Modeling Information Systems as Systems of Systems

Paolo Salvaneschi
University of Bergamo, Dep. of Management, Information and Production Engineering and Salvaneschi & Partners
Bergamo, Italy
paolo.salvaneschi@unibg.it

***Abstract*—Large industrial information systems are composed of dozens of inter-operating software applications. Each of them implements a relatively independent set of functions but also participates to business processes involving more applications. Applications may be COTS acquired from different vendors or custom-developed. The network of them evolves and grows in time. Many development groups/vendors manage the network. All these characteristics support the idea that large information systems may be interpreted as Systems of Systems (SoS) and that this view may provide value to IT managers. In the paper, we present an experience report of SoS concepts application to the information system of a large retail company. In this System of Systems, a critical problem is the absence of documents providing SoS views of the whole information system: the set of applications, their relationships and the impact of business processes on the network of applications. To mitigate the problem, we developed a set of models using the *minimalist* approach (the selection of the minimal set of documents based on a cost/benefit analysis) and the ArchiMate modeling standard and tools. A static view (software applications and relations) and dynamic views (business processes and their interaction with software applications) model each business area. We discuss the current use and benefits of the models and the forecast improvements.**

Keywords—Information System, System of Systems, ADLs, Archimate

I. INTRODUCTION

Large industrial information systems are composed of dozens of inter-operating software applications. The applications may be custom-developed or COTS components acquired from the market and, if needed, adapted. Each application delivers functions implementing a specific set of business activities, but the business processes flow through more applications. The set of applications evolves and grows in time and usually different development groups or vendors drive the evolution process, each working on parts of the whole system.

We suggest that the application of concepts of Systems of systems (SoS) may be useful to the management of such type of systems.

Systems of systems are described as a collection of relatively independent but interconnected systems [1] where a strong central control of evolution may not exist and the global behaviors (both desirable and undesirable) emerge from the interaction between the composing systems.

In this paper, we describe an industrial experience where we apply the concepts of SoS to the management of the information system of a large retail company.

The application of SoS concepts has been mainly in other areas [2] than industrial information systems, even if e-commerce applications (part of the information system of our case study) are considered a typical example of SoS [3].

Section 2 discusses how information systems share the characteristics of SoSs to be considered as SoSs. We contend that the SoS view may advance the state-of-the-art of information systems management.

Section 3 presents the information system that is subject of the study.

In section 4 we define the research question this work answers: how to develop and maintain a knowledge base of the most important aspects of the whole information system at the abstraction level of a SoS. To this aim, we developed a model of the information system.

Sections 5 and 6 define the modeling technique and tool, exemplify the content of the model and discuss how the model was developed.

Section 7 describes how different stakeholders used and get benefits of the model.

Section 8 concludes, discusses the scope of the industrial experience and the possible improvements. Finally, Section 9 provides an overview of related work.

II. INFORMATION SYSTEMS AS SOS

Information systems share the characteristics of SoSs ?

Nielsen [3] claims that a SoS is characterized by the following dimensions:

- *Autonomy of constituents*: a constituent system's behavior is governed by its own rules while also participating in the SoS.
- *Independence*: the capacity of constituent systems to operate when detached from the rest of the SoS.
- *Distribution*: constituent systems are dispersed so that some form of connectivity enables communication or information sharing.
- *Evolution*: SoSs are long lasting and subject to change, whether in the functionality, the quality of that

functionality, or in the structure and composition of constituent systems.

- *Dynamic Reconfiguration*: the capacity of an SoS to undertake changes to its structure and composition, typically without planned intervention.
- *Emergence of Behavior*: emergence refers to the behaviors that arise as a result of the interaction of constituents.
- *Inter-dependence*: refers to the mutual dependency that arises from the constituent systems having to rely on each other in order to fulfill the common goal of the SoS.
- *Inter-operability*: refers to the ability of the SoS to incorporate a range of heterogeneous constituent systems.

A large information system is composed of many software applications (custom, COTS, adapted). We may interpret each application as a "component" of a system architecture (according to the paradigm of components and connectors description of a software architecture) at a larger scale than the architecture of a single application. Various servers host the applications and the information system may cooperate with others information systems (for instance suppliers and clients). All these servers are typically geographically distributed (*Distribution*).

From the functional point of view, applications are relatively independent functional entities (*Autonomy of constituents* and *Independence*) but inter-operate with other relatively independent applications (*Inter-dependence* and *Inter-operability*). Business processes may be orthogonal to many software applications. Each software application may implement a set of functions and share part of them with various business processes.

Information systems are subject to an evolutionary process (*Evolution*). Evolution projects (for example the evolution required by a change in a business process or a new process) may involve many applications. In this case, the specification and design documents of the project define a set of local changes generating a global desired behavior (*Emergence* of desired behaviors). Consequently, designing and implementing changes require the understanding of many existing applications and their relations. The evolutionary process, composed by a sequence of local changes, may also produce global undesired changes like the "architectural erosion" [4] or undesired behaviors (another type of *Emergence*).

If we consider the management aspect, projects may involve more development teams or COTS providers. Projects run concurrently and are managed in a relatively independent way. Each project team follows a "local optimum" goal with limited interactions with the remaining parts of the system not included in the project. Therefore, it is difficult for the IT management to have a global control of evolution.

From a cognitive point of view, each development team or provider has a local knowledge related to a subset of applications. Consequently, none of the teams has a reasonably complete knowledge of the whole information system. Even if the specification and design documentation of each application is of sufficient quality, it does not offer a useful knowledge supporting global understanding of applications relations and behaviors. This causes extra costs in the specification phase of an evolution project due to the effort required to collect knowledge from people and available documents. This also causes costs due to reworks caused by the delivery of changes not completely and correctly analyzed during the specification phase.

The above considerations show that large industrial information systems may share a large part of the properties stated by Nielsen [3] for SoSs.

We support the idea that a SoS view of industrial information systems has value for the information systems managers, business analysts, architects, developers and testers. Therefore, is not enough to apply software engineering concepts, practices and tools to single applications and systems. We need to apply an engineering approach at the larger scale of the whole information system as a System of Systems.

In the following case study, we apply these ideas to a specific information system.

The case study deals with the problem of maintaining a documented knowledge of the information system at the SoS level of abstraction. At this level, the system is viewed as a network of software applications and systems and business processes emerge because of relations between these systems.

The aim of the study is not only to present a specific case but also to show how this approach may improve the management practice of information systems.

Lack of documented knowledge is a common practice in information systems, but the problem is typically considered at the level of single applications or projects [5].

We show that this knowledge level does not consider critical aspects that are arising from the complexities of modern information systems. Therefore, the explicit consideration and modeling of the SoS level mitigate critical problems of information system evolution and may advance the state-of-the-art of information systems management.

III. Case Study

The case study concerns the information system of a large Italian retail company. The company manages about 100 physical stores and an e-commerce store. The information system is composed of about 70 applications and 2 large databases. About one-half of applications is custom-developed (about 1.500.000 LOC of Java-JSP and RPG code). Software products acquired from vendors and adapted/integrated compose the remaining half.

Examples of the constituent software applications/systems: selling in a physical store, manage e-commerce orders, transport management, warehouse management.

During the years, the amount of COTS/adapted applications increased and the global landscape of the architecture moved from a shape similar to repository-style (databases and a cloud of applications working on the databases) to a shape more similar to a network.

Consequently, the number of business processes involving more applications and the number of projects with more than one development team or vendor increased. During the last year, out of 54 evolutionary projects, 21 were composed of more than two different organizations, with the maximum of 7.

This evolutionary process increased the problems caused by the lack of a global view of the information system (a SoS view). Projects requiring the implementation or change of business processes need the cooperation of different teams. Each development/vendor team has a local knowledge (a specific application or a limited set of applications), but is difficult for each team to have a deep understanding of the interactions implementing the business processes. Business process analysts have difficulties to specify the required changes, because the global model of business processes and their interaction with the information system components is fragmented into many documents and oral tradition. Testers have the same problems.

The IT organization established an "Architecture office" whose aim is to develop feasibility studies, design the changes and deliver architectural and technical standards for each development team or vendor. This office too needs a global view of the information system.

The management of the information system evaluated that the priority to mitigate these problems was the availability of documented knowledge concerning the set of applications, their relationships and the impact of business processes on the network of software applications (a model of the whole information system at SoS level). The aim is that the organizations running the projects will share this knowledge and the different stakeholders involved in the information system evolution will use it as reference.

IV. THE GOAL

The information system maintains a wiki-based documentation system storing the most important knowledge for each application and the data models, but the documentation models each application as a relatively independent system and the interaction with other systems is limited to definition of interfaces.

We focused our work to add a new layer of documentation based on a model of the most important aspects of the whole information systems architecture at the abstraction level of SoS.

The model will support each project with global views of the whole set of components and behaviors implementing the information system.

Different group of users can benefit of it:

- Business process analysts
- Information system architects
- Development teams
- Testers

Business process analysts and information system architects will use the model to understand the existing business processes and their implementation in the various systems composing the whole information system. The model will support the feasibility studies to evaluate the effort required for implementing a new or changed business process. It will also support the impact assessment of technological changes (for example porting an application on a new software infrastructure requires the understanding of the interacting applications) or the design of the architectural solution to implement a business process change.

The development team will use the model to understand the high-level view of structure and behaviors of the systems to modify. The testing team will also use the model to implement the testing scenarios.

V. THE MODEL

In this section, we describe the characteristics of the implemented model.

A. Minimalist documentation

IT organization developed the documentation system using a *minimalist* approach [6] and the method defined in our previous work [8]. The method select a *minimal set* of documents that can support the information system actors during the evolution process.

The basic idea of the minimalist documentation is that knowledge is a property emerging from a system where people and documents interact. Documents do not require being "complete" (rich of details in each part), it is sufficient that they be "good enough" [7]. For example, developers use an architectural document as a "landscape" for understanding where to read the details in the code.

During the evolution phase, we must update each document and this need generates a cost. Even the lack of a document may generate a cost: the effort of software understanding during the evolution. A *minimal set* is a set of documents showing the most efficient cost-benefit ratio of the maintenance cost of the documents and the costs that can be generated if the documents are not available. Details of the cost-benefit selection rules may be found in [8]. Therefore, we select and maintain only this set of documents during the evolution.

For example, an architectural model (a document with a low number of pages) has a limited cost of maintenance (measured in number of pages) whereas the lack of this model may generate a high cost required to extract the architectural knowledge from code. Consequently, we should maintain this document during the evolution.

On the contrary, for example, a detailed specification of interactive screens (many pages) has a high cost of maintenance whereas his absence generates a low cost (the programmer may understand the specification observing the behaviors of the implemented software product). Therefore, we should not maintain this document during the evolution.

To select the SoS models, we followed the same minimalist approach used for the existing documentation.

The selection was decided during the meetings with people of the "Architecture office". They decided that the most important documents at SoS level (using the cost/benefit criteria previously considered) were the SoS architecture and the relation between the main business processes and the architecture. They judged the cost of developing and maintaining these documents significantly less than the costs caused by the need to recover the necessary knowledge and the extra costs caused by poor knowledge during each evolution projects.

The first step was to list the more important business areas. Examples of business areas are:

- Selling in the physical store;
- Selling in the e-commerce store;
- Supply chain;
- Manage product data and prices.

For each business area, we developed:

- A static model describing the software applications used inside the business area (components) and the relations between the applications (connectors). The modeled relations are data flows between components (only the flows of the specific business area).
- Dynamic models describing how the set of components and connectors implement the main business processes of the area (activities of components and flows of data between activities).

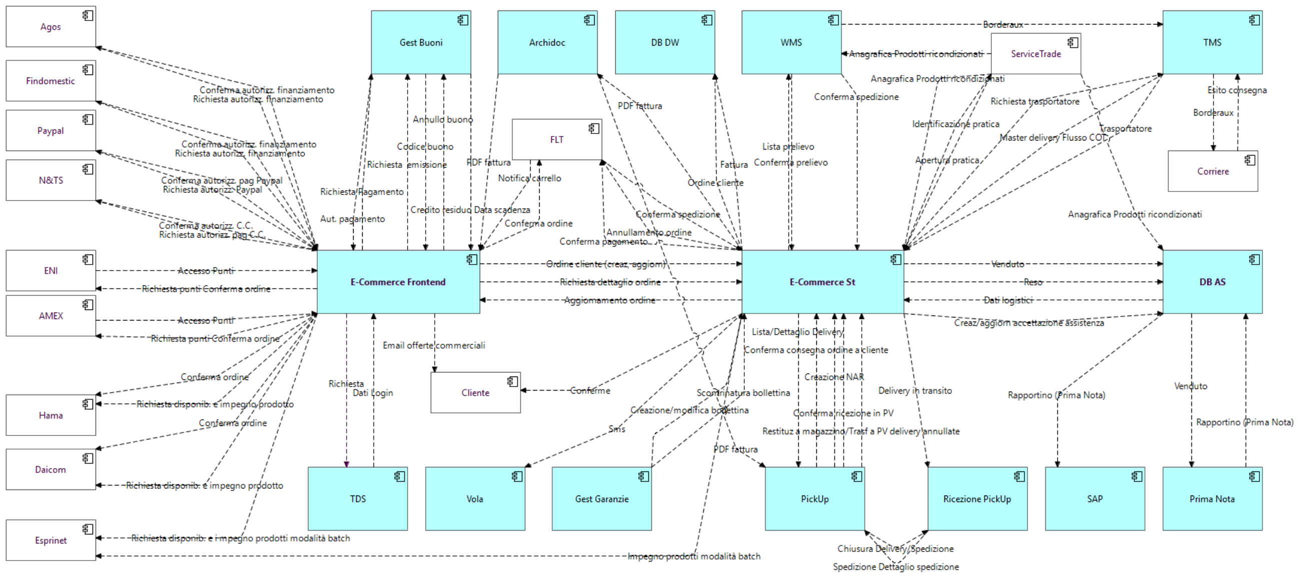

Fig. 1. Static model of the business area “Selling in the e-commerce store”

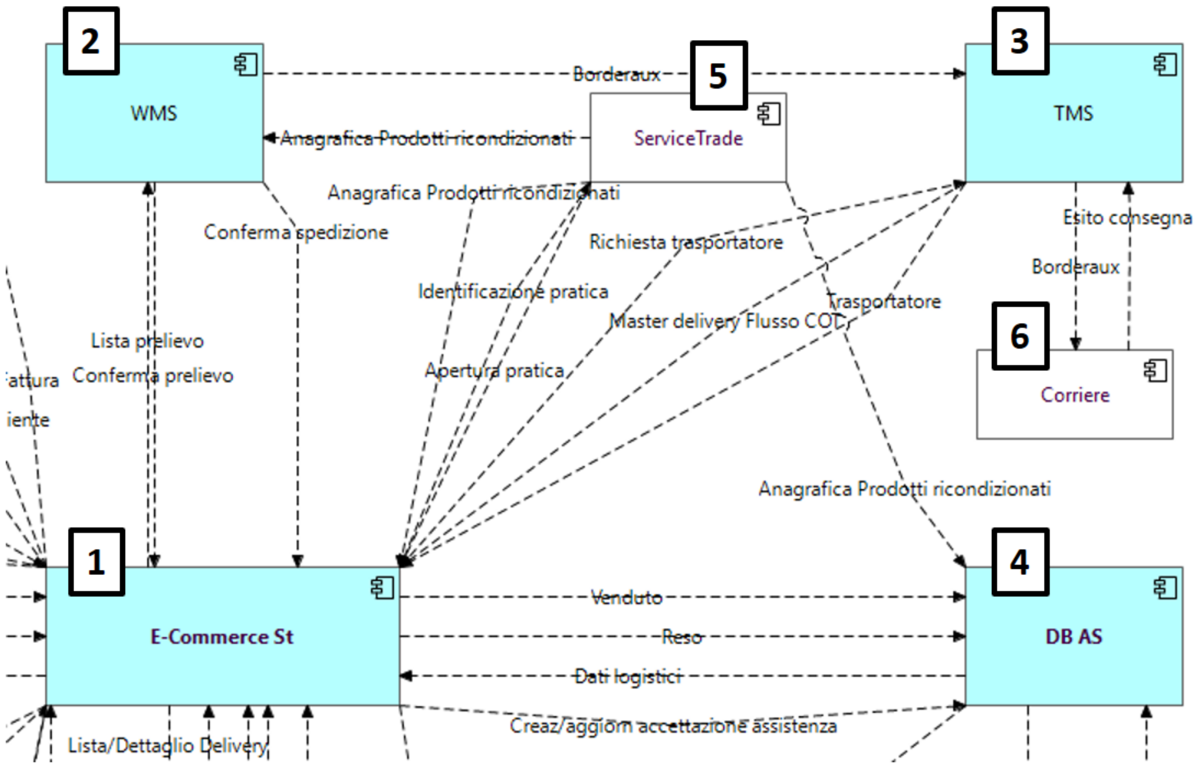

Fig. 2. Static model of the business area “Selling in the e-commerce store” - detail

For example, fig 1 shows the static model of the business area "Selling in the e-commerce store". Boxes are software applications (components) and arrows are connectors. The label of each connector is the name of the data flow. The business area includes 28 software applications. 13 of them (boxes with white background) are external to the organization.

Fig 2 shows a detail of the model. In this fragment, four systems (1 – order management system, 2 – transport management system, 3 – warehouse management system and 4 – operational database) collaborate exchanging the data flows described by oriented arcs. The remaining two systems (5 – technical assistance for the sold products and 6 – courier service) are part of other information systems.

Each system is a relatively independent component and has its own functions and role in the organization, but the e-commerce business processes emerge from the interaction of the constituent systems.

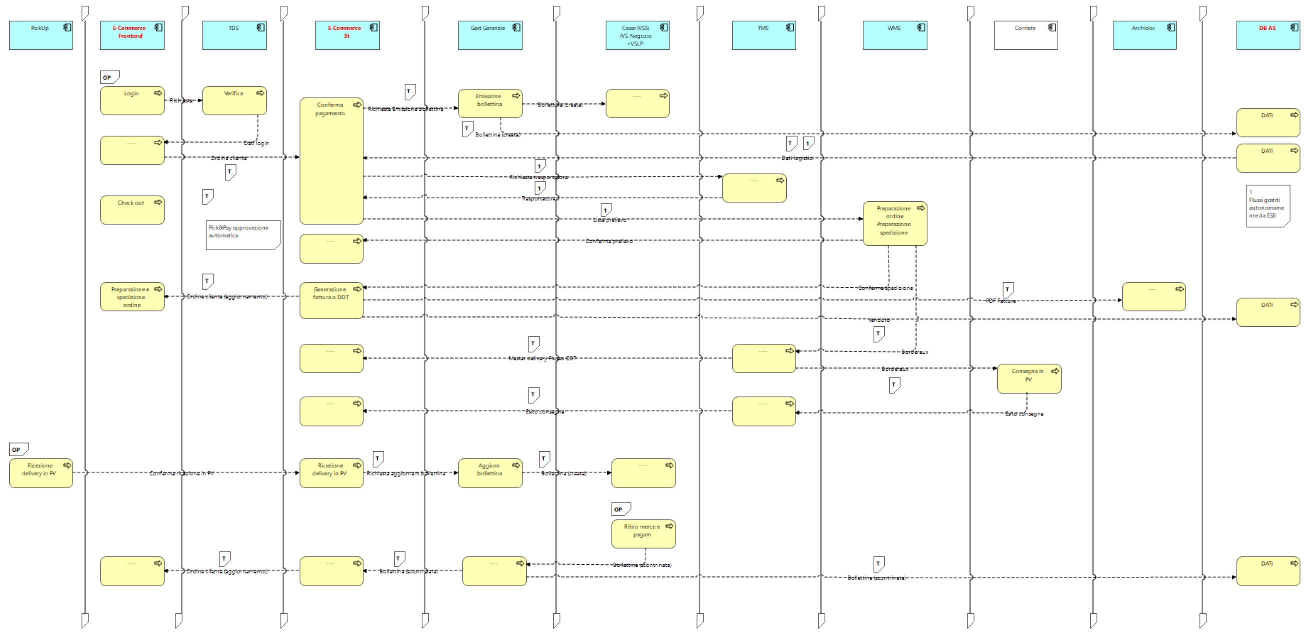

Fig. 3. Dynamic model: "Buy with the e_commerce site, pick and pay in the physical store"

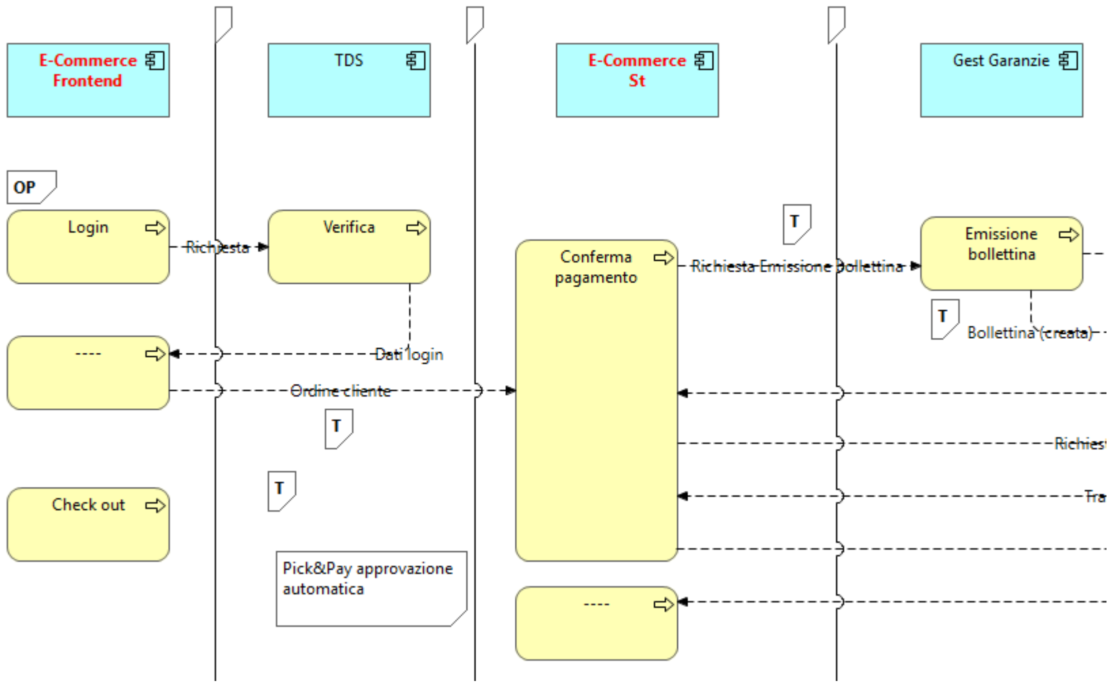

Fig. 4. Dynamic model: "Buy with the e_commerce site, pick and pay in the physical store" - detail

During the evolution, both the functions of single systems and the business processes emerging from the interaction of more systems may evolve.

For this static model, we developed 41 dynamic models describing the most important business processes implemented by part of components and connectors of the static model.

Figure 3 and 4 show one of the dynamic models "Buy in the e_commerce site, pick and pay in the physical store" and a detail of the model. The model represents one of the behaviors emerging from the interaction among the set of constituent systems of the SoS. Considering all the application of fig.1, eleven of them cooperates for this business process.

Each swim lane is associated to the component on top of the diagram (a subset of the static model). Rounded boxes describe activities of the business process. Some activities are labeled with the activity content, while some others are not specifically identified. Labelled arrows are data flows. Additional icons add information about the control flow (for example interactive activity – the "OP" icon or periodically scheduled – the "T" icon). The model describes a partial ordering (from top to bottom, from left to right) of the application data flows. The ordering is partial due to the icon "T" that identifies a timed schedule with the time constraint not defined by the model. According to the *minimalist* approach the model is not complete at a defined level of abstraction but is "good enough" to help people understanding the business process.

B. Tool

We implemented the model with the tool Archi [10]. The tool allows defining a database of model elements (for example components and relations) and apply different views to the database. Fig 5 is the global static view. This view includes all the components of the different business areas with their data flows. We may query and navigate the model to explore the model elements, relations and attributes (name-value pairs) associated to them.

VI. Model development

We developed the model with the "Architecture office", a team of three people, the most experienced professionals of the information system with the widest available knowledge on the existing systems and business processes. The development also involved three experienced people of the testing team that accumulated, during the testing projects, a significant knowledge of systems and business processes. Another important source of knowledge was the documentation system that maintains the basic knowledge for each application composing the information system. This was specifically useful to define the interfaces of each application.

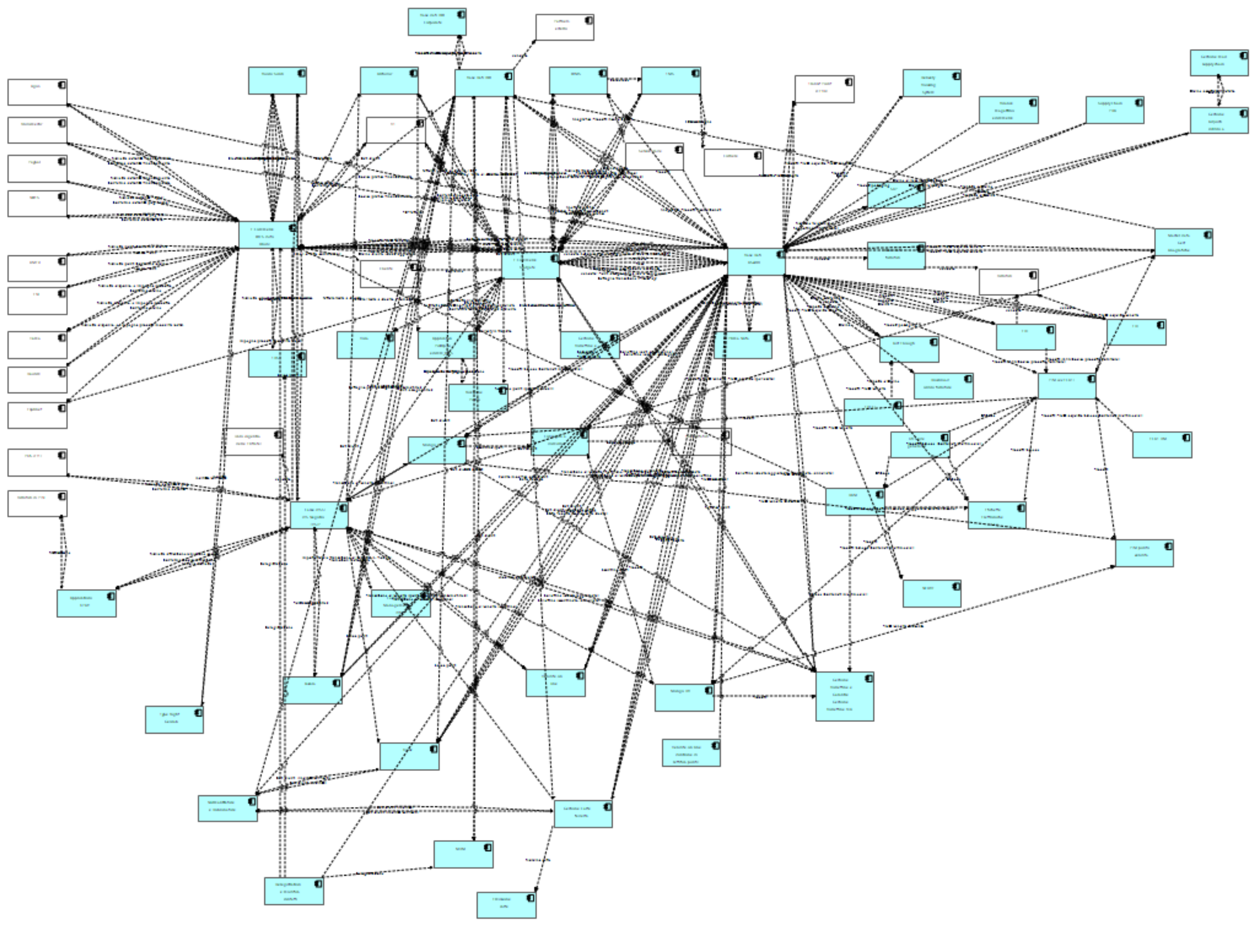

Fig. 5. Static view of the whole information system

The development required about 60 days distributed in a period of about six months.

A significant problem we found during the construction of models was that, due to the fragmentation of knowledge, even the architecture office people and the testers had problems in defining precisely all the business processes and the interactions between systems. Consequently, in many cases, we designed a draft of the model and verified/improved it with the help of project leaders knowing specific parts of the information system. Therefore, the development required many review cycles to verify and refine the models.

The result was a body of knowledge previously not available. No single development team or expert had these integrated views of applications and business processes.

VII. LESSONS LEARNED

Are there real evidences that the models delivered significant benefits to the management of the information system?

The model in now used by the “Architecture office” for the feasibility studies. The office interacts with business people requiring business processes changes, new functionalities or new processes.

The model allows the understanding of the existing business processes and the implementation of them into the systems and is the basis to design the changes and evaluate the impact. A more detailed knowledge related to each system may be found in the pre-existing documentation system.

Previously, the analysis required a costly and risky work of collecting fragments of knowledge from different people to reconstruct the set of involved systems and their interactions.

The development teams had benefit of the model. It is now the basic documentation to understand the systems and the required changes. The IT organization also used the model for training new development teams (for example in the e-commerce business area). The inclusion in complex projects of a new working group was difficult. New people slowly accumulated a more global knowledge from the experience of implementing changes of SoS parts. Now a new team is trained through the explanation and discussion of the SoS models. This decision mitigated the difficulties, increased the new team’s efficiency and reduced the risk of code defects caused by poor knowledge of the whole context.

The testing team also used the process models. For example, the models of the business area “Selling in the e-commerce store” were the reference to develop a set of about 100 automatic non-regression scenario-based tests. A set of test cases verifies each modeled process (the test procedures run each of the 41 dynamic models with some variations of input data). Each test case follows the process through all the involved systems, from an initial set of input data (for instance an order from a customer) to the complete interaction (and intermediate verifications) with all the systems.

Were these models enough to represent the studied SoS?

The static and dynamic models were judged, during the development phase, the priority knowledge of the SoS.

Obviously, the idea of maintaining only a minimal set of documents has its own assumptions and limitations. These limitations come from the decision to select the modeled knowledge according to cost-benefit criteria and reduce the maintenance effort of the model during the SoS evolution.

For example, static models only define data flows and lack of views for technical information (the implementation views of connectors). Consequently, for example, the model does not provide the information if a flow of data is implemented by a simple invocation with parameters or the execution of a complex publish-subscribe pattern delivered by the enterprise bus.

VIII. CONCLUSION

In the paper, we presented the development of models of an information system based on the view of Systems of Systems. In our experience, this type of models showed significant benefits for the management of the information system evolution.

We developed the first version of the model with a limited cost and we estimate that the periodical updating of the model will require relatively moderate costs.

On the contrary, we estimate that the availability of the model saved significant extra costs of many projects due to the lack of this knowledge.

Furthermore, a significant question is the scope of our experience. We studied a single information system of a specific industrial sector. Beside direct experience, we base our confidence that the presented results can generalize to other information systems on the following considerations.

The experience concerns a large information system that is representative of many others. It includes many software applications, large databases and a growing set of COTS-based components. In this context is usual the cooperation of different teams internal to the IT organization, outsourced to external software houses or coming from product vendors.

The difficulty of maintaining a useful documentation of the information system and the knowledge fragmentation is a widespread and recognized problem by the IT managers. Consequently, the SoS-based explicit knowledge we presented may mitigate a common problem in industrial information systems.

Furthermore, we think that the real problem for a widespread use of these approaches based on the SoS point of view is not the cost. The cost of developing and maintaining the models is low if compared to the extra costs caused by the lack of this knowledge (costs of understanding the existing software, costs caused by poor quality of the developed solutions and need of reworking). The key point is the management culture and the need to introduce more mature management approaches based on sound software engineering practices and cost/benefit evaluations.

A future improvement could be the use of models and tool to control the growing complexity of the information system structure. One of the aims of the architecture office is to publish the design rules of the information system architecture and to enforce their adoption. Therefore, the office can use the models and the tool to control the architectural erosion during the evolution process.

For example, we can navigate the structural models to measure the coupling between components. We also may add technical attributes to the data flow oriented arcs. A useful information may be the type of connectors. This can allow controlling the design rules for the choice of connectors, discovering violations and refactoring the architecture.

Clearly, this is not an easy task. If we examine figure 5, in this architecture, even if the local systems are carefully designed, the resulting SoS does not support none of the basic good design principles (for example cohesion and coupling rules) we teach in university courses of software engineering.

IX. Related Work

Software engineering architectures developed a rich set of concepts, practices and tools [11]. SoS research extended these concepts to environments where the delivered functionalities and processes result from the composition and interaction of many relatively independent systems [12].

In software intensive SoSs, that is SoSs in which software plays an essential role in design, development, and evolution, it is recognized that an adequate representation of SoS software architectures is important [12]. In this context, different architecture description languages (ADLs) have been proposed like UML or SysML [13]. The definition of the most suitable ADLs for SoS is still a research subject [12].

In the area of enterprise information systems architectural frameworks, as for example TOGAF [14], were developed enterprise models considering various aspects of the enterprise: business functions and processes, data, software applications and the underlying technical infrastructure. Archimate [9] is a standard modeling language to support the description, analysis and visualization of enterprise architectures, taking into account multiple aspects of architectural frameworks. In our industrial application, we used this modeling language to apply the concepts of SoS to the considered information system. The reason of the choice was that the language is a recognised standard in the information systems community and has expressive power to model different aspects like business processes and software architectures. These aspects are important for the different users of the models: business analysts, information systems analysts, architects, developers and testers.

Application of SoS, originally identified in the defense environment, is now much broader and still expanding [15]. The relevance of SoS concepts for any organization, public or private, seeking to attain competitive advantage through leveraging of information technology systems has been early recognized [16]. In spite of that, as far as we know, the application in the context of industrial information system is not a common practice.

What is significant in our experience report is not the use of novel concepts, methods or tools, but the example of application of SoS to a field that can strongly benefit. Our experience report is specific, but the structure of the information system and the management problems coming from the complexities of many interacting and relatively independent software applications are common to many IT departments of other industrial sectors.

References

[1] J. Boardman and B. Sauser, "System of systems - the meaning of *of*". In Proceedings of IEEE/SMC Int. Conf. on System of Systems Engineering, pp. 118-123, 2006.

[2] I. G. Vargas, T. Gottardi, and R. T. Vaccare Braga, "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 '16). ACM, pp. 32-38, 2016.

[3] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, J. Peleska, "Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions". In ACM Comput. Surv. 48, 2, pp. 1-41, 2015.

[4] B. Merkle, "Stop the software architecture erosion". In Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion, OOPSLA '10, ACM, pp 295–297, 2010.

[5] D. Oprea, G. Mesnita, The Information Systems Documentation - Another Problem for Project Management. Managing Information in the Digital Economy: Issues and Solutions. In Proceedings of the 6th Ibima Conf., Khalid S. Soliman, ed., pp. 332-338, IBIMA, 2006

[6] J. M. Carroll, "The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill". MIT Press, Cambridge, MA, USA, 1990.

[7] A. Rüping, "Agile Documentation - A Pattern Guide to Producing Lightweight Documents for Software Projects", John Wiley & Sons, 2003.

[8] P. Salvaneschi, "The evolution of Information Systems a case study on document management". In Proceedings of IEEE 27th International Conference on Software Maintenance, ICSM, Williamsburg, VA, USA, pp 428 - 437, 2011.

[9] ArchiMate® 2.1 Specification 2012-2013 The Open Group. http://www.opengroup.org.

[10] https://www.archimatetool.com/

[11] P. Kruchten, H. Obbink and J. Stafford, "The Past, Present, and Future for Software Architecture", IEEE Software, Vol. 23, Issue 2, pp. 22-30 , March-April 2006.

[12] F. Oquendo, "Software Architecture Challenges and Emerging Research in Software-Intensive Systems-of-Systems", Proceedings of Software Architecture 10th European Conference, ECSA, pp. 3-21, 2016.

[13] M. Guessi, E. Cavalcante, L. B. R. Oliveira, "Characterizing Architecture Description Languages for Software-Intensive Systems-of-Systems". In Proceedings of IEEE/ACM 3rd International Workshop on Software Engineering for Systems-of-Systems SESoS '15, pp 12-18 2015.

[14] TOGAF® Version 9.1, http://www.opengroup.org

[15] FP7 CSA Road2SoS (Roadmaps to Systems-of-Systems Engineering): Survey on Industrial Needs and Benefits of SoS in Different SoS Domains: Multi-site Industrial Production Manufacturing, Multi-modal Traffic Control, Emergency and Crisis Management, Distributed Energy Generation and Smart Grids. http://road2sos-project.eu/

[16] P.G. Carlock, R.E. Fenton, "System-of-Systems (SoS) Enterprise Systems for Information-Intensive Organizations," Systems Engineering, Vol. 4, No. 4, pp. 242-261, 2001