The SERA ecosystem: Socially Expressive Robotics Architecture

for autonomous human-robot interaction

Tiago Ribeiro[1] **André Pereira**[2] **Eugenio Di Tullio**[1] **Ana Paiva**[1]

[1]INESC-ID & Instituto Superior Técnico,
Universidade de Lisboa, Portugal
tiago.ribeiro@gaips.inesc-id.pt

[2]Yale University
New Haven, CT, USA

Abstract

Based on the development of several different HRI scenarios using different robots, we have been establishing the SERA ecosystem. SERA is composed of both a model and tools for integrating an AI agent with a robotic embodiment, in human-robot interaction scenarios. We present the model, and several of the reusable tools that were developed, namely Thalamus, Skene and Nutty Tracks. Finally we exemplify how such tools and model have been used and integrated in five different HRI scenarios using the NAO, Keepon and EMYS robots.

Introduction

Human-robot interaction (HRI) systems are starting to spread as a new form of human-computer interaction. In these systems, the concept of ”computer” can expand into several integrated devices like a robot, touch and mobile devices, and perceptual devices like cameras or microphones. While existent in the academia for some time now, during the most recent years we've started to see hints that these systems are being launched into the real world and will some day become an established industry. In order for that to be possible, and to keep on par with the state of the art technology created at universities and research facilities, the HRI community must get together and establish standards, common platforms and reusable tools that can encompass all the different needs of a system that is as heterogeneous as the HRI ones.

On this paper we describe the SERA ecosystem: a model and tools for integrating an AI agent with a robotic embodiment, in an HRI scenario. SERA provides both a recyclable model, and reusable tools that were developed with consideration for both technical developers (e.g., programmers) and also non-technical developers (e.g., animators, interaction designers, psychologists).

The SERA model and tools are about merging techniques from computer animation, intelligent virtual agents (IVAs), and robotics (see Fig. 1). The CGI techniques allow us to integrate tools that professional animators are familiar with, into our workflow. The IVA techniques provided a grounding for the modular integration of a complex, situated and embodied agent - one that uses components for AI, perception, behaviour management, animation and also interaction

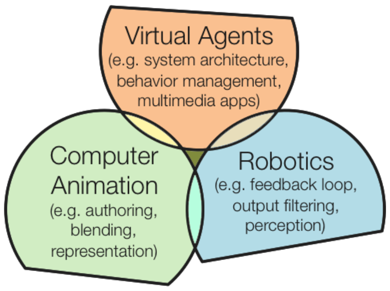
Virtual Agents
(e.g. system architecture, behavior management, multimedia apps)
Computer Animation
(e.g. authoring, blending, representation)
Robotics
(e.g. feedback loop, output filtering, perception)

Figure 1: Our methodology as an intersection of CGI animation, IVA and robotics techniques.

through virtual interfaces (such as video games). Working alongside psychologists also allowed us to understand how such IVA techniques could be adapted and used in a way that they could contribute directly to the development of the interaction and robotic behaviours.

After presenting some related work, we describe the SERA ecosystem along with some of its reusable tools. The paper is complemented with a set of real cases in which such architecture and tools were used.

We hope that by sharing this with the community, we will contribute towards establishing generalized practices and systems for autonomous HRI, towards the advancement of HRI into real world scenarios and eventually towards the establishment of a new technological industry.

Related Work

In this section we refer to the major previous works that have inspired and lead to the current state of our technology. While there may be other works sharing some similarities or purposes, these are the ones that directly taught us the most important lessons and provided us with paths for innovation.

Breazeal created Kismet, the first expressive anthropomorphic robotic character that aimed at presenting lifelike behaviour and engaging people in natural face-to-face interaction (Breazeal 2003). It is considered the pioneering work in our field. Breazeal and collegues also presented the Interactive Theatre (Breazeal et al. 2003). This is one of the first robotic systems to be developed mostly with procedural, interactive animation in mind, by blending AI and an artistic perspective. In his basilar work, van Breemen, defined robot animation as *'the process of computing how the robot should act such that it is believable and interactive'* (Breemen 2004). These works laid down the bricks for the idea of robot animation and interactive animated robots as a field of study and technological explotation.

Gray et al. have provided the first description of tools and methodology that allowed the creation of HRI system with collaboration between animation artists and robot programmers (Gray et al. 2010). The authors drove from experience on animating at least seven different robots with tools that allow animators to take a role on the process. The process they describe is composed of several phases. First, the role of the animator was mostly to design animations and postures which were later played and blended into the character in real-time. This process of defining the blending was lead by another person, a behavior architect who combines the work of the animators, considering different levels of autonomy, and connection with other sources of motion. The resulting behavior would often mix animations and postures by an animator, with functional gestures (e.g. object manipulation or gaze fixation) and procedural expressive gestures (e.g. breathing motion, eye blinking). The authors took inspiration from CGI animation into their workflow, in order to provide an appropriate pipeline that could handle the animation setup at the animator's environment, down to the blending of motion with procedural controllers, and finally rendering the animation safely on the physical robot.

Hoffman and Weinberg have been creating interactive robots that behave in a musical environment. Shimon is a gesture based musical improvisation robot that plays marimba (Hoffman and Weinberg). Its behavior is a mix between his functionality as a musician, for which he plays the instrument in tune and rhythm, and being part of a band, for which he performs expressive behavior by gazing towards his band mates during the performance. Travis is a robotic music listening companion also created by Hoffman, that acts as an interactive expressive music dock for smart phones (Hoffman 2012). The robot plays music through a pair of integrated loudspeakers while autonomously dancing to the beat-matched rhythm.

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 (Quigley and Gerkey 2009). 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 middle-ware (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.

Pereira and colleagues showcased an EMYS robot that continuously interacts with both users and the environment while playing a multi-player board-game, in a way to provide a more lifelike experience[1]. This was the first autonomous robot to interact simultaneously with several human players through a video game running on a large touchtable (Pereira, Prada, and Paiva 2014), and kick-started a series of HRI scenarios developed within the GAIPS group at INESC-ID, up to the work presented in this paper.

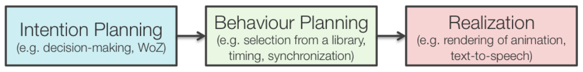
Intention Planning
(e.g. decision-making, WoZ)
Behaviour Planning
(e.g. selection from a library, timing, synchronization)
Realization
(e.g. rendering of animation, text-to-speech)

Figure 2: The SAIBA model for virtual agents (Kopp, Krenn, and Marsella 2006).

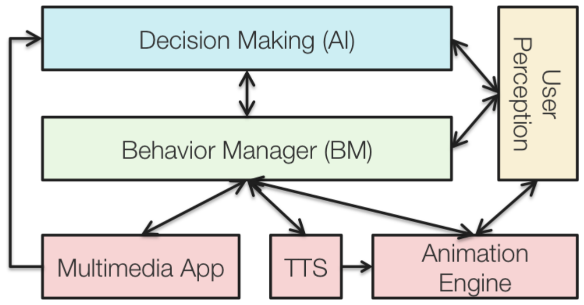

Figure 3: The SERA model. In compliance with SAIBA, the AI is the Intention Planning level; BM is the Behaviour Planning level; All the others (including User Perception) are the Realization level.

The social nature of this application required EMYS to be able to blend several animation modalities in real time, such as gazing towards a person while performing an expressive emotional animation, or changing the overall look of its idle behavior in order to portray its internal emotional state, while still reacting to the presence of the other players.

The SERA Ecosystem

The SERA ecosystem was created following on the SAIBA model which is very popular within the virtual agents community (Kopp, Krenn, and Marsella 2006). Figure 2 shows the original SAIBA model, while Figure 3 illustrates the general components of the SERA model. Coloring of the components establishes a relationship between both figures. In overall, our architecture aims at providing a reusable structure and collection of modules, that can work for different scenario applications and robots.

Starting on Figure 2, the *Intention Planning* components perform the decision-making based on high-level perceptions (e.g. User performed action *X*; User provided some form of input). Those decisions (i.e, intentions) are fed to a Behaviour Planner (BP), which decomposes such intention into more atomic behaviour actions (e.g., speech, gesture) and schedules them for properly synchronized execution. The realization level is responsible for actually performing those individual behaviours and serving as the interface to the user.

[1]EMYS playing Risk: `http://vimeo.com/56200151`

Figure 3 shows how we place components along this SAIBA structure. In general there is some Decision Making (or AI) component controlling the overall interaction, thus representing the Intention Planning. A Behaviour Manager (BM) implements the Behaviour Planning level. This BM can be as simple as a behaviour scheduler, or be modular and contain extra components such as gazing management or rapport.

For a general approach, we decompose the realization level into a Text-to-Speech (TTS) engine, realization of animation, and some multimedia application through which the user can interact with the system and receive feedback from it. Actions selected by the user in such application can be interpreted by the AI as perception of user actions. User perception has not been traditionally included in the SAIBA model. However, on previous work adapting that model to HRI, we have include a transversal Perception layer that runs across all other levels (Ribeiro et al. 2014b). Our experience has shown it to be useful to provide both high-level perception of the user to the AI (e.g., facial expression, gestural actions), and also lower-level perception to be used both for the generation of some types of behaviours at the BM component (e.g. rapport), and for adaptation of behaviour at both the BM and Animation component (e.g., tracking a user's face).

Implemented Reusable Components

Within the SERA ecosystem, there are several components that we have developed in order to be generic and reusable across applications and robotic embodiments. The major components are described in the next subsections. Those are Thalamus, which is the backbone integration middleware, Skene which is a semi-autonomous behaviour planner, and Nutty Tracks which is a symbolic animation engine.

Thalamus

Thalamus is a high-level integration framework aimed especially at developing interactive characters that interact both through virtual and physical components. It was developed in C# to accommodate social robots into such a framework, while remaining generic and flexible enough to also include virtual components such as multimedia applications or video games running on a touch table (Ribeiro et al. 2014a). It follows on the concepts of asynchronous messaging middle-ware and on well-defined message structures (based on MOM as ROS does) to provide a seamless plug-and-play-modules functionality (Figure 4). However, being a higher level middle-ware (in comparison to ROS) it works "out of the box", without requiring any installation on the host system, and also includes graphical interfaces. It aims at being easy to use and to share, to be portable and adequate for collaborative development.

Thalamus breaks the sense-think-act loop by not specifying any particular layer structure. The idea behind it is that a Thalamus Character is an agent built out of agents. These agents are Thalamus modules that exchange perceptions and actions between them, so while any module may actually contain a sense-think-act loop, holistically the Thalamus Character does not. That allows it to simultaneously

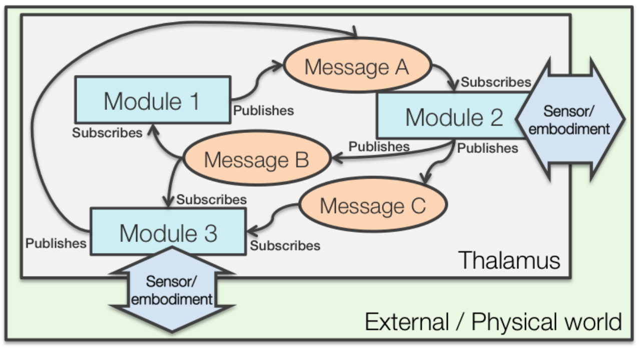

Figure 4: Exemplification of how several Thalamus modules coexist in the same virtual space, exchanging messages through a publish/subscribe mechanism. Notice that any module may act as an external interface (e.g., a TTS, a Microsoft Kinect[3], an embodiment).

contain several modules that deal with behaviour, or with perception, or even with decision making, as long as the combination of them all produces the expected overall behaviour. These Characters can be used seamlessly across embodiments (virtual or robotic) and applications, by just switching or tweaking some of the modules. An example of that is a robot interacting with users through an application running on a touch-table, and using a Microsoft Kinect to track the user's face. Contrary to traditional agents that contain a "body", all the those three components represent the physical interface between the users and the system. User perception is informed by the Kinect, which is independent of both the robot and the touch-table; user actions are perceived by the application (e.g user clicks), and behaviours are both executed expressively by the robot, and task-wise through the application (e.g. the agent can invoke the application to pop-up a screen while the robot points at it).

It is importante to notice at this point that every component we develop for our system (including the ones on the following sections) is developed in order to function as a Thalamus module, and therefore, to be able to coexist within the SERA ecosystem along with all the other ones we develop.

Skene

Skene is a semi-autonomous behaviour planner that translates high-level intentions originated at the decision-making level into a schedule of atomic behaviour actions (e.g. speech, gazing, gesture) to be performed by the lower levels (Ribeiro et al. 2014b). Its development is still ongoing, and it was created with situated robots in mind that can also interact through multimedia/virtual interfaces (like a large touch-table). As such, it is the place where most of the other components meet in order to integrate behaviour with the environment. Some of its features are:

- Containing an explicit representation of the virtual and physical environment, by managing coordinates of relevant targets at which a robot can point or gaze at;
- Autonomously performs some contingent gazing be-

haviour, such as gaze-aversion and establishing gaze (the opposite of aversion), using an internal gaze-state machine (GSM);

- Gaze-tracking a target that is marked as a Person using the GSM;
- Automatically gaze-track screen-clicks using the GSM (for use with multimedia application running on touch-tables);
- Maintaining, managing and allowing other components to control utterance libraries.

Skene Utterances are the actual representations of the aforementioned intentions and were mostly inspired by the FML-BML pair used in virtual agents and the SAIBA model (Kopp, Krenn, and Marsella 2006). They are composed of text, representing what the robot is to say, along with markups both for the TTS, and for behaviour execution. The behaviour markup can be used to control Gazing, Glancing, Pointing, Waving, Animating, Sound, Head-Nodding and even Application instructions. The following is an example of a Skene Utterance:

```
<GAZE(/currentPlayerRole/)>I'm unsure if
\rspd=70\ it's a good idea \rspd=100\ to
<HEADNODNEGATIVE(2)> build industries
near <WAVE(throughMap)> the populated
areas. <ANIMATE(gestureDichotomicLeft)>
<GLANCE(Eco)> What do you think?
<GAZE(clicks)>
```

The behaviours contained in the markup are non-blocking, meaning that while the speech is executed, the TTS engine sends events whenever it reaches a marked-up position, so that Skene can concurrently launch the execution of that mark-up behaviour. While this seems like a pliable solution, it actually allows the further Realization components to perform their own resource management. Thus, if for example, the robot needs to gaze somewhere and perform an animation at the same time, the animation engine is be the one to either inhibit or blend the simultaneous forms of expression.

These Skene utterances have been developed mostly by well informed psychologists that take part in the development cycle as interaction designers. In order to facilitate such collaboration, Skene Utterance Libraries are stored and loaded directly as Microsoft Excel Open XML spreadsheets[4]. Such feature hugely facilitates the interaction designers to collaborate between them and with the technical development team by authoring such files using online collaborative tools such as Google Spreadsheets[5].

Nutty Tracks

Nutty Tracks (Nutty) is a symbolic animation engine based on CGI methods that allows to animate both virtual and robotic characters (Ribeiro, Paiva, and Dooley 2013). It is simultaneously a design-time and run-time environment, i.e., it is used both for designing and programming animation, as well as to execute it in real-time during interaction.

Using Nutty provides us with high flexibility regarding the design, blending and modulation of animations on any robot. It allows to use professional animation tools (e.g. Autodesk 3ds Max [6]) to design animations and postures, and provides a generic translation layer between the character's animation parameters and the actions and parameters that arrive from other components in the system.

One of the principles of Nutty is to work on animation at a symbolic level. This means that while the system is aware of the hierarchy of the robot, its animation isn't processed at the level of the actual joints, but on symbolic joints (similar to (Gray et al. 2010)). These symbolic joints can actually be mapped to a real robotic joint, or to a set of joints, thus working as an aggregated joint (e.g. we can animate a 1-DoF joint called VerticalGaze which is later decomposed into several real motors of the real robot's neck).

The composing of animation programs in Nutty Tracks follows a box-flow type of interface greatly inspired in other programming tools commonly used by artists, such as the Unreal Development Kit (UDK)[7], Pure Data[8] or Houdini[9]. Animation Controllers (ACs) are connected into a chain of execution that generate and compose animation either procedurally or using animations and postures that were pre-designed (e.g. with Autodesk 3ds Max). These chains of ACs are further composed into a hierarchy of layers that can be activated and deactivated during interaction in order to either blend or override their animated degrees-of-freedom with each other[10].

These ACs can be programmed separately from Nutty Tracks and also loaded as plugins, and shared within the community. This features turns Nutty into not only a highly flexible animation software for robots, but also a highly extensible one.

Nutty Plugins for each robot

In order to control a new robot, a specific Nutty Output Plugin module is developed. By fitting into Nutty Tracks as a plugin, it is loaded during execution, allowing the user to select which output (and robot) should be used. It contains a BodyModel, representing the robot's hierarchical structure, along with parameters that specify each joint's axis of rotation and limits. It also contains the code that translates and executes a generic Nutty Animation Frame into the robot's control API. The referential in Nutty was developed based on the one found in Autodesk 3ds Max, and joint rotations are generally specified as floating point degree angles. We also consider the zero-pose (when all angles are set to zero) to be having the robot facing straight, neutral and forward.

Creating a Nutty Output Plugin for each new specific robot requires some work and a lot of expertise. However once it is created, it can be reused throughout all future

[4]XLSX: http://fileformat.wikia.com/wiki/XLSX

[5]Google Spreadsheets: https://www.google.com/sheets/about/

[6]3ds Max: http://www.autodesk.com/products/3ds-max/overview

[7]UDK: http://www.unrealengine.com/

[8]Pure Data: http://puredata.info/

[9]Houdini: http://www.sidefx.com/

[10]Nutty Tracks: http://vimeo.com/67197221

projects. Moreover, any plugin for any robot can be shared with the community. The main advantage is, of course, that one might not need to develop the plugin for a robot if it is already available in some public repository. The second major advantage is that in case of a robot's API upgrade, only this plugin needs to be replaced, while all the animation data and logic remains.

Nutty-Keepon example: We take as example the development of the Nutty-Keepon plugin. The first step was to understand how the Keepon is controlled in its own API. It is especially important to outline what units and reference system it uses. The Keepon used in our system was modified with controllable servos[11] and connects to a computer using an Arduino[12] board. Each servo is controlled by specifying a target position which is represented by an integer value ranging from 0 to 180 for the Pan, Roll and Tilt servos, and 0 to 100 for the Bop servo (Figure 5).

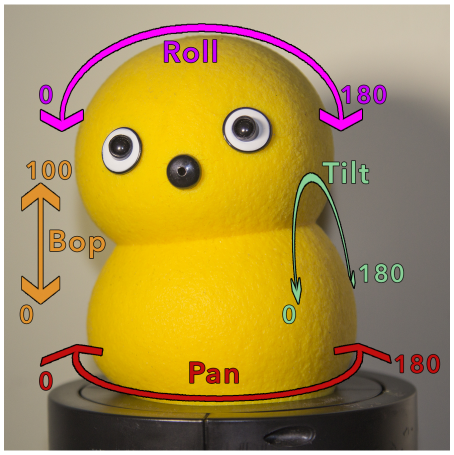

Figure 5: A real Keepon robot and range of execution of its Arduino-hacked servos.

Because in Nutty Tracks animation is normally specified as degree angles, the Nutty-Keepon's robot representation performs a translation to set all zero-angles (0°) to correspond to servo values of 90 for Pan, Roll and Tilt. As to Bop, it was kept as a value ranging from 0 to 100, representing a percentage. To test and verify this, a virtual version of the Keepon was made using Autodesk 3ds Max for modelling, and Unity3D[13] for realtime rendering (Figure 6). Nutty Tracks can also be used to control this virtual version by setting the used BodyModel to the Keepon (which is loaded from the Keepon plugin), while using as output a built-in frame streamer based on JSON[14], which send frames via TCP sockets from Nutty Tracks to the Virtual-Keepon application.

Animatable CGI model of the robot

We used Autodesk 3dsM ax as a host animation software to

[11]Keepon Hack: http://hennyadmoni.com/keepon/
[12]Arduino: https://www.arduino.cc/
[13]Unity3D: http://unity3d.com/
[14]JSON: http://www.json.org/

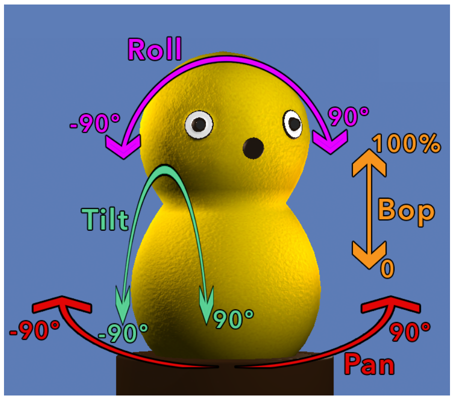

Figure 6: A screenshot of a Virtual-Keepon built in Unity3D with the angular range of movement of its degrees of freedom. Note the mapping from the real values in Figure 5 to the angular coordinates used in Nutty Tracks.

load a different type of Nutty Tracks plugin. This should not be confused with the Nutty plugins described on the previous section. While the Nutty Tracks environment works by loading plugins, it can also become a plugin itself, to a host animation software such as 3ds Max. That means that instead of Nutty Tracks being ran as a standalone application, it is programmed in Maxscript[15] to run as a 3ds Max plugin. From there, Nutty can optionally use the available Body-

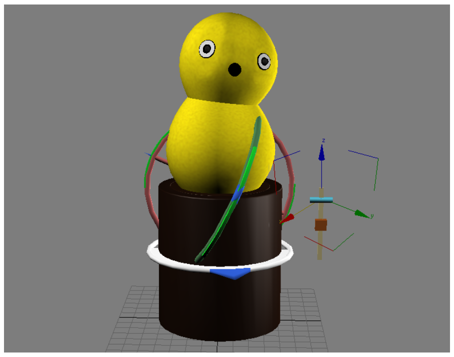

Figure 7: The animatable CGI Keepon robot in Autodesk 3dsmax.

Model to automatically create a simple animatable structure based on the representation provided. However, because 3ds Max already provides a complete set of modelling and animation tools, an expert animator was able to create a proper animation rig along with a 3d mesh of robot's embodiment with a deformable modifier in order to more accurately represent how the poses and animations look in the real robot (Figure 7). Also, because the actual Nutty Tracks engine is running within 3ds Max, the animator can animate while watching the result rendered on the real robot in real-time[16].

[15]Maxscript: www.autodesk.com/3dsmax-maxscript-2012-enu/
[16]Nutty Tracks: http://vimeo.com/67197221

Other Generic Reusable Modules

A reusable standalone TTS component was also developed, as it is independent of both embodiment and application. However it serves only as a bridge to the operating system's own TTS (generally Windows SAPI). As to perception, the main general component we have been using is a Microsoft Kinect interface, which is used to track people's faces and direction/intensity of speech (note that Kinect's speech recognition is a different thing, and has not been used yet). Some other perception components have also been used in particular applications to complement Kinect's capabilities (e.g. using a web-camera for accurate detection of user's facial expressions).

Real Cases

Here we present brief descriptions of different robots and HRI application created with the SERA ecosystem.

EMOTE's NAO robotic tutor

The EU FP7 EMOTE project[17] has been developing an autonomous empathic robotic tutor aimed to teach topics about sustainable development to children in schools. This project represents the first use of the SERA ecosystem. A NAO torso robot[18] plays an educational video game called Enercities[19] on a large touch-table along with one or two children (Figure 8).

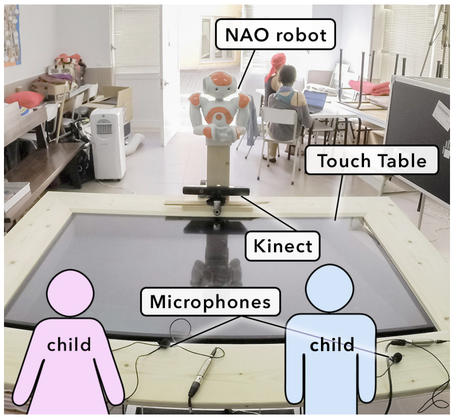

Figure 8: The physical setting of the EMOTE scenario.

Figure 9 illustrates the components used in the EMOTE scenario. All components were developed as Thalamus modules, with Skene acting as the main central point of the system. As the earliest scenario using the SERA model, it still didn't use Nutty Tracks as the animation system. Instead, a specific NAO Robot module was written to connect Thalamus with the NAO's api (NAOqi framework[20]). However, this scenario was developed during the whole course of the EMOTE project, and as such, it was also a sandbox for experimentation on, for example, the role of Skene, and what kind of perception information could be generalized in such a system. Skene was used to manage all the gazing, between both children and also towards specific on-screen targets. Because it includes a model of the environment, it is able to translate screen coordinates provided by the Enercities Game (in X,X form) to angles that are then used to generate gazing commands. Because NAO includes its own TTS, there was no need to include it as a separate component.

Note there are two lavalier microphones connected to the system. These were used for more accurate perception of when and which student was speaking, so that the robot could properly gaze towards that student, or wait for silence before starting any speech behaviour. The specification of perception messages however, are abstracted in such a way that, in the absence of the microphones, such role can be performed by the Kinect, even if with less accuracy.

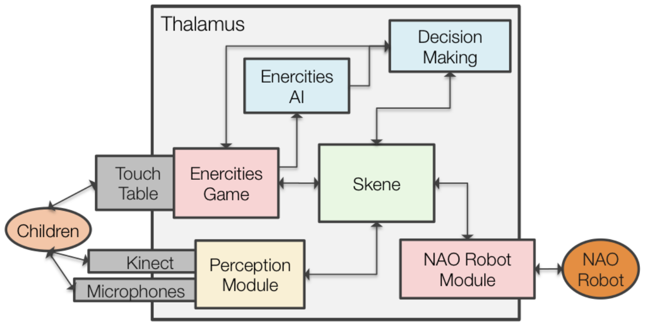

Figure 9: System used in the EMOTE scenario. Coloring of the Thalamus components are meant to match the ones of the SAIBA and SERA models (Figures 2 and 3)

E-Fit Keepon

E-Fit is an adaptive HRI application that keeps early adolescents engaged in physical activity over long periods of time (Grigore 2015). The Keepon robot (Kozima, Michalowski, and Nakagawa 2009) interacts with participants once a day for approximately 5-10 minutes. Each day, the robot asks a series of questions and collects data from an off-the-shelf fitness sensor worn by participants. The robot's back-story unfolds over time. It is a robot-alien, named EfiT, that landed on Earth and needs the adolescent's help to return home. If the user accomplishes daily physical activity goals he will be helping the robot get back to its home planet.

Figure 10 shows the structure of the E-Fit scenario. As it follows the SERA model, it looks very similar to the EMOTE project's architecture in Figure 9. The major difference is that the decision making is performed solely by a Dialogue Manager, while in EMOTE there was a separate AI just for calculate game moves. There is also an EfiT App in the system, which runs on a smart-phone instead of a touch-table and provides task-related interaction to the children. The Perception is simplified to using the Kinect only for face-tracking, because children interact freely with the

[17]EMOTE project: `http://www.emote-project.eu`
[18]NAO Robot: `http://www.aldebaran.com/en`
[19]EnerCities: `http://www.enercities.eu/`
[20]NAOqi: `http://doc.aldebaran.com/1-14/dev/naoqi/index.html`

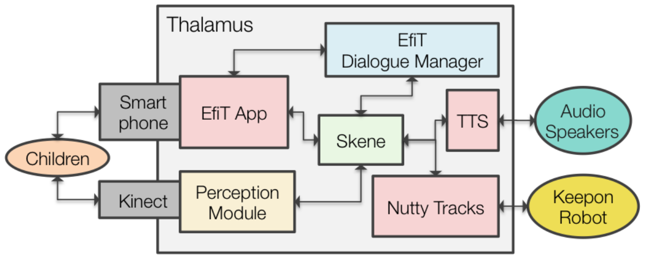

Figure 10: System used in the E-Fit scenario.

system, whereas in EMOTE they were ”attached” to the table and as such it was possible to place lavalier microphones on them.

Here the NAO robot was also replaced with the Keepon robot, which is now controlled with Nutty Tracks (as explained in the previous section).

PArCEIRO: EMYS plays Cards and Tabletop games

The PArCEIRO project[21] is an aglomerate of several individual projects developed by undergraduate students at INESC-ID’s GAIPS lab[22]. It aims at developing and creating social robots that will interact with humans in entertaining activities, such as tabletop card games. Until now, these have all been implemented using very similar versions of the SERA ecosystem. The common components between them are that they all use Thalamus, Skene and Nutty Tracks to control the EMYS robot[23]. They are all based on a multimedia game played on a touch-table. Some are single-player, others are multi-player. The single-player games use the Kinect to track the person that is interacting, while the multi-player scenarios don’t. The reason for this is that due to the placement of the players, the current Perception module using a Kinect was not able to capture and track them all. Instead, the system relies on contextual information (e.g., to know who’s turn it is), and uses that information to, for example, gaze or speak at the current player. It also uses all the information and events provided through the game applications to know that a player has touched the screen (to generate gaze commands), or performed a game action.

Split or Steal

In the Split or Steal scenario, the EMYS robot plays an adaptation of the british daytime game show Golden Balls[24]. The architecture of this scenario can be see in Figure 11 and is very similar to the E-Fit one. Instead of running an application on a smart-phone, the Split or Steal game is ran on a large touch-table (the same one as the EMOTE scenario, previously shown in Figure 8). Here the EMYS robot is controlled by Nutty Tracks, which in turn receives behaviour commands from Skene, in a similar fashion to what was described previously in the EMOTE scenario. In this case however, the game is played by EMYS and a single human player, which is placed face-to-face with the robot in opposite sides of the touch-table. Because players play in a standing pose, a Kinect was used solely for tracking the face of the player so that EMYS can gaze correctly towards them.

A new component that was introduced here is FAtiMA. FAtiMA (Fearnot AffecTIve Mind Architecture) is an Agent Architecture with planning capabilities designed to use emotions and personality to influence the agent’s behaviour (Dias and Paiva 2005), and has actually been widely used within the GAIPS lab. While it is not a new piece of software, this was the first time it was integrated within a Thalamus environment in order to be used with the SERA ecosystem. Since then we have been using FAtiMA as the ”mind” for all SERA systems.

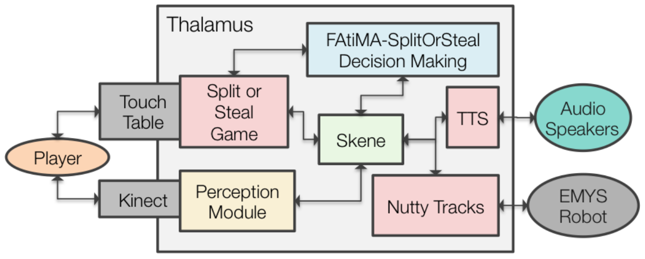

Figure 11: System used in the Split-or-Steal scenario.

Sueca

In the Sueca scenario, the EMYS robot plays a traditional portuguese card-game called Sueca[25]. This HRI scenario is aimed at the elder population, where the Sueca card game is very popular. As such, both the game and behaviour of the robot were designed following an initial user-center design study lead by psychologists, involving members of the elderly population (Petisca et al. 2015). Figure 12 shows the components of the scenario. As described previously, the system relies on Thalamus, Skene and Nutty to integrate and manage the behaviour of the EMYS robot. As in the EMOTE scenario, the decision making was split between two components: because Sueca is a game, there is a Sueca AI dedicated to calculating game moves. Along with that, the main decision making is performed by FAtiMA.

Coup

The final scenario we add is Coup. This is a multimedia adaptation of the Coup card-game[26]. This scenario was developed with both a single-human version (human vs EMYS) and a multi-human version, in which five human players plus EMYS all play the game. For the same reason mentioned with the Sueca scenario, the Kinect and Perception module are used only for the single-human version.

[21]PArCEIRO: http://gaips.inesc-id.pt/parceiro/
[22]GAIPS lab: www.gaips.inesc-id.pt
[23]EMYS robot: http://flash.ict.pwr.wroc.pl/
[24]Golden Balls: https://en.wikipedia.org/wiki/Golden_Balls
[25]Sueca: https://en.wikipedia.org/wiki/Sueca_(card_game)
[26]Coup: https://boardgamegeek.com/boardgame/131357/coup

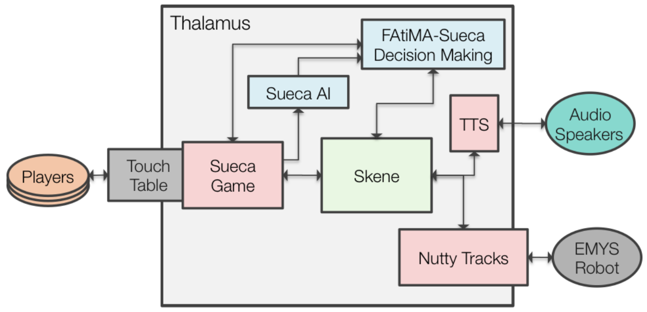

Figure 12: System used in the Sueca scenario.

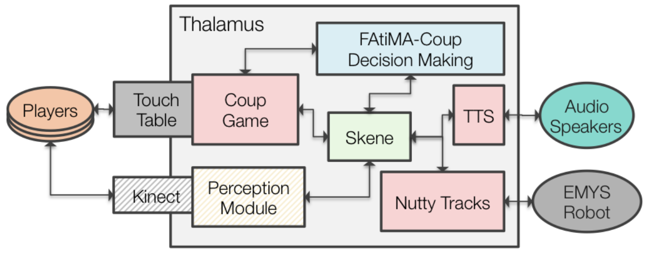

Figure 13: System used in the Coup scenario.

Conclusions

Through the development of several different HRI scenarios, using different robots, we are reaching a point of standardization of our practices. Most of the software developed was created with reusability in mind, and the integrated solution has proven to work. Our focus has mostly been on scenarios that feature a stationary robot interacting with one of more human users through some multimedia application running on a large touch-screen. However, we believe that our practices can be extrapolated to other settings, even if that requires development and integration of new components (e.g. natural interaction, locomotion, navigation). Furthermore, we are dedicating maximum effort so that any new components we may need to develop are also developed so that they can also be reused and coexist in the SERA ecosystem.

Following on our experience, we believe that it is possible to develop a general set of tools for HRI that can be modularly integrated depending on the functionalities required by the application. The SERA ecosystem is currently being used in all HRI application developed at the GAIPS lab at INESC-ID, and also in several scenarios being developed at Yale University's Social Robotics Lab. While the Skene component still requires some tweaking and further developments, the Nutty Tracks animation engine has provided stable, flexible and sufficient for the needs of complex multimodal animation of different types of robots.

Our next steps will be to continue expanding this system to more robots such as the Baxter robot by Rethink Robotics[27]. Following on the very positive experience with the Arduino-hacked Keepon robot, we are also starting to explore the use of SERA for other DIY[28] or printable robots which are generally supplied with just assembly instructions, and no software. As most of these robots rely on Arduino, we already have a working system for them.

So far the major challenges are to make sure the animation rig and Nutty Tracks representation match the resulting robot pose. In most cases that it solved through trial-and-error while tweaking the translation methods from Robot-to-Nutty and vice-versa. Although implementing an interface for a new robot requires expertise regarding both the robot's control system, and CGI character animation, we have found that after this step is accomplished, the more relevant step of designing the robot's expression can further take an artistic approach. With the animation software controlling the actual robot in real-time, even during design-time, an animator is able to carefully refine all the animations. By following on reusability-factors from intention-to-animation, we expect this type of system to become a standard for HRI scenarios in which the role of the robot is to stand as a believable and expressive interactive character.

[27]Baxter robot: `http://www.rethinkrobotics.com/baxter/`

[28]DIY: ”Do it yourself”

References

Breazeal, B. C.; Brooks, A.; Gray, J.; Hancher, M.; Mcbean, J.; Stiehl, D.; and Strickon, J. 2003. Interactive Theatre. 46(7):76–84.

Breazeal, C. 2003. Emotion and sociable humanoid robots. *International Journal of Human-Computer Studies* 59(1):119–155.

Breemen, A. V. 2004. Bringing robots to life: Applying principles of animation to robots. *CHI2004* 4:1–5.

Dias, J., and Paiva, A. 2005. Feeling and reasoning: A computational model for emotional characters. In *Progress in artificial intelligence*. Springer Berlin Heidelberg. 127–140.

Gray, J.; Hoffman, G.; Adalgeirsson, S. O.; Berlin, M.; and Breazeal, C. 2010. Expressive, interactive robots: Tools, techniques, and insights based on collaborations. *HRI 2010 Workshop on What do Collaborations with the Arts Have to Say About Human-Robot Interaction*.

Grigore, E. C. 2015. Modeling motivational states through interpreting physical activity data for adaptive robot companions. In *User Modeling, Adaptation and Personalization*. Springer. 379–384.

Hoffman, G., and Weinberg, G. Gesture-based Human-Robot Jazz Improvisation.

Hoffman, G. 2012. Dumb robots, smart phones: A case study of music listening companionship. *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication* 358–363.

Kopp, S.; Krenn, B.; and Marsella, S. 2006. Towards a common framework for multimodal generation: The behavior markup language. In *Intelligent Virtual Agents*, 205–217.

Kozima, H.; Michalowski, M. P.; and Nakagawa, C. 2009. Keepon. *Adv. Robot.* 1(1):3–18.

Pereira, A.; Prada, R.; and Paiva, A. 2014. Improving social presence in human-agent interaction. In *Proceedings of the 32nd ACM Conference on Human Factors in Computing Systems*, 1449–1458. ACM.

Petisca, S.; Correia, F.; Maia, N.; and Paiva, A. 2015. Social robots for older adults : Framework of activities for aging in place with robots. In *International Conference on Social Robotics*.

Quigley, M., and Gerkey, B. 2009. ROS: an open-source Robot Operating System. *ICRA workshop on open source software.* 3(3.2).

Ribeiro, T.; Di Tullio, E.; Corrigan, L. J.; Jones, A.; Papadopoulos, F.; Aylett, R.; Castellano, G.; and Paiva, A. 2014a. Developing Interactive Embodied Characters using the Thalamus Framework: A Collaborative Approach. In *14th International Conference on Intelligent Virtual Agents*.

Ribeiro, T.; Pereira, A.; Di Tullio, E.; Alves-Oliveira, P.; and Paiva, A. 2014b. From Thalamus to Skene: High-level behaviour planning and managing for mixed-reality characters. In *Intelligent Virtual Agents - Workshop on Architectures and Standards for IVAs*.

Ribeiro, T.; Paiva, A.; and Dooley, D. 2013. Nutty tracks: symbolic animation pipeline for expressive robotics. *ACM SIGGRAPH 2013 Posters* 4503.

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 UID/CEC/50021/2013 and by the scholarship SFRH/BD/97150/2013. We also thank the rest of the people from INESC-ID's GAIPS lab who were somehow involved in these projects, namely Patrícia Alves-Oliveira and Sofia Petisca for their psychology expertise, to João Dias for the help on integrating FAtiMA, and to our undergraduate students Filipa Correia and Nuno Maia for the Sueca scenario, to André Pires for the Split or Steal scenario, and to Filipe Fortes and Rui Soares for the Coup scenario. We must also thank the remaining of the EMOTE team who contributed to the overall development of the presented scenario throughout the project. Last but not least, we must thank the Yale University and in particular Brian Scassellati and Iolanda Leite for their invitation to visit and collaborate with them at Yale's Social Robotics Lab "ScazLab" on the E-Fit Keepon project. This paper represents them sum of collaboration with multiple different people, and without all of them it would not have been made possible.