AgentSlang: A New Distributed Interactive System
Current Approaches and Performance

Ovidiu Şerban and Alexandre Pauchet
LITIS Laboratory, Normandy University, INSA de Rouen,
Avenue de l'Université - BP8, 76801 Saint Étienne du Rouvray Cedex, France

Keywords: Distributed Interactive System, Embodied Conversational Agent, System Benchmark.

Abstract: This paper proposes a generic platform for developing fast and reliable Distributed Interactive Systems. The modelling is based on a component design approach, with element structure simple and versatile enough to allow the integration of existing algorithms. The AgentSlang platform consists in a series of original components integrated with several existing algorithms, to provide a development environment for Interactive Systems. There are several original parts in our approach. First, the platform is based on a data and component oriented design, which integrates into a unified system the concept of Feedback Management, Dialogue Management and a flexible component architecture. Second, the Syn!bad language, is integrated as a component of AgentSlang. Third, the message exchange speed is superior of any existing platforms, even in the context of providing extra features, such as action execution feedback and data type consistency check.

1 INTRODUCTION

Building a virtual environment, in which the participants can interact naturally, is very challenging. Virtual conversational agent (also called Embodied Conversational Agent: ECA (Ogan et al., 2012)), due to a very realistic appearance, can increase the users' expectations up to the point that they are disappointed by the actual agent capabilities. This phenomenon is called the "*uncanny valley*" (Mori, 1970).

Due to the complexity of complete dialogue systems, most of the existing ECAs only integrates basic dialogue management processes, such as a keyword spotter within a finite-state or a frame-based approach (for instance, see the SEMAINE project (Schröder, 2010)). Therefore, dialogue management remains inefficient in existing ECAs (Swartout et al., 2006).

With the emergence of virtual environments and especially with participative digital storytelling systems, situations with child-ECA interaction are increasing. In such an environment, providing children with natural reactions from an ECA becomes critical as their social capabilities are still in development. Moreover, the new generation of children is now used to virtual environments and therefore expects the best for his/her interaction with a system.

Dealing with rich interaction, given by multiple sources, in a very fast and reliable way has become our current challenge. The human-computer interaction problem is complex and requires work on multiple levels, such as speech and feedback recognition, natural language processing, dialogue management or speech generation. Each of these issues has been processed independently, or at most a combination of the two, but to our knowledge, a system dealing with all of them has not been proposed yet. The integration problem becomes more complex when a certain genericness is required.

We choose to focus our design around the storytelling environments, where a narrative and classic dialogue model need to be introduced. The narrative states are a set of fixed points where the only control the user has is to stop or resume the narration, where as the dialogue is a more dynamic phase, where the user can ask or can be asked questions. Since none of the existing ECA systems focus on these issues, we decided to focus more on dialogue management.

This paper is structured as follows: we present a brief state of art for the Distributed Interactive Field in Section 2. Section 3 presents several components of the AgentSlang platform, followed by our proposed benchmark (Section 4). We conclude this paper with a brief discussion over the proposed platform.

È.Ÿerban O. and Pauchet A..
AgentSlang: A New Distributed Interactive System - Current Approaches and Performance.
DOI: 10.5220/0004907305960603
In *Proceedings of the 6th International Conference on Agents and Artificial Intelligence* (ICAART-2014), pages 596-603
ISBN: 978-989-758-015-4

2 RELATED WORK

An Interactive System (IS), as a complex system, contains multiple critical components. The components processing the user's inputs can be formalized as Knowledge Extractors, the Dialogue Manager as an Interaction Manager and the output components are Behaviour Generators associated to Players. For each of these categories, the problem has been approached by multiple research projects. In this chapter, we focus on platforms that propose an architecture to model the integration of all these components into a Distributed Interactive System (DIS).

The component that interprets the behaviour, the Player, can be either a simple Speech Interface, an Embodied Conversational Agent (ECA) or a Robot, for instance. We restrict the presentation to projects that propose a system with a player that are not strongly linked with the interpreters and are generic enough to be used in multiple environments and scenarios. All the presented platforms follow the evolution of communication paradigms between components for large Agent-based Architectures.

2.1 MULTIPLATFORM

The first example is an ad-hoc communication, formalized over a PVM Protocol (Geist et al., 1994). The implementation of the protocol was used in the MULTIPLATFORM project (Herzog et al., 2004) and served as a component integration platform for two well known projects: Verbmobil (Wahlster, 2000) and Smartkom (Wahlster, 2006). However, the message protocol is outdated and this project is not actively maintained any more.

2.2 Psyclone

Another generation of systems is based on the Psyclone middleware platform (CMLabs, 2007), which implements a classic blackboard communication protocol. This enables quick integration, but the middleware tends to be rather slow for platforms that use many updates of the shared blackboard space, as the Semaine Middleware benchmark shows (Schröder, 2010). We present two platforms using the Psyclone protocol: Mirage and GECA.

2.2.1 Mirage

Thorisson et al. (Thórisson et al., 2004) propose a constructionist methodology to build a component based architecture. This can be adapted to design ECA systems, by modelling reusable components that can be plugged in a unified architecture. The platform aims at identifying existing software that could be reused and integrated into the project. The system allows the integration of different components written in C++ or Java. The message format is a series of well documented hard-coded protocols, that are distributed across the network on multiple machines in order to obtain real-time performance. The platform has no source code publicly released.

2.2.2 GECA

The Generic Embodied Conversational Agent Framework (GECA) (Huang et al., 2008) aims at building ECAs that interact with human users in face-to-face conversations, while being able to recognize verbal and non-verbal input, generate speech, gesture or posture behaviour and perform basic conversational functions (i.e. utterance turn taking and feedback detection). The platform supports C++ and Java integration through the OpenAIR protocol, integrated in the latest revision of Psyclone (CMLabs, 2007). The messages are sent using several distributed blackboards. The system proposed various formats for the data messages, such as standard existing XML formats. Moreover, the authors propose a generic markup language GECAML (Huang et al., 2008) which unifies all the features offered by various Automatic Speech Recogniser and Text to Speech API.

2.3 Companions: A Generic ECA Project using a Middleware Platform

Companions (Cavazza et al., 2010) is not only an ECA but also a companion, engaged into a long term interaction process to forge an empathic relation with its user. The Companions ECA is built around the scenario *"How was your day ?"*, which acts as a general interaction theme with an open dialogue. Unfortunately, due to a proprietary technology, the platform does not offer many research or technical details, such as validation, licence or performance aspects.

From the technical perspective, the system uses several proprietary platforms, designed by the industrial partners: a middleware platform, Inamode, developed by Telefónica I+D[1]; an Automatic Speech Recognition (ASR) and a Text To Speech (TTS) engine, developed by Loquendo[2]; and a Virtual Character, developed by As An Angel[3].

[1] http://www.tid.es/en/
[2] http://www.loquendo.com/en/
[3] http://www.asanangel.com/

2.4 Message Oriented Communication

The idea of components communicating using a message oriented protocol is well developed in industrial applications (O'Hara, 2007). Recent developments lead to a unified stable model called the Advanced Message Queue Protocol (AMPQ) (O'Hara, 2007). This protocol provides more flexibility than Psyclone in terms of communication models, such as publish/subscribe and request/reply patterns. Moreover, these operations are optimised for large and fast data exchange. There are various implementations of this protocol and two platforms are very commonly used in Open Source projects: ActiveMQ (Snyder et al., 2011) and ZeroMQ (also spelled ØMQ) (Hintjens, 2013). Two well known projects currently integrate the ActiveMQ platform: Semaine and VHToolkit.

2.4.1 Semaine

Semaine (Schröder, 2010) is a Sensitive Artificial Listener (SAL), built around the idea of emotional interaction. The project focuses on a Virtual Character that perceives human emotions through a multimodal set-up and answers accordingly. The response is not always a direct reaction to the affect perceived, as a certain level of planning is supported. Several virtual characters with different personalities are proposed, each having a different reactive model to the perceived emotion. The system introduced the idea of component based, distributed interactive system, where each algorithm or component could act independently. The affect detection part is a fusion of low level speech features extracted using OpenSMILE (Eyben et al., 2010) and face gestures classified using iBug (Soleymani et al., 2012). The behaviour of the agent is managed by two components developed by the team, which is sent to a Text-to-Speech synthesiser: MaryTTS (Pammi et al., 2010). The speech is transmitted to a speech and gesture synthesis component, which converts the data into Greta BML code (Poggi et al., 2005).

2.4.2 VHToolkit

Virtual Human Toolkit (VHToolkit) (Hartholt et al., 2013) is a generic platform designed to support ECA systems, developed around a component-based design methodology. It has been used successfully in many applications varying from e-learning to military training. It provides a collection of components for all the major tasks of an interactive system: speech recognition, text-to-speech, dialogue management (using NPCEditor component (Leuski and Traum, 2011), non-verbal body movement generator (Lee and Marsella, 2006) and an uniform perception layer (formalized as PML (Scherer et al., 2012)). The project uses the SmartBody Embodiment (Shapiro, 2011) as a visual BML interpreter for verbal and non-verbal behaviour.

2.5 Summary

The key functionalities for an efficient IS system are: a fast Data-Oriented Design, Dialogue Management and Affect Detection. First of all, except for Mirage, all the presented platform do not discuss the choice of internal data representation. Moreover, almost all the systems use heavy XML messages or ad-hoc string formats, which does not offer the possibility of future extensions of these systems. Then, only Semaine and Companions projects currently propose Affect Oriented features. Finally, the Dialogue Management is offered by VHToolkit, Companions, Mirage and GECA.

Virtual Human Toolkit is the most complete platform for building ECA systems that we found so far. Unfortunately, even if the platform offers a multimodal perception component, its output is linked directly to the smart body behaviour, making the system purely reactive rather than "emotion-aware". The dialogue manager and the non-verbal behaviour generator do not take into account such perception.

Semaine is a Sensitive Artificial Listener, where the sensitive part refers to the affect recognition and simulation aspect. The listening key word refers to the ability to perceive certain emotions, but due to the lack of dialogue/interaction management, the agent is not able to reply with semantically adequate behaviour to the human companion.

Companions offers both dialogue management and affective interaction, but with very few details about this aspect. In fact, due to the proprietary licence of the system, no component or source code has been publicly released. This is making very difficult an evaluation of the quality of these features. Moreover, the system is centred around the *"How was your day ?"* scenario, which restricts even more the field of application.

Mirage is used for an augmented reality application, where the agent is participating to the environment. The dialogue capabilities of the system are very basic, by providing some reactive model linked with the perception. GECA does not offer support for affect oriented design, but supports basic dialogue models. Both Mirage and GECA use different versions of Psyclone (OpenAir is the previous version of Psyclone), which are not actively maintained.

In the following, we describe our proposition, AgentSlang, that encapsulates some of the good points of these platforms, while adding better performance, Object-Oriented Data representation, simplified dialogue modelling and Affective Feedback detection.

3 OUR PROPOSITION: AgentSlang

AgentSlang is a collection of components, created on top of the MyBlock middleware platform, which enables to build rich, distributed and fast Interactive Systems. MyBlock ensures the component-to-component communication, dealing with data transmission in an efficient way. It adds a transparent layer of communication so that AgentSlang components do not have to deal with all these concepts.

The choice of a certain Message Queue protocol implementation has been done on licence availability, popularity and performance. In particular, we benchmarked ZeroMQ over ActiveMQ, to confirm that ZeroMQ (our choice) is a faster and reliable candidate, which supports multiple connection types, different communication patterns, the possibility to send binary data and the absence of a broker component, which slows down a distributed pipeline architecture.

In a component-based approach, the data exchange between components becomes critical. Therefore, we discuss the aspect of Data Oriented Design by introducing the current existing issues, followed by our proposition.

3.1 Data Oriented Design

Even if the formats do not have to be strictly identical between linked components, the compatibility has to be ensured at least. There are two main directions in this area: 1. Design data to have a small transfer size and memory footprint 2. Generic data representation, written in standard formats (i.e. JSON or XML)

Ad-hoc feature representations are very popular in early system integration, but since no specification is used, the data becomes very difficult to maintain. An alternative to this process is represented by Google Protocol Buffers (Google, 2012), which formalises all the data messages into a strict syntax which is translated into messages and data types in various programming languages. This approach seems to be secure and flexible enough for the usual data exchange between services and is also very strict with data type inheritance, a concept supported by all the major Object Oriented programming languages.

On the other hand, generic data formats have been formalised in the recent years, due to the increase in popularity of Semantic Web technologies. Due to increasing popularity of establishing strict interchangeable formats which a web service could "understand", several formats have been proposed. World Wide Web Consortium (W3C) is the authority that deals with current and future web standards, including the current web service data formats for the Semantic Web. Whereas the standardisation of several data formats is an ongoing process, this approach seems to be more suitable for large scale web services, such as the Semantic Web, rather than using them for small scale, fast conversational agents.

Google Protocol Buffers (Google, 2012) presents a comparison between their own serialisation format and the XML parsing and conclude that their format is 10 to 100 times faster than XML. MsgPack (Sadayuki, 2012) follows the same direction of small memory footprint data representation, by offering better performance than Google Protocol Buffers, while maintaining the multi-language binding and multi-platform support.

We therefore propose to define our own Object-Oriented data representation. Thanks to object hierarchies, objects are extendible and independent from the data serialisation level. This allows us to change the serialisation level in the future if needed. Currently, the serialisation is done with MsgPack due to the best performance. The data has a small memory footprint than in the case of XML or JSON, has very fast serialization mechanisms and offers the advantages of working with native Object Structures rather than XML trees or JSON maps.

On top of ZeroMQ and using our Object-Oriented design for data exchange, MyBlock is designed to be a middleware for a distributed pipeline processing. It had to be small, flexible and fast enough to support the development of a rich platform, capable of exchanging information in real-time. The main platform has a three level separation, such as in most of the modern architectures, to support an easy management and understanding of the components.

3.2 MyBlock Components

A component is an atomic structure for the MyBlock platform. The component processes a given set of data types and forward the output to the next component in the chain. The internal flow of a component can have two different aspects: either it is a reactive output to the input, or an active component that can produce output based on the internal states without any input. Special components are elements which

only consume (Sink) or produce (Source), without any mixed function.

At this level, data types are an important aspect. The data exchanged need to be compatible between linked elements. A component formally defines preconditions and postconditions in terms of data types, in order to be linked with other components to form complex processing pipes. The communication protocol between two components is a simple publish-subscribe architecture to ease data exchange.

3.3 MyBlock Services

Similarly to the component, we define the services. A service is designed is to respond to requests that can be triggered by any component or other service. The communication protocol used for services is a synchronous request-reply.

3.3.1 Architecture Modelling

On the architecture level, the actual pipeline is constructed by passing the configuration parameters to the components and services, and by building the dynamic links between all the elements. The important features of this level are highlighted by the ability to change dynamically the structure of the processing pipeline, without changing the logic of the components. If the data exchanged between several elements is compatible, the order of processing is not important. Moreover, in comparison with other platforms which use the processing pipeline paradigm, MyBlock does not need special components for data multiplication or joining.

3.4 AgentSlang Components

All the principles enumerated for the MyBlock platform are valid for AgentSlang. We the propose a short presentation of the main components currently implemented in the AgentSlang platform. Most of the basic components are based on existing libraries, such as: Google Speech API (Google, 2013) for Automatic Speech Recognition, Cereproc Voice (CereProc,) for speech synthesis, various Part-of-Speech Taggers (SENNA (Collobert et al., 2011), TreeTagger (Schmid, 1995)) and MARC (Courgeon et al., 2008) as an Embodied virtual Agent. AgentSlang also includes various models for Natural Language Understanding, Dialogue Management and Affective Feedback Detection, as important parts of an IS. Figure 1 displays a summary of the current AgentSlang capabilities.

3.4.1 Natural Language Understanding: Synonym-based Key Word Spotting

This component is based an home-made regular expression language called Syn!bad , which is used to extract information from sentences. Syn!bad is an acronym of *Synonyms [are] not bad*, which suggests that the main concept of Syn!bad is centred among synonym processing. Syn!bad uses some POSIX Regular Expression structures (Alfred, 1990) extended to integrate synonyms. Synonyms are independent structures, grouped in different sets, according to their meaning. The most common grouping currently known is the WordNet synsets (Miller, 1995), which consist of sets of different words according to their semantics and part of speech. Each synset has a unique id to enable an easy retrieval.

A Simplified Description of the Problem. In IS, the knowledge extraction process is usually slowed down by the complexity of the rules describing a certain concept. Using regular expressions associated to variable structure is an alternative. For instance, to match the following sentence: `Bob do you have water`, one can use `<name> do you <verb> <object>`. The variable extraction is already supported by certain implementations of regular expressions. The problem becomes more complex when restrictions are added to the matched variables, especially in the case of `<verb>` and `<object>`. To our knowledge, the syntax of matching only variable structures while having a certain part of speech is not supported by any regular expression implementation.

Syn!bad Example. Based on the rules described above, the following Syn!bad pattern introduces most of the features of the language:

```
$name <#*>? do you <VB*>* [some|RB*]
          [water#object]
```

where: 1. `$name` represents a context free variable, which matches any single word and retrieves it as the *name* variable. 2. `<#*>?` is an optional token that can match any punctuation mark. The *#** represents a generic part of speech group matching punctuation marks. 3. *do* and *you* are words matched by the expression. 4. `<VB*>*` is a none-or-many token matcher, which restricts the element to match only a selected part of speech, in this case a verb. 5. `[some|RB*]` represents a matcher for a synonym of the word *some*. A restriction over the part of speech is added, which matches only adverbs. 6. The `[water#object]` token is similar to the previous one, but matches a synonym

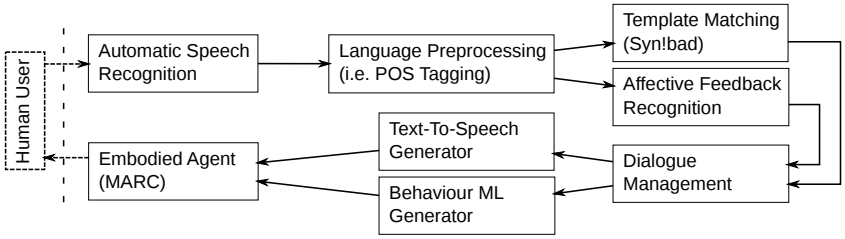

Figure 1: The current status of the AgentSlang platform.

of *water* and recovers the value of the detected word into the *object* variable.

The matching process applied to the sentence: `Bob, do you want any aqua`, produces the following result: $*name* ← *Bob* and #*object* ← *aqua*, while `<#*>?` matches the comma, `<VB*>*` matches the verb *want* and *any* is matched by the token `[any|RB*]`.

3.4.2 Dialogue Manager and Natural Language Generation

Our dialogue model is a simplified Deterministic Finite Automata (DFA), that deals with the agent presentation and state-transition in a narrative environment. This structure includes a series of hand-crafted patterns, collected from our previous experiments in story-telling environments. The component receives a Syn!bad pattern identifier as an input and a series of variables, and generates either a transition to the next narrative state or a response to a question.

The reply action represents a sentence or a command (in case of pointing certain key aspects in to the story) that is interpreted afterwards. The reply action patterns allow variable substitution, similar to the Syn!bad language: `item1 item2 $variable1 item3`, where `item1,item2,item3` are actual words used to generate the reply and `$variable1` is a variable. In the current implementation, since the reply action is mapped directly to the pattern identifier, the variable names are mapped as well to the variables extracted previously. If no such variable can be extracted, the process fails and a default pattern is generated.

3.4.3 Affective Feedback

The Affective Feedback component relies on an extended version of affective information detection based on a fusion of affective contextualised dictionaries (Serban et al., 2013). The component takes as an input a sentence already annotates using a POS Tagger. The valence is computed as a context overlap of the phrase and annotated contextonym (Serban et al., 2013). A contextonym is a word relation structure, similar to the synonyms, which describes the appearance of words in a similar context (a phrase). This structure makes the search of the proper context very easy in comparison to the classic synonymic relations. Finally, the valence of the phrase is given by the average valence of all matched contextonyms.

4 EVALUATION

We benchmarked AgenSlang over main IS platforms, in order to evaluate the performance of future IS based on this infrastructure. The machine used for this test is a I7 Intel machine, at 1.6 GHz per core and 3.9 Gb of RAM. The operating system used for the test is an Ubuntu 12.04 Linux, with Oracle Java 1.7.

The setup is to send a series of random messages, of a given size, from one component to another. We choose to send random sequences in order to prevent the caching speed of ActiveMQ, which applies a set of heuristics in case the same message is sent over the network. Moreover, in order to prevent local traffic peaks, we sent 100 messages and presented only the average time. For the message throughput[4], we represented the number of messages that pass between the two components in one second. Figure 2 represents this measure, side-by-side, on a logarithmic scale.

AgentSlang, as integrating MyBlock, is presented in two versions: with Automatic System Feedback (ASF) and non-ASF. This mechanism enables AgentSlang to send a feedback message each time an action is successfully executed. This is executed in the case of sending and receiving a message. SEMAINE and VHToolkit does not provide a similar mechanism, therefore, in order to achieve a fair comparison of the two systems, this feedback has also been disabled.

The conclusion of this experiment is that AgentSlang is faster than SEMAINE and VHToolkit, when having the ASF disabled. For messages longer than 10,000 characters, the ASF does not increase the

[4]The exact message throughput ratio is measured as the number of messages per second that pass from one component to another.

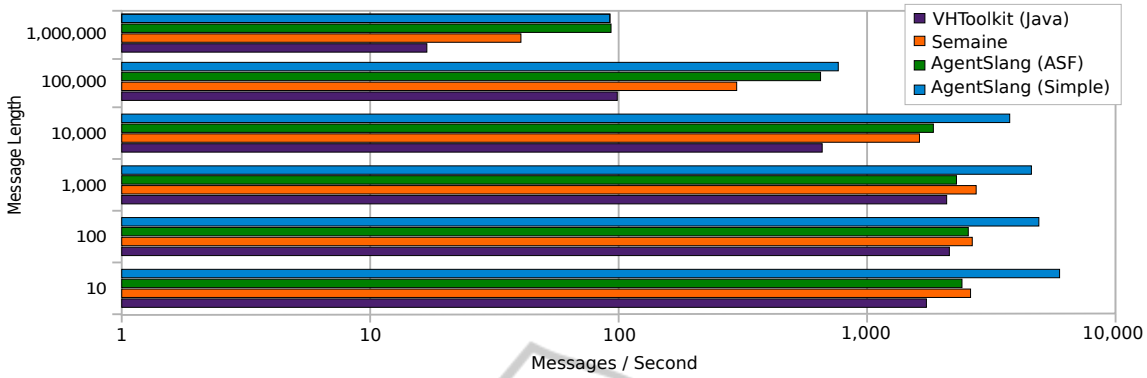

Figure 2: The performance comparison between SEMAINE, VHToolkit and AgentSlang, for message throughput representation. The representation is done on a logarithmic scale.

sending time. Since AgentSlang targets a large spectrum of data types, both scenarios can be used in practical situations. The choice of a platform depends on the application. To achieve the best speed while sending data, AgentSlang simple (non-ASF) is a good choice. To ensure that a message has been successfully processed, AgentSlang ASF is currently the only choice. In conclusion, the two versions of AgentSlang are better that the current implementations of SEMAINE and VHToolkit and the choice of a version depends on the scenario.

The core functions of AgentSlang are the Data Oriented design, Dialogue Management and Affect Oriented design. Only AgentSlang, Semaine and Companions projects propose Affect Oriented features. For Dialogue Management, AgentSlang, Companions, VHToolkit, Mirage and GECA offers it. On the system management aspect, only AgentSlang, VHToolkit and Semaine provides such a tools, whereas real-time system messages that provide informations about the state and status of all the components, are supported only by AgentSlang.

From the Natural Language Understanding and Dialogue Management perspective, our proposition introduces a solution to the natural language variability, by using synonyms, which solve well the input variability issue of spoken language. The dialogue model is a simplified DFA, which is easy to adapt in any situation and proves very efficient in narrative situations. Nevertheless, with proper annotated data, VHToolkit can resolve this task efficiently as well.

Two of the major differences between AgentSlang and other platforms are that we focus on data-oriented design rather than the source and by not using blackboards. We consider that the data source is not important for Distributed Interactive System, since trust among components is guaranteed by the builder of the system. At most, components could provide a level of confidence for certain tasks. Moreover, the blackboard as a fundamental mechanism for knowledge sharing in AI is replaced by the services, modelled by AgentSlang. A service offers similar functionality through the request/reply protocol, with flexibility and robust data synchronisation mechanisms.

5 CONCLUSIONS AND FUTURE WORK

Currently, AgentSlang is the only ECA system that aims at building realistic story-telling environments. We shown in our previous work that such a system needs to be able to deal with feedback from multiple sources: dialogue, affective feedback and narrative actions. We choose to combine narration to classic dialogue management in our approach in order to produce better interaction, measured in an increased satisfaction, especially on children. Finally, from the performance point of view, we have shown that the performance of AgentSlang is better than Semaine and VHToolkit.

Nevertheless, such a complex platform has its limits. First, the narrative states and answers to the user's questions are based on a series of patterns, chose randomly based on matching rules. More complex Natural Language Generation techniques could be applied in order to create non-repetitive and naturalistic content.

Second, the Non-Verbal Behaviour is fixed and attached to each narrative state or answer. In the future, this could be done in a dynamic way, based on the answer, the context of the interaction (i.e. user profile, environment) and the personality of the agent.

Moreover, a multi-modal Feedback Detection Algorithm need to be implemented and tested. Currently our approach is based on text features mainly, but offering a generic platform for Distributed Interactive

System development enables to benchmark several models on a common platform.

REFERENCES

Alfred, V. (1990). Algorithms for finding patterns in strings. *Handbook of Theoretical Computer Science: Algorithms and complexity*, 1:255.

Cavazza, M., de la Camara, R. S., and Turunen, M. (2010). How was your day?: a companion eca. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1629–1630. International Foundation for Autonomous Agents and Multiagent Systems.

CereProc. Cerevoice sdk. http://www.cereproc.com/.

CMLabs (2007). Psyclone. http://www.mindmakers.org/.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Courgeon, M., Martin, J.-C., and Jacquemin, C. (2008). Marc: a multimodal affective and reactive character. In *Proceedings of the 1st Workshop on AFFective Interaction in Natural Environments*.

Eyben, F., Wöllmer, M., and Schuller, B. (2010). Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the international conference on Multimedia*, pages 1459–1462. ACM.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1994). *PVM: Parallel virtual machine: a users\'guide and tutorial for networked parallel computing*. MIT Press.

Google (2012). Protocol buffers. https://code.google.com/p/protobuf/.

Google (2013). http://developer.android.com/reference/android/speech/.

Hartholt, A., Traum, D., Marsella, S. C., Shapiro, A., Stratou, G., Leuski, A., Morency, L.-P., and Gratch, J. (2013). All together now: Introducing the virtual human toolkit. In *International Conference on Intelligent Virtual Humans*, Edinburgh, UK.

Herzog, G., Ndiaye, A., Merten, S., Kirchmann, H., Becker, T., and Poller, P. (2004). Large-scale software integration for spoken language and multimodal dialog systems. *Natural Language Engineering*, 10(3-4):283–305.

Hintjens, P. (2013). *Zeromq: Messaging for Many Applications*. O'Reilly Media.

Huang, H.-H., Cerekovic, A., Tarasenko, K., Levacic, V., Zoric, G., Pandzic, I. S., Nakano, Y., and Nishida, T. (2008). Integrating embodied conversational agent components with a generic framework. *Multiagent and Grid Systems*, 4(4):371–386.

Lee, J. and Marsella, S. C. (2006). Nonverbal behavior generator for embodied conversational agents. In *6th International Conference on Intelligent Virtual Agents*, Marina del Rey, CA.

Leuski, A. and Traum, D. (2011). NPCEditor: a tool for building question-answering characters. In *International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta.

Miller, G. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Mori, M. (1970). The uncanny valley. *Energy*, 7(4):33–35.

Ogan, A., Finkelstein, S., Mayfield, E., D'Adamo, C., Matsuda, N., and Cassell, J. (2012). Oh dear stacy!: social interaction, elaboration, and learning with teachable agents. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 39–48. ACM.

O'Hara, J. (2007). Toward a commodity enterprise middleware. *Queue*, 5(4):48–55.

Pammi, S., Charfuelan, M., and Schröder, M. (2010). Multilingual voice creation toolkit for the mary tts platform. *Proc. LREC. Valettea, Malta: ELRA*.

Poggi, I., Pelachaud, C., Rosis, F., Carofiglio, V., and Carolis, B. (2005). Greta. a believable embodied conversational agent. *Multimodal intelligent information presentation*, pages 3–25.

Sadayuki, F. (2012). Msgpack. http://msgpack.org/.

Scherer, S., Marsella, S. C., Stratou, G., Xu, Y., Morbini, F., Egan, A., Rizzo, A., and Morency, L.-P. (2012). Perception markup language: Towards a standardized representation of perceived nonverbal behaviors. In *The 12th International Conference on Intelligent Virtual Agents (IVA)*, Santa Cruz, CA.

Schmid, H. (1995). Improvements in part-of-speech tagging with an application to german. In *In Proceedings of the ACL SIGDAT-Workshop*. Citeseer.

Schröder, M. (2010). The semaine api: towards a standards-based framework for building emotion-oriented systems. *Advances in Human-Computer Interaction*, 2010:2–2.

Serban, O., Pauchet, A., Rogozan, A., and Pecuchet, J.-P. (2013). Modelling context to solve conflicts in sentiwordnet. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, pages 393–398. IEEE.

Shapiro, A. (2011). Building a character animation system. In *The Fourth International Conference on Motion in Games*, Edinburgh, Scotland.

Snyder, B., Bosanac, D., and Davies, R. (2011). *ActiveMQ in Action*. Manning Publications.

Soleymani, M., Pantic, M., and Pun, T. (2012). Multimodal emotion recognition in response to videos. *Affective Computing, IEEE Transactions on*, 3(2):211–223.

Swartout, W. R., Gratch, J., Jr., R. W. H., Hovy, E. H., Marsella, S., Rickel, J., and Traum, D. R. (2006). Toward virtual humans. *AI Magazine*, 27(2):96–108.

Thórisson, K. R., Benko, H., Abramov, D., Arnold, A., Maskey, S., and Vaseekaran, A. (2004). Constructionist design methodology for interactive intelligences. *AI Magazine*, 25(4):77.

Wahlster, W. (2000). *Verbmobil: foundations of speech-to-speech translation*. Springer verlag.

Wahlster, W. (2006). *SmartKom: Foundations of Multimodal Dialogue Systems (Cognitive Technologies)*. Springer-Verlag New York, Inc.