Mandala: An Agent-Based Platform to Support Interoperability in Systems-of-Systems

Altair Mendes, Stefano Loss, Everton Cavalcante, Frederico Lopes, Thais Batista
Federal University of Rio Grande do Norte
Natal, RN, Brazil
{altairbmendes,momoloss10}@gmail.com,everton@dimap.ufrn.br,fred@imd.ufrn.br,thais@ufrnet.br

ABSTRACT

A particular challenge to the construction of systems-of-systems (SoS) is the high heterogeneity of their constituent systems, thereby making interoperability an important issue to be tackled. This paper introduces Mandala, a platform to support interoperability in SoS. Mandala aims to offer a software layer to integrate heterogeneous, independent information systems without significantly changing their implementation or even knowing details about each system. The proposal relies on (i) business process models to represent activities associated to SoS global missions and (ii) software agents to support asynchronous communication among autonomous components. This paper presents an evaluation of Mandala and its interoperability mechanisms through a case study using information systems within a smart city scenario.

CCS CONCEPTS

• **Information systems** → **Information systems applications**; • **Software and its engineering** → **Designing software**; • **Computer systems organization** → *Distributed architectures*; • **Computing methodologies** → Multi-agent planning;

KEYWORDS

Information systems, systems-of-systems, interoperability, middleware, business processes, software agents

ACM Reference format:
Altair Mendes, Stefano Loss, Everton Cavalcante, Frederico Lopes, Thais Batista. 2018. Mandala: An Agent-Based Platform to Support Interoperability in Systems-of-Systems. In *Proceedings of SESoS'18: IEEE/ACM 6th International Workshop on Software Engineering for Systems-of-Systems, Gothenburg, Sweden, May 29, 2018 (SESoS'18),* 8 pages.
https://doi.org/10.1145/3194754.3194757

1 INTRODUCTION

Technological advances have led to a great diversity on how information systems are developed. At the same time, users have demanded a plurality of information that might not be satisfied by a single information system. Unconsciously, it is not just information that people require, but information systems that can be integrated and able to meet their needs as if they were a single system.

When attempting to integrate information systems, recurrent problems arise and hence hamper the development of value-added applications. A first one is that these systems are typically heterogeneous, developed with different technologies and data formats, and owned by distinct organizations. Another problem is that *interoperability* is a concern not properly addressed at their design or it becomes more relevant only after their development or deployment [1]. At last, it might be required to integrate legacy systems, which are often not intended to be integrated with other systems and/or make use of obsolete technologies.

The integration and interaction among different systems towards providing new functionalities through emerging behaviors result in the so-called *systems-of-systems* (SoS) [19]. In particular, it has also been possible to emergence of *systems of information systems* (SoIS)[1], a specific class of SoS oriented to business processes in which constituent systems are information systems belonging to different organizations [13, 20]. Besides the features shared with SoS, SoIS possess some distinct characteristics, namely (i) information flows among constituents, (ii) a significant process-driven nature, and (iii) the creation of value-added information from the interoperation among the involved constituent information systems that could not be obtained when their constituents work separately [28].

One of the major challenges to the development and deployment of SoS (and SoIS, by extension) regards the nature of their constituent systems, which are inherently heterogeneous and both operationally and managerially independent. For this reason, the interoperability among constituent systems within an SoS has been a quite relevant topic addressed in this context [4, 11, 18, 21, 31]. Therefore, making these systems interoperable while preserving their operational and managerial independencies is an imperative issue to be tackled. This is determinant for promoting emergent behaviors and hence making an SoS to achieve its global missions.

A way of addressing these issues is developing a platform capable of mediating the interoperation among existing heterogeneous information systems, which may lead to the emergence of new functionalities resulting from this collaboration [17]. In line with such a strategy, this paper introduces *Mandala*, a platform to allow for interoperability among heterogeneous information systems. The purpose of the Mandala is offering a software layer to integrate constituent systems of a directed SoIS without significant changes

[1] For the sake of simplicity, the terms SoS and SoIS will be herein interchangeably used to express both singular and plural.

SESoS'18, May 29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5747-0/18/05…$15.00
https://doi.org/10.1145/3194754.3194757

in their original implementation or requiring managers to know details about the other participating systems. The communication framework used in Mandala relies on *software agents* [29] for asynchronous message exchange between autonomous components and it is guided by business processes represented in the Business Process Model and Notation (BPMN) [23]. Such a model representing the business process is interpreted by Mandala and each step of the flow is assigned to constituent systems of the SoIS, which perform the activities defined in the global mission expressed by the business process.

The remainder of this paper is organized as follows. Section 2 illustrates a motivating scenario with the need for interoperability among heterogeneous information systems within a smart city environment. The architecture and implementation of Mandala are detailed in Section 3. Section 4 describes an evaluation of the current prototype of Mandala and its interoperability mechanisms through a case study. Section 5 discusses related work. Finally, Section 6 brings some concluding remarks and directions to future work.

2 MOTIVATING SCENARIO

Looking forward to increase smartness, today's cities are evolving into environments with a myriad of distributed systems engaged in complex relationships including the integration and interaction with other systems towards providing new functionalities. In this context, both public and private heterogeneous and independent systems shall be integrated to achieve effectiveness and efficiency for citizens, besides being producers and consumers of information to each other [7]. Therefore, smart city SoIS can rely on collaborating information systems that interact, communicate, and share information with each other, allowing for cross-domain usages of services and their synergistic integration.

Aiming at presenting the benefits from integrating heterogeneous information systems, this section illustrates a scenario involving three systems in a smart city. Managed by the City Hall, System 1 is made up of sensors and periodically collects and informs both level of occupation and content weight of garbage containers across the city. Managed by a non-governmental organization, System 2 is an eco-feedback system responsible for providing the population with information about garbage production in the city, but such an information is produced a few days late since System 1 does not provide information at real-time. The mission of System 3 is generating routes to garbage trucks responsible for collecting garbage from the containers, so that the number of routes and cars for the garbage collection depends on container information provided by the City Hall. Considering that these three systems are independent, isolated, developed using different technologies, and managed by different agencies of the city, some sort of integration needs to be developed to enable them to communicate with each other towards better supporting public cleaning in the city.

This scenario presents a situation in which all systems run in isolation and have their own individual missions. However, if they were integrated and able to interoperate, new functionalities could arise to benefit city operations:

Example 1. Integrating System 1 and System 2 would allow knowing the exact amount of waste in all neighborhoods or regions of the city. This information could be compared with historical

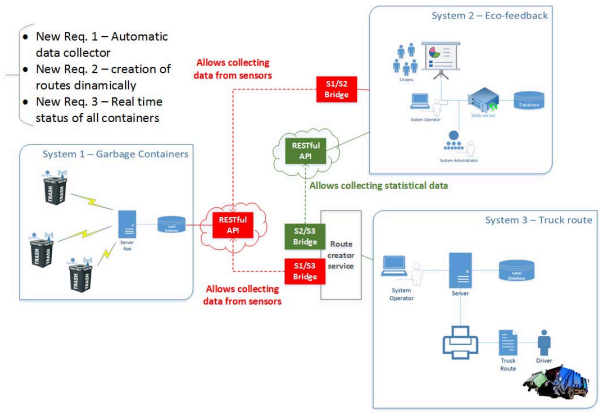

Figure 1: Scenario involving the integration of three systems to support garbage collection in a smart city.

data to decide if it is necessary to adjust the amount of containers in a specific region of the city. Working in a isolated way, such an adjustment would be difficult to achieve by either System 1 or System 2.

Example 2. System 3 generates routes to garbage trucks using a simple division of the number of containers by the number of available cars. If a new container appears, the route will not be automatically or immediately generated, resulting in garbage accumulation in streets as the information of this new container will take time to be provided by the City Hall to the container owner company. If System 1 and System 3 were integrated, then the company responsible for route generation would know when there is a new collection site.

Example 3. A more comprehensive situation achievable by the proper interoperability among the three systems would be generation of optimized routes according to the level of occupation and content weight of garbage containers. In addition, it would be possible to present the real-time collection and which and how many cars are being used. System 1 would be responsible for proving the level of occupation of each container while System 3 would determine optimized routes per region based on information registered at System 2 by verifying which containers are geographically close and the amount of garbage in containers. This would prevent garbage cars from leaving for collection without enough garbage to fill them or using several cars when less cars would be needed. This situation only would be achieved through interoperability among the aforementioned systems.

After identifying the individual missions of each system and the benefits resulted from such an integration, the issue regards on how these systems would provide these benefits. An alternative is to directly perform the integration (as shown in Figure 1), i.e., each system negotiating how to communicate with the others. This situation is possible to be realized, but there are clear disadvantages:

- assuming that a fourth system (e.g., a system of a company that deals with recycled garbage and needs up-to-date information about routes) is integrated to the environment, each existing system already in operation should communicate with this new system;

- replacing a system that has been retired from the environment to another one able to perform the same functionalities would likely impact all involved systems;
- development teams of each system within the environment need to be aware of all other involved systems.

Due to these issues, it is evident that an attempt of directly integrating these different heterogeneous systems is not the most viable alternative to the dynamic scenario of smart cities. Mandala arises to address such a lack of interoperability by offering a way of seamlessly integrating information systems towards having SoIS able to better support operations in dynamic environments.

3 ACHIEVING SYSTEM INTEROPERABILITY WITH MANDALA

This section introduces Mandala as means of promoting interoperability among information systems to form a SoIS while abstracting away details related to the technologies used by them and other specificities. As each constituent system has its individual mission and may not be originally designed to interoperate with other information systems, Mandala would be responsible for such an interoperability. It is important to emphasize that this interoperability is not merely related to data integration, but the idea is providing new functionalities resulting from the interaction of these systems, the so-called emergent behaviors within the SoIS. Section 3.1 describes the architecture of Mandala and Sections 3.2 and 3.3 respectively present details about its prototype implementation and operation.

3.1 Architecture

To a general extent, Mandala aims to minimize difficulties raised from the need of interoperating heterogeneous information systems developed with different technologies. As described in the following, Mandala is composed of four main components and a communication structure (see Figure 2).

SoS Composer. This composition component is responsible for creating the SoIS and associating constituent systems and execution flows. In this component, the developer queries candidate systems to participate in a SoIS. If it is possible to create a SoIS with the systems registered at the platform by their managers, they are associated to the SoIS (and now can be considered as constituents) and an execution flow is generated. Afterwards, the SoIS is created.

SoS Manager. This component stores information registered at the SoS Composer as well as information about candidate systems to constituents provided by their respective managers. It is also responsible for publishing a Web service associated to the SoIS and making it available to clients.

SoS Server. When a SoIS is created by the SoS Composer, a Web service is published and the functionalities of the new SoIS become available for clients. The SoS Server component is responsible for the client interface to the external environment.

Broker. This entity is responsible for bridging Mandala and constituent systems. The structure of Broker was created to extract information from constituent systems without directly dealing with their respective development technology. The constituent systems developers must provide an interface to enable the Broker to query

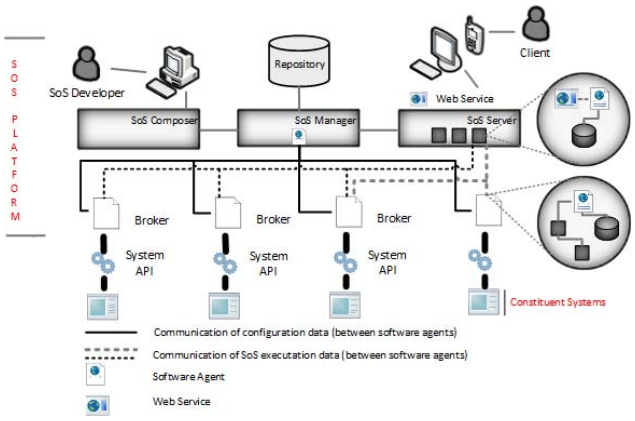

Figure 2: Mandala architecture.

the features registered at the SoS Manager. A Broker instance is associated to each constituent system, allowing it to communicating with the platform.

The communication structure aims to make the connection between the Mandala's components and the Broker that communicates with constituent systems. To provide dynamism, autonomy, and intelligence to such a communication, Mandala makes use of *software agents* for exchanging messages. A software agent is a computational entity placed into an environment and capable of actuating over it to achieve their design goals in an autonomous, flexible way [29]. Typical features of software agents are autonomy, communication, cooperation capabilities, intelligence, reactivity, and proactivity, and mobility [6]. In this perspective, a software agent can perform tasks based on asynchronous message exchanges with other local and/or external agents using a standardized specific language. These characteristics have led us to adopt such an approach in Mandala, thus fostering a fast, asynchronous, distributed communication within the platform.

As depicted in Figure 2, data are stored in three different components. The repository within SoS Manager contains metadata information related to the structure of the SoIS to be executed, such as the address of each Broker, both SoIS and constituents' missions, and the SoIS execution flow. Another repository is located into the SoS Server, which stores specific data about each client connection to the Web service available at this component. The third repository is within the Broker, which stores information related to the communication with constituent systems. When a SoIS is under execution, the Broker associated to each participating system stores the last step that was performed as well as the next action to be executed.

3.2 Implementation

The current prototype of Mandala was implemented using the Java programming language for the sake of portability and compatibility with the technologies used for most of its components. As previously mentioned, Mandala makes use of software agents because it is into a distributed environment and requires fast, efficient asynchronous communication, as well as a certain autonomy to perform some tasks. The communication structure of Mandala could have

been implemented through Service-Oriented Architecture (SOA) principles and Web services [31], but such an integration would be limited to syntactic compatibility. In turn, agents possess proactivity characteristics and can allow for a greater dynamism when executing a SoIS atop the platform.

In the context of agent-based systems, JADE [14] is a Java framework designed to create and manage the life cycle of intelligent agents. Besides facilitating the development of systems made up of multiple interoperable agents, JADE provides (i) an environment for the distributed execution of agents, (ii) classes and libraries to create agents, and (iii) a set of graphical tools to monitor and manage the execution of intelligent agents. In JADE, agents communicate with each other using the Agent Communication Language (ACL) and each one executes a behavior implemented by the developer. As the interaction of Mandala with clients takes place through a Web service, the JADE Gateway is used to support processing requests made by clients and responses to the activities defined in the business processes that represent SoIS global missions.

The agent technology plays a central role in the implementation of Mandala as it allows abstracting away the communication among its components. There are agents in each Broker instance associated with each constituent system, in the SoS Manager, and in the SoIS under execution. When the SoIS Web service receives a request, the SoIS agent communicates with the agent of each of the Broker instance associated with the constituent systems, initiating the execution of the business process regarding the SoIS global mission. The SoS Manager agent is also part of the communication cycle and it is used to communicate with the Broker instance to inform of its participation in a newly created SoIS.

When integrating constituent systems within SoIS, individual and/or global missions may have associated *business processes* defining a sequence of interdependent activities to be performed by capabilities of the constituent information systems, besides involving the information flows among these systems [13]. Although there is still no specific modeling notation for SoIS missions, it is possible to use Business Process Modeling (BPM) to complement mission specification in this type of systems, allowing for a better understanding of its business-oriented nature in terms of activities and the relationships among them [32].

Business processes can be graphically represented in BPMN through Business Process Diagrams (BPDs) with flow objects (events, activities), connection objects (sequences, messages, associations), swimlanes to organize activities in different categories or responsibilities, and artifacts aggregating additional information about the business process. Some studies in the literature have analyzed the use, benefits, and drawbacks of BPMN in comparison to the Software Process Engineering Metamodel (SPEM) notation [22], but they lack of conclusive results and suggest situations in which one of these notations would be best suited to use, even in combination [8, 25]. Despite SPEM is more expressive to represent business processes, BPMN is easier to understand and it is better supported in terms of execution tools. For these reasons, BPMN was adopted in the context of this work.

Figure 3 illustrates a BPD regarding *Generate Optimized Routes*, the global mission of the SoIS described in Section 2. The execution

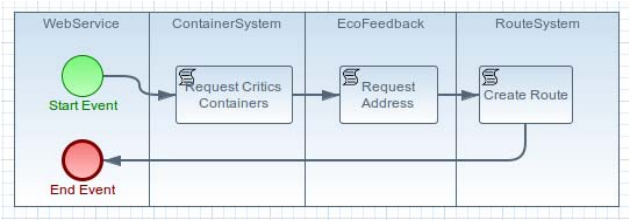

Figure 3: Representation of the *Generate Optimized Routes* global mission as a business process in BPMN. Each activity is assigned to a constituent information system (represented by a swimlane) of the SoIS.

sequence is determined by connection objects represented by arrows: the sequence is started and finished by the Web service that serves as the interface of the SoIS with its clients. The intermediary executors are the constituent systems of the SoIS.

3.3 Operation

Before being available for use, a SoIS needs to be designed, registered, and have associated candidate information systems to become constituents that together can make it to achieve its global mission. The Broker instance that will communicate with the candidate constituent system is registered by implementing an interface provided by Mandala to developers. Relevant information about the Broker instance is agent address, constituent system address, individual missions, interface, functionalities, and input/output data types. This information is stored into the repository of the SoS Manager.

Once the SoIS developer has information about the candidate systems, he/she can group systems and associate them to a SoIS as well as define a global mission by creating an execution flow. Such a flow is generated from a BPD transformed into an XML schema whose elements are mapped to objects in the Java programming language and stored into the repository of the SoS Manager. After this composition process, part of the information registered at the SoS Manager is replicated to the repository within the SoS Server and to the repository of each Broker instance. Next, a Web service is generated and published to enable clients to consume information.

The SoS Server represents the Mandala's client interface. When a client makes a request to the SoIS Web service, the JADE Gateway creates a temporary software agent within the SoS Server for each received request. After retrieving the addresses of the software agents of each *Broker* instance associated with the constituent systems, the execution of the SoIS is started. The Broker instance responsible for the execution of the activity specified in the business process communicates with the constituent system interface to perform the activity. At the end of the activity execution, a message is sent to all Broker instances of the SoIS and the one responsible for the next activity is identified and so on. When all activities of the business process representing the SoIS mission have been executed, the Broker instance responds to the SoS Server, which requests the Web service to forward execution results to the client. Figure 4 illustrates the whole process to realize the SoIS mission.

It is noteworthy that the Mandala operation resembles the traditional Web service orchestration, which has been advocated as

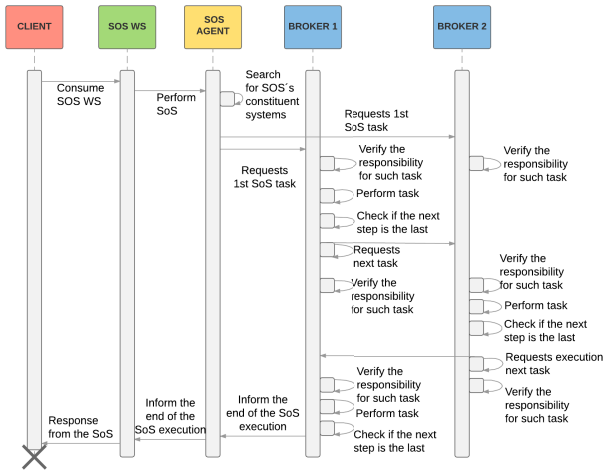

Figure 4: UML Sequence Diagram representing the process to realize the global mission of a SoIS.

a promising technology to support interoperability using process-driven application integration [15, 35]. In Web service orchestration, a single centralized executable business process (the orchestrator) coordinates the interaction among different services, resulting in a workflow that consists of the combined services and in what order they will be invoked to realize a business process [10, 24]. Mandala works as an orchestrator in terms of coordinating the interaction among the constituent systems, but the platform goes beyond as SoS evolutionary development and emergent behaviors are features not properly addressed in the traditional Web service orchestration. Indeed, SoS arise from the interaction among different constituent systems, but the result of such an interaction is said to be more than the sum of the constituents or the mere combination of their functionalities, as it is often observed in composite services.

4 EVALUATION

Mandala and its interoperability mechanisms have been evaluated through a case study with information systems in the smart city scenario described in Section 2. A *case study* is an observational, primarily exploratory empirical method intended to investigate phenomena in its natural environment using multiple sources of evidence [26, 34]. Even though case study research is of flexible design, this does not mean that a rigorous process is not required. Well-known guidelines in the literature [16, 27, 33] describe five major steps to be considered when conducting a case study, as illustrated in Figure 5: (i) *case study design/planning*, in which rationale and purpose are established and the case study itself is planned; (ii) *data collection preparation* in terms of procedures and protocols to conduct the case study and methods to collect data; (iii) *data collection* on the studied case; (iv) *analysis of collected data*, either quantitatively and/or qualitatively; and (v) *reporting/discussion* as means of communicating the findings of the case study. The conduction of these steps in the context of the case study is described in the following.

Figure 5: Case study process.

4.1 Case Study Design

Runeson et al. [27] and Wohlin et al. [33] elicit the basic elements to be considered in the design of a case study, such as objectives, case under study, research questions to be investigated, and methods for data collection and analysis. The objective of the case study was to explore how Mandala is able to integrate heterogeneous constituent information systems based on the premises defined for SoIS. The case[2] refers to the SoIS described in Section 2, which intends to support optimized garbage collection in a smart city. Two research questions (RQs) are of particular interest in this case study:

RQ1: Is Mandala able to support interoperability among constituent systems of the SoIS and present correct results to users? This RQ aims to verify whether Mandala can successfully integrate heterogeneous constituent systems.

RQ2: Is Mandala useful as means of integrating information systems in SoIS? This RQ aims to address how Mandala complies with the defining characteristics of SoIS.

4.2 Data Collection Preparation

In many case studies, data collection techniques are mostly focused on qualitative data, but quantitative data are also important in a case study. Wohlin et al. [33] suggest that the definition of what data to collect should be based on a goal-oriented measurement technique, such as the Goal/Question/Metric (GQM) method [30]. In this perspective, a metric was associated to each RQ as follows.

Metric M1: Amount of failures during Mandala execution. The successful integration of heterogeneous constituent systems (RQ1) can be assessed through the absence of errors in the execution of Mandala. This can be measured by the amount of failures (crashes, stops, incorrect results) observed during the execution of the platform when integrating systems towards achieving the global mission of the SoIS.

Metric M2: Number of SoIS characteristics addressed by Mandala. As previously mentioned, SoIS possess the five distinguishing features of any SoS, namely operational independence, managerial independence, geographical distribution, evolutionary development, and emergent behaviors [19]. Besides these characteristics and inherent dynamicity, SoIS specifically exhibit information flows among constituents, business process orientation, and the generation of value-added information resulting from interactions among constituent systems. To verify whether Mandala is indeed useful in the context of SoIS (RQ2), it is necessary to check how many characteristics are properly addressed by Mandala.

[2] Yin [34] distinguishes between *holistic case studies*, in which the case is studied as a whole, and *embedded case studies*, in which multiple units of analysis are studied within a case to be observed. According to such a terminology, the case study in scene here can be classified as a holistic one, the unit of analysis being the case itself.

4.3 Data Collection and Analysis

In terms of data collection and analysis, different methods were used for each metric defined in Section 4.2. Apart from these differences, all methods have considered the SoIS consisting of the systems described in Section 2. The global mission of this SoIS is to *Generate Optimized Routes*, as shown in Figure 3.

The correctness in requests to the SoIS (metric M1) can be verified by comparing the response returned by each request to the expected response. To provide more confidence on the reliability of the results, multiple requests were simulated by making use of Apache JMeter [3], a Java-based open-source tool for analyzing and measuring the performance of Web services. A JMeter test plan with ten threads (each one simulating a virtual user) was built to run 30 times in a computer with Intel Core i5 2.6 GHz processor, 4 GB of RAM, and Linux Ubuntu 16.04 as operating system. Performing these multiple executions is required to ensure an adequate statistical significance for the observed results since Mandala was implemented in the Java programming language and may be subjected to influences of the Java Virtual Machine (JVM). In turn, Mandala and System 1 (container management) were deployed on the same computer where JMeter was running while System 2 (eco-feedback) and System 3 (route generation) were deployed on a computer with Intel Core i5 2.5 GHz, 4 GB of RAM, and macOS High Sierra as operating system.

The compliance of Mandala with the inherent characteristics of SoIS (metric M2) can be verified through a simple checklist contrasting the features provided by the platform with such characteristics. More specifically, the following concerns shall be addressed:

- **Operational and managerial independencies.** Constituent systems only provide an access interface, there are no changes in their implementation, and their participation in an SoIS managed by Mandala is transparent and does not affect their individual execution.
- **Geographical distribution.** There is at least one constituent system placed in a computational node different of where Mandala is.
- **Evolutionary development.** Changes in the constituent systems by their own are properly handled by the Mandala while supporting the SoIS execution.
- **Emergent behaviors.** The SoIS capable of providing new functionalities from the integration solely using system interfaces provided to Mandala.
- **Business process orientation.** Business processes can properly represent the global missions of the SoIS, the activities to be realized by the integrated constituent systems, as well as execution flows among these systems.

4.4 Reporting and Discussion

RQ1. The tests described in Section 4.3 run with no failures in the Mandala execution[3], making the value for metric M1 be equals to zero. Therefore, it possible to conclude that Mandala is feasible to promote the integration among different constituent systems, well succeeding in the studied SoIS. Nonetheless, other cases with different scenarios and/or a greater number of constituent systems need to be further verified as a single one can show effects in a typical situation, but it cannot be generalized to every possible situation [16]. In more realistic environments with geographically distributed constituent systems accessed through networks, some delay is likely to be observed, but this is more related to the communication infrastructure rather than the operation of Mandala.

[3] The average response time for 30 executions of the JMeter test plan was 3.743 seconds.

Table 1: SoIS characteristics vs. Mandala features

SoIS characteristics	Fulfillment
Operational and managerial independencies	√
Geographical distribution	√
Evolutionary development	×
Emergent behaviors	○
Business process orientation	√

√ = complete fulfillment, ○ = partial fulfillment, × = no fulfillment

RQ2. Table 1 summarizes how the features provided by Mandala comply with the inherent characteristics of SoIS. In a score scale ranging from zero to five with complete fulfillment scored as 1.0, partial fulfillment as 0.5, and no fulfillment as 0.0, the value for metric M2 would be 3.5 out of 5.0. Executing the SoIS in the studied scenario has led to some important observations. First, constituent systems were not affected in their individual operation or implementation and their integration to the SoIS was completely transparent, thereby preserving their operational and managerial independencies and fulfilling the Mandala's purpose of abstracting away heterogeneities. Second, BPM has shown to be useful to represent the global mission of the SoIS, the activities to be performed by the integrated constituent systems, and execution among them (see Figure 3). This model and further transformations towards agent-based communication and execution within Mandala make the platform suitable to cope with concerns related to the inherent business process orientation of SoIS.

The conducted case study has revealed some limitations of Mandala with respect to evolutionary development and emergent behaviors. The platform currently is not able to handle changes in the integrated constituent systems and supporting emergent behaviors significantly depends on developers' implementations at the SoIS side that be able to make use of information coming from constituent systems. Nonetheless, it is still possible to conclude that Mandala is on the way of being a useful platform when integrating independent, heterogeneous systems in SoIS.

5 RELATED WORK

The implementation of Mandala makes use of technologies already known in commercial or academic solutions, but their combination in a single solution has still been few explored so far. Indeed, a recent systematic literature review conducted by Vargas et al. [31] reports that there was still a lack of tools to assist the integration of systems in the context of SoS. As the literature is scarce in terms of proposals adopting the same concepts and paradigms used in Mandala, this section briefly discuss existing work related to some aspects considered in the platform.

Linked DataSpace [9] is a tool comprising modules to manage heterogeneous data generated by several sources and to share and reuse these data. This proposal aims to achieve interoperability by creating a global information space in which heterogeneous

data coming from different systems are stored and Semantic Web techniques are used to structure and share data. Linked DataSpace promises creating SoS, but it focuses on data rather than functionalities and it does not concern data flows.

Afoutni et al. [2] proposes an approach based on multi-agent systems and ontologies to handle technical interoperability among information systems related to product lifecycle management (PLM). Important features to be considered are: (i) not intrusive interoperability, i.e., the participating systems are not modified in their usual features; (ii) the addition of a system should not cause important developments, and; (iii) enable user to seamlessly access heterogeneous and distributed resources of PLM. However, the work is still under development and only a conceptual architecture has been presented so far.

Other relevant works found in the literature are Holons [5] and OverStar [12]. Holons is actually a proposal towards modeling integration of distributed systems that form an SoS and hence there is still no implementation. In turn, OverStar is a framework through which heterogeneous middleware services can be composed in an interoperable way to achieve a performance-optimized service configuration to serve applications. Unlike these proposals, Mandala aims to promote interoperability among heterogeneous information systems within SoIS.

6 CONCLUDING REMARKS

This paper presented the architecture and implementation of Mandala, an agent-based platform to support interoperability among independent information systems. Mandala provides a software layer to integrate heterogeneous constituent systems of a SoIS while relieving developers from the burden of significantly change the original implementation of these systems or the need of knowing details about the other participating systems. In Mandala, the global missions to be achieved by a SoIS are represented as business processes, deriving activities to be performed by the integrated constituent systems. Another outstanding feature of the platform is the use of software agents to allow for asynchronous communication among its components. Mandala has been evaluated through a case study with information systems within a smart city SoIS, showing that it fulfills its purpose of integrating information systems while meeting the inherent characteristics of SoIS. The work presented in this paper seeks to contribute to minimize the slow maturation and lack of systematic approaches to SoS integration still noticed in the literature. Even though Mandala uses several already consolidated technologies adopted for various purposes, not much has been done so far about using these technologies together to achieve interoperability in SoIS.

Mandala currently has some limitations that must be addressed in future work. One of the most important ones is that the platform concerns only directed SoS, in which there is a central control entity and the behavior of its constituent systems is subordinated to such a central control and its purposes. The other typical categories of SoS (namely, acknowledged, collaborative, and virtual) may be addressed by bringing the proactivity of software agents to the platform. In this case, global missions could be autonomously realized through emergent behaviors resulting from interactions among the constituent systems available in the environment. Finally, other concerns raised during the conduction of the case study and that shall be subject of future work refer to improvements on the support for evolutionary development and emergent behaviors, which are important characteristics of any SoS.

REFERENCES

[1] Hadil Abukwaik and Dieter Rombach. 2017. Software interoperability analysis in practice: A survey. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE 2017)*. ACM, New York, NY, USA, 12–20.

[2] Zoubida Afoutni, Julien Le-Duigou, Marie-Hélène Abel, and Benoit Eynard. 2017. Towards a proactive interoperability solution in systems of information systems: A PLM perspective. In *Proceedings of the 14th IFIP WG 5.1 International Conference on Product Lifecycle Management (PLM 2017)*, José Ríos, Alain Bernard, Abdelaziz Bouras, and Sebti Foufou (Eds.). IFIP Advances in Information and Communication Technology, Vol. 517. Springer International Publishing, Cham, Switzerland, 580–589.

[3] Apache Software Foundation 1999. Apache JMeter™. (1999). http://jmeter.apache.org/

[4] Thiago Bianchi, Daniel Soares Santos, and Katia Romero Felizardo. 2015. Quality attributes of systems-of-systems: A systematic literature review. In *Proceedings of the 3rd International Workshop on Software Engineering for Systems-of-Systems (SESoS 2015)*. IEEE, USA, 23–30.

[5] Gordon Blair, Yérom-David Bromberg, Geoff Coulson, Yehia Elkhatib, Laurent Réveillère, Heverson B. Ribeiro, Etienne Rivière, and François Taïani. 2015. Holons: Towards a systematic approach to composing systems of systems. In *Proceedings of the 14th International Workshop on Adaptive and Reflective Middleware (ARM 2015)*. ACM, New York, NY, USA.

[6] Walter Brenner, Hartmut Wittig, and Rüdiger Zarnekow. 1998. *Intelligent software agents: Foundations and applications*. Springer-Verlag Berlin Heidelberg, Germany.

[7] Everton Cavalcante, Nélio Cacho, Frederico Lopes, and Thais Batista. 2017. Challenges to the development of smart city systems: A system-of-systems view. In *Proceedings of the 31st Brazilian Symposium on Software Engineering (SBES 2017)*. ACM, New York, NY, USA, 244–249.

[8] Mario Cervera, Manoli Albert, Victoria Torres, and Vicente Pelechano. 2012. *A comparative analysis of SPEM 2.0 and BPMN 2.0*. Technical report. Universidad Politécnica de Valencia, Valencia, Spain.

[9] Edward Curry. 2012. System of systems information interoperability using a linked dataspace. In *Proceedings of the 7th International Conference on System of Systems Engineering (SoSE 2012)*. IEEE, USA, 101–106.

[10] Florian Daniel and Barbara Pernici. 2008. Web service orchestration and choreography: Enabling business processes on the Web. In *E-business models, services and communications*, In Lee (Ed.). IGI Global, USA, 250–273.

[11] David A. Fisher. 2006. *An emergent perspective on interoperation in systems of systems*. Technical report. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA.

[12] Paul Grace, Yérom-David Bromberg, Laurent Réveillère, and Gordon Blair. 2012. Overstar: An open approach to end-to-end middleware services in systems of systems. In *Proceedings of the 13th ACM/IFIP/USENIX International Middleware Conference (Middleware 2012)*, Priya Narasimhan and Peter Triantafillou (Eds.). Lecture Notes in Computer Science, Vol. 7662. Springer Berlin Heidelberg, Germany, 229–248.

[13] Valdemar Vicente Graciano Neto, Everton Cavalcante, Jamal El Hachem, and Daniel Soares Santos. 2017. On the interplay of Business Process Modeling and missions in systems-of-information systems. In *Proceedings of the Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (SESoS/WDES 2017)*. IEEE, USA, 72–73.

[14] JADE 2015. Java Agent DEvelopment Framework. (2015). http://jade.tilab.com

[15] Marijn Janssen, Jeffrey Gortmaker, and René W. Wagenaar. 2006. Web service orchestration in public administration: Challenges, roles, and growth stages. *Information Systems Management* 23, 2 (Dec. 2006), 44–55.

[16] Barbara Kitchenham, Lesley Pickard, and Shari Lawrence Pfleeger. 1995. Case studies for method and tool evaluation. *IEEE Software* 12, 4 (Jul. 1995), 52–62.

[17] Frederico Lopes, Stefano Loss, Altair Mendes, Thais Batista, and Rodger Lea. 2016. SoS-centric middleware services for interoperability in smart cities systems. In *Proceedings of the 2nd International Workshop on Smart Cities: People, Technology and Data*. ACM, New York, NY, USA.

[18] Azad M. Madni and Michael Sievers. 2014. System of systems integration: Key considerations and challenges. *Systems Engineering* 17, 3 (Sep. 2014), 330–347.

[19] Mark W. Maier. 1998. Architecting principles for systems-of-systems. *Systems Engineering* 1, 4 (Feb. 1998), 267–284.

[20] Saleh Majd, Abel Marie-Hélène, and Mishra Alok. 2015. An architectural model for system of information systems. In *On the Move to Meaningful Internet Systems: OTM 2015 Workshops*, Ioana Ciuciu et al. (Ed.). Lecture Notes in Computer Science, Vol. 9416. Springer International Publisher, Cham, Switzerland, 411–420.

[21] Rebeca Campos Motta, Káthia Marçal Oliveira, and Guilherme Horta Travassos. 2017. Rethinking interoperability in contemporary software systems. In *Proceedings of the Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (SESoS/WDES 2017)*. IEEE, USA, 9–15.

[22] OMG 2008. Software & Systems Process Engineering Meta-Model Specification, version 2.0. (2008). http://www.omg.org/spec/SPEM/2.0/

[23] OMG 2011. Business Process Model and Notation (BPMN), version 2.0. (2011). http://www.omg.org/spec/BPMN/2.0/

[24] Chris Peltz. 2003. Web services orchestration and choreography. *Computer* 36, 10 (Oct. 2003), 46–52.

[25] Carlos Portela, Alexandre Vasconcelos, Antônio Silva, Ariane Sinimbú, Elder Silva, Maurício Ronny, Wallace Lira, and Sandro Oliveira. 2012. A comparative analysis between BPMN and SPEM modeling standards in the software processes context. *Journal of Software Engineering and Applications* 5, 5 (2012), 330–339.

[26] Colin Robson and Kieran McCartan. 2016. *Real world research* (4th ed.). John Wiley & Sons Ltd., Chichester, United Kingdom.

[27] Per Runeson, Martin Höst, Austin Rainer, and Björn Regnell. 2012. *Case study research in Software Engineering: Guidelines and examples*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

[28] Majd Saleh and Marie-Hélène Abel. 2015. Information systems: Towards a system of information systems. In *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015)*. SciTePress, Setúbal, Portugal, 193–200.

[29] Katia Sycara, Anandee Pannu, Mike Williamson, Dajun Zeng, and Keith Decker. 1996. Distributed intelligent agents. *IEEE Expert* 11, 6 (Dec. 1996), 36–46.

[30] Rini van Solingen and Egon Berghout. 1999. *Goal/Question/Metric Method: A practical guide for quality improvement of software development*. McGraw-Hill, USA.

[31] Iohan Gonçalves Vargas, Thiago Gottardi, and Rosana Teresinha Vaccare Braga. 2016. Approaches for integration in system of systems: A systematic review. In *Proceedings of the 4th International Workshop on Software Engineering for Systems-of-Systems (SESoS 2016)*. ACM, New York, NY, USA, 32–38.

[32] Mathias Weske. 2012. *Business Process Management: Concepts, languages, architectures* (2nd ed.). Springer-Verlag Berlin Heidelberg, Germany.

[33] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer-Verlag Berlin Heidelberg, Germany.

[34] Robert K. Yin. 2017. *Case study research and applications: Design and methods* (6th ed.). SAGE Publications, Inc., Thousand Oaks, CA, USA.

[35] J. Leon Zhao and Hsing Kenneth Cheng. 2005. Web services and process management: A union of convenience or a new area of research? *Decision Support Systems* 40, 1 (Jul. 2005), 1–8.