Designing Proactive Interfaces for Cooperation using Systems of Systems

Gregory Moro Puppi Wanderley, Marie-Hélène Abel, Emerson Cabrera Paraiso

Designing Proactive Interfaces for Cooperation using Systems of Systems

Gregory Moro Puppi Wanderley*, Marie-Hélène Abel* and Emerson Cabrera Paraiso†
*Sorbonne Universités, Université de Technologie de Compiègne,
CNRS, UMR 7253 Heudiasyc, Compiègne, France
Email: {gregory.wanderley, marie-helene.abel}@utc.fr
†Pontifícia Universidade Católica do Paraná
PPGIa - Graduate Program in Informatics, Curitiba, Brazil
Email: paraiso@ppgia.pucpr.br

***Abstract*—Cooperation is a fundamental trait of Systems of Systems (SoS). In this paper we discuss the design and implementation of proactive vocal interfaces with the goal of improving and stimulating cooperation between users and constituent systems. Our approach is based on intelligent personal assistants displaying proactive behavior and capable of interacting vocally. The paper discusses the main challenges for integrating proactive behavior into SoS interfaces, as well as for using them. The proposed approach is generic and can be applied to different domains.**

***Index Terms*—Systems of Systems, Vocal interface, Collaborative work, Multi-agent system**

I. Introduction

Recently, a booming interest in Systems of Systems (SoS) has led to the design of a range of different applications belonging to several domains such as health care, e-commerce, transportation or production (Nielsen et al. [1]).

Formally, a System of Systems can be defined as a set of independent constituent systems cooperating to achieve a common purpose through the synergism between them (Ackoff [2], Nielsen et al. [1]). In an SoS, the constituent systems are owned and operated by independent users. According to Maier [3], five main features characterize an SoS: (i) operational independence of constituent systems; (ii) managerial independence of constituent systems; (iii) geographic distribution; (iv) evolutionary development; and (v) emergent behavior.

Since an SoS is complex and difficult to be designed, one faces some challenges like limitations on the exchange of information and on collaboration, caused for instance by users' competing interests and priorities (Dahmann and Baldwin [4]). This can compromise the expected results of the SoS goal. According to Nielsen et al. [1], there is a need to employ methods and tools that support collaboration in an SoS.

The research reported in this paper aims at improving the collaboration between the users and constituent systems of an SoS. Recently, we have proposed an architecture (Wanderley et al. [5]), named Memory-Broker-Agent (MBA), for simplifying the development of an SoS. MBA relies on several types of agents such as Service Agents (SA) and Personal Assistant (PA) agents. In our approach, the particular skills of a PA are devoted to understanding its master and presenting the information intelligently and in a timely manner. The main goal of such an agent is to reduce the user's cognitive load (Paraiso and Barthès [6]). In this paper, we present the design of a new feature to be added to the PA interface: a Proactive Vocal Interface for improving the collaboration between users and constituent systems of an SoS. Our approach is based on intelligent personal assistant agents, displaying proactive behavior when assisting SoS users, and capable of interacting vocally.

This type of approach can improve the quality of assistance that personal assistants can offer. First, in our approach the PAs act proactively anticipating problems and needs of SoS users for improving cooperation to achieve the expected results. Second, our PAs can provide customized support to their users, by keeping models (ontologies) which contain knowledge about users, systems, or the SoS domain. Moreover, using traditional interface approaches like point-and-click may rapidly become difficult to implement and to maintain in an SoS, due to its evolutionary nature. On the contrary personal assistant agents can provide a flexible interface when using natural language, which could stimulate and better motivate users (Song et al. [7]). In addition, when using vocal interfaces users can keep working while interacting and cooperating with the SoS.

Our main contribution in this paper is to provide a detailed description of our proactive personal assistant agent, designed for interfacing users to an SoS. First, we summarize related work. Then, we present a scenario to provide an overview of the vocal interaction between users and personal assistants. After that, we detail our approach. Next, we apply our approach to a scenario, showing how it works in practice. Finally, we conclude with a discussion on the advantages and disadvantages of our approach and present possible directions for future work.

II. Related Work

There has been much research in designing natural language interfaces for a wide range of applications and domains. However, we could not find any work for doing it for Systems of Systems. Most works in the SoS literature focus mainly on the integration or connection between constituent systems, but do not take into account the user dimension. Let us briefly review some of the work dealing with interfaces.

Jokinen et al. [8] intended to provide reconfigurable interfaces for users of SoS consisting of production systems. The interfaces took into account user privileges and data permissions to allow users to monitor and receive data from the systems, presented in dashboards and charts. The approach relies on a central component, to distribute and provide information to the interfaces. However, the necessary services for providing information are hosted in a single system, and the interfaces do not offer any means for collaboration among the different users of the SoS.

Some researchers have exploited the use of Web portals to let users obtain information from the SoS. Nativi et al. [9]) have designed a Web portal for retrieving information from an SoS involving systems for Earth observation, monitoring and information. The visualization of the information adopted a strategy involving filters and zooming. Mazetti et al. [10] proposed an SoS for a Virtual Research Environment (VRE), mostly focused on monitoring and retrieving information concerning volcanoes. The Web portal designed in this work lets users publish, discover, and access datasets, according to their profile. The interfaces are point-and-click, being composed of several menus, buttons and bars. Such an approach may become difficult when implementing and maintaining an SoS, as the SoS consists of several distributed systems that may change over time. Besides, such interfaces do not favor the exchange of information or the collaboration among users.

The main difference between our approach and what we found in the literature is that we provide a proactive vocal interface based on intelligent personal assistants, for improving the collaboration between users and the systems. Moreover, our approach fits an SoS globally and does not rely on a centralized component, preserving the distributed and independent features of the SoS.

III. Vocal Interface and Proactive Behavior

To better explain our approach, we first present a scenario used to design and build an SoS with our MBA architecture. Then, we present and exemplify the vocal interaction used to interface users and personal assistants in such an SoS.

A. Collaborative Software Development Scenario

During software development, team members use several different systems and tools. At the same time, they need to pay attention to the quality of the code being developed (Wanderley et al. [11]). Because the systems used by team members, including the ones to measure the code quality, are not integrated, it is difficult to use all of them at the same time and share information among systems and participants. Thus, in this scenario we designed and implemented an SoS using the MBA architecture, aiming at improving the quality of code being developed. Among the constituent systems of the SoS, we have a version control system to host and manage different versions of code, a system to analyze the quality of the code, coding systems used by *developers* for coding, and project management systems for *managers*. In such a context, managers and developers are interfaced to the SoS through proactive vocal interfaces.

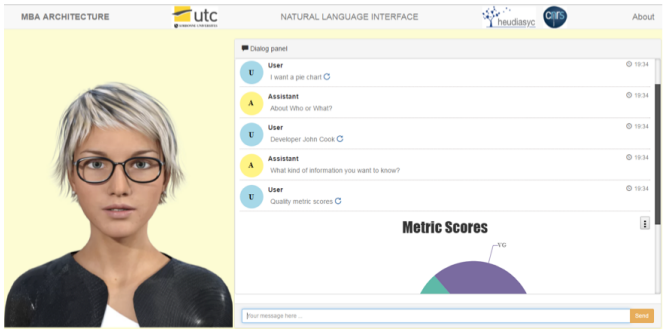

Fig. 1. The MBA Browser interface used for communication between a user and her personal assistant.

B. Example of Vocal Interface and Proactive Behavior

In our SoS team members can interact vocally with their personal assistants, requesting functionalities from other systems for improving the quality of their code through collaboration. Vocal interaction requires a special interface between team members and their personal assistants. Our approach already provides such an interface, named *MBA Browser* (Fig. 1). An MBA Browser has speech-to-text and text-to-speech engines, for translating speech into text and for synthesizing text into voice. Moreover, the interface has dialog panels displaying the text of the vocal interaction, and a dynamic avatar (Bot Libre [12]) that embodies the PA and displays the synthesized text. Besides, as an option, the user can also interact with her PA by typing in text instead of using voice.

To give an example of the vocal interaction and the proactive behavior of PAs, we focus on the role of software *managers*. Table I illustrates the vocal request made by a manager to her PA using the MBA Browser of Fig. 1. The manager asks her PA to show the global code quality of a project. The utterance is captured and treated using the speech-to-text engine of the MBA Browser. Next, the PA asks the manager for additional information, using text-to-speech engine. When results are available, the PA checks the quality of the project. Then, if the quality is not good enough, it acts proactively to find information that can be help the user to improve it. Once the task is completed the PA provides a chart showing the results along with helpful information for the manager.

TABLE I
An example of vocal interaction between a software development manager and her personal assistant.

Interlocutor	Utterance
Manager	Show me the global quality of the project.
Personal Assistant	What's the project name?
Manager	ACE4SD.
Personal Assistant	I have the results, please see the chart.
Personal Assistant	...
Personal Assistant	There are some quality issues. Please, find below information that may be helpful.

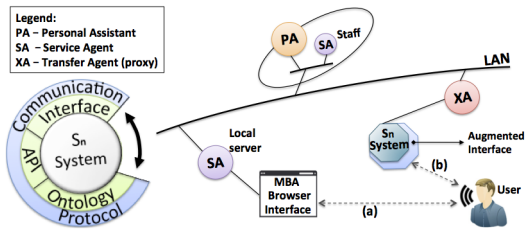

Fig. 2. A proactive vocal interface integrated in a constituent system of the MBA architecture.

IV. Designing Proactive Vocal Interfaces for Systems of Systems

In this section, we present the design of proactive vocal interfaces for Systems of Systems. First, we present how to integrate vocal interfaces into constituent systems, and then we show how to add proactive behaviors to them.

A. Integrating Vocal Interfaces into Constituent Systems

In our approach, proactive vocal interfaces are designed in an "SoS fashion," i.e., they are integrated and kept encapsulated in constituent systems, preserving the distributed and independence features of the SoS. Fig. 2 shows a proactive vocal interface integrated in a given constituent system of an SoS developed using the MBA architecture.

In Fig.2 the user's system is connected to a *proxy agent* which is a transfer agent (XA) that acts as a gateway, taking care of communication with the interface. Here, the user of the constituent system has a Personal Assistant (PA) agent that acts proactively on her behalf. The PA obeys the concept of digital butler proposed by Negroponte [13]. It has a *staff* agent [14] for dealing with more specialized services. In this research, the staff has a model represented by an ontology, consisting of concepts describing the constituent system, the user, and the domain of the SoS. We use the model for supporting the proactive behavior of the PA (details in the next section).

When integrating the interface into the constituent system, the designer can choose between two approaches for the interaction between the PA and the user of the system. The first (see Fig. 2 (a)) is based on a generic browser interface already provided by the MBA architecture, and the second (see Fig. 2 (b)) consists in augmenting the user's system.

The *MBA Browser* interface, detailed in the section III-B, is shown in Fig. 1. The messages exchanged between the user and her PA are handled by a local server that is an SA, named "Local server," also provided by our MBA framework. To integrate the MBA Browser into a constituent system, the designer needs only to connect the agents (PA, staff and Local server) of the browser to the same LAN (Local Area Network) containing the proxy agent of the constituent.

Conversely, the second approach that can be used for integrating the proactive interfaces is the *augmented* one. In this case instead of the browser interface, the designer is required to augment the user's system. For instance, the designer could create a specific window by using the system API, and then adding some modules for the natural language interaction such as text input/output, or if also desired coupling speech-to-text and text-to-speech engines to the window for vocal interaction. After that, the designer needs to connect the interface agents, now only the PA and its staff (see Fig. 2), to the same LAN of the proxy agent of the constituent system. The messages between the PA and the augmented system pass through the proxy agent. Note that the MBA architecture requires that systems have an API to be part of the SoS (Wanderley et al. [5]). Thus, building augmented interfaces is not constrained when using our approach.

After integrating the interface using either one or other approach, the designer is required to developed the content of the dialogs for the interaction between PAs and their users. Usually such dialogs are focused on requesting SoS functionalities (details in section V). Moreover, note that one should decode the interactions in natural language between users and PAs. This is done with the help of the proxy agent ontology (Wanderley et al. [5]) which is an ontology containing concepts describing the system, its user, and the SoS domain. The ontology is built when the requirements of the SoS are defined. It is beyond the scope of this paper, thus we do not detail it.

B. Adding Proactive Behavior to the Vocal Interfaces

In this section we indicate how to add proactive behavior to the Personal Assistant (PA) agents offering the vocal interfaces integrated in constituent systems. In our approach, the PAs display proactive behavior by assisting their users to achieve the expected results of the SoS goal, regarding the SoS functionalities requested through their systems. To add proactive behavior to a PA, a designer needs to follow the activities of a method we call *ProPA* for "Proactive Personal Assistant." The ProPA has two activities, named "Create User Model" and "Act Proactively." For a better comprehension of the method, we give examples using the manager of software development (section III-B) during the descriptions of the ProPA activities.

The "Create User Model" activity intends to create a User Model (UM) to describe the user of the system in which the PA is integrated. Note that this task is related to the domain of the SoS. The activity consists of three steps, named "Describe User," "Define User Scores" and "Define Shared Information."

The "Describe User" receives as input the proxy agent ontology and focus on extending a concept for describing the user of the system. For instance, for the scenario of software development the designer could create a concept for the manager describing her name, address, or email. The output of this step is the ontology updated with the user's concept.

The "Define User Scores" receives the ontology of the previous step, aiming at extending a concept to define scores along with thresholds for SoS functionalities the user can request through her system. The goal is to define scores showing how well the user is performing tasks related to the SoS functionalities. For instance, for the example of the manager requesting analysis for code quality, it could be scores

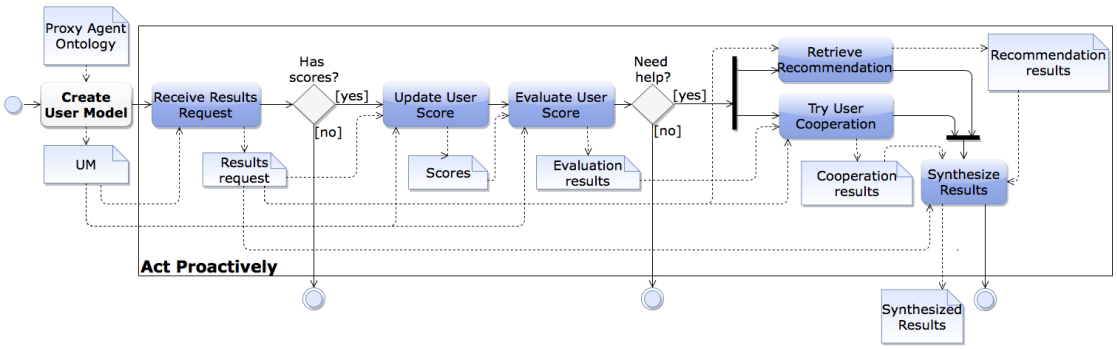

Fig. 3. The steps of the "Act Proactively" activity of the ProPA method for adding proactive behavior to the personal assistants of vocal interfaces.

and thresholds related to quality metrics (Wanderley et al. [11]) measuring the quality of the code of her projects. The output of this step is the ontology updated with the user's scores.

The "Define Shared Information" receives the ontology from the previous step, aiming to extend a concept describing shared information related to the user for supporting SoS cooperation and the expected results of its goal. For instance, for the software manager example the shared information could be her contact to keep in touch with other users, or the scores of her projects for supporting the code quality improvement. The output of this step and also of the"Create User Model" activity is the UM which is kept within the staff agent of the PA.

The "Act Proactively" activity of the ProPA method aims to let the PA acting proactively based on the UM that was created. The activity consists of the steps: "Results User Request," "Update User Score," "Evaluate User Score," "Retrieve Recommendation," "Try User Cooperation" and "Synthesize Results," as shown in Fig. 3.

In the "Receive Results Request," the PA receives the results for an SoS functionality requested by its user. Then, it verifies through the UM if there are user's scores associated with such a functionality. In case negative, the PA steps out of the method and sends the results to her user. Conversely, if there is a score, then the PA performs the step "Update User Scores" taking the UM along with the results for the request, and then updating the user's scores. The output of this step are the new scores.

Next, in the "Evaluate User Scores" the goal is to evaluate the user's scores outputted in the previous step. To do that, the PA takes the scores and apply the thresholds associated with them. If the results indicate that the user is going well with the tasks related to the functionality of such scores, then it steps out of the method sending the results of the request to its user. Conversely, if the evaluation indicates that the user needs further assistance, then the PA can execute in parallel (but not obliged to) the steps "Retrieve Recommendation" and "Try User Cooperation" which receive as input the results for the request performed and the results for the score evaluation.

In the "Retrieve Recommendation," the PA tries to fetch recommendations from the SoS memories for helping its user to improve her tasks for achieving the SoS goal. The recommendations could be found, for instance, through keywords describing the tasks. For the code quality context of the manager, the recommendations fetched could be, for instance, texts like "Reduce the depth of inheritance tree of classes," or links to external pages providing useful information for improving the code quality. The output of this step are the recommendations fetched.

In the step "Try User Cooperation" the PA aims to find useful information for supporting its user by cooperating with other PAs. To do that, it exchanges messages with the other PAs trying to establish Contract Nets through Work Orders, i.e., the protocol of the MBA architecture (details in Wanderley et al. [5]). The information from the cooperation among PAs is retrieved from the shared information of the UMs of their users. Note that, the information coming from other SoS users could create an intrinsic cooperation among them. For the example of the software manager, the information received by her PA could be the scores of the developers with the worst code quality. Such an information could be useful for letting her support and cooperate with such developers to improve their code quality. The output of this step are the information received from the cooperation among PAs.

In the last step of the "Act Proactively" activity and the ProPA, named "Synthesize Results," the PA synthesizes the results from the "Receive Results Request," the "Retrieve Recommendation" and "Try User Cooperation" steps. After that, it sends the synthesis to its user.

V. Applying Proactive Interfaces

In this section, we apply our approach using the SoS built for the scenario of collaborative software development introduced in section III. First, we detail the implementation of our approach. Then, shown the results of the interactions for the request made by the software manager in the example of section III-B.

A. Implementation

To test our proactive interfaces, we implemented both the MBA Browser and the augmented approach (section IV-A). For the MBA browser we used HTML5 and JavaScript. The vocal interface of the browser was handled by the Google Web Speech API[1]. Conversely, we augmented the coding system

[1]https://www.google.com/intl/en/chrome/demos/speech.html

used by developers which was Eclipse[2] by implementing a plugin with Java. Then, we connected to the plugin a simple application in C# used for handling the vocal interface through the Microsoft System.Speech API[3]. All agents were implemented in Common Lisp through the OMAS platform (Barthès [14]). For other systems of our scenario, the versioning system was GitHub[4]. The analysis system was a multi-agent system applying rules based on quality metrics (Wanderley et al. [11]) to compute the quality of the produced Java code.

The vocal interaction between a user and her Personal Assistant (PA) was made by task selection. The reason was because usually in an SoS the interactions between systems and users involve mainly performing requests for functionalities and then transferring back results. In our approach when a user makes a request, her PA tries to select a possible task from a library of tasks. The library consists of tasks related to the functionalities the system is able to request. Once the PA finds a task, a dialog associated with such a task is triggered in order to acquire missing information necessary to execute the task. When enough information is gathered, the PA sends the request associated with the task to find providers in the SoS. Then, when providers are found they perform the task and send the results to the PA that in turn presents them, using voice or text, to the user. In this research, we implemented our task dialogs through the OMAS platform.

B. Interactions for the Software Development Scenario

For matters of space, to show the results from vocal requests and the proactive behaviors in our approach, we focus on detailing the interactions for the request made by the software manager in the example of section III-B. The sequence diagram of Fig. 4 shows the interactions for the vocal request of project quality made by the manager. After, translating the utterance using speech-to-text the MBA Browser sent it to the PA of manager ("PA-Mgr"). Then, the PA sent a work order (the MBA protocol) for retrieving the code of the project, based on its name, from the Version Control System. Once the code was retrieved, the PA sent a work order requesting a quality analysis for it. The Code Quality Analyzer received it, performed the analysis and sent the results for the PA. After receiving the results, the PA requested its staff agent for retrieving the User Model (UM) of the manager. Based on the UM it checked if the functionality for analyzing the code quality has scores associated with it. For the manager we defined a score, based on the code quality of the project determined by the analysis system, and on the current date. For instance, if the analysis system had determined that the project quality was about 65% and the current date was 28/07/2017, then the new score would be as follows in JSON standard format: {"score" : {"project-quality" : 0.65, "date" : "28/07/2017"}}

Then, the PA asked the staff to update the UM based on the new score. After that, the personal assistant sent a request to its staff to evaluate the score. In this case, the staff agent

[2]https://www.eclipse.org

[3]https://msdn.microsoft.com/en-us/library/office/hh361625

[4]https://github.com

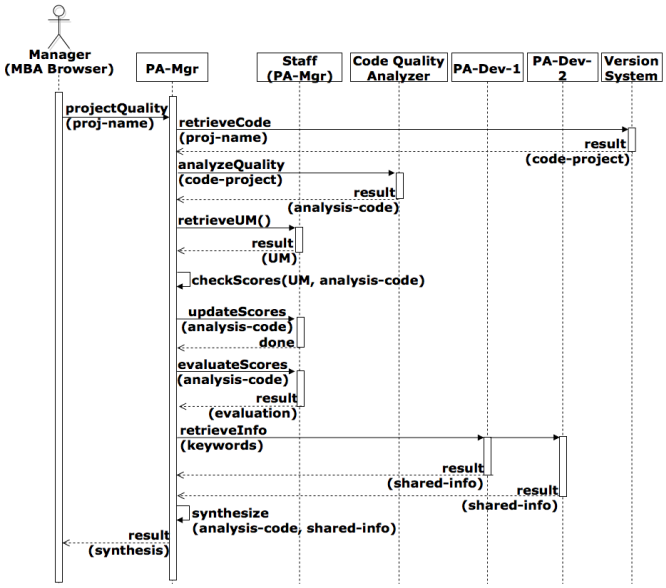

Fig. 4. The interactions resulted from the proactive behavior of the personal assistant of the software development manager.

applied a threshold based on project milestones for evaluating the scores. That is, in the interface we built, the staff had a list consisting of several dates indicating milestones for the manager's project. If the score had a quality below 70% and the current date was less than two weeks for the next milestone such as 02/08/2017, both defined empirically, then the PA started to act proactively on behalf of the manager. For that, the PA sent a work order for trying to retrieve helpful information for the manager, containing keywords that indicated information required such as "code" and "quality. The work order established a Contract Net with the other SoS PAs (in the case of developers, "PA-Dev-1" and "PA-Dev-2"), aiming to gather shared information related to the keywords. In this case the shared information was quality scores, base on software metrics like Cyclomatic Complexity, indicating how well each developer was coding. After receiving the information the PA of the manager selected the worst scores, synthesized the information with the project quality analysis, and then displayed it to the manager, warning her through voice.

VI. Discussion

Implementing and testing the MBA Browser and the augmented approach for the proactive interfaces has allowed us to learn important lessons. One of the main advantages of using the MBA Browser as interface is that it is generic and can be reused by different systems and SoS domains. To change between domains, the designer only needs to develop the content of the dialogs for the interaction between a PA and its user. However, a disadvantage can be the switching among different different windows, i.e., system and browser, though when interacting vocally, setting up a new window may not be necessary and could be discarded. Conversely, the augmented interface is tailored and specialized to a given system.

Moreover, besides augmenting a system with a new window, if vocal interaction is desired, the designer is also required to link the engines for the speech interface. Nevertheless, such an approach may require more effort and time to set up, it allows users to stay in their (constituent-) system environment, avoiding switching among different windows.

Another relevant point concerns the vocal interaction. Although it facilitates the communication between users and the SoS, it has some requirements involving the speech recognition and synthesis. In the SoS implemented for the collaborative software development scenario, we tested two approaches for speech-to-text and text-to-speech modules, named here as "server-based" and "local-based." The "server-based" approach was implemented using Google Web Speech API which performs the recognition and translation in servers distributed over the Web. This approach has performed very well and with good quality for the voice recognition and the text-to-speech. The main drawback is that it requires an Internet connection. The second approach we implemented was "local," using the local modules of the Microsoft Speech API for speech-to-text and text-to-speech. Such an approach has the advantage of being used locally without requiring Internet connection. However, to obtain a good recognition it needs that each user train the speech-to-text module, or we need to adopt a constrained grammar.

VII. Conclusion and Future Work

In this paper we presented the design of proactive vocal interfaces for Systems of Systems, aiming at improving cooperation when using them. Our approach has some advantages, for instance, each user has her own personal assistant agent tailored to her and her system, according to the SoS requirements and domain. The PA acts proactively and cooperates in the SoS providing customized support to its user, in order to achieve the expected results of its goal. Moreover, in our approach personal assistants are activated by voice, thus allowing users to keep working while interacting and cooperating with the SoS.

We can summarize the contributions of this research in: (i) an original and generic method for adding proactive behavior to SoS user interfaces, meaning that to change between domains the designer needs only to customize User Models according to the user of a system; (ii) a generic vocal interface ready to be integrated to SoS systems, i.e., the MBA Browser; and (iii) detailed insights into integrating our proactive interfaces, and challenges faced by different approaches.

With all that in mind, perhaps one may still think what are the differences between our personal assistants and the embodied agents used in other fields like virtual reality. First, the Personal Assistant agents of our approach are digital butlers focusing on assisting human beings, i.e., the SoS users. Moreover, the PAs are cognitive and use their knowledge (UM) for providing personalized support to their users. Furthermore, in our approach the natural language interaction between SoS users and PAs is much more simple, as in SoS usually the interactions involve requests for SoS functionalities for achieving the expected results of the SoS goal. Conversely, usually embodied agents of virtual reality take a more social and psychological perspective, approaching aspects such as emotions and trust (Callebert et al. [15]). Furthermore, the conversations also tend to be longer in order to exploit such an aspects.

In the future, we plan to improve the presentation policy of personal assistants, for instance, to not interrupt users that are busy. We are also interested, to work with more languages besides English, as well as apply and test the approach in other domains.

VIII. Acknowledgment

Gregory Moro Puppi Wanderley would like to thank CNPq-Brazil (grant 233137/2014-9) for its support in this research.

References

[1] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of systems engineering: basic concepts, model-based techniques, and research directions," *ACM Computing Surveys (CSUR)*, vol. 48, no. 2, p. 18, 2015.

[2] R. L. Ackoff, "Towards a system of systems concepts," *Management science*, vol. 17, no. 11, pp. 661–671, 1971.

[3] M. W. Maier, "Architecting principles for systems-of-systems," in *INCOSE International Symposium*, vol. 6, no. 1. Wiley Online Library, 1996, pp. 565–573.

[4] J. S. Dahmann and K. J. Baldwin, "Understanding the current state of us defense systems of systems and the implications for systems engineering," in *Systems Conference, 2008 2nd Annual IEEE*. IEEE, 2008, pp. 1–7.

[5] G. M. P. Wanderley, M.-H. Abel, J.-P. Barthès, and E. C. Paraiso, "A core architecture for developing systems of systems," in *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 141–146.

[6] E. C. Paraiso and J.-P. A. Barthès, "An intelligent speech interface for personal assistants in r&d projects," in *Computer Supported Cooperative Work in Design (CSCWD), 2005 IEEE 9th International Conference on*. IEEE, 2005, pp. 804–809.

[7] D. Song, E. Y. Oh, and M. Rice, "Interacting with a conversational agent system for educational purposes in online courses," in *Human System Interactions (HSI), 2017 10th International Conference on*. IEEE, 2017, pp. 78–82.

[8] J. Jokinen, M. B. Ambat, and J. L. M. Lastra, "Condition monitoring for distributed systems with reconfigurable user interfaces and data permissions," in *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*. IEEE, 2016, pp. 5705–5710.

[9] S. Nativi, P. Mazzetti, M. Santoro, F. Papeschi, M. Craglia, and O. Ochiai, "Big data challenges in building the global earth observation system of systems," *Environmental Modelling & Software*, vol. 68, pp. 1–26, 2015.

[10] P. Mazzetti, G. Puglisi, L. DAuria, R. Roncella, D. Reitano, R. Merenda, and S. Nativi, "The med-suv virtual research environment for enabling the geo geohazard supersites in italy," *Earth Science Informatics*, pp. 1–13, 2017.

[11] G. M. P. Wanderley, M.-H. Abel, J.-P. Barthès, and E. C. Paraiso, "An advanced collaborative environment for software development," in *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 002 917–002 922.

[12] Botlibre, "Bot Libre," https://www.botlibre.com, 2017, online; accessed 2 November 2017.

[13] N. Negroponte, "Agents: From direct manipulation to delegation," in *Software agents*. MIT Press, 1997, pp. 57–66.

[14] J.-P. A. Barthès, "Omas a flexible multi-agent environment for cscwd," *Future Generation Computer Systems*, vol. 27, no. 1, pp. 78–87, 2011.

[15] L. Callebert, D. Lourdeaux, and J.-P. Barthès, "Trust-based decision-making system for action selection by autonomous agents," in *Computer Supported Cooperative Work in Design (CSCWD), 2016 IEEE 20th International Conference on*. IEEE, 2016, pp. 4–9.