Developing Interactive Embodied Characters Using the Thalamus Framework: A Collaborative Approach

Tiago Ribeiro[1], Eugenio di Tullio[1], Lee J. Corrigan[2], Aidan Jones[2], Fotios Papadopoulos[2], Ruth Aylett[3], Ginevra Castellano[2], and Ana Paiva[1]

[1] INESC-ID & Instituto Superior Técnico, Universidade de Lisboa, Portugal
tiago.ribeiro@gaips.inesc-id.pt
[2] University of Birmingham, United Kingdom
[3] School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK

Abstract. We address the situation of developing interactive scenarios featuring embodied characters that interact with users through various types of media easily presents as a challenge. Some of the problems that developers face are on collaborating while developing remotely, integrating all the independently developed components, and incrementally developing a system in such way that the developed components can be used since their incorporation, throughout the intermediate phases of development, and on to the final system. We describe how the Thalamus framework addresses these issues, and how it is being used on a large project that targets developing this type of scenarios. A case study is presented, illustrating actual development of such scenario which was then used for a Wizard-of-Oz study.

Keywords: Embodied characters, component integration framework, social robotics, BML.

1 Introduction

As scientists, we envision the creation of autonomous artificial characters that are able to interact with humans in a natural way. It seems that interactive technologies are evolving enough to provide this; however we still strive to figure out how to integrate artificial characters, virtual environments, and our physical world in a seamless interaction. Current applications that feature interactive embodied agents tend to integrate several technologies together, into larger holistic systems.

One particularly interesting field of application that features these kind of holistic systems are the ones involving socially interactive robots [3]. Our work is part of the EU FP7 EMOTE Project[1]. The project aims at developing empathic robotic tutors that can interact with school children through multimedia applications in order to improve learning.

[1] http://www.emote-project.eu/

T. Bickmore et al. (Eds.): IVA 2014, LNAI 8637, pp. 364–373, 2014.

In this field of human-robot interaction, users interact with an agent, embodied as a robot. This robot is normally regarded by the user as the actual artificially intelligent being. However, it frequently turns out that technically, the robot acts solely as an embodiment and does not actually contain what is regarded as the artificial intelligence. This intelligence will most likely be running on separate computers which communicate with and control the robot [4]. It is also becoming common to have external third-party components extending the interaction environment, such as capture devices (e.g. Microsoft© Kinect©[2]) or touch-tables. Current mobile devices can also be used to extend the interaction, providing the physical robot with a ubiquitous virtual form, in a process called migration [9].

The requirements for this type of tutor and interaction environment thus aims us at exploring component-based embodied agents in which components can be reused in different scenarios.

This paper describes Thalamus, our component integration framework, developed in order to support the development of interactive agents that can seamlessly integrate the agent's logic with components for both various embodiments (virtual or robotic) and mixed environments (virtual and physical). We describe how it is being used in EMOTE for collaboratively developing the overall system that includes a robotic character (tutor), a touch-based video-game, perceptual components, and high-level behaviour control. We present a case study in which the tutor is controlled in a Wizard-of-Oz setting, and how the same components can still be used when we replace the wizard with an autonomous agent.

2 Related Work

Several authors have faced the kind of problems we refer to, and as such, have proposed other architectures before. Schröeder developed the SEMAINE API, which was used in the EU FP7 Semaine Project[3]. This is a component integration framework, based on the principles of asynchronous messaging middleware. Its architecture, however, has a pipeline message flow, meaning that it follows in the traditional sense-think-act loop of interactive agents. The author points out two key requirements for a framework of this kind: Infrastructure, meaning that components must be able to run on different programming languages and operating systems; and Communication, meaning that components must follow suitable representation formats, which should be standards where possible[8].

CMION was developed in the context of the EU FP7 LIREC[4]. It is a mind-body framework for integrating sensors and actuators through various degrees of abstraction. It was designed especially for allowing agent migration (transferring the agent's identify to a different embodiment). As such, it abstractly encapsulates functionalities of an embodiment into what they call competencies. These competencies share information through a blackboard component. By defining

[2] http://www.microsoft.com/en-us/kinectforwindows/
[3] http://www.semaine-project.eu/
[4] http://lirec.eu/

an embodiment as a set of competencies, agents can then migrate to other embodiments, as long as those implement the same type of competencies[2].

Thalamus was first also developed for the same LIREC project[6]. I was initially developed as an embodiment-independent BML scheduler which was used to run BML on robots. It also provided interaction between high-level perception structures (PML) and the BML plans, so that an agent's behaviour could be planned to interact with asynchronous events from the environment.

More recently, the same authors have presented the Censys Model[7]. Censys serves as a theoretical-to-technical foundation on how developers can design and structure agents following some concepts taken from philosophy and neuroscience, in order to break the sense-think-act loop of traditional agents. What Censys proposes is that there is no need to explicitly define a Mind or a Body in an agent. The Mind process can be built out of several interacting processes, which exchange information. The Body processes would be all the processes that are capable of turning the higher-level behaviour instructions onto low-level body actions, and the low level perceptual data into higher-level representations that can be understood by other components. The behaviour realization components do not event have to be the same as the perception components. This allows to more easily reuse components in different applications, by replacing only specific parts of the system. In a Censys architecture, the flow of communication is asynchronous and does not follow a predefined path (pipeline). This allows several modules to perform lower-level autonomous behaviour, while other modules simultaneously process and provide higher-level information.

ROS - Robot Operating System is a popular middleware for robotics that provides a common communication layer to enable different types of sensors, motors and other components to exchange data[5]. ROS is module-based, meaning that a ROS-based robot actually runs several different modules, being each one of them responsible for controlling one or more components of the robot. They communicate based on a message oriented middleware (MOM). This is accomplished through a publish-subscribe pattern, in which each module specifies the type of messages it wants to receive (subscription), so that each time another module produces that message (publication), the subscribed modules receive it.

3 Thalamus as a Modular Framework for Interactive Embodied Agents

Thalamus was initially developed as a cross-media body interface for artificial embodied characters[6]. It was based on the SAIBA framework [1], and developed mostly as a BML scheduler, with the additional capability of supporting abstract perceptions which could interact with BML actions, in order to allow for continuous interaction with a character. By providing only the scheduling and not the realization functionality of BML, it can be used with different embodiments, both virtual and robotic.

3.1 Motivation

We have now adapted Thalamus to follow on the Censys model[7], thus turning it into a more general component-integration framework. Traditionally, the body of an embodied agent framework contains all the physical interfaces of the character, both in terms of actuation and perception. However, we consider that the interface with the environment can be composed of several components.

An example of this would be an interactive scenario featuring an expressive robot and a Kinect© camera for perceiving the user. An interactive system should be modular enough to allow replacing the robot by another one for the expressive function, while keeping the Kinect© for the perceptual function.

By building on Censys, we do not designate any specific component as being either the *body* or the *mind*. This also makes it easier to have a character that interacts both with the physical environment and with a virtual one.

Taking as example an interactive setting with the robot, the Kinect©, and a touch surface/screen, these three components all provide an interface with the user and the physical environment. However, the touch screen will most likely be running another application which provides a virtual environment. On such a setting, we do not consider it appropriate to strictly define the body as a specific component of our system.

The main new feature we have introduced into Thalamus is the MOM mechanism, which is designed to integrate with the scheduler. This integration between scheduler and MOM allows to have the asynchronous and abstract sides of communication given by the MOM, while still supporting synchronously distributed behaviours that run in a BML-like manner. However, the Thalamus scheduler is more abstract than BML, which allows it to synchronize actions and events that do not only originate from BML-based behaviour.

3.2 Architecture

Figure 1 shows an overview of how Thalamus is currently structured. The Thalamus Master (Master) is the main node which centralizes all communication that runs between different Thalamus Modules. Both the Master and each of the Modules can run either in the same or in different computers, as the communication is established over some type of network protocol. As described on the Censys model, each Module can subscribe and publish both Actions and Perceptions.

In Thalamus, both of these are treated as Events. We distinguished them mostly for easier design, development and comprehension of the agents. The figure also shows the MOM-based manager as the central part of the framework, and how it is tightly linked with the scheduler.

The Master maintains a proxy to each of the Modules that are connected, in order to manage the communication between these two parts. Each Module actually communicates with it's specific Module Proxy, both to subscribe, announce, publish and receive events. Every time the configuration of the connected modules changes (i.e, some module connects or disconnects from the system), all the

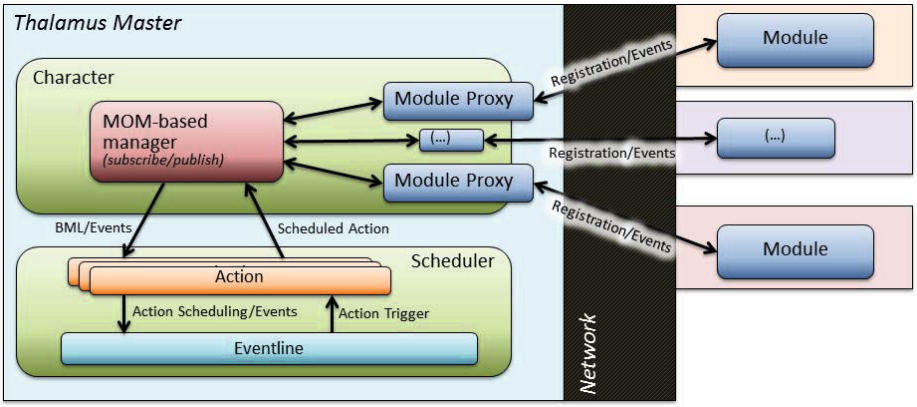

Fig. 1. The current Thalamus Framework architecture

available Event definitions are broadcast to all Modules. This allows Modules to know what is currently available in the system, and if necessary, adapt the way they behave.

Models that exchange the same type of Events must all follow a pre-defined Event structure in order to consider the same parameters for the transmitted Event. These are specified outside of any Module, in shared libraries. Each library can contain several groups of Events, which we called Interfaces. These Interfaces are defined externally by the agent developers, and should include all the Events that are necessary for implementing a specific kind of functionality.

Figure 2 shows an example in which several Modules subscribe and publish to Events which are defined by Interfaces in separate libraries. By subscribing to a specific Interface, the Module subscribes to all Events defined in it. Each Module can subscribe or publish to as many Interfaces as the agent developers consider the Module to be responsible for.

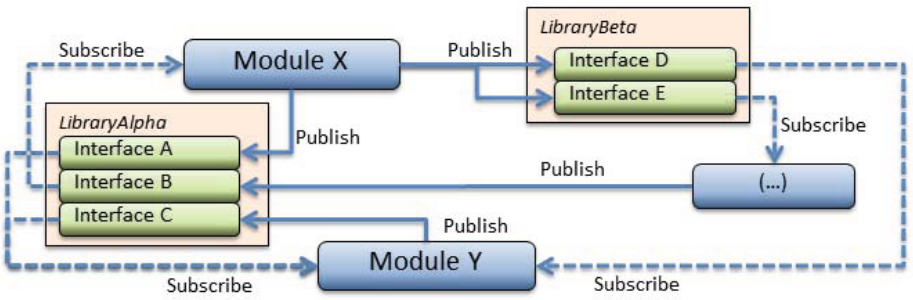

Fig. 2. Example scheme of how different modules publish and subscribe to events

3.3 Implementation and Development Workflow

Thalamus is currently implemented in C#/dotnet. It can run either as a library, or as a standalone application, with its own GUI. The Master node generally runs in the standalone form. All the other Modules run as separate applications by implementing the ThalamusClient class. In order to publish events, each Module also contains an instance of a ThalamusPublisher class. We are currently using the XML-RPC.Net[5] library for remote message invocation between the Modules and the Master.

The framework is open-source, and is currently available through a Mercurial repository in Sourceforge[6]. There is a README.txt file which includes the basic workflow with Thalamus, and a Documentation folder with instructions on how to start writing modules.

Thalamus also includes a simple GUI which provides features like:

- Creating a Character, which will automatically receive connections from any Modules;
- Viewing the Modules that are connected to such Character;
- Manually triggering Actions and Perceptions, for debugging and testing purposes;
- Event viewer with filters.

4 Case Study: The Modular Wizard-of-Oz

Thalamus is currently being use as the backbone integration platform in the European FP7 EMOTE project. Several partners are using it collaboratively to develop different Modules that communicate with each other in order to achieve the project's goals. The project aims at developing empathic robotic tutors that can interact with school children through multimedia applications in order to improve learning.

4.1 Collaborative Platform for Large-Scale Agent Development

Several things in Thalamus were developed with collaborative development in mind. We highlight the method of defining Event Interfaces as separate libraries that can be shared by different Modules. The Thalamus Master has no knowledge of these Interfaces and Events. They are all abstracted when sent to the Master, so that the Master relies only on the type of Event that is being transmitted, and the rules that it has for who to broadcast it to. This has allowed developers to work independently on different Modules that communicate with each other by simply sharing a library which contains the necessary Events.

By having anticipated that a large scale interactive scenario would include several logical, virtual and physical components, we have made the integration

[5] `http://xml-rpc.net/`
[6] `http://sourceforge.net/projects/thalamus/`

process as easy and flexible as possible. Besides the shared Interfaces, all Modules and the Master have a network communication layer that allows them to find each other and connect automatically across a local network, and maintain such connection even when one Module, or even the Master fails for some reason (normally during development and debugging of new features). It would be extremely tedious to require setting up connections manually, and having to restart all the modules every time something failed.

4.2 Interactive Scenarios

In EMOTE, the tutor is being developed for two different interactive scenarios, both using a robotic embodiment, and a large touch-table. The touch-table runs a multimedia interactive application (like a video-game). The user interacts with the system by using the game application. The system also interacts back through the robotic embodiment, which provides a character with expressive behaviour.

In a large-scale project, it is common to run tests with initial versions of the system in order to collect data about how users interact with all the components that are being developed. In our case, before implementing the final autonomous robotic tutor, we are going through several mock-up and wizard-of-oz (WoZ) experiments. This section briefly describes how Thalamus was used to integrate the components used in a recent WoZ experiment.

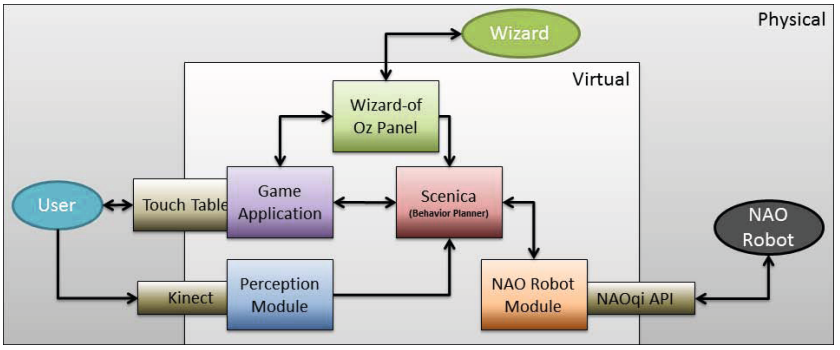

Fig. 3. Structure of the Wizard-of-Oz setting. The physical and virtual components are distinguished.

Figure 3 shows how the current system is structured for the WoZ experiment. Our physical components are a NAO robot[7], a multi-touch table (MTT), and a Microsoft© Kinect©.

The MTT provides both a virtual environment (Game Application) that is shared by the agent and the user, and is also used for input from the user. The Kinect captures the user. Currently it is used only for head-tracking.

[7] `http://www.aldebaran.com/en`

The NAO Robot provides an embodiment that exhibits expressive behaviour towards the user. Such expressive behavior is generated and managed by a behavior planner module which we call Scenica. This module is constantly updated with information from the Perception module (which interfaces with the Kinect). It also receives coordinate information from the Game Application, in order to be able to instruct the robot to gaze, point or wave towards specific points on the screen. Scenica also provides some semi-autonomous behaviour. It manages gazing behaviour so that the Wizard does not have to deal with all that.

The Wizard uses the WoZ panel to control the flow of the game, to parametrize some of Scenica's semi-autonomous behaviour, and to manually select high-level FML utterances. These utterances are dialogue acts which were previously written and tagged both with non-verbal and game instructions. The FML is broken down in Scenica into BML actions and game-actions. The actions are then sent through Thalamus to be scheduled and/or routed to other modules (NAO Robot Module and Game Application).

As stated earlier, all the modules communicate over a network. That allows for easy deployment of different Modules across different machines. The NAO Robot Module runs on the actual robot, using Mono[8]. The Game Application runs on the MTT, while the remaining modules (Scenica, Perception and WoZ) run on another computer. As we are implementing more intensive algorithms for the Perception module, in the future we may decide to deploy this to a dedicated machine with specific hardware.

4.3 Feedback from the Developers

EMOTE's technical team has been collaboratively developing the described system. All of them have experience and degrees related with Computer Science, and are either pursuing a PhD degree, or working as post-doc in the project. They were questioned regarding the strengths and weaknesses of Thalamus. Their qualitative feedback is reported in the following paragraphs.

A. Strengths

Development

- Modules can easily be added, removed and reused;
- Easy to debug with the event viewer;
- Simple concept of events and actions;
- Underlying mechanics are abstracted. The modules' internal logic are separated from the communication level;
- Easy to specify what information a Module publishes and easy to discover what you can subscribe to;
- Made it easy to work with NAO robot by using BML;
- Distributed computing allows distributing resources;
- Concentration on developing primary code without actually knowing which other modules the interface would need to communicate with;
- Run on different OS and easy to interop with different languages.

[8] `http://www.mono-project.com/Main_Page`

- Each Module synchronizes its internal clock with the Master, in order to provide logging with near-matching time-stamps;

Collaboration

- Collaboration with other developers enhanced thanks to the independence of each Module;
- Each developer is abstracted away from the concerns of others;
- Development is easily separated, implementation within modules can be changed simply and easily without affecting other modules;
- Each developer needed only to agree on the specification of their requirements for the Events' format (name and parameters);
- Development was not drastically affected by partners changing requirements on a near daily basis;
- After specifying the interfaces, each developer works on its own module, and once ready, things fit easily.

Networking

- A module can crash and then reconnect to Thalamus without causing any issues to other modules;
- No need for set-ups e.g. ip-addresses or names;

B. Weaknesses

Development

- Some limitations on the type of data that can be sent or received from the clients;
- Complex classes or enumerators are not well managed (even though they can be used);
- Sometimes Thalamus Standalone crashes;
- All modules should be running the same version of Thalamus (from sourceforge) otherwise they can't communicate;

Networking

- The discovery of the Thalamus Master can be improved.
- Impossible to select which network adapter to use;
- On some versions of Windows©, requires removing UAC[9] and firewall protection to work;
- Network failure sometimes caused some repeated messages;

5 Conclusions and Future Work

We have described how the Thalamus framework is being used in a large project in which different developers working remotely are able to collaborate on building an interactive scenario featuring a robotic character. The developers have pointed out some of the main benefits and handicaps collected during their experience of working together. We believe that on projects integrating so many technologies, there should be focus on how the backbone tools and frameworks are adequate for the needs of the developers. International projects often require development across different countries, habits, and time zones. We also address the re-usability

[9] User Access Control.

of components in different scenarios as a benefit of planning ahead with an easily extensible and modular framework. The kind of abstraction and high-level integration we provide with Thalamus is also directed towards more recent applications, which often integrate virtual and physical environments, with virtual and physical characters.

There are, of course, points we still consider to be missing. One of them is a mechanism for managing conflicts, in the case of several Modules publishing or subscribing to the same type of Messages. This will highly depend on what is the purpose of the agent, which Modules are being used, and how each of them works. As such, we intend to provide Thalamus only with mechanisms for establishing rules, while the actual rules should be established by the developers.

There are still some network issues to be solved. In the future we may want to replace the XML-RPC-based communications layer with a more stable and efficient implementation.

Acknowledgments. This work was partially supported by the European Commission (EC) and was funded by the EU FP7 ICT-317923 project EMOTE and partially supported by national funds through FCT - Fundação para a Ciência e a Tecnologia, under the project PEst-OE/EEI/LA0021/2013.

References

1. Kopp, S., Krenn, B., Marsella, S.C., Marshall, A.N., Pelachaud, C., Pirker, H., Thórisson, K.R., Vilhjálmsson, H.H.: Towards a common framework for multimodal generation: The behavior markup language. In: Gratch, J., Young, M., Aylett, R.S., Ballin, D., Olivier, P. (eds.) IVA 2006. LNCS (LNAI), vol. 4133, pp. 205–217. Springer, Heidelberg (2006)
2. Kriegel, M., Aylett, R., Cuba, P., Vala, M., Paiva, A.: Robots Meet IVAs: A Mind-Body Interface for Migrating Artificial Intelligent Agents. In: Vilhjálmsson, H.H., Kopp, S., Marsella, S., Thórisson, K.R. (eds.) IVA 2011. LNCS, vol. 6895, pp. 282–295. Springer, Heidelberg (2011)
3. Leite, I., Martinho, C., Paiva, A.: Social Robots for Long-Term Interaction: A Survey. International Journal of Social Robotics 5(2), 291–308 (2013)
4. Pereira, A., Prada, R., Paiva, A.: Socially present board game opponents. In: Nijholt, A., Romão, T., Reidsma, D. (eds.) ACE 2012. LNCS, vol. 7624, pp. 101–116. Springer, Heidelberg (2012)
5. Quigley, M., Gerkey, B.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software, vol. 3(3.2) (2009)
6. Ribeiro, T., Vala, M., Paiva, A.: Thalamus: Closing the mind-body loop in interactive embodied characters. In: Nakano, Y., Neff, M., Paiva, A., Walker, M. (eds.) IVA 2012. LNCS, vol. 7502, pp. 189–195. Springer, Heidelberg (2012)
7. Ribeiro, T., Vala, M., Paiva, A.: Censys: A Model for Distributed Embodied Cognition. In: Aylett, R., Krenn, B., Pelachaud, C., Shimodaira, H. (eds.) IVA 2013. LNCS, vol. 8108, pp. 58–67. Springer, Heidelberg (2013)
8. Schröder, M.: The SEMAINE API: A component integration framework for a naturally interacting and emotionally competent Embodied Conversational Agent. Ph.D. thesis (2012)
9. Segura, E.M., Cramer, H., Gomes, P.F., Paiva, A.: Revive! Reactions to Migration Between Different Embodiments When Playing With Robotic Pets Categories and Subject Descriptors, pp. 88–97 (2012)