An Integrated Development Model for Character-based games

Marconi E. Madruga Filho, Allan J. S. Souza, Patrícia C. A. R. Tedesco, Danielle R. D. Silva, Geber L. Ramalho

Universidade Federal de Pernambuco, Centro de Informática, Brazil

Abstract

Character-based games are strongly dependent on the quality of their Non Player Characters' behavior. Developing games of this nature usually requires a high investment of time and money on Artificial Intelligence techniques, in order to provide better credibility to the characters. For the independent game developers' community, affording this kind of extra complexity may be highly expensive. Therefore, this article proposes a way of developing character-based games while maintaining high quality and low-cost. This is done by using a modular Artificial Intelligence technique called Rule-Based Systems, integrated with a game development support tool, XNA. Since Java is better provided with Rule-based System tools, the IKVM application is used to allow this integration. All mechanisms and tools used are free. A comparison of free Rule-based Systems is presented, as well as an application of the proposed model on a real game project.

Keywords: virtual characters, artificial intelligence, character-based games, game development process

Authors' contact:
{memf,ajss,pcart,drds,glr}@cin.ufpe.br

1. Introduction

The development cost of a commercial game has been steadily increasing. More and more, predictions come up stating that developing for the next-generation consoles will cost more than double the current-generation value [Morris 2009]. Furthermore, the demand for more realistic games also increases. This realism is not only about good-looking scenarios and game environments, but also about game characters' behavior and intelligence. This is the case of character-based games, like the The Sims [Maxis 2000] series or even Role-playing games, which are games that rely mainly on their characters' illusion of life shown to the player. This search for behavioral realism normally entails a demand for a more sophisticated game AI to define the characters. Furthermore, the AI sophistication creates new challenges for the game development, as well as a higher game development cost.

For independent game developers, this extra cost may cause the development of good character-based games to be too expensive. These developers always look for ways to produce games while executing the hard task to maintain the balance between high quality and low monetary and time cost. One way of doing so is to use less expensive AI techniques that still have great power to simulate characters' behavior, as is the case of Rule-based Systems (RBS). In this case, RBS need to have good performance, documentation and tool support, and easy integration, in order to give a better support to character-based game development.

Therefore, this article shows a comparative study of various available RBS tools, based on the requirements related to character-based game development. By using the results of this study, the article proposes a process that supports the modeling of characters with rich behavior. This is done by using free platforms and tools. This approach is also base for a character-based training game, VTEAM [VTEAM 2009], whose goal is to improve the players' capacity of managing human resources. Then, the characters are modeled richly using true personalities which are commonly found on working teams, in order to create conflicts and situations to be surpassed by the player.

This paper is organized as follows. Section 2 illustrates the use of Artificial Intelligence in character-based games, especially Rule-based systems. Section 3 discusses choosing an adequate RBS and game development technology. The character-based game development model is proposed on section 4. The study on applying it on a real case is presented on Section 5 and section 6 shows conclusions about the research and possible future works.

2. Character-based games and Independent Development

Character-based games are the type of game that rely strongly on the behavior of its non-player characters (NPCs). These are games like Black & White [Lionhead 2001], by Lionhead Studios, where the player interacts with an autonomous character which may behave differently, depending on the player's actions. The illusion of life and intelligence passed by the game's autonomous characters captivates the player and makes the game more interesting. However, to achieve this effect it is necessary to reach a certain level of realism on the character's behavior. Thus, character-based games require a strong focus on their character's modeling and on thinking about how these

characters are going to make the player feel like dealing with a living being.

This credibility is shown when the character's behavior is adaptable. Some attempts have been done to develop this kind of behavior in games. In the beginning this was done using simple AI techniques such as Finite-state Machines (FSM) [Millington 2006], as done in classic games such as Pacman [Namco 1980]. However, the amount of work required to model character behavior using FSM depends on the size of the problem. As the demand for believability increased, modeling behavior using FSMs became more complicated.

Therefore came up the need to incorporate more sophisticated AI techniques such as Bayesian Networks, machine learning, fuzzy logic, rule-based systems and others. Among the available approaches, rule-based systems are, probably, the most used [Bourg and Seemann, 2004]. One of their main advantages is that they model the way how people usually think, since the rules are written in declarative language. For this reason, they are easy to work with, as well as flexible and powerful to solve problems of various natures. Rule-based systems have been used in many games [Cavazza 2000], such as Civilization: Call to Power [Activision 1999] and Rainbow Six [Red Storm 1998].

On the other hand, investing in RBS or any other artificial intelligence technique might add an extra cost to game development. This goes against the demand from independent developers' community for low-cost solutions and mechanisms to develop character-based games. For this reason, even when choosing RBS for their technical advantages, it is necessary to consider how much one has to spend in order to include this approach in one's project.

2.1 Rule-based Inference Engines

Rule-based systems (RBS), also known as Production Systems, are already a very popular AI technique, due to being easy to implement and powerful. The control of the RBS rules is made by an inference engine, the heart of the system. The rules are, in general, condition-action pairs; the inference engine checks the rule conditions and, if they are true, the rule is fired, which means the rule actions are executed. One can opt to implement an inference engine from scratch to use in one's game, but there is also already a great variety of implementations available, such as Drools [JBoss, 2009] and Microsoft's BizTalk [Microsoft 2009], even though they are not directed specifically to game development. When choosing an inference engine, the developer must keep in mind the character's demand for realism and believability.

Based on our previous development experience, on character-based games, we have elicited some requirements for choosing RBS in Character-Based games. They are:

- **Embeddability**: The language in which the rules are written must be easy to comprehend. It may be close to natural language or to a standard programming language. This rule language should be implemented integrating concepts of object orientation and production rules, a reusable and easily integrated approach known as Embedded Object Oriented Production Systems (EOOPS) [Pachet 1995]. On EOOPS, due to their integrated behavior, objects from the programming language might be used on the rule conditions and actions, preserving and dealing with the aspects of object-oriented language (encapsulation, inheritance, and so on). For instance, object's fields can be checked on the rule conditions and object's functions can be called on the rule actions. This facilitates rule writing and reuse.
- **Performance:** Since the goal is to develop games, performance seems to be RBS's main issue. It is not useful for a game to have great character behavior realism if it will sacrifice the game's performance. The RETE algorithm is the pattern matching method most commonly used and has already been shown to be efficient [Forgy 1982].
- **Development support:** Even when there is good performance, readability and integration, a fine support for developers is essential. This includes precise documentation, additional tools, development support features such as rule debugging. In addition, integrated environments and support tools are also helpful and should be taken into account. Besides that, it is preferable to choose active RBS projects, for they have more intense updating, bug correction and community support.
- **License cost:** A free engine is vital to avoid adding additional costs associated with the rule engine. This is fundamental when development is independent or academic.

	Platform	Embeddability	Performance	Docs & Tools	License
Drools.NET	.NET	High	+(RETEOO)	None	Open-source
NxBRE	.NET	Poor	Unknown	Medium	Open-source
Simple Rule Engine	.NET	Medium	Unknown	Poor	
Drools Expert	Java	High	+(RETEOO)	High	Open-source
JEOPS	Java	High	+(RETE)	Medium	Open-source
Jess	Java	Poor	+(enhanced-RETE)	High	Academic use
Soar	Independent	Poor	+(RETE)	High	Open-source

Table 1: Comparison between free rule-based systems

Besides the above criteria, the choice of the inference engine is also associated with the development platform on which the game will be developed. The choices of an adequate RBS as well as of the development platform are discussed on the next section.

3. Developing a Character-based game

Character-based game development clearly stands for a hybrid of game and intelligent system development. Hence, requirements come from both sides. From games, comes the need to use a proper game development platform, with support to audio and input control, as well as other game development features. Besides, from intelligent systems, comes the need for choosing an appropriate AI technique to implement, on this case, Rule-based systems. The selected RBS and game development platform must be decided, in order to develop an idea of development process for this kind of games.

3.1 Choosing an RBS

There are many RBS available. A great number of them, though, are proprietary, like Microsoft's Business Rule Framework [Microsoft 2009], ILOG Rules for .NET [ILOG 2009], and Jinni [BinNet Corporation, 2005] and do not conform to the requirements of independent or academic development. Among the free rule-based inference engines available, seven were selected and they are described in the list below. These RBS were evaluated using the requirements listed on section 2.1; the result is detailed below and compiled in the Table 1.

- **Drools.NET (.NET):** Drools.NET [Drools.NET 2006] is an open-source .NET implementation of a java inference engine named Drools. It is out-of-date, compared to its java companion, has poor unfinished documentation and does not have tool support. Its strong point is the embeddability, since it is an EOOPS and hence has high integration with code. Its rule language is a clean union of C# and predicate logic. Drools aims to have good performance by implementing the RETE algorithm. It has not been updated since 2007.
- **NXBRE (.NET):** NXBRE [NxBRE 2009] is an inactive rule-based inference engine for .NET. Unlike Drools.NET it is well documented and its rule language is XML-based, but it has no tool support either. Objects referenced by the rules must be written in XML, which means the developer cannot make a direct integration between the objects used inside the code and the objects used by the rules. Hence, NXBRE is not an EOOPS, an essential feature when regarding embeddability. It is open-source too, but its matching algorithm is not known.
- **Simple Rule Engine (.NET**): Simple Rule Engine (SRE) [Sierra digital solutions 2005] is another open-source rule-based inference engine written in .NET. It is, like NXBRE, based on xml, but its objects may come from C# code, making it easier to embed. This project claims to be better in performance than NxBRE. SRE is another case of nearly abandoned project with extremely poor documentation and tool support.
- **Drools Expert (Java):** Drools Expert [JBoss 2009] is an object-oriented rule engine which is also open-source. It uses the RETE-OO algorithm - an optimization of Forgy's original algorithm using Object Oriented Programming concepts, intending to have high performance. It has the option of writing rules in XML as well as in the "native" rule language DRL (Drools Resource Language). It also allows the user to create DSLs (Domain Specific Languages) and is fully integrable with Java code. Drools has great development support, with rule debugging and rule authoring tools and an extensive documentation. It is possibly the most complete option available.
- **JEOPS (Java):** JEOPS [Figueira and Ramalho 2000] is an embedded object production system, whose rule language syntax is Java-based. Unlike the other rule engines, in JEOPS, the rule base file is compiled to a Java file to be used in the application. It is also open source and well

documented, but it has no tool support. It implements the classic version of the RETE algorithm.

- **Jess (Java):** Jess [Friedman-Hill 2008] uses an enhanced version of the RETE algorithm to process rules, claiming to have better performance than most engines. It also manipulates and reasons about Java objects, which demonstrates high embeddability. The rules can be expressed in XML-like or Lisp-like languages, both designed for the Jess engine. It has extensive documentation and some additional tools such as IDE integration. Although it is not free, Jess was included in this evaluation since it is available at no cost for academic use.
- **Soar:** Soar [Laird et al. 1987] Soar is a cognitive architecture for applications with intelligent behavior, in general. It has tool support and is extensively documented. Besides, it is platform independent, working as service that can be accessed by any system. However, this broadness makes it necessary to develop a protocol of communication between Soar and the application, which raises integration work significantly.

Regarding documentation and tools, Drools Expert and Jess stand out, while the .NET counterpart of Drools stays in the last position. Most of the verified engines implement the RETE algorithm or enhanced versions of it, making performance comparisons harder. In addition, Drools and JEOPS are the engines that can be easily classified as EOOPS. Although Soar has tool support and documentation, it has some integration issues. Thus, it is not hard to notice that Java inference engines are the ones that better fit the elicited requirements for developing a character-based game with low cost. This is natural, given the existence of a great variety of open-source consolidated inference engines in Java [Stamper 2008]. However, to choose an inference engine it is still necessary to define which game development technology will be used.

3.2 The game development platform

There are many game development platforms available today, but none of them is specifically directed to character-based game development. Hence it is necessary to choose, from the general purpose development options, one that is free, easy to use and integratable with the inference engine. Among the various options, Microsoft's XNA has attracted attention of the independent game developers' community, even though it is not a game development industry standard. This is due to XNA's good learning curve (coming from C#.NET's ease of use), game development support as well as to it being free. In addition to this, the XNA creators club [XNA 2009], the official community of XNA game developers, has over 300 published games and 100,000 users.

XNA is a set of managed code development libraries that intends to make it possible for game developers to be more productive when creating games for Windows and the Xbox 360. It encapsulates low-level technological components involved in coding a game, allowing developers to focus on the content and gaming experience. The XNA Framework is part of XNA Game Studio toolset, which works with Visual C# Express and Directx SDK, all free tools provided by Microsoft [MSDN 2009].

The best-fitting inference engines, though, are not written in .NET. But using the .NET inference engines, which are less adequate, in order to match XNA may compromise the main characteristic of character-based games, which is the credibility of the NPCs' behavior. Soon comes up the need to integrate XNA's pro-game technology and the simulation capabilities of the Java inference engines.

4. The proposed model

When designing a development model proposal that matches character-based games, it is necessary, as pointed out previously, to integrate Java inference engines and XNA game development framework. The latter is responsible for supporting game development and the former for assuring the quality of the character-based game development main requirement: character behavior credibility. In order to achieve this integration, IKVM.NET [Frijters 2008] toolset was used. It can compile Java packages (JARs) to .NET libraries and also allows .NET code to access the Java API, promoting the communication between both sides.

In the proposed approach, hereafter referred to as Cross-platform Reasoning Support Model, the compiler is used to convert the Java inference engine and its related classes (packed in a JAR file) to a .NET library (as described in Figure 1), which can be accessed in the .NET environment. Thus, the code written in C# over the XNA platform can communicate directly with the inference engine. Hence, the developer can benefit from the strong points of both sides, being able to fulfill the requirements of character-based games.

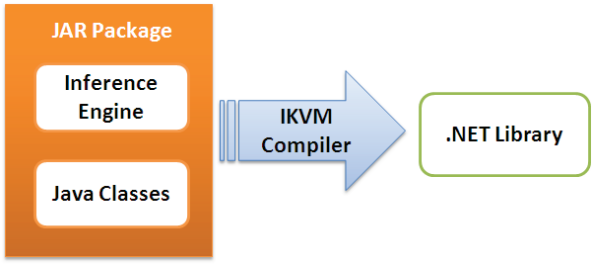

Figure 1: Compiling a Java Engine to .NET

This approach has weak spots, though. It may sacrifice one of the criteria used to evaluate de quality of inference engines: the tool support. This occurs

because the Java engine-related tools only work in the Java environment. By compiling the engine to .NET, all game components stay in .NET, becoming unable to access the Java toolset. And during the development phase it is extremely important to have features such as rule debugging, rule logging, authoring tools and so on.

In order to bypass this problem and be able to use all the engine advantages, a specific model directed to the development phase of the game is also proposed.

4.1 Cross-Platform Solution

For the purpose of using the tools provided by the Java engines to help the development, it is necessary to run parallelly the Java and .NET environments. Therefore, specifically during the game development stage, the already mentioned approach is not appropriate, since it focuses on fully developing in the .NET environment.

The solution to avoid this is a new approach that keeps the .NET and Java components apart and implements communication between them using the Java Remote Method Invocation (RMI) [Sun 2006]. It is a way of communication that makes it possible for two objects in different virtual machines to contact each other, locally or remotely. By doing so, through the use of RMI service provided by IKVM, the components in .NET are able to communicate with the Java engine, and the two environments can work simultaneously and individually. This can be done simply by using a remote engine interface in .NET that will work as a communication protocol. The process is described in Figure 2.

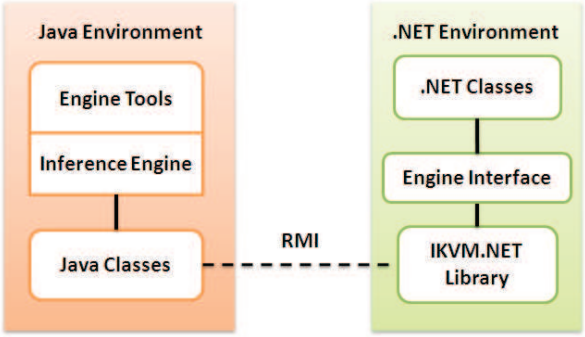

Figure 2: Communication between Java and .NET environments

By using this process, it is possible, for instance, to debug rules, and thus raise the chance of bug resolution during the development phase of the game. However, since communication using RMI may have a negative impact on the game performance, it is recommended to use this model only during the development phase. At the release phase, compiling the engine to .NET maintains the necessary functionality and attenuates the effects of adding the inference engine to the game.

Next section presents Virtual Team, a game prototype where the proposed approach was successfully applied.

5. VTEAM

Figure 3: Main game screen of VTEAM

Virtual Team [VTEAM 2009] (Figure 3), a.k.a. VTEAM, is a serious game prototype created to assist software project managers' training. During the game, the player incorporates a project manager whose goal is to finish a project while trying to deal with personal issues of the development team. The player has to assign tasks, help the team when needed, promote outstanding members, and arrange meetings, among other actions, in order to finish the project successfully.

It focuses on improving the manager's ability to deal with problems related to Human Resources Management. VTEAM's goal is to provide for the manager being trained a greater experience of organizational, methodological, cultural and personal processes that are generally characteristic to software development teams. This mechanism offers to both trainer and trainee a richer experimentation scenario when compared to traditional methods and simulators used to teach those concepts.

VTEAM is also a character-based game, since the interaction between player and virtual characters is its main feature. In this game, the characters are represented by Synthetic Actors - intelligent agents with special features, such as emotions, personality and beliefs [Silva 2009]. As in any character-based game, the NPCs (team members) need to have behavior credibility, especially when this will have great impact on the player learning curve. Hence the need for a strong focus on the character's AI.

RBS were chosen to model the characters' behavior, mainly due to the ease of representing character knowledge using rules. Moreover, RBS usually do not require the developers to be experts on the subject, because declarative language is easy to understand. For the purpose of guiding the game development, the approach proposed in this game was applied.

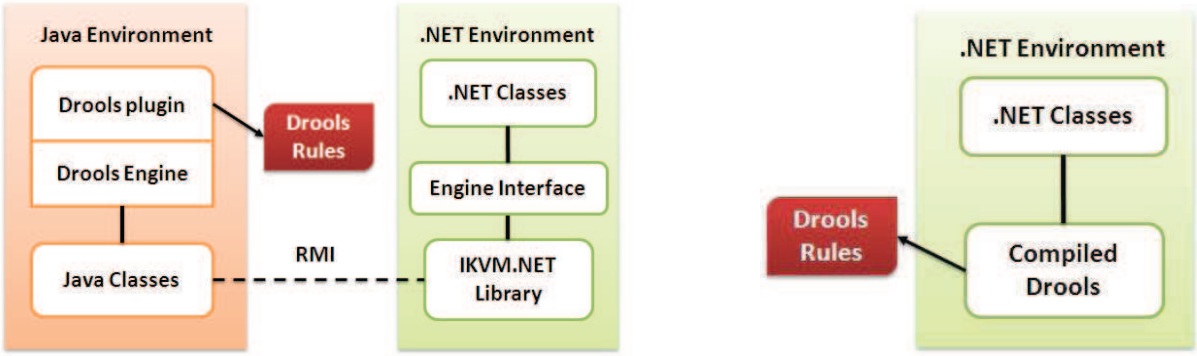

Figure 4: VTEAM's communication with Drools on development phase (left) and release phase (right)

5.1 Cross-platform Reasoning Support Model on VTEAM

In order to apply the proposed approach to VTEAM's prototype, it was necessary to look for an adequate inference engine. At the first version of the game, developed in C++, the Soar engine was used to model the character's behavior. Soar works like a black box, completely separated from the game code, and in order to use it properly components of communication were implemented. Besides that, the Soar rule language is not very readable, since it uses its own concepts regarding operators and automatic state generation.

Thus, on this version of the project, a different engine was chosen, Drools Expert. Drools rule language is a hybrid between Java and declarative programming, making it easy for developers to understand it. Besides that, Drools can be used via an integrated eclipse environment, which adds important features such as on-line working memory viewing and rule debugging - the latter is important, since VTEAM deals with a large number of variables and has an intense need for game balance. Figure 4 illustrates the implementation of the proposed approach on VTEAM.

VTEAM's prototype was developed on the .NET framework using the Visual C# Express IDE, due to the game development support provided by XNA. In order to integrate XNA technology and the Drools engine, the proposed process was used. During the development phase, Drools is executed from eclipse, while the game code is in Visual C# Express. For deploying, Drools is compiled into a .DLL, used by the C# code. So far, it is possible to say that the application of this article's proposal to VTEAM was successful.

One of the main concerns when working with an RBS is its impact on game performance. In order to investigate this issue, the prototype FPS rate was measured using the proposed approach. There were no changes in FPS (Frames per second) rate when using the release phase model. And, as expected, the use of RMI caused the FPS to fall during the development phase. But to bypass this problem, the game and the inference engine interface were implemented in two different threads, running in parallel.

6. Conclusion

As the demand for high quality and innovative games increases, so does the game development cost. Due to this state of affairs, independent game developers look for low-cost ways to produce high quality games. In the case of character-based games, this quality is deeply associated with the characters realism. Not only visual details, but also, and more importantly, believable behavior transmitted by the characters, provided by Artificial Intelligence. Thus, it is necessary to find ways of including this extra complexity avoiding a negative impact on financial cost.

The proposed approach allows the creation of a character-based game, based on a powerful AI mechanism: Rule-based systems. This is done using only free tools in a cross-platform way, bypassing the need for additional cost on development. However, this approach does have some drawbacks. In order to implement a cross-platform approach, it is necessary for the development team to know both languages involved, a demand that might be a problem. In this research's case, Java and C# are very similar languages [MSDN 2009], which attenuates this problem

As future work, it is important to test the study with a complete game and a large amount of rules implemented, so that the real impact on performance might be verified. Besides, it is also important to run tests with various magnitudes of games, which have different requirements on AI. Character-based games that call for interaction between virtual characters, like The Sims, for instance, might have different needs compared to those where the character only interacts with the player and the scenario, like Black & White and Nintendogs [Nintendo 2005].

This article study might also be used on developing games that have a rule module, but not necessarily associated to intelligent agents, such as The Distributor Game [van Houten et al 2005], which uses rule-based

scenarios. Besides, the model that used RMI, might also allow that one Java inference engine works as a server for many XNA game clients. These approaches have yet to be tested, though.

Acknowledgements

First of all the authors would like to thank the VTeam development team and project coordinators for all the effort and support. Also thanks to the project partners and sponsors: FINEP, Jynx playware, Valença & Associados, Qualiti, CNPq and CIn-UFPE.

References

MORRIS, C., 2009. *The Next Generation of Gaming Consoles* [online] CNBC. Available from: http://www.cnbc.com/id/31331241 [Accessed in 24 July 2009]

MAXIS, 2000. *The Sims.* [computer game]. Redwood City, USA: Electronic Arts.

LIONHEAD, 2001. *Black & White*. [computer game]. Redwood City, USA: Electronic Arts.

NAMCO, 1980. *Pac-man*. [computer game]. Tokyo, Japan: Namco.

MILLINGTON, I. 2006. Artificial intelligence for games. San Francisco: Morgan Kaufmann.

BOURG, D. M. AND SEEMANN G., 2004. AI for game Developers. Sebastopol, USA: O'Reilly.

ACTIVISION, 1999. *Civilization: Call to Power* [computer game] Santa Monica, USA: Activision.

RED STORM ENTERTAINMENT, 1998. *Tom Clancy's Rainbow Six* [computer game] Morrisville, USA: Red Storm Entertainment.

CAVAZZA, M., 2000. AI in computer games: Survey and perspective. *Virtual Reality* 5 (4), 223-235.

PACHET, F., 1995. On the embeddability of production rules in object-oriented languages. *Journal of Object-Oriented Programming*, 8 (4), 19-24.

FORGY, C. L., 1982. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19 (1), 17-37.

MICROSOFT, 2009. *Microsoft Biztalk Server 2009: Business Rule Framework* [online]. Available from: http://www.microsoft.com/biztalk/en/us/business-rule-framework.aspx/ [Accessed in 24 July 2009].

ILOG, 2009. *ILOG rules for .NET* [online]. Available from: http://www.ilog.com/products/rulesnet/ [Accessed in 24 July 2009].

BINNET CORPORATION, 2005. *Jinni: Java INference Engine and Networked Interactor* [online]. Available from: http://www.binnetcorp.com/Jinni/ [Accessed in 24 July 2009].

DROOLS.NET, 2006. *Drools.NET home* [online]. Available from: http://droolsdotnet.codehaus.org/ [Accessed in 24 July 2009].

NXBRE, 2009. *NxBRE home* [online]. Available from: http://nxbre.org/ [Accessed in 24 July 2009].

SIERRA DIGITAL SOLUTIONS, 2005. *Simple Rule Engine* [online]. Available from: http://sourceforge.net/projects/sdsre/ [Accessed in 24 July 2009].

JBOSS, 2009. *Drools: Business Logic Integration Platform* [online]. Available from: http://www.jboss.org/drools [Accessed in 24 July 2009].

FIGUEIRA FILHO, C. AND RAMALHO, G., 2000. JEOPS - The Java Embedded Object Production System. In: Monard M. and Sichman J., eds. *Advances in Artificial Intelligence*, 1952. London: Springer-Verlag, 52-61.

FRIEDMAN-HILL, E., 2008. *Jess, the Rule Engine for the JavaTM Platform* [online]. Available from: http://www.jessrules.com/ [Accessed in 24 July 2009].

CARNIEL, M., 2006. *JRuleEngine – OpenSource Java Engine* [online]. Available from: http://jruleengine.sourceforge.net/ [Accessed in 24 July 2009].

STAMPER, J., 29 July 2008. *The 10 Best Open Source Rules Engines. Jason Stamper's Blog* [online]. Available from: http://www.businessreviewonline.com/os/archives/2008/07/10_best_open_so.html [Accessed in 24 July 2009].

XNA, 2009. *XNA Creators Club* [online]. Available from: http://creators.xna.com/en-US/ [Accessed 23 June 2009].

MSDN, 2009. *XNA Developers Center* [online]. Available from: http://msdn.microsoft.com/en-us/xna/default.aspx [Accessed 23 June 2009].

SUN, 2006. *Java Remote Method Invocation* [online]. Available from: http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp. [Accessed 24 July 2009]

VTEAM, 2009. *Projeto VTEAM* [online]. Available from: http://vteam.cin.ufpe.br. [Accessed 24 July 2009].

SILVA, D. R., 2009 *Atores Sintéticos em Jogos Sérios: Uma abordagem baseada em Psicologia Organizacional*. PhD Thesis. Universidade Federal de Pernambuco.

MSDN, 2009. *The C# Programming Language for Java Developers* [online]. Available from: http://msdn.microsoft.com/en-us/library/ms228602.aspx [Accessed 24 July 2009]

NINTENDO, 2005. *Nintendogs*. [computer game]. Kyoto, Japan: Nintendo.

FRIJTERS, J.., 2008. *IKVM.NET* [online]. Available from: http://www.ikvm.net/ [Accessed 24 July 2009]

LAIRD, J., E., NEWELL, A. AND ROSENBLOOM, P.S., 1987. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1), 1-64.

VAN HOUTEN, S.P.A., VERBRAECK, A., BOYSON, S., AND CORSI, T., 2005. Training for Today's Supply Chains: An Introduction to the Distributor Game. *In: Proceedings of the 2005 Winter Simulation Conference,* 04-07 December 2005 Orlando. 2338-2345.