DESIGNING A TOOLSET FOR THE PRIME VIRTUAL BUSINESS ENVIRONMENT

PAVEL BOYTCHEV[1], BOYAN BONTCHEV[1], ROUMEN NIKOLOV[1], MANUEL OLIVEIRA[2], IVAN KOYCHEV[3]

[1)] *Sofia University, FMI, Dep. of Inf. Technologies, Sofia, Bulgaria, {pboytchev, bbontchev, roumen}@fmi.uni-sofia.bg*

[2)] *Alfamicro Ltd., Lisbon, Portugal manuel.oliveira@alfamicro.pt*

[3)] *Dep. of Inf. Research, Inst. of Mathematics and Informatics, BAS, Sofia, Bulgaria, ikoychev@math.bas.bg*

ABSTRACT

The PRIME (Providing Real Integration in Multi-disciplinary Environments) project[1] aims in modeling, creating and evaluation of a Virtual Business Environment (VBE) for learning the entire life cycle of products and processes for all involved stakeholders and for gaining practical experience in strategic manufacturing through serious gaming. The paper presents the overall architecture of the PRIME system. Further it discusses in more detail the design of the game toolset, which provides the user interface to system models such us: the life cycle of products and processes and all involved stakeholders. By providing various tools for management of business process semantics and dynamics, the toolset facilitates defining different end user scenarios upon scenarios templates within the VBE.

INTRODUCTION

The PRIME project aims in modeling, creating and evaluation of a Virtual Business Environment (VBE) for learning the entire life cycle of products and processes for all involved stakeholders and for gaining practical experience in strategic manufacturing through serious gaming (PRIME, 2005). The game model of PRIME represents a complex set of business processes modeled on real-life processes in industrial enterprises. The business processes are prioritized by their importance regarding strategic manufacturing and are described by sequences of decisions. For example, the Make-or-Buy decision is a complex one that is equally important (or have same priority) to the less complex Protecting Innovations/R&D decision. A workflow built by such sequences of decisions (Cutumisu, 2005) is not explicitly represented in PRIME, i.e. it is not imposed on real players as a gaming framework but will be incorporated in the realization of artificial stakeholders (players) playing a role either of "proxies" of real players when last are offline or being competitors to real players. So, instead of restricting real players doing their business only by following predefined sequences, PRIME gives the players freedom do fulfill any available decision allowed by its pre-conditions, without guiding players step by step through a sequence. Thus, the game leaves the players to learn how to make decisions in non-determined situations proactively acquiring the needed information and using it.

[1] Providing Real Integration in Multi-disciplinary Environments (PRIME), FP6-IST, NMP-2004-IST-NMP-1 - Integrating Technologies for the Fast and Flexible Manufacturing Enterprise, Project Reference: 16542

Once business processes of strategic manufacturing are described by means of strategic decisions and game object model is defined in a stable and consistent way, there appears crucial the need of a flexible mean for managing the data model of the Virtual Business Environment (VBE) which is supposed to be the core of the game itself. This mean is planned to be the game toolset which should provide useful tools used for creation, control and moderation of the PRIME game. Modern games rely on toolsets for general management of a virtual world represented by the game (Jacob, 2005). They are managed and monitored by means of set of tools with which you can build (configure, set up) any instance of game in the chosen category (Flanagan, 2005). Usually, there are two types of roles having right to access and control the game by the toolset, namely game administrators and moderators. PRIME is not going to make exclusion. On other hand, moderators are something atypical of PRIME that may not exist in other games. There are no rules for the support staff of an online game such as in (JungHyun, 2005) but user support is a major investment that greatly exceeds technological maintenance. However, while most of the modern serious games use toolset for creation, control and moderation of the structures and processes inside the game virtual world, the PRIME toolset has to provide another set of tools for controlling the artificial stakeholders available in the game and used for creating critical mass of players in order to simulate the market in a believable way. This incurs additional complexity and the toolset design has to deal with it, because in tools for Artificial Stakeholders (AS) such as (Riedl, 2003) we have to deal with the complexity of configuring the knowledge database and the decision process along with the personality of the AS (Silverman, 2001).

Moreover, the toolset should be flexible and extendible enough in order to allow addition and modification of tools as the PRIME game is supposed to evolve very dynamically. Some of the tools will be simpler enough in terms of operational complexity regarding changes of the VE data model, however, there are supposed to be included tools which needs a higher complexity of interventions with the data model and VBE itself. This is the reason to introduce a script language for designing patterns for the tools and, on other hand, a script engine, which will deal with execution of scripts on the server side.

The paper describes design of a toolset as a solution of the problems stated above. First at all, there have to be determined the roles of the actors using the toolset, and their use cases. Having these use cases defined, we will be able to define exactly the context of the toolset, i.e. which types of manipulations will be supported by its tools. Next, there will be defined the software architecture of a flexible and extendible toolset solution. There will be outlined the syntax of a declarative script engine powerful enough to del with all the PRIME concepts including control over the business units, regional markets, artificial stakeholders, etc. Finally, the conclusions will trace out the way of implementing the designed toolset.

MOTIVATION

The term *toolset* is used in the paper as a container of a set of tools (*toolbox*) running not at the server side but at the client, while the scripting engine runs on the server. Before starting with definitions of the tools to be included into the toolset, there are needed to be defined the actors (users) operating the toolset system.

Toolset actors

The PRIME conceptual model includes two basic user roles: System and Player. In the System role there are two types of users depending on there responsibility:

- VBE Administrators - initialize and manage the whole VBE via the toolset. They can manage the game agents (i.e. AS and VBE agents).
- Moderators (Mentors) - monitor the game and players. Via the toolset they can manage AS as well, thus becoming “players” via proxy.

The Players are also two kinds:

- Real (Human) Players - play the game and manipulate game world through exposable Graphical User Interface (GUI)
- Artificial Stakeholders - intelligent agents operating either as competitors to the players or as cooperative partners

The role that each Artificial Stakeholder have to play is to act as an artificial player, created automatically for each Business Unit (BU) in the game. The ASs can act different roles depending on their business relationship to the players e.g.:

- competitor - managing a similar BU;
- collaborator - as a partner in a joint-venture or as a supplier.

Furthermore the paler can delegate responsibilities to an AS to care about his business, when the player goes offline.

There are defined two system roles – moderators and VBE administrators. Both they can control different aspects of the PRIME game accessing the data model through a specialized GUI component – the PRIME toolset.

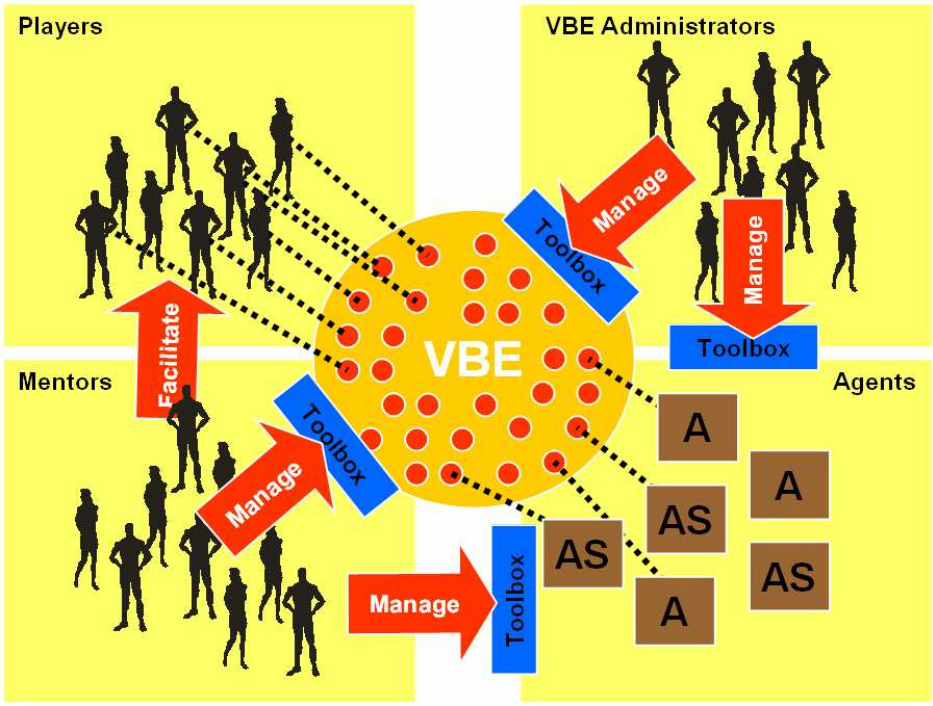

Figure 1: Users responsibilities in PRIME

Figure 1 presents users responsibilities at conceptual level. Both mentors (moderators) and VBE administrators manage VBE and AS through the toolset. However, there are substantial differences between specifics of their management, based on their scope. The moderator is like a windows system power user whilst the VBE administrator has full access rights.

In terms of responsibilities, we do not differentiate between global and local scope, since meaning of global and local is rather relative. So one would choose, for example, raw materials, then select iron and then the world would reveal where he/she have reservoirs of iron. He/she could then add, remove or modify those reservoirs. As with any drawing program, users can zoom in/out and define the granularity of their brush.

The set of tools

In all the cases encountered in previous section, management includes (C)reation, (V)iew, (E)dit and (D)eletion (C/V/E/D) of correspondent items. There needs to be checks in place to see if the changes don't break the consistency of the data model. As well, we should also consider to have (C)ommit an (R)ollback – if the introduced changes do break the consistency of the data model. For this purpose, there should be checking whether this occurs or not. We can define cases where we apply all or some of the C/V/E/D actions. Each tool within the toolset is represented by metadata (its name, description, permissions, etc.), GUI, scripts (patterns) and, eventually, control logic of usage.

The toolset of PRIME is intended for two main purposes:

1) to manipulate the data model
2) to introduce changes and personalization/parameterization features into the behavior of the Artificial Stakeholders (AS)

Both of the tasks over suppose controlling and managing the dynamic issues of the game. Static issues build two main groups:

1) Structural elements - fixed mainly within the structure of the relational data model and are not going to be changed. Structural elements are represented by tables and columns and are not subjects of modification.
2) Parameters - static issues which are represented as records within in the data model and are fixed for the total game play, such as the geographical regions, KPI, etc.

Toolset design issues

The design of the toolset should to be powerful enough in order to support all the scenarios. Potential toolset users have to agree what type of scenario is meaningful and useful for them and next, the toolset developers have to confirm what is possible to be done in the time determined for the PRIME project. Thus, it can be resumed in two main points:

1) to have a very flexible design – easy support of new tools just by adding new tool descriptions; tools can be edited by changing the object descriptions so the properties of an object field attribute can be changed (i.e. from Read Only to Read/Write) which will be reflected by the Object Viewer itself;
2) to provide a sufficient level of simplicity and generality.

On other side, the tools and their dependency and hierarchy are not dependent from the usage scenarios but from the entity relations defined by the game object model and (partially yet) in the database model. These relations will define the tools' hierarchy; however, if we follow exactly the levels of relationships in the data model we might achieve a vertical and narrow toolset hierarchy, with few tools at each level. Of course, having a deep tools hierarchy with many nested levels will not be useful regarding user interface, thus, we have to restrict the tools levels.

TOOLSET ARCHITECTURE

Basic principles

Here we encounter the following important toolset issues:

- The toolset uses a part of common client-server API to communicate to the server and, thus, to the scripting engine
- As well, it will use a proprietary toolset-server API – as it has functionalities different from the PRIME client; there will be some toolset calls to the server, which are not to be used by the client as they are reflecting the nature of the toolset and thus, they are useless for the client. Such a decision was made the previous meeting and in

addition to the advantages of discriminating the functionality, it would also be good from security point of view.

- Provides means for adding new tools and modification, freeze or delete existing tools
- Means for syntax
- VBE client does not depend on the toolset
- VBE server does not depend on the toolset. However, the toolset need a running server in order to access the data model.

The toolset inside the overall PRIME architecture

Each tool within the toolset is represented by metadata (its name, description, permissions, etc.), GUI, scripts (patterns) and, eventually, control logic of usage. The tool metadata is defined in XML or XMI and describes the main properties of the tool.

The user interface for the first prototype will be a set of tabs having HTML input elements for entering information (entry fields, select boxes, etc.) and control elements for requesting data from the VBE server; for the second prototype we plan to reuse components from the PRIME client, e.g., components for selection of regions, companies, etc.

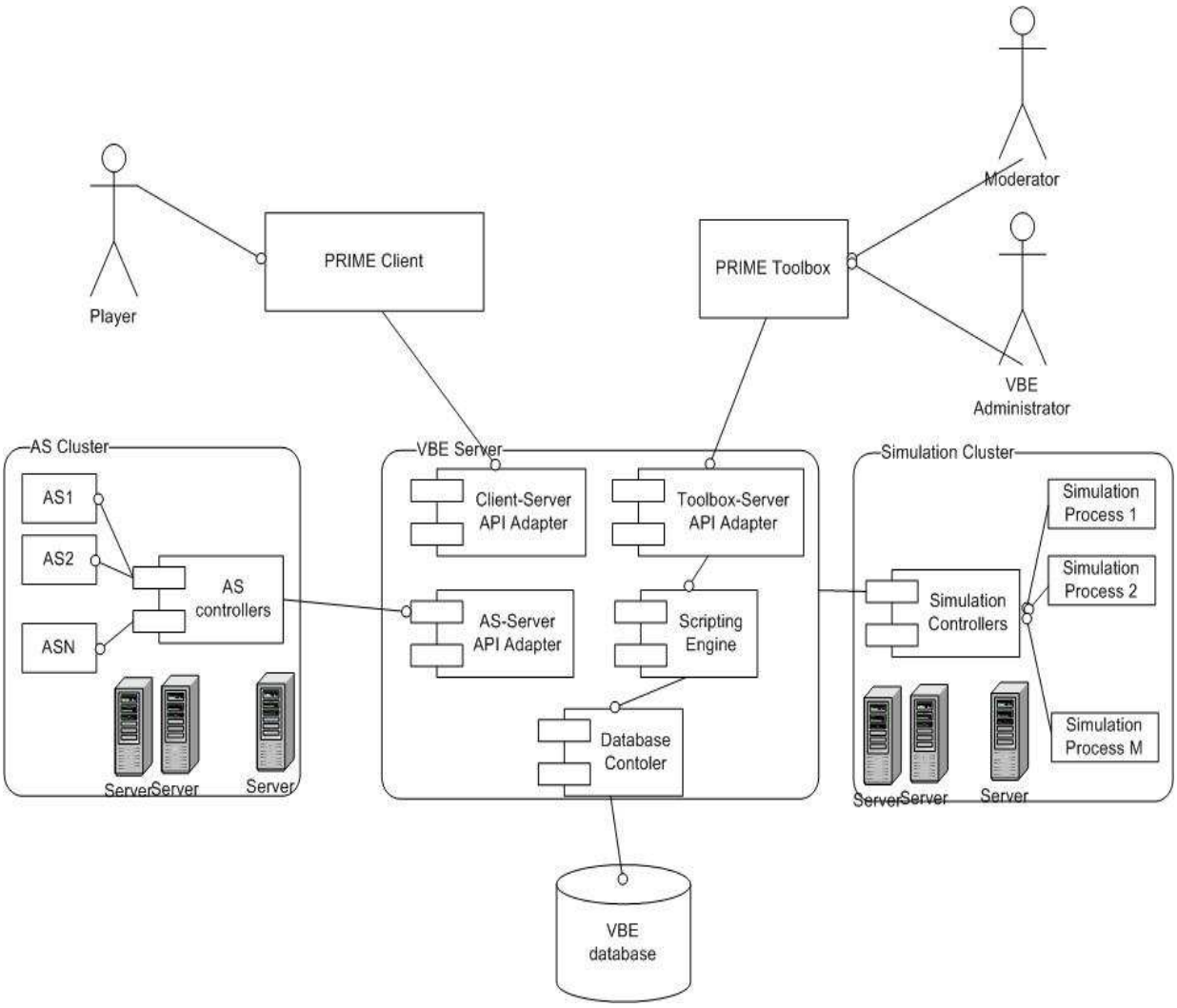

Figure 2: Overall software architecture of the PRIME game

The scripts are written in a declarative scripting language ensuring sufficient power in order to describe the game entities in a descriptive, human-readable way, However, the scripting language does not modify the structure of the data model, it just populates it. Scripts can add new rules and behaviors, but no new tables, neither columns in the underlying data structure.

Figure 2 shows the overall software architecture of the PRIME game with a special focus to the toolset and its interactions with the VBE server. The player's client communicates to AS (in order to customize it) only through the VBE client-serve API. The client and the toolset both use a common VBE client-server API.

AS and Simulation cluster are controlled by specialized controllers. The hierarchal simulation model resides on the VBE server, which then keeps track of all the simulation nodes that may, or may not, reside on the same machine. AS controllers and Simulation controllers use specific interfaces for communication to the server, different from the common VBE client-server API.

Although many modules (the client, the Toolbox, each AS) access the data in the database all access passes through the database controller of the VBE Server.

The toolset anatomy

The architecture of the toolset is represented in Figure 3. Each of the tools is built based on its tool description and is manipulated by a general object viewer. Thus, the toolset contains two main components: *GOM (Game Object Model) Description* and *Object Editor*. The Object Editor is a universal viewer and editor for game entities using the GOM Descriptor as a pattern for navigation and screen layout. The Object Editor is the static component of the Toolset, i.e. when finished it will be able to process any entity in the GOM even those created in the future. New objects descriptions can be added by the Tool Manager. The GOM Description is the dynamic component of the Toolset. It will evolve together during the next phases of the PRIME development.

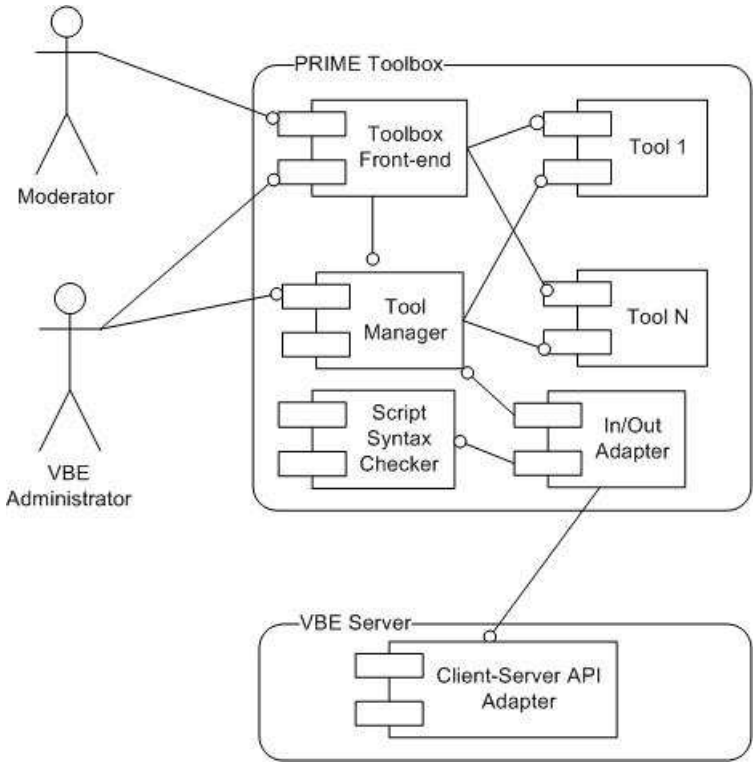

Figure 3: Architecture of the PRIME toolset

The purpose of the GOM Description is to provide description of the GOM model in a useable by the Object Editor way. The Object Editor will use the GOM model to display data about each entity; to provide information about how to extract the data from the PRIME server; to provide navigational paths from one entity to another and to describe the modification methods for each entity. Additionally it will allow a flexible tuning of the toolset and unification of the definition of all objects in the PRIME game. Physically the GOM Description could be a text file, a configuration file, a database, or in-memory array, whatever is convenient for the Object Editor.

The Object Editor is a universal object viewer and editor. It is the actual software of the toolset. It traverses the GOM description mapped onto the actual data from the PRIME Server. The main functions of the Object Editor are to give access to the hierarchy of entities in the PRIME game; to provide easy navigation to related entities and their modification. Navigation can be described as traversing the entities' tree in three main directions:

- to lower-level entities (i.e. going deeper in the GOM)
- to higher-level entities (i.e. going up in the GOM hierarchy)
- to sibling entities (i.e. not immediate relatives, but share common properties)

Navigation to siblings is important, because it will provide means to extract related information. For example, while examining the properties of an earthquake in a specific region, the navigation to sibling entities could access to data about all companies which have offices in this region, or the access to other events at the time of the crisis, or some other group of entities, which have a common parameter with the earthquake.

The modification of entities can be classified in four groups depending on the properties which will be modified.

- Modification of simple properties – numbers, texts, dates. This can be done by a simple text box where users edit the data, or with user-friendly choosers, e.g. a Calendar Chooser for dates, Region Chooser for geographic location, etc.
- Modification of collection properties – these are sequences of simple data organized visually in a table or spreadsheet. Collections may represent historical data, aggregated data, or multi-instances data.
- Modification of objects. Nested objects provide the means of navigation to lower-level GOM hierarchical elements
- Modification of object collection (add, change and delete of individual objects)

Because the structure of the objects is defined in the GOM Description, the Object Editor is not able to modify it by adding new elements or changing the type of existing elements. When a new entity is introduced in the Game Model, it should be first included in the PRIME Server, then in the GOM Description. There will be no need to change the Object Editor as long as it does not favor any of the entities in the GOM Description, and it works uniformly on any level in the hierarchy.

Apart from managing data, the Object Editor will automatically create the visual appearance of screens in the toolset based on the current GOM entity, which is viewed. All simple properties can be grouped in a single table; collections and objects can be displayed as single entries in this table (i.e. collapsed view) or can be visualized as standalone tables (i.e. expanded view). The creation of screens is done on-the-fly depending on the currently viewed entity. Changes in the Object Editor will be needed only if there is a significant change in the GOM Model which cannot be expressed with the then-existing GOM Definition.

EXAMPLE OF THE TOOLSET GOM DEFINITION

In this section we present a simple example of a fragment of the Toolset GOM Definition representing server-side game objects. It is not complete. The actual physical structure could be different, however, for the purpose of demonstration we will use a tree-like visualization.

The entries in the definition have two system attributes * and ^{1}. X* means that X may exist as 0, 1 or more instances (for example, there might be 0, 1, or more WARS under CRISES). X^{+} means that X may exist as 1 or more instances (a WAR can span over many regions, thus having at least one LOCATION). X without any attribute means that X must exist as a single instance (for example, for each WAR there is only one STARTDATE).

```
VBE
  ├─CRISES                    ; enumerator for types of crisis
  │   ├─WAR*                  ; description of wars (0, 1 or many)
  │   │   ├─STARTDATE         ; for each war there is a single starting date
  │   │   ├─ENDDATE           ; and a single end date
  │   │   ├─LOCATION+         ; and a list of  1 or many locations
  │   │   └─IMPACT            ; and a coefficient of business impact (100%=total destruction)
  │   └─EARTHQUAKE*           ; description of earthquakes (0, 1 or many)
  │       ├─DATE              ; there is a single date for each earchquake
  │       ├─LOCATION          ; and a single location (epicenter)
  │       └─GRADE             ; and Rihter-scale grade
  ├─REGIONS                   ; definition of a region
  │   └─SUBREGIONS*           ; 0, 1 or more nested subregions
  │       ├─GOVERNMENT        ; region's government
  │       └─LABOR_MODEL       ; labor model for the region
  └─BUSINESS_UNITS            ; definition of a business unit
      ├─NAME                  ; name of the business unit
      └─SITE+                 ; 1 or more sites
          ├─LOCATION          ; site location
          ├─SALES_UNIT*       ; 0, 1 or more sales units
          │   ├─EMPLOYEES     ;  employees in the unit
          │   └─CONTRACTS*    ; 0, 1 or more contracts
          ├─DISTRIBUTION_UNIT*      ; 0, 1 or more distribution units
          │   ├─EMPLOYEES     ;  employees in the unit
          │   ├─DISTRIB_CENTER1     ; 1 or more distribution centers
          │   └─NETWORK       ; distribution network
          └─FINANCE_UNIT*     ; 0, 1 or more finance units
              ├─EMPLOYEES     ;  employees in the unit
              ├─ASSETS        ; assets
              └─CAPITAL       ; capital
```

The toolset design presented in the paper provides a powerful and flexible solution for controlling the virtual business environment of the PRIME game. Each screen of the toolset GUI will provide functionality for opening more detailed information, drill down for further details, going back to a higher level in the hierarchy, editing (manually or with wizards, adding and removing elements of collections which have multiple number of instances.

Most of these functions can be interfaced by buttons or mouse clicks. It is a problem of a more detailed design what method will be used. The GOM definition will contain system information related to not only the nature of each element in the description, but also to how this element is retrieved. If there is a specific API function for each element, the GOM Description may have one API name attached to element in the GOM Description. The Object Editor will use this API function to retrieve the data of the element.

AS will also be defined in the GOM Description. The moderator will have to include different number of AS from each prototype, according to scenario he wants to provide. The players will have to outperform AS competition, which involves flexible tactics and considering each situation in its full context. In contrast, AS are straightforward and act according to different models (Silverman, 2002) such as:

- *Aggressive* – takes all possible opportunities
- *Individualist* – does not consider teaming but tries to do everything himself
- *Cautious* – acts only when chances for positive outcome are very good
- *Mating* – looks for the best player to have as a partner
- *Equalizing* – unites with others to beat best player

CONCLUSIONS

The development of modern game tools is a process that imposes several seemingly conflicting goals, such as rapid development usually frequently overlapping with development of a game itself, easiness of use, good flexibility and maintainability, etc. Often these tools must work with or produce data for legacy code or engines, or be at a quality level such that they can be released alongside a game to aid the moderators' community. The paper presented design principles of the PRIME toolset being currently under development. The toolset of PRIME is going to manipulate the data model of the PRIME virtual business environment and to introduce changes and personalization/ parameterization features into the behavior of the artificial stakeholders.

The presented toolset design provides a great deal of flexibility and extendibility as far as each of the tools is built based on its tool description and is manipulated by a general object viewer. By means of the Tool Manager, game administrators will be able to add new objects descriptions as far as the description of the game object model is the dynamic component of the Toolset. It will evolve together during the next phases of the PRIME development. On other side, administrators will use the object editor as a universal viewer and editor for game entities.

REFERENCES

PRIME - Providing Real Integration in Multi-disciplinary Environments - Project proposal, FP6-IST, NMP-2004-IST-NMP-1, Proposal no.: FP6-016542, May 24, 2005.

Cutumisu M. (2005), Generating Ambient Behaviors in Computer Role-Playing Games, LNAI 3814, Springer-Verlag (Intetain Conference 2005), pp. 34-43.

Jacob T. (2005), A Toolset for Creating Iconic Interfaces for Interactive Workspaces, Proc. of INTERACT 2005 - IFIP TC13 International Conference, Rome, Italy, September 12-16, 2005, also in Lecture Notes in Computer Science 3585, Springer, ISBN 3-540-28943-7, pp. 699-712.

Flanagan M., (2005), Nissenbaum H. Values at play: design tradeoffs in socially-oriented game design, Proc. of the SIGCHI conference on Human factors in computing systems table of contents, Portland, Oregon, USA, pp. 751-760.

JungHyun H. (2005), Multi-platform Online Game Design and Architecture, Proc. of INTERACT 2005 - IFIP TC13 International Conference, Rome, Italy, September 12-16, 2005, also in Lecture Notes in Computer Science 3585, Springer, ISBN 3-540-28943-7, pp. 1116-1119.

Silverman B. (2001) Toward a Human Behavior Modeling Anthology for Developing Synthetic Agents, Proc. of 10th Conf. On Computer Generated Forces and Behavioral Representation, SISO, May. 2001 , pp. 213-228.

Riedl M. (2003), Character-Focused Narrative Generation for Execution in Virtual Worlds, Proc. of the Second Int. ICVS Conf., Toulouse, France, November 20-21, 2003, Proceedings. Lecture Notes in Computer Science 2897 Springer 2003, ISBN 3-540-20535-7, pp. 47-56.

Silverman B. (2002), "Human Behavior Models for Game-Theoretic Agents", Cognitive Science Quarterly, vol. 2, no. 3-4, pp. 273-301.