GAMBAD: A Method for Developing Systems of Systems

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

GAMBAD: A Method for Developing Systems of Systems

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

***Abstract*—Despite the great number of Systems of Systems (SoS) being developed, building them still remains hard and difficult. Currently, there is a lack of methods capable of supporting architects for building an actual SoS. In this paper we introduce an original method called GAMBAD for developing an SoS from a practical point of view. Our method guides the development of SoS on top of a multi-agent layer supported by ontologies. We tested GAMBAD by building an SoS in the domain of Health Care. Early results show that by using our method, architects can develop an SoS faster and more accurately.**

***Index Terms*—Systems of Systems, Method, Multi-Agent System**

I. Introduction

Today, complex collaborative applications require to let several systems that have been developed independently work together, leading to the concept of Systems of Systems (SoS). According to Maier [1], 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.

Nowadays, building an SoS has become quite popular and demanding, leading to a number of collaborative applications developed in several distinct domains aiming inter alia to support military activities (Olivier et al. [2]), or to improve understanding of medication prescriptions (Wanderley et al. [3]). However, developing an SoS still remains a hard and difficult task facing a number of challenges, most of them resulting from the SoS features (Nielsen et al. [4]). Although different frameworks and architectures have been proposed to support the development of SoS, no method for guiding architects has been provided, that would offer a clear and logical path for building an SoS from a practical point of view.

In this research, we aim at facilitating the development of SoS. Recently, we developed the MBA framework (Wanderley et al. [5]) that is based on a Memory-Broker-Agent architecture (Wanderley et al. [6]) to simplify the process of building Systems of Systems in practice. Our MBA framework builds SoS on the top of a multi-agent layer supported by ontologies. The result of our experience in using our framework for building an SoS in the domain of collaborative software development allowed us to define key points, gathered to form general guidelines for building MBA SoS. We then used such guidelines for developing an original method called *GAMBAD* (Guide for Actual MBA Development) to support architects for developing SoS faster and more accurately from a practical point of view.

In this paper, we introduce the GAMBAD method. First, we survey related work. Then, we present the MBA framework describing its main concepts and elements. After that, we introduce the GAMBAD method. Next, we provide a discussion comparing two SoS we built, one without using our method and the other one using it. Moreover, because our method is used to develop an SoS through the MBA framework that in turn uses multi-agent systems, we also compare GAMBAD with the well-known Gaia methodology (Zambonelli et al. [7]) for developing multi-agent systems. Finally, we end up with conclusions and future work.

II. Related Work

There has been much research aiming at supporting the development of Systems of Systems (SoS). A number of approaches have proposed architectural support and frameworks for building SoS.

Several authors in the literature take a more conceptual view when providing architectural support for SoS. Some propose languages to formally describe the architecture of an SoS (Oquendo [8]), or techniques to optimize and provide decision-making about SoS architectures (Agarwal et al. [9]); others propose architectural patterns (Ingram et al. [10]). Some authors like (Ge et al. [11]) use well-known frameworks like DoDAF, originally developed for the military domain, to support their approaches. However, such frameworks represent the architecture statically and focus excessively on what should be described rather than on concrete problems (Hu et al. [12]). All approaches are mostly done on a conceptual level, i.e., describing and documenting the architectures, rather than giving a practical point of view for supporting the SoS development.

Other authors however, like Varga et al. [13] provide a framework and architectural support from a more practical point of view. Varga et al. [13] implemented a framework for

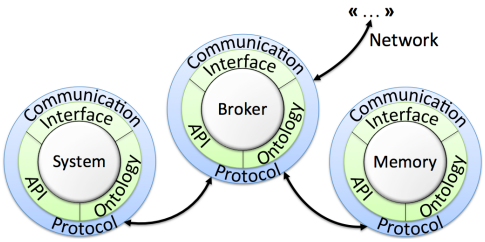

Fig. 1. A minimal example of the MBA architecture.

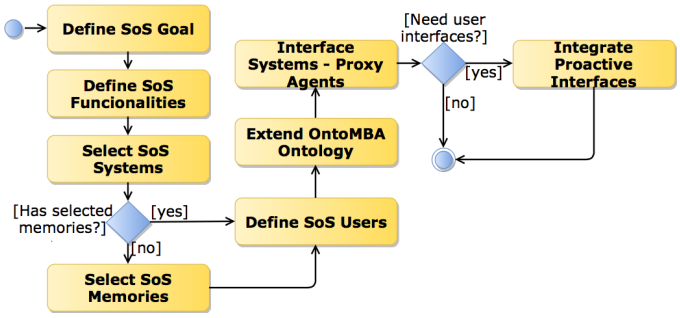

Fig. 2. The GAMBAD method which provides general guidelines for developing MBA SoS.

developing and deploying an SoS. The approach was based on a Service-oriented architecture (SOA).

However, none of the works offers a method for supporting SoS architects with an accurate and clear sequence of steps for developing Systems of Systems in practice. To the best of our knowledge, the approach we introduce in this paper is the first one offering such a method focusing on building an SoS in practice.

III. A Method for Developing Systems of Systems

In this section we present the proposed method for developing Systems of Systems. First, we present the main concepts and elements of the MBA framework then introduce our method.

A. The MBA Framework

We developed the MBA, Memory-Broker-Agent, framework (Wanderley et al. [5]) to simplify the process of building Systems of Systems (SoS). We consider that an SoS can be organized as a set of independent components using a standard protocol for exchanging information through the use of real or virtual brokers. The component systems can be upgraded as agents, users can have a Personal Assistant (PA) and memory can be used for knowledge management. The approach is P2P, making the resulting SoS both robust and extensible.

The MBA architecture is a domain-independent core architecture, meaning that it is extended according to the domain, goals and systems of the SoS being developed. The approach consists in building systems of systems on top of a multi-agent layer (MAS) that will provide the needed mechanisms for handling the interoperability issues both syntactic and semantic. The main idea is to render the component systems interoperable by using proxy agents, one for each component system. Then, we link the agents to brokers (Park et al. [14]) that will take care of organizing and transmitting (Gardikiotis et al. [15]) exchanges using a standard protocol. The MAS platform also offers the possibility of adding Personal Assistant agents to interface users, and agents in charge of recording information necessary for managing knowledge. Fig. 1 shows a minimal example of an MBA SoS comprising a system, a broker and a memory component.

The system element represents any type of independent system. After being interfaced through a proxy agent, the system will be augmented by a communication protocol, ontologies, and optional interfaces. The communication protocol of the MBA framework, called *Work Order* (details in Wanderley et al. [6]), provides syntactic and semantic interoperability between constituent systems of an MBA SoS. Ontologies play a crucial role, modeling the SoS domain and the users of constituent systems, handling the semantics of the communication protocol, and decoding interactions in natural language. The optional interfaces (detailed in Wanderley et al. [16]) are proactive interfaces in natural language handled by Personal Assistant agents, interfacing users to the SoS.

The broker element is intended to receive requests from component systems. It tries to find potential providers, and then once the tasks are allocated, transfers the results back to the callers. The broker provides loose coupling between systems making the MBA architecture more robust. Brokers use a Contract-Net protocol (Smith [17]).

The memory element is there to capitalize and manage knowledge concerning the SoS and its domain. In our approach, it is possible to have as many memories as needed or required by a given domain, including redundant elements. Memories are usually external systems. The MBA architecture provides a generic interface for memories through a proxy agent with an *Abstract Statement* for handling four basic memory operations, named *insert*, *update*, *remove*, and *select*. The advantage of doing this, is that our approach facilitates the work of SoS architects, requiring them only to customize such a proxy agent according to the SoS being developed.

B. The GAMBAD Method

In this section we provide general guidelines for developing an MBA SoS. We present the key points of our guidelines through the activities of a method we call *GAMBAD* (Guide for Actual MBA Development). Fig. 2 shows the GAMBAD method consisting of the following activities:

1) Define SoS Goal
2) Define SoS Functionalities
3) Select SoS Systems
4) Select SoS Memories
5) Define SoS Users
6) Extend the OntoMBA Ontology
7) Interface Systems - Proxy Agents
8) Integrate Proactive Interfaces

The first key point we identified for developing an MBA SoS is "**Define SoS Goal**." This point is straightforward, meaning that the SoS architect needs to define the goal that the SoS intends to achieve.

After defining the SoS goal, the second point is "**Define SoS Functionalities**." In this key point, the architect defines the SoS functionalities, i.e., the inter-system functionalities constituents of the SoS can request and provide to each other, needed to achieve the SoS goal.

Next, the "**Select SoS Systems**" intends to effectively select the systems offering the SoS functionalities in practice.

The next activity is "**Select SoS Memories**." If memories are not among the selected systems, the architect is now invited to do that for providing means for knowledge capitalization and management during SoS operation.

The "**Define SoS Users**" defines the SoS users and their roles when interacting and using the SoS. Usually, such users operate constituent systems of the SoS. However, they could also be users of the SoS domain operating external systems, for instance, interfaces allowing them to request and receive SoS information.

After that, in the "**Extend OntoMBA Ontology**" the architect takes the OntoMBA core ontology (Fig. 3), and extends it according to the SoS being developed. Defining such an ontology has several roles such as modeling the SoS domain and the users of constituent systems, handling the semantics of the communication protocol, and decoding interactions in natural language. The goal of the OntoMBA is to be used as a point of departure by SoS architects when developing the ontologies of an MBA SoS. The extended ontology is the *global ontology* of the SoS.

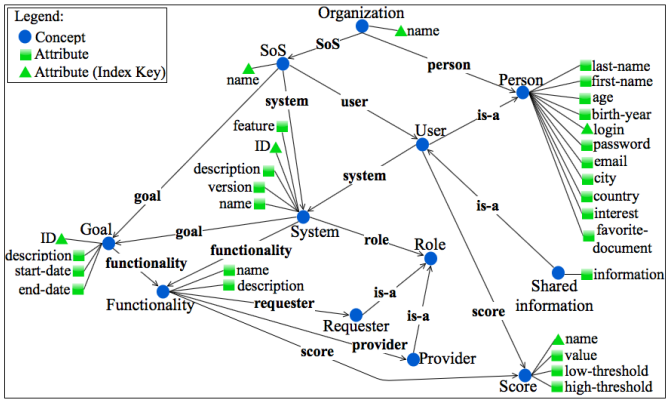

Fig. 3. The OntoMBA core ontology.

The next key point is "**Interface Systems - Proxy Agents**." In this point, the architect interfaces the systems selected in "Select SoS Systems" and "Select SoS Memories," with proxy agents. The goal is to allow constituent systems to exchange information and cooperate to achieve the SoS goal. For interfacing the systems, we recommend the architect to follow the steps shown in Fig. 4, namely:

- "Prepare System" to prepare the system (if needed). For instance, it could be setting up tables in a relational database.
- "Connect System - Proxy Agent" to connect systems that are not memories with proxy agents. For instance, depending on the API provided by a given system, network standards can be used to connect it with its proxy agent.
- "Connect Memory - Generic Interface" to connect memories with proxy agents focused on a generic memory interface (see Section III-A).
- "Derive Proxy Agent Ontology" to derive an ontology from the global SoS ontology for each proxy agent. The goal of the proxy agent ontologies is to handle the semantics of the work order, i.e., the MBA protocol (detailed in Wanderley et al. [6]), and also to decode interactions in natural language when optional user interfaces (see Section III-A) are used. The proxy agent ontologies focus on the idiosyncrasies of the systems they interface.
- "Store Global Ontology" to store the global ontology in the proxy agents of memories. The goal is to keep the global ontology within the SoS memories.
- "Define Skills" to define skills in the proxy agents for handling the SoS functionalities requested and provided by the systems connected to them. Such skills usually contain work orders for requesting the SoS functionalities.

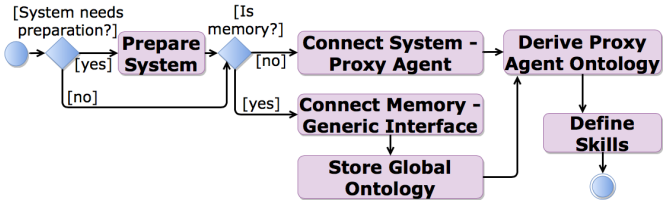

Fig. 4. The steps of the "Interface Systems - Proxy Agents" activity of the GAMBAD method.

If the SoS users defined in the "Define SoS Users" do not require interfaces for interacting with the SoS, then the architect can skip the next activity and the GAMBAD is finished. Otherwise, the "**Integrate Proactive Interfaces**" integrates proactive interfaces into the systems of SoS users for supporting them, for instance, to request or receive information from the SoS.

The proactive interfaces allow interaction in natural language between SoS users and Personal Assistant (PA) agents. Moreover, users can receive proactive and customized support from their PAs. In such an approach, PAs follow a *digital butler* approach (Negroponte [18]), and may have staff agents in charge of more specialized services like keeping User Models used for customized support. Currently, the MBA framework provides two approaches for integrating proactive interfaces, namely: (i) *MBA Browser* which is a system already provided by the framework; and (ii) *Augmented* which means that a given constituent system of an SoS is augmented, for instance, with a window for allowing the interaction between users and Personal Assistants. When integrating the interfaces,

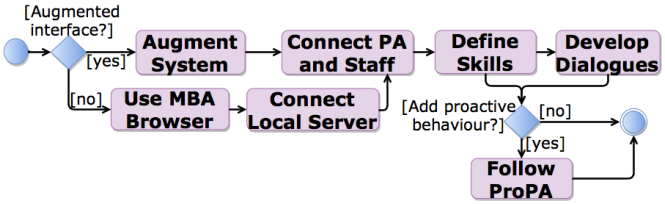

Fig. 5. The steps of the "Integrate Proactive Interfaces" of the GAMBAD method.

we recommend to follow the steps shown in Fig. 5, which are:

- "Augment System" to augment the user's system for creating a Personal Assistant (PA) interface, if chosen.
- "Use MBA Browser" to use MBA Browsers as the PA interface, if chosen.
- "Connect Local Server" to connect a local server handled by an agent (for MBA Browsers, if chosen) to the same LAN (Local Area Network) of the proxy agent interfacing the user's constituent system. In the MBA approach, the server is provided as part of the Browser system.
- "Connect PA and Staff" to connect a Personal Assistant and staff agents to the proactive interface. The goal in this step is to connect a PA and its staff agents to the same LAN of the proxy agent interfacing the user's constituent system.
- "Define Skills" to define skills in the PA for handling the message exchanged with its user, for instance, possible requests and results for SoS functionalities related to the system to which the PA is integrated.
- "Develop Dialogues" to develop the PA dialogues for natural language interaction with the user of the system. Usually, in an SoS the dialogues between the PA and the user are related to the SoS functionalities the system can request or provide (Wanderley et al. [16]).
- "Follow ProPA" to follow the *ProPA* (Proactive Personal Assistant) method (Wanderley et al. [16]) which is a method for adding proactive behavior to the interfaces, if desired.

After performing the GAMBAD method, the output is an SoS built in practice and supported by the MBA architecture.

IV. Discussion

In this section we discuss the advantages and the disadvantages of our GAMBAD method. First, we compare two SoS we built, one without and the other using our method. After that, we examine GAMBAD in contrast to the Gaia methodology.

A. Global Discussion

The guidelines provided by the GAMBAD method were the result of lessons learned from the development of an SoS in the domain of collaborative software development (Wanderley et al. [5]). This means that such an SoS was built through the MBA framework, but without using a clear sequence of steps that are specified by GAMBAD.

To test and show the results of using the guidelines of GAMBAD, we have used them for building a brand new MBA SoS in the domain of Health Care (Wanderley et al. [3]). Through such a new experience we have drawn a number of conclusions.

Comparing to the software development SoS that was built without the method, in the Health Care SoS we could have been much more accurate when performing the tasks required for the development. For instance, because we did not have a clear sequence of steps to follow in the first SoS, we defined the SoS global ontology at the wrong time, and thus we were obliged to modify and update it many times during the SoS development. On the other hand, for the second SoS (Health Care) we did not experience such a problem, actually we were careful to define the ontology at the correct time, which allowed us to avoid redoing unnecessary tasks and wasting time.

Moreover, when using GAMBAD we were much more focused on the correct tasks at all times. For example, during the development of the first SoS we tried to integrate proactive interfaces at the same time we were still interfacing systems with proxy agents. This was not a good idea because we found proactive interfaces may rely on the skills of proxy agents for sending and receiving work orders, or derive their ontologies, i.e., User Models, from the proxy agents one. For the Health Care SoS we did not suffer from such a problem, as the guidelines prescribed to interface proxy agents before integrating proactive interfaces.

Furthermore, the expansions of the guideline key points "Interface Systems - Proxy agents" (Fig. 4) and "Integrate Proactive Interfaces" (Fig. 5) were very helpful when developing the second SoS. Because the two key points are complex, the fact of providing expansions showing their steps in sequence has kept us on the right track focusing on the right tasks to do.

However, the method we propose is specific for developing SoS using our MBA framework, which imposes some restrictions, although the MBA framework is generic, meaning that it can be used for building SoS independent of the target domain. By building the Health Care SoS we also tested and validated the generic feature of our framework. For adapting the architecture of our framework from the collaborative software development to the Health Care domain we only needed to: (i) adapt ontologies; (ii) interface systems with proxy agents; and (iii) integrate proactive interfaces, as an option.

B. Comparison with the Gaia Methodology

Because the GAMBAD method is used to develop SoS obeying the MBA framework that in turn has a key component that are multi-agent systems (MAS), we briefly compare GAMBAD with the well-known Gaia methodology (Zambonelli et al. [7]) for designing MAS, referring to the revised version of the Gaia methodology described by Zambonelli et al.

In the Gaia methodology, the authors take the perspective of using MAS as a software engineering paradigm, i.e., an agent-oriented approach to software development for the analysis and

design of multi-agent systems. Gaia provides guidelines for building an MAS through a process of organizational design. The method begins after detailed requirements of the MAS system. Gaia has two main phases: (i) analysis phase to collect and organize the specifications of the agent environment, roles and interactions; and (ii) a design phase that derives agents, services, and acquaintance models describing the high-level responsibilities of the organization. In both phases, the architect describes and specifies a set of core abstractions provided by the methodology.

Compared to GAMBAD, the Gaia methodology is situated on a more abstract level, providing descriptions and models of the components of the given MAS being designed. For instance, the interaction between agents is described through a very abstract model consisting of fields like inputs, outputs or initiator. Moreover, Gaia does not provide models or any specific support for inserting human users. Furthermore, according to the authors, the output of Gaia is a set of agent classes to be implemented and instantiated. Conversely, the GAMBAD method focuses on a more concrete and practical level. For instance, when building an SoS using GAMBAD, an architect connects proxy agents with prospective systems in practice. Also, when using our method the architect selects the prospective SoS systems concretely. Moreover, the output of the GAMBAD method is an actual SoS following the MBA architecture. One of the results is that the interactions between the components systems of an SoS are performed through a well-defined protocol (work order), which provides syntactic and semantic message exchanges. Moreover, the GAMBAD method handles the integration of proactive interfaces for supporting SoS users.

However, one could still ask what are the differences between building an MAS system with Gaia, and an SoS with GAMBAD as our approach develops the SoS on top of an MAS layer. We can say that one of the most substantial differences is that in a pure MAS (like the ones built with Gaia), the agents are not capable of working in complete isolation, i.e., they are part of an overall system and depend on the cooperation with the other MAS agents, meaning that such an approach does not fit the Maier's operational independence (Section I) feature of an SoS, which is handled by GAMBAD by interfacing true independent systems with MBA proxy agents.

V. Conclusion and Future Work

In this paper we introduced an original method called GAMBAD for developing Systems of Systems. With GAMBAD we aimed at supporting SoS architects with a clear and accurate logical sequence of steps for building SoS from a practical point of view. The GAMBAD method provides a path for building SoS on top of a multi-agent layer, by using our MBA (Memory-Broker-Agent) framework. Our results showed that developing an SoS with the GAMBAD is much more accurate, preventing architects from redoing tasks, thus saving time.

Currently, we are studying possible refinements in our method to better support SoS architects, for instance, by reducing the steps of the expansions (interfacing systems with proxy agents, and integrating proactive interfaces).

Acknowledgment

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

References

[1] M. W. Maier, "Architecting principles for systems-of-systems," *INCOSE International Symposium*, vol. 6, no. 1, pp. 565–573, 1996. [Online]. Available: http://dx.doi.org/10.1002/j.2334-5837.1996.tb02054.x

[2] J. P. Olivier, S. Balestrini-Robinson, and S. Briceno, "Approach to capability-based system-of-systems framework in support of naval ship design," in *Systems Conference (SysCon), 2014 8th Annual IEEE*. IEEE, 2014, pp. 388–395.

[3] G. M. P. Wanderley, É. Vandenbergh, M.-H. Abel, J.-P. A. Barthès, M. Hainselin, H. Mouras, A. Lenglet, M. Tir, and L. Heurley, "CONSIGNELA: A multidisciplinary patient-centered project to improve drug prescription comprehension and execution in elderly people and parkinsonian patients," *Telematics and Informatics*, 2017.

[4] 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.

[5] G. M. P. Wanderley, M.-H. Abel, E. C. Paraiso, and J.-P. A. Barthès, "MBA: A framework for building systems of systems," in *2018 13th Annual Conference on System of Systems Engineering (SoSE)*. IEEE, 2018, pp. 358–364.

[6] G. M. P. Wanderley, M.-H. Abel, E. C. Paraiso, and J.-P. A. Barthès, "MBA: A system of systems architecture model for supporting collaborative work," *Computers in Industry*, vol. 100, pp. 31–42, 2018.

[7] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multi-agent systems: The gaia methodology," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 12, no. 3, pp. 317–370, 2003.

[8] F. Oquendo, "Formally describing the software architecture of systems-of-systems with sosadl," in *System of Systems Engineering Conference (SoSE), 2016 11th*. IEEE, 2016, pp. 1–6.

[9] S. Agarwal, L. E. Pape, C. H. Dagli, N. K. Ergin, D. Enke, A. Gosavi, R. Qin, D. Konur, R. Wang, and R. D. Gottapu, "Flexible and intelligent learning architectures for sos (fila-sos): Architectural evolution in systems-of-systems," *Procedia Computer Science*, vol. 44, pp. 76–85, 2015.

[10] C. Ingram, R. Payne, and J. Fitzgerald, "Architectural modelling patterns for systems of systems," in *INCOSE International Symposium*, vol. 25, no. 1. Wiley Online Library, 2015, pp. 1177–1192.

[11] B. Ge, K. W. Hipel, K. Yang, and Y. Chen, "A novel executable modeling approach for system-of-systems architecture," *IEEE Systems Journal*, vol. 8, no. 1, pp. 4–13, 2014.

[12] J. Hu, L. Huang, X. Chang, and B. Cao, "A model driven service engineering approach to system of systems," in *Systems Conference (SysCon), 2014 8th Annual IEEE*. IEEE, 2014, pp. 136–145.

[13] P. Varga, F. Blomstedt, L. L. Ferreira, J. Eliasson, M. Johansson, J. Delsing, and I. M. de Soria, "Making system of systems interoperable–the core components of the arrowhead framework," *Journal of Network and Computer Applications*, vol. 81, pp. 85–95, 2017.

[14] H. Park, J. Tenenbaum, and R. Dove, "Agile infrastructure for manufacturing systems: A vision for transforming the us manufacturing base," in *Proceedings of the Defense Manufacturing Conference*, 1993.

[15] S. K. Gardikiotis, V. S. Lazarou, and N. Malevris, "Employing agents towards database applications testing," in *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, vol. 1. IEEE, 2007, pp. 173–180.

[16] G. M. P. Wanderley, M.-H. Abel, and E. C. Paraiso, "Designing proactive interfaces for cooperation using systems of systems," in *Computer Supported Cooperative Work in Design (CSCWD), 2018 IEEE 22st International Conference on*. IEEE, 2018.

[17] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on computers*, no. 12, pp. 1104–1113, 1980.

[18] N. Negroponte, *Being digital*. Vintage, 1996.