Pogamut Toolkit

Jakub Gemrot
Charles University, Faculty of Mathematics and Physics
Ke Karlovu 3
Prague 2, 121 16, Czech Republic
jakub.gemrot@gmail.com

Michal Bída
Charles University, Faculty of Mathematics and Physics
Ke Karlovu 3
Prague 2, 121 16, Czech Republic
michal.bida@gmail.com

Cyril Brom
Charles University, Faculty of Mathematics and Physics
Ke Karlovu 3
Prague 2, 121 16, Czech Republic
brom@ksvi.mff.cuni.cz

ABSTRACT
All experiments using intelligent virtual agents, sooner or later, ask for a specific virtual environment that would fit their setup. Seeking such environment is a daunting task accompanied with the need for an appropriate agent adapter that provides infrastructure for mediation of virtual body senses and actions thereby enabling remote high-level agent control. This demo presents Pogamut toolkit, which provides out-of-box programmer tools for creating virtual agents for Unreal Tournament 2004, Unreal Development Kit and Defcon virtual environment. Pogamut's virtual world abstraction is compatible with many agent oriented languages and architectures including Jadex, GOAL, POSH, Soar or ACT-R, which makes it highly suitable for research on intelligent virtual agents.

Categories and Subject Descriptors
D.2.13 [**Reusable Software**]: Reusable libraries

General Terms
Design, Experimentation.

Keywords
IVA toolkit, Virtual environments, Action-selection

1. INTRODUCTION
The development of intelligent virtual agents (IVA) is still far from being easy as every IVA application calls for complex chain of tools and libraries that must work together to enable quick and efficient IVA production. IVA production typically comprises several cycles, during which researchers:

(a) design,
(b) implement, run, observe & debug,
(c) test & validate their IVAs.

Technically, IVA applications can be conceived as consisting of three parts (see Picture 1):

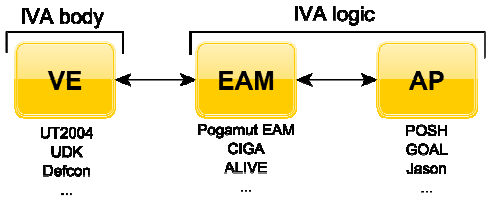
(1) a virtual environment (VE),
(2) an environment-agent middleware (EAM),
(3) an agent platform (AP).

Furthermore, as every research have to implement & debug (Point (b)) and test & validate the application (Point (c)), a researcher needs:

(4) implementation tools,
(5) debugging tools,
(6) testing & validation tools.

As there is no mature standard yet that would cover the whole IVA development process or provide research methodology guidelines and technology interface standards (contrary to "classical" agents, cf. e.g. FIPA), every IVA application setup requires a proprietary solution that combines Parts (1) – (6). Here, we present Pogamut toolkit, a result of 5 years of work, which aims at providing complete solutions for building IVAs for various virtual environments. Pogamut toolkit currently supports development of IVAs for (i) Unreal Tournament 2004 (UT2004), (ii) Unreal Development Kit (UDK) and (iii) Defcon. Unreal Tournament 3 (UT3) is a work-in-progress. The toolkit complements similar attempts, such as [1] capitalizing on BML.

Picture 1. A typical IVA application



2. FEATURES OF IVA TOOLKITS
Instead of listing Pogamut features, it is better to review IVA production cycle (Points (a) – (c)) with respect to IVA application Parts (1) – (3). That will provide the list of features that every IVA toolkit should possess.

2.1 Designing IVAs (Point (a))
Process of designing an IVA is typically sensitive to the selection of Parts (1) – (3). A researcher has to understand capabilities, limitations and options of every part involved. She must understand a VE (1) to be able to create its particular instance suitable for the application; she has to work with an EAM (2) encoding agents' reflexes and complex sensory and motor primitives; finally, she will work with an AP itself (3), which will accommodate agents' plans and strategic decision making.

The support from the IVA toolkit here is to have getting-started tutorials, be well documented and provide a lot of executable example agents that exemplify various features provided by Parts (1) – (3).

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

2.2 Implementing & Debugging (Point (b))

Once an IVA application is designed and the tool chain is understood, the implementation can take place. This phase itself will contain a lot of iterations of Point (b) (see Table 1).

Unfortunately, all steps of Point (b) will happen in each of VE, EAM and AP so the toolkit must provide (ideally integrated) (4) & (5) to help the researcher along the way. The list of desired features is presented in Table 1.

2.3 Testing & Validating IVAs (Point (c))

Once an IVA is implemented, it needs to be run through series of tests that provide data for answering experiment hypotheses, e.g., for comparison to other existing IVAs fulfilling the same goal. Usually, it means to run the IVA multiple times (e.g., 20x or 100x) to gain statistical validity of the obtained data.

The IVA toolkit has two roles in this process (Part (6)). First, it should provide means for gathering such data, e.g., stubs for agent observers that can collect data of agent actions, reasoning, decision making and a VE itself. Second, it should provide tools (GUIs, libraries, scripts) for automatic testing, so that the researcher does not need to run every test manually or create such tools.

Table 1. The list of IVA platform features that ease implementing & debugging of IVAs

	(1) VE	(2) EAM	(3) AP
Implement	VE editor	IDE for coding reflexes and complex sensory and motor primitives	IDE for creating agent plans
Run	Means for quick (re)starting of the whole tool chain (startup scripts or GUI).		
Observe and debug	VE visualizer	Interactive coding, sync. breakpoints with VE, logs.	Interactive coding, sync. breakpoints with EAM, logs.

2.4 Technical dependencies

Unfortunately, there are technical dependencies between a VE, an EAM and an AP. Thus every complete tool chain will contain a lot of "glue" code that adapts VE-EAM and EAM-AP. As there are no mature standards how VEs, EAMs and APs should look like, no one can expect (for instance) that existing tools for AP will provide much insight into interoperability between EAM and AP or even VE and AP. For example, an automated IVA testing tool that operates over UT2004-Pogamut-SPOSH (as part of (6)) will not work for Defcon-Pogamut-Jason setup as it will contain much of UT2004-Pogamut-SPOSH specific code.

This is not surprising but leads to another observation that every IVA toolkit should state which tools it provides with respect to concrete VE-EAM-AP chosen.

3. FEATURES OF POGAMUT

Tables 2 and 3 provide an overview of existing and implemented tool chains for creating IVAs for UT2004, UDK and Defcon environment by the Pogamut toolkit.

Table 2. Bindings that Pogamut as EAM provides.

VE / EA	Java	POSH	Jason	ACT-R
UT2004	Yes	Yes	No	Yes
UDK	Yes	Yes	No	No
Defcon	Yes	No	Yes	No

4. USAGE

In this paper we have presented a list of general features that are (has to be) common to every IVA toolkit aiming to support development of IVA applications. The crucial point is that Pogamut supports these features with respect to three different virtual environments. Furthermore, Pogamut already proved its applicability by being used for international IVA competition, research and education.

5. ACKNOWLEDGMENTS

This research was partially supported by project P103/10/1287 (GACR), by student grants GA UK No. 0449/2010/A-INF/MFF 655012/2012/A-INF/MFF, and by SVV project number 263 314.

6. REFERENCES

[1] Thiebaux, M., Marshall, A., Marsella, S., Kallmann, M. SmartBody: Behavior Realization for Embodied Conversational Agents. In: Proc. of Autonomous Agents and Multi-Agent Systems (2008).

Table 3. Existing tutorials and features of Pogamut toolkit for respective VE/AP combinations.

	Designing				Implementing & Debugging	Testing & Validating
VE / AP	Installer	Getting started doc.	Tutorials	Commented examples	IDEs / Tools	IDEs / Tools
UT2004+Java	Yes	Yes	Yes	Yes	NetBeans IDE, Debug GUI	Experiment runner lib.
UT2004+POSH	Yes	Yes	Yes	Yes	NetBeans IDE with POSH Editor and POSH Debugger	Experiment runner lib.
UT2004+ACT-R	Yes	No	Yes	Yes	NetBeans IDE	X
UDK+Java	Yes	Yes	Partially	Yes	NetBeans IDE	X
UDK+POSH	No	No	Partially	No	NetBeans IDE with POSH Editor and POSH Debugger	X
Defcon+Java	No	No	Yes	Yes	NetBeans IDE for coding, Auto deploy & run Ant scripts	X
Defcon+Jason	No	No	Yes	Yes	Auto deploy & run	X