Project no.
035086
Project acronym
EURACE
Project title
An Agent-Based software platform for European economic policy design with heterogeneous interacting agents: new insights from a bottom up approach to economic modelling and simulation

Instrument STREP

Thematic Priority IST FET PROACTIVE INITIATIVE "SIMULATING EMERGENT PROPERTIES IN COMPLEX SYSTEMS"

Deliverable reference number and title
D1.3: Graphical user interface to build and develop complex agent-based models in economics
Due date of deliverable: 31.01.2008

Actual submission date: 31.01.2008

Start date of project: September 1st 2006 Duration: 36 months

Organisation name of lead contractor for this deliverable
TUBITAK National Research Institute of Electronics and Cryptology – TUBITAK UEKAE
Revision 2

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Graphical User Interface to Build and Develop Complex Agent-based Models in Economics*

EURACE Deliverable D1.3

Kaan Erkan, Mehmet Gençer, Bülent Özel, Vehbi Sinan Tunalıoğlu

Contents

*This research was funded by the European Commission as part of the FP6-STREP project EURACE: "An agent-based software platform for European economic policy design with heterogeneous interacting agents: new insights from a bottom up approach to economic modeling and simulation".

1 Introduction

Agent based simulations is the choice of approach in EURACE project aiming at a better understanding of European Economy and production of software for use in policy design. During planning phase of the project it was decided that the existing agent based simulation tools from FLAME project (Holcombe et al., 2006) would be used as a base to build the necessary software upon. Although FLAME has been used in several types of agent-based simulations, it lacked some capabilities needed for a realistic simulation of European economy and had no tools to account for requirements of use targeted in EURACE.

The EURACE project objectives required implementation of several software tools to present a complete platform, in addition to adaptation of FLAME for needs of the project. One of these tools is a graphical interface to aid agent model designers so that for different types of agents in the European Economy (e.g. households, firms, banks) knowledge and behavior of agents could be filled in in a convenient way. By use of this tool the problem should be accessible to the designers (economists) in ways which is compatible with common conceptualization of such models by economy experts, and it should also enable use of these models in FLAME. Following this degisn step one would need to create an instance of simulation using the agents designed, by creating a population of agents and their relations, then test the overall design. Finally the design should be presented to the policy designers with adequate tools which allow them to run what-if analysis they need.

The deliverable presented here covers the first piece of the user interface software in EURACE, which is the graphical user interface (GUI) for agent model design.

2 Architecture of the overall user interface software system

Currently adopted software architecture, shown in Figure 1, reflects the three types of uses and users for which the complete software platform is targeted for. The functionality that is used in several places was organized into several libraries to allow for code reuse and parallel implementation. Implementation of software tools has started from pieces that are at the top of the illustration and proceeds downwards. Deliverable No 1.3 presented in this document summarizes the Agent design GUI and its library. This GUI was named XMML-Editor and its library was named libxmm.

A historical note on naming is necessary here as it relates to FLAME. The FLAME framework uses X-machines at its core to represent agents. As a result the file format to store agent models in FLAME was named XMML

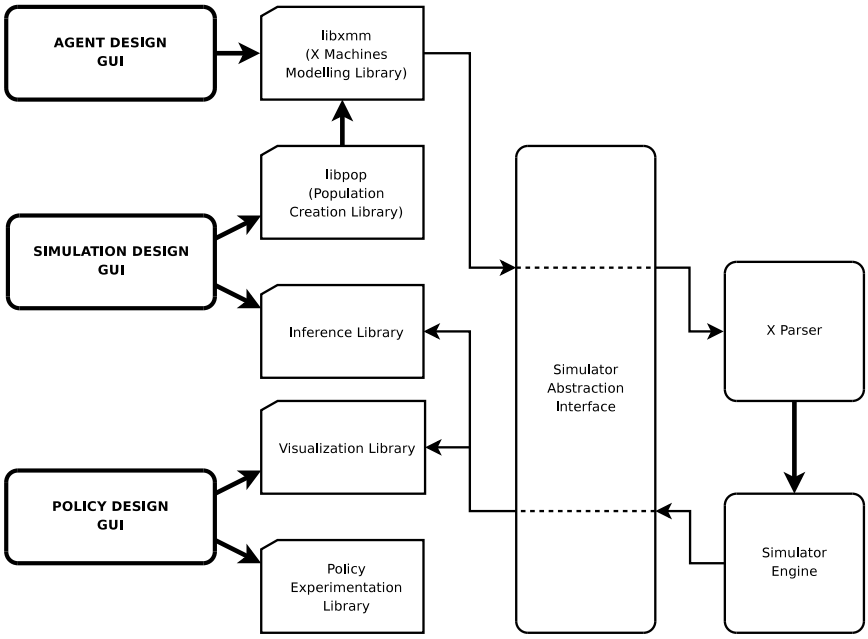

Figure 1: EURACE Software Architecture

which is short for "X Machines Modeling Language", and the computer program which processes these models was called Xparser. The GUI which was produced as part of Deliverable 1.3 ultimately passes on the agent models to FLAME in XMML format, and therefore we have chosen the name XMML-Editor for it.

The XMML-Editor was expected to enable the economy experts to reflect an agent design they have in mind in a form amenable to computer simulation. Therefore it was necessary to follow generally accepted conceptualizations and terminology in computational economy in GUI design, while at the same time ensuring that the design can be represented in canonical form in the XMML format of FLAME, which most probably needed to be updated to do so.

TUBITAK-UEKAE researchers have worked closely with both economy experts from various partner organizations in EURACE project, and also with software specialists at University of Sheffield (USFD) and Council for the Central Laboratory of the Research Councils (CCLRC) for better satisfaction of these criteria during design and implementation of XMML-Editor and libxmm. During the discussions and additional concern was identified, that of evaluating the agent designs for possible impediments to parallelization, as it was a central issue when one needs to run millions of agents in simulations.

Key design prospects that came out of this collaborative effort can be summarized as follows:

- Abstractions of agent models which were adopted by economy experts (Catalano et al., 2006; van der Hoog and Deissenberg, 2007) should be followed in XMML-Editor to the extend which is both generic enough to be used in other complex systems projects and is also trivial for use in the EURACE project itself.

- Since some of the functionality overlaps with other pieces of EURACE software platform agent model manipulation functionality of XMML-Editor was designed as a separate library, called libxmm.

- In addition several features were added to XMML-Editor for easier detection of mistakes in agent models and to assist improvement of parallelization to achieve more efficient simulations.

The rest of this report presents detailed design perspective and features of XMML-Editor and libxmm. The software itself and its documentation can be reached online at `http://eurace.cs.bilgi.edu.tr`.

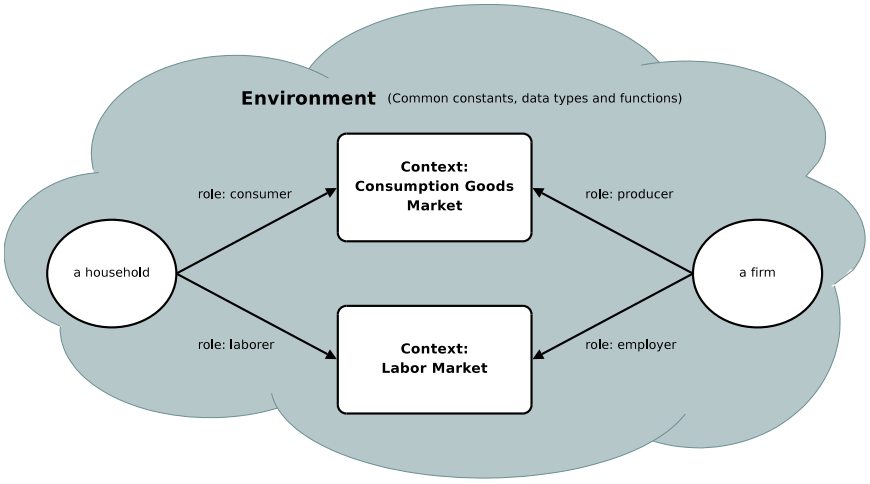

Figure 2: Example layout of interactions in a complex economic system

3 GUI tools for agent design

In many multi-agent simulation frameworks, agent designers are expected to express agent behavior and memory in a format and programming language of framework's choice. Since agent populations in most cases consist of one or two types of agents with relatively simple behavioral specifications, this approach has been successfully applied in many domains of applications. However when agent design process is considered in the EURACE project for simulation of the European Economy, this simplistic approach fails as there are several classes of agents with complex behaviors and various contexts of action corresponding to different market sections of the economy.

The XMML-Editor was created to ease the design process for agent model designers by providing (1) a convenient graphical interface in which designers can arrange their model in a natural way, (2) facilities for visualizing inter-dependencies between models of several agents to facilitate easier error detection and improvement, and (3) means to generate XMML files which can be used to run their design in the FLAME multi-agent simulation framework. Therefore FLAME, XMML-Editor, and a third component to be built in the course of EURACE project to ease management of populations provides a complete bundle for running multi-agent simulations, although inference tools are also indispensable for use of the system. Interaction of these various pieces was visualized in the software architecture in Figure 1.

Although starting point for XMML-Editor has been an economic model, its design build on a more generic perspective in order to make it useful in other domains of application, without jeopardizing its suitability for the very complex uses in EURACE project. The core of this approach is based on arrangement of agent behavior into agent roles which are in turn related

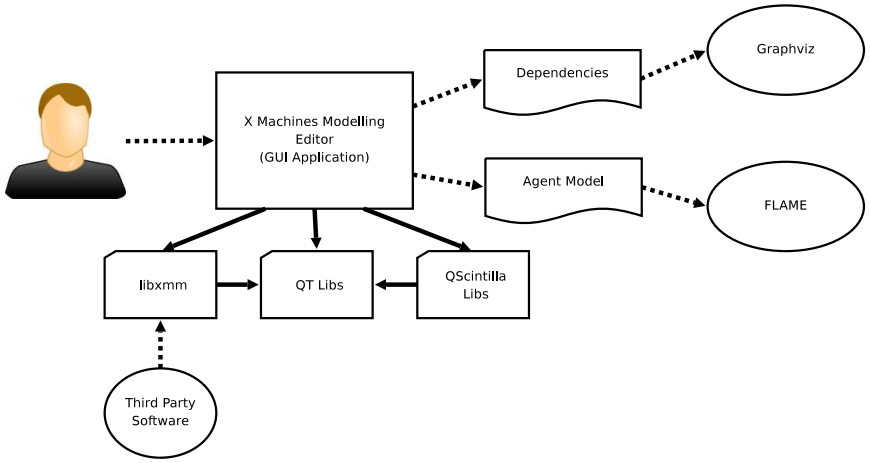

Figure 3: Dependencies in XMML-Editor and libxmm

to more-or-less independent contexts of activity.

The resulting abstraction of agent model design rests on the concepts of interaction contexts, environment, and agent roles. As shown in Figure 2, interaction between agents in a complex system like European economy can be better laid out if they are divided into different contexts such as labor markets, consumer goods markets, or financial markets. The term 'environment' was used in XMML-Editor in reference to common things shared by all agent designs such as constants, functions and data types specific to the simulated system (rather than the emergent situation in any run of simulations).

Primary way the software is used is through a graphical user interface (GUI). Several technologies were evaluated for use in GUI implementation, in terms of platform independence, license conditions, and capabilities. As a result the GUI component is written in C++ language for platform independence, and utilizes Trolltech Inc.'s Qt and Riverbank Inc.'s QScintilla libraries for their capabilities and suitable licensing terms. In addition functionality related to manipulation of agent models is encapsulated in a C++ library which can be interfaced from other third party software if desired. This should enable, for example, enhancements to XMML-Editor to be implemented in a very short time if desired, or ease development of EURACE related software in future. A graphical representation of dependencies between XMML-Editor's pieces is shown in Figure 3.

One of the capabilities the GUI offers is visualization of dependencies between actions of agents. When users attempt to show dependencies, XMML-Editor will export them into a dot file which can be displayed by Graphviz software. If Graphviz is not available this functionality will not work.

Once designers are finished with agent model, XMML-Editor can export agent model specification in the XMML format suitable for FLAME.

4 Components and Features

4.1 Graphical User Interface

Presentation of agent design which is natural for the agent model designers has been a topic of long discussion within EURACE project. Eventually it is agreed[1] that agent behavior will be arranged into agent roles, and system messages will be arranged into contexts. The main screen of XMML-Editor GUI reflects this design decision, a screen-shot of which is shown in Figure 4. Each item in the two lists of Agents and contexts tabs, one showing agents, roles and activities, and other showing contexts and messages in a hierarchical tree view, can be right clicked to display a menu which allows modifying or deleting items. Similarly the environment tab shown in the second screen shot captures all things relevant to environment specification of agents.

[1] Bielefeld Workshop, June 2007

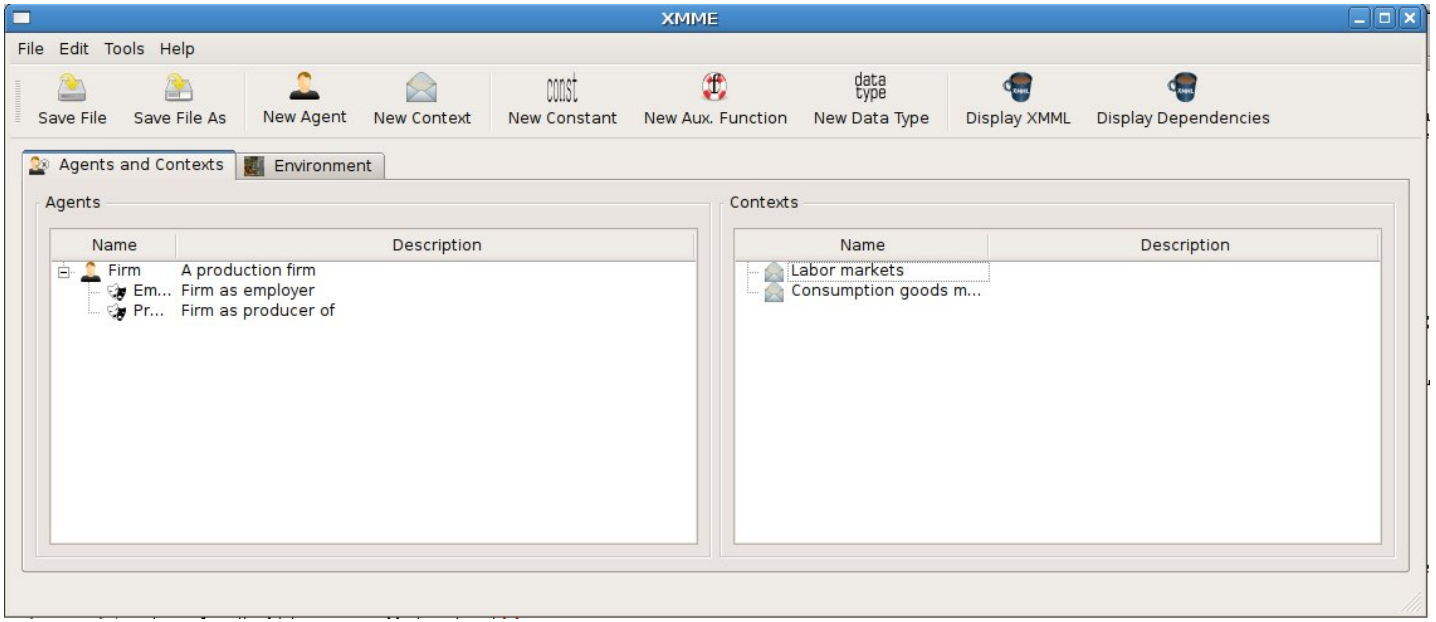

Figure 4: Main screen of XMML-Editor showing agent role and context arrangements, and a second tab

Defining agents consists of entering agent's memory variables and activities, in addition to its name and an optional description. Memory variables have a type (such as integer, floating point number, etc.) and name, but also they have a so called init-form which can be used by designer to indicate a random model to be used when creating populations of agents. These init-forms are in a syntax such as random(0,100), which indicates a uniform random distribution between two values. There has been no initial requirement to extend of init-forms for the XMML-Editor. Indeed, the program is indifferent to content of this entry by designers. Instead these values will be interpreted by future software components of EURACE project that will actually create the populations. However, a recommendation list for init-forms is to be provided in the future with the user manual of XMML-Editor for convenience. The toolbar of the main window allows users to create any type of entity with just one click.

When defining agents, designers also describe activities and their dependencies on other activities and messages. This was required in various meetings on the matter as ordering of activities reflect the internal logic of agent's activities, such as plans for production increase affecting whether a firm should hire more workers or not, etc.

Defining contexts is less complicated and consists of defining messages in addition to context name and description. For messages one must enter the type and name of message, as similar to agent's memory variables.

For both agent memory variables and messages, designers are free to use 'types' which are compound types such as array of integers, etc. There is no interpretation or whatsoever in XMML-Editor regarding these types. Rather their 'validity' is to be determined during compilation in FLAME framework. However it is strongly recommended that one uses the basic types listed in the user manual or data types defined in environment specification in XMML-Editor.

The user interface can show dependencies between agent activities with the help of external Graphviz program, using the button on the main window. This feature was added with the expectation that such visualization can help designers detect mistakes in entry by visual check.

Once finished with design one can both save it in XMML-Editor's native format, XMME, and export it to XMML format for use in FLAME, from file menu. The same menu also allows saving the design, with different file name if desired. The XMME format captures agent model design as it is represented in XMML-Editor, and has extra fields which are not exported to XMML format. A formal XML specification of XMML format is shown in Figure 5. Merging of the two formats into a unified scheme is planned, but currently is not an obstacle to functioning of the system.

In summary the main window toolbar, main menus, and item menus displayed when one right clicks on agents, roles, activities, contexts and environment entities provide access to following actions:

```
<!ELEMENT xmodel (name, version, lastEditTime,
          lastEditor, description, xagent*, xcontext*)>
<!ELEMENT xagent (name,description,xmemvar*,xrole*)>
<!ELEMENT xmemvar (type,name,description,init)>
<!ELEMENT xrole (name,description,xactivity*)>
<!ELEMENT xactivity (name,description,code,dependency*)>
<!ELEMENT xcontext (name,description,xmessage*)>
<!ELEMENT xmessage (name,description,xmsgvar*)>
<!ELEMENT xmsgvar (type,name,description)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT lastEditTime (#PCDATA)>
<!ELEMENT lastEditor (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT init (#PCDATA)>
<!ELEMENT code (#PCDATA)>
<!ELEMENT dependency (#PCDATA)>
```

Figure 5: XMML-Editor native format for storing models

New Agent The toolbar button is used for creating new agents.

Modify or Delete Agents Right clicking on agents shown in the tree view displays a menu to modify or delete agents.

New Context The toolbar button is used for defining new contexts.

Modify or Delete Contexts Right clicking on the contexts list in the tree view displays a menu to modify or delete contexts.

New Constant The toolbar button allows one to add new constant definitions to the environment.

Modify or Delete Constants Right clicking on constants displays a menu to delete the constant. They can be edited in place in the list.

New Auxiliary Function The toolbar button allows creation of new Auxiliary functions.

Modify or Delete Auxiliary Functions Right clicking on the Auxiliary function in the list displayed in environment tab displays a menu to modify or delete the function.

New Data Type The toolbar button allows creation of new data types. The pieces of the data structure is in turn added in the displayed dialog screen.

Modify or Delete Data Types Similar to auxiliary functions, right clicking on data type in the list will display a menu to modify or delete them.

New Agent Actions Right clicking on the agent roles displays a menu to create new agent activities. This displays a detailed screen with five tabs to define activities description, enter the C code for activity, define dependencies on other activities and messages, and agent memory variables read or modified by the activity.

Modify or Delete Activities Right clicking on he activity displays a menu for modifying or deleting activities.

Export to or Import from XMML Format Tools menu contains commands to export to or import from agent designs in FLAME's XMML format.

Save in XMME Format File menu contains commands to save the design in XMML-Editor's native format: XMME. This format captures all aspects of presentation of agent model to designer.

Visualize Dependencies Tools menu also contains a command to export dependencies in Graphviz .dot format and in turn run Graphviz program —if available — to display the dependencies graphically.

4.2 Agent Model Manipulation Library: libxmm

Manipulation of agent models in XMML-Editor is encapsulated in a C++ library, called libxmm[2]. This is extremely useful if different user interfaces (e.g. a web based one) is desired as an alternative to current XMML-Editor GUI, or additional facilities will be provided as add-in modules or programs to manipulate models. The library was built on Trolltech's Qt library and distributed with GPL license. Therefore it is free for use in GPL software applications. As Qt library is highly portable, libxmm can be used for various target platforms including, but not limited to, Unixes, Linux, MacOS, and Microsoft Windows.

Abstraction of agent models in libxmm library is general enough to allow its use in different application contexts for simulations of complex systems, although it is primarily designed to account for economic models in EURACE project. Here we summarize how the classes in libxmm map to agent model definitions. For programming examples and full library API documentation readers are recommended to look at the XMML-Editor website.

[2]Its full API documentation is available at XMML-Editor website at `http://eurace.cs.bilgi.edu.tr` in HTML and PDF formats.

The classes in libxmm correspond to entities one encounters when using the XMML-Editor GUI. Following is a summary of what these classes represent and how they are used:

XModel Represents the complete model. A model contains a single environment (XEnvironment), multiple agents (XMachine), and multiple contexts (XContext). This class can be constructed from an existing XMME document or string representation of model for convenience. It also provides convenience methods for export to and import from XMML format.

XEnvironment Represents environment for the model, containing multiple constants (XConstant), auxillary functions (XAuxFunc), and data types (XDataType).

XMachine and XMemVar Represents an agent. It can contain multiple roles (XRole) and memory variables (XMemVar).

XContext, XMessage and XMsgVar Represents a context, containing multiple messages (XMessage). An XMessage in turn contains multiple variables (XMsgVar).

XRole and XFunction A role contains multiple activities of type XFunction.

XConstant, XAuxFunc, XDataType, and XDTField These represent entities in an environment. A data type contains multiple data fields (XDTField).

For better schematization, object-property relations for some classes are shown in Figure 6.

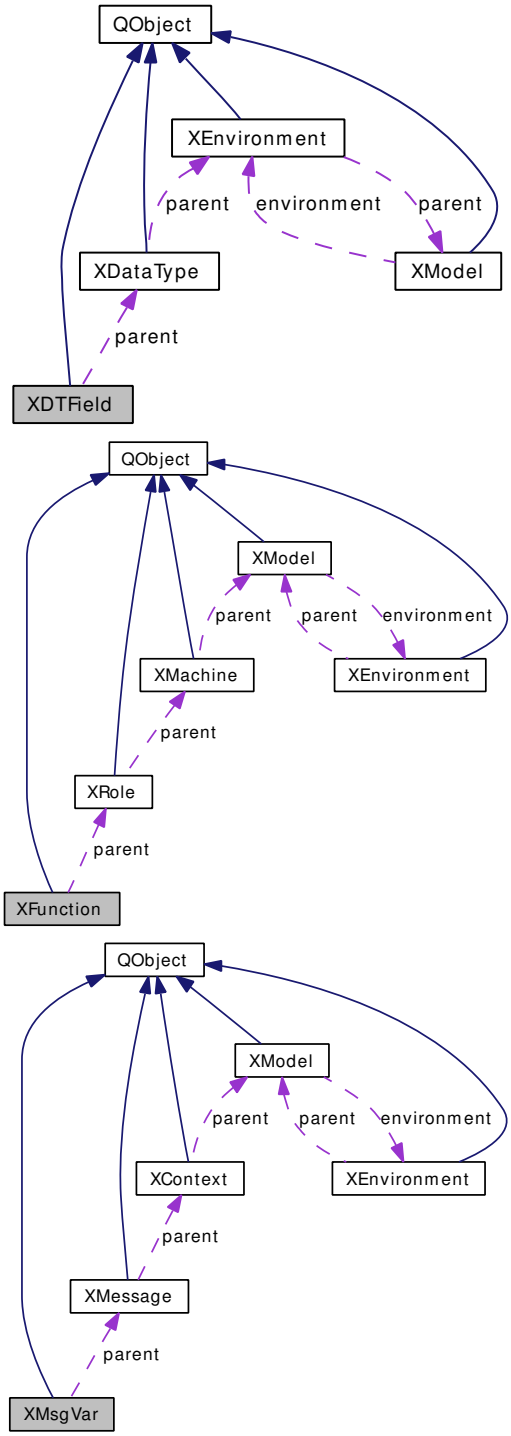

Figure 6: Object-property class relations in `libxmm`

5 Present status and future work

This report presented the agent design GUI and its library to build and develop complex agent-based models in economics. It targets economy experts who want to reflect FLAME based agent design in a form amenable to computer simulation. Therefore currently implemented GUI design follows generally accepted conceptualizations and terminology in computational economy while at the same time it ensures that the design can be represented in canonical form in the XMML format of FLAME.

Presented component of GUI helps economists to create an instance of simulation using the agents designed. Subsequent components as it is depicted within body of this report will be used to create population of agents and their relations, and then test the overall design, and will provide tools which allow policy designers to run what-if analysis they need. Current GUI design approach is quite modular and allow us to improve/add new features when new requests and necessities emerge.

Prior to release of agent design GUI, modelers have implemented their agent models by editing raw XML suitable for FLAME framework. Since the initial release of agent design GUI modelers are actively using it and providing feedback for improvements and bug fixes. Due to the complexity of models accumulated so far this transition goes parallel with improvement of models and development of other parts of the user interface and simulation software. Software teams in EURACE projects are currently working to implement the second part of these software tools which is for creating populations and handy management of inference on simulation output. This will provide a complete platform to economy experts and allow us to conclude the transition.

References

Catalano, M., Clementi, F., Gatti, D. D., Guilmi, C. D., Gaffeo, E., Gallegati, M., Giulioni, G., Napoletano, M., Palestrini, A., and Russo, A. (2006). The C@S project. EURACE Working Paper.

Holcombe, M., Coakley, S., and Smallwood, R. (2006). A general framework for agent-based modelling of complex systems. EURACE Working paper WP1.1, Department of Computer Science, University of Sheffield.

van der Hoog, S. and Deissenberg, C. (2007). Modelling specifications for EURACE. EURACE Working paper WP2.2.