has improved the performance of Java applications (Ghahramani and Pauley 2003).

The schedulers search for available and suitable resources required by the agents to be run on remote nodes and manage the execution, collecting simulation data by

![Figure 3. Grid computing infrastructure for ABSS](image-url)
retrieving agents’ outputs if necessary. Normally, the Grid infrastructure provides a registry service that contains index information of services and resources available and those we can localize through a service discovery method. We will take advantage of this characteristic of Grid infrastructure as possible to implement schedulers.

The Java execution Grid service is a computation service that executes a submitted Java agent by starting up a Java Virtual Machine (JVM) to run the agent. It provides interfaces for basic monitoring of agents execution status.

The agent information Grid service is responsible of agents’ localization; it acts like a registry of simulation agents belonging to a specific simulation process. Once the agents are initialised, they register in the agent information grid service to notify and to receive notifications of other agents’ localizations. Also this service could maintain more sophisticated information like agents’ execution status reported by the agents themselves, or information provided by the schedulers like agents they manage.

Lastly, the simulation monitors have two goals. The first goal is to present information of agents’ execution status to end users; the second goal is to act like an observer that communicates with simulation agents and with the environment to collect data and present intermediate results about the simulated process to users.

Because MAS have no central control (Wooldridge and Jennings 1995), the global behaviour of the system is said to emerge from agents and their interactions based on agents’ autonomous actions. Therefore, we do not provide a central process that collects information from each agent and then decides what action each agent should take; doing so would be in an opposite position to some MAS principles. A similar argument has been made by (Davidsson 2001). Based on this, our approach assumes that simulations that are going to be carried out are time driven distributed simulations.

4 Basic Steps of Deployment

Given the structure of the ABSS framework, the deployment of an ABSS system on the Grid can be summarized as the following steps:

1. The modeller develops an agent-based model of a social system with IDK adapted editor and generates Grid enabled agents with the code generator.
2. User logs in to the simulation manager. The simulation manager authenticates the user and loads the user’s personal settings or credentials like the user’s certificate to get access to services and resources in the Grid.
3. The user sets any appropriate configuration specific to every simulation process through the simulation manager, which provides a graphical user interface (GUI) for agents’ submission and management.
4. Distribute agents in the Grid nodes with the simulation manager that in turn invokes a scheduler to serve the agents. The scheduler searches for suitable resources from the indexing service and dispatches them to the suitable Java execution services to be executed.
5. Starts the distributed simulation process and retrieves the simulated data for its processing and evaluation with the aid of the simulation monitor.

The security of the ABSS system will be assured in all Grid nodes through delegation of proxy certificates that enable single sign-on mechanism.

The framework provides all necessary interfaces to interact in the Globus protected environment, which supports authentication, authorization, and message protection through standard protocols like SSL/TLS and standard X.509 certificates and other extensions to standard protocols.

5 Conclusions

We proposed the development of a software platform based on IDK for the agent-based modelling and simulation of complex social systems. The platform will provide modellers with a customizable graphical editor for each application domain. In this way, modellers can centre on their modelling activity.

The simulation is supported by tools that generate code for the agents and deploy them on the Grid environment. The modellers are supported as much as possible in the implementation details by an automatic process.

Added low level services support execution of simulations on the Grid environment, and high level services support monitoring of those simulations and gathering of results.

Thus, we ease agent-based simulation providing customizable tools for modellers with basic programming knowledge, and supporting the development of scalable models hiding the complexity of the Grid system from the users.

We are in the first stages of this project. We expect to accomplish a case study to contribute with results on performance, taking into consideration that we are more concerned in providing tools that ease the design and implementation of agent-based social simulations, and in case performance gets compromised, we could always consider techniques to minimise communication overheads like the grouping of agents that interact intensively.

Compared with other simulation tools we reviewed before, our approach addresses issues not considered all together by these tools. Actually, they are more concerned with the programming task and do not take into consideration the conceptual modelling task. Our approach supports the whole research process from conceptual modelling to simulation analysis based on the principles and tools associated to the INGENIAS development process.

We consider fundamental to establish a common language and procedure to describe agent-based models applied to the social sciences, independently of the implementation platform, mainly because it is necessary in the scientific community as a means of communication. Also, but not less important, we have identified the need in many users for being able to replicate models with different platforms.

Replication contributes to the reliability of the model results and better understanding of the system. As (Axelrod 1997) stated, replication is one of the hallmarks of cumulative science. It is needed to confirm whether the claimed results of a given simulation are reliable in the sense that they can be reproduced by someone
starting from scratch. Without this confirmation, it is possible that some published results are simply mistaken due to programming errors, misrepresentation of what was actually simulated, or errors in analyzing or reporting the results.

Finally, to achieve the necessary methodological rigour demanded to any scientific work we have to satisfy the Axelrod’s three main goals in the development and programming of an agent-based social model: \textit{internal validation}⁴, \textit{usability} (it must be easy to run the model and analyse the results) and \textit{extensibility} (it must be possible to adapt and reuse the model). Thus, our approach is compromised in supporting these issues providing adaptable tools that perform the correct transformation of the conceptual models to program code.

6 References


---

⁴ Internal validation, also known as \textit{verification} (Gilbert and Troitzsch 1996) in the simulation field, is concerned with the correctness of the transformation from the abstract representation (the conceptual model) to the program code (the simulation model). That is, with ensuring that the program code faithfully reflects the behaviour that is implicit in the specifications of the conceptual model (Axelrod 1997).
IDK "INGENIAS Development Kit." [http://grasia.fdi.ucm.es/ingenias](http://grasia.fdi.ucm.es/ingenias)