**KnowSoS: A Software Architecture for Knowledge Discovery
in Systems-of-Systems**

Bruno Sena da Silva

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências
de Computação e Matemática Computacional (PPG-CMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Bruno Sena da Silva

KnowSoS: A Software Architecture for Knowledge Discovery in Systems-of-Systems

Master dissertation submitted to the Institute of
Mathematics and Computer Science – ICMC-USP,
in partial fulfillment of the requirements for the
degree of Master of Science – Computer Science and
Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Profa. Dra. Elisa Yumi Nakagawa

USP – São Carlos
March 2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

S474k Sena, Bruno
KnowSoS: A Software Architecture for Knowledge
Discovery in Systems-of-Systems / Bruno Sena;
orientador Elisa Yumi Nakagawa. -- São Carlos, 2020.
127 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2020.

1. Systems-of-Systems. 2. Knowledge Discovery.
3. Software Architecture. 4. Big Data. I. Nakagawa,
Elisa Yumi, orient. II. Título.

Bruno Sena da Silva

KnowSoS: Uma Arquitetura de Software para a Descoberta de Conhecimento em Sistemas-de-Sistemas

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestra em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Elisa Yumi Nakagawa

USP – São Carlos
Março de 2020

To my beloved mother, Rose, for everything she sacrificed to get me here.

Acknowledgements

Before starting this Master's program, I thought it would be easy. I thought I knew everything. I thought I was ready. I thought it was going to be simple. I thought it was going to be quick. But, over time, I realized that this journey was much more than just developing a project. This journey was about personal growth. It was about intellectual development. It was about the humility of accepting that I don't know anything. It was about learning to deal with different types of people. It was about receiving "no" and not giving up. But above all, it was about strengthening bonds with those who were by my side, and it is to these people that I offer my sincere gratitude.

First of all, to God, for giving me wisdom to embrace opportunities, patience to await victories, and strength to face difficulties. My heart will be forever grateful for your compassion.

To my mother, Rosemeire, for being by my side at all times, for laughing and crying with me, for the advice when I needed it and for believing that one day I would be here. You're everything to me!

To my whole family, and especially, to Talita Fontanezi and Feliphi Sena, for the company, love and words of affection.

To my advisor, Elisa Yumi Nakagawa, for adopting me when I knew nothing, and teaching me much of what I know today.

To my best friend, Carolina "Buzz", for comforting me in every moment of crisis, for not letting me give up at any time, and for always using the right words to support me. You are the exactly definition of what a best friend should be.

To the friends and co-supervisors that this journey gave me, Maria Lydia Fioravanti and Ana Paula Allian, who advised me, guided me, and supported me throughout my (eternal) drama. I couldn't have done it without you.

To the friends who were patient with me, even when I didn't deserve it, Ana Paula Bertho, Camila Dias de Oliveira, Pedro Henrique do Valle, and Raul Donaire. To my colleagues at START and LabES, who helped along the project and made this journey easier.

To the support of Brazilian funding agencies: Grant 2017/22237-3, São Paulo Research Foundation (FAPESP), and Grant 132691/2018-3, Coordination for higher Education Staff Development (CAPES), that allowed the goals of this project to come true.

Finally, I express my sincere gratitude to all those who have contributed, directly or indirectly, to the development of this work.

First, think. Second, dream. Third, believe. And finally, dare. All our dreams can come true, if we have the courage to pursue them.

Walt Disney

Abstract

SENA, B. **KnowSoS: A Software Architecture for Knowledge Discovery in Systems-of-Systems**. 2020. 127 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Systems-of-Systems (SoS) have been playing an important role in the industry and academia as an answer to the ever growing complexity of software systems. This class of systems emerges from the interoperability of operational and managerial independent systems (called constituents) that work together to achieve more complex missions, not achievable by any of them individually. Constituents of an SoS generate at runtime a huge amount of data with high velocity and that is sometimes highly variable, characterizing a big data scenario. However, SoS have not taken advantages from the knowledge that could be discovered from such data, which could provide an overall understanding of the SoS behavior, support missions fulfillment, and enable the system to react to it. In this scenario, the main contribution of this Master's project is to provide means to support knowledge discovery in SoS, that is, to support the entire data cycle in SoS, from the collection in the constituents to the extraction of knowledge that will be useful for the system operation. For this, we propose KnowSoS, a Software Architecture for Knowledge Discovery in Systems-of-Systems, which assists in the design of a system that can be integrated into the SoS to control data flow and enable knowledge discovery. Besides that, we propose a set of guidelines that support SoS understanding and the development of the system designed by using KnowSoS. To evaluate this proposal, we present a proof of concept using the architecture and the guidelines to design and develop a knowledge discovery system in two different SoS domains, i.e., human resources and emergency management domain. As main results, the system improved the recommendation of job openings to candidates who are looking for a job in the human resources domain, and the precision of alerts for natural disasters to emergency stakeholders (i.e., police, hospitals, and firefighters) in the emergency management context. Such results indicate that, even as an initial approach, KnowSoS can be used to bring benefits in different domains where knowledge discovery is necessary to fulfill SoS missions. As future works, we intend to apply this approach in new domains and conduct more robust experimental studies.

Keywords: System-of-Systems, Software Architecture, Knowledge Discovery, Big Data.

Resumo

SENA, B. **KnowSoS: Uma Arquitetura de Software para a Descoberta de Conhecimento em Sistemas-de-Sistemas**. 2020. 127 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Sistemas-de-Sistemas (do inglês, Systems-of-Systems, ou SoS) têm desempenhado um papel importante na indústria e na academia como resposta à crescente complexidade dos sistemas de software. Essa classe de sistemas emerge da interoperabilidade de sistemas que são operacionalmente e gerencialmente independentes (chamados de constituintes), e que trabalham juntos para alcançar missões complexas, impossíveis de serem alcançadas por qualquer um deles individualmente. Os constituintes de um SoS geram em tempo de execução uma enorme quantidade de dados com alta velocidade e, em muitas vezes, alta variabilidade, caracterizando um cenário de big data. No entanto, muitos desses sistemas não estão utilizando o conhecimento que poderia ser descoberto a partir desses dados, o que poderia fornecer uma compreensão geral do comportamento do SoS, apoiar na realização das missões, e permitir que o sistema reagisse a tal conhecimento. Nesse cenário, a principal contribuição deste projeto de mestrado é fornecer meios para apoiar a descoberta de conhecimento em SoS, isto é, apoiar o ciclo completo dos dados em SoS, desde a coleta dos dados nos constituintes até a extração de conhecimento que poderá ser útil para a operação do sistema. Para isso, é proposta KnowSoS, uma Arquitetura de Software para Descoberta de Conhecimento em SoS, que auxilia no design de um sistema que será integrado no SoS para controle do fluxo de dados e a descoberta de conhecimento. Além disso, nós propomos um conjunto de guidelines que auxiliem no entendimento do SoS e no desenvolvimento do sistema construído por meio da KnowSoS. Para avaliar essa proposta, é apresentada uma prova de conceito usando a arquitetura e as guidelines para definir e desenvolver um sistema de descoberta de conhecimento em dois domínios diferentes, o domínio de recursos humanos e gerenciamento de emergências. Como principais resultados, o protótipo melhorou a recomendação de vagas para candidatos que procuram emprego no domínio de recursos humanos e a precisão de alertas de desastres naturais para as entidades interessadas em crises (ou seja, policiais, hospitais e bombeiros) no contexto de gerenciamento de emergências. Tais resultados indicam que, mesmo como uma abordagem inicial, KnowSoS pode ser usado para trazer benefícios em diferentes domínios em que a descoberta de conhecimento é necessária para cumprir as missões de SoS. Como trabalhos futuros, espera-se aplicar essa abordagem em novos domínios e realizar estudos experimentais mais robustos.

Palavras-chave: Sistemas-de-Sistemas, Arquitetura de Software, Descoberta de conhecimento, Big Data.

List of Figures

Figure 1 – SoS types and its inherent arrangement.	32
Figure 2 – A conceptual model for SoS.	33
Figure 3 – Quality attributes for SoS.	38
Figure 4 – An unmanned aerial vehicle architecture	40
Figure 5 – An electric road SoS	41
Figure 6 – Data-information-knowledge hierarchy	42
Figure 7 – Fayyad process.	44
Figure 8 – CRISP-DM process.	45
Figure 9 – Number of studies per type of publications, years of publication, and authors	47
Figure 10 – Wordcloud of the studies.	48
Figure 11 – Number of studies that addressed each quality requirement.	50
Figure 12 – General process of KDD in SoS.	55
Figure 13 – Architectural Design Activities.	60
Figure 14 – Domain Requirements	64
Figure 15 – Usecase View	68
Figure 16 – Process View	70
Figure 17 – Architecture Overview	71
Figure 18 – Data Collection Layer	73
Figure 19 – Data Storage Layer	74
Figure 20 – Discovery Layer	75
Figure 21 – Decision Making Layer	76
Figure 22 – Orchestrator Layer	77
Figure 23 – Guidelines for using KnowSoS.	79
Figure 24 – Workflow of Strategy 1	87
Figure 25 – Workflow of Strategy 2	87
Figure 26 – Descriptive of jobs and users	88
Figure 27 – Knowledge Discovery System Architecture proposed for Human Resources SoS	90
Figure 28 – Wordcloud of Jobs.	91
Figure 29 – Descriptive of jobs and users	92
Figure 30 – Data collection window.	98
Figure 31 – Knowledge Discovery System Architecture proposed for Emergency Man- agement SoS	100

Figure 32 – Amount of messages detected through the days. 102

Figure 33 – Amount of “high impact” messages through the days. 104

Figure 34 – Search conduction results. 125

List of Tables

Table 1 – Systems-of-Systems vs. Monolithic systems. Extracted from Oberndorf and Sledge (2010).	30
Table 2 – Patterns used for integration	52
Table 3 – KnowSoS Requirements	67
Table 4 – Recommendations using Strategy 1	94
Table 5 – Recommendations using Strategy 2	94
Table 6 – Recommendations using Strategy 3	95
Table 7 – Statistics of each dataset	95
Table 8 – Results in the total dataset and in each cluster	96
Table 9 – 10-fold cross validation: 1st Execution	102
Table 10 – 10-fold cross validation: 2nd Execution	103
Table 11 – 10-fold cross validation: 3rd Execution	103
Table 12 – Final list of primary studies	126

Contents

1	Introduction	21
1.1	Problem Statement	23
1.2	Objectives and Contributions	23
1.3	Dissertation Outline	24
2	Background	27
2.1	Systems-of-Systems	28
2.1.1	Classification of Systems-of-Systems	30
2.2	Systems-of-Systems Software Architectures	34
2.2.1	Software Architectures	34
2.2.2	SoS Software Architectures	35
2.2.3	Quality Attributes	37
2.2.4	Examples	39
2.3	Knowledge Discovery	41
2.3.1	Fayyad Process	43
2.3.2	CRISP-DM Process	45
2.4	Knowledge Discovery in SoS: A Systematic Mapping	46
2.4.1	Results	46
2.4.2	Discussions	55
2.5	Final Remarks	58
3	KnowSoS: A Software Architecture for Knowledge Discovery in Systems-of-Systems	59
3.1	Method	59
3.2	KnowSoS	61
3.2.1	Architectural Analysis	61
3.2.2	Architectural Synthesis	67
3.3	Guidelines for using KnowSoS	77
3.3.1	Phase 1: Identification of the SoS constituents	78
3.3.2	Phase 2: Identification of the SoS Missions	79
3.3.3	Phase 3: Selection of the Knowledge Discovery Approach	80
3.3.4	Phase 4: Application of KnowSoS	80
3.3.5	Phase 5: Validation of the Knowledge System	81
3.4	Final Remarks	81
4	KnowSoS Evaluation	83

4.1	Human Resources Domain	84
4.1.1	Phase 1: Identification of the SoS constituents	84
4.1.2	Phase 2: Identification of the SoS missions	85
4.1.3	Phase 3: Selection of the Discovery Approach	86
4.1.4	Phase 4: Application of KnowSoS	88
4.1.5	Phase 5: Validation of the Knowledge System	92
4.2	Emergency Management Domain	96
4.2.1	Phase 1: Identification of the SoS constituents	96
4.2.2	Phase 2: Identification of the SoS missions	97
4.2.3	Phase 3: Selection of the Discovery Approach	98
4.2.4	Phase 4: Application of KnowSoS	99
4.2.5	Phase 5: Validation of the Knowledge System	100
4.3	Final Remarks	104
5	Conclusions	107
5.1	Research Contributions	108
5.2	Research Limitations	108
5.3	Future Works	109
5.4	Resulting Publications	109
	References	111
	APPENDIX A Knowledge Discovery in Systems-of-Systems: A Systematic Mapping Protocol	123
A.1	Planning	123
A.1.1	Research Objectives and Research Questions	123
A.1.2	Search Strategy	124
A.1.3	Selection Criteria	124
A.1.4	Selection of Studies	124
A.2	Threats to Validity	125

Introduction

OVER the past years, there has been a growing interest in a class of complex systems whose constituents are themselves complex (JAMSHIDI, 2008). Interoperability among a group of heterogeneous systems aiming at achieving a common goal has become the focus of various application domains, including military, security, space, manufacturing, aviation, environmental systems, and disasters management (WOJCIK; HOFFMAN, 2006; BIANCHI *et al.*, 2015; MORDECAI *et al.*, 2016; BAYRAK *et al.*, 2018). In this context, Systems-of-Systems (SoS)¹ have been continuously growing in popularity, due to their complexity and adaptive characteristics (JAMSHIDI, 2011; VERÍSSIMO, 2006; BONDAVALLI *et al.*, 2016). SoS are complex systems composed of a collection of distributed systems (constituents), which gather their resources and capabilities together to provide new and enhanced services aiming at accomplishing missions that are not feasible by any of the constituents working independently (ISO/IEC-24765, 2010; NIELSEN *et al.*, 2015; MITTAL; RAINEY, 2015; ZOPPI *et al.*, 2017).

Maier (1998a), DeLaurentis (2005) and Nielsen *et al.* (2015) define SoS as a type of complex systems that have five characteristics: (i) operational independence: each constituent has its individual purpose and continues to operate even disassembled from the SoS; (ii) managerial independence: each constituent is developed and maintained by independent organizations; (iii) evolutionary development: SoS development is evolutionary and adaptive, in which structures, functions, and purposes change according to the needs of the SoS; (iv) distribution: constituents are dispersed and must provide a communication protocol to interoperate to the SoS; and (v) emergent behavior: behavior of the SoS emerges as a result of the interdependent collaboration of its constituents.

This scenario leads to the need of building SoS that must be developed and maintained in a way that they can evolve more easily, and in which the impact of changes must be minimized

¹ For sake of simplicity, SoS will be used interchangeably to express singular and plural.

by the operational and managerial independence of the constituents (NIELSEN *et al.*, 2015; ZOPPI *et al.*, 2017). However, the challenges for building SoS emerge due to the unpredicted collaborations among their constituents (DERSIN PIERRE E TRANSPORT, 2014). SoS must deal with the loss, inclusion, and replacement of constituents at runtime, besides adjusting their behavior to available resources (MAIER, 1998a). Important characteristics for the success of SoS, such as interoperability, maintainability, and dynamic evolution, can be accomplished with an appropriate software architecture (MICHAEL *et al.*, 2009) and a set of architectural patterns, which represent a pre-defined set of components and relationship among them and can help in structuring the basic characteristics of a SoS (INGRAM *et al.*, 2015; NICHOLS; DOVE, 2011).

Not only in SoS, but in several software systems, software architectures have had a fundamental role, promoting several characteristics of quality, e.g., interoperability, performance, portability, adaptability, and maintainability (BIANCHI *et al.*, 2015; NAKAGAWA *et al.*, 2015). Software architectures can be summarized as a structure (or a set of structures) of the systems that comprises software elements, the externally visible properties of those elements, and the relationships among them (BASS *et al.*, 2013). Software architectures are also the key element to design SoS, due to their support in SoS development, maintenance, and specially evolution. SoS software architectures also promote the inherent characteristics of such systems and surely their quality (NAKAGAWA *et al.*, 2015).

Besides the aforementioned characteristics of SoS (MAIER, 1998a; ISO/IEC-24765, 2010; NIELSEN *et al.*, 2015), SoS and their constituents have been also responsible for generating and managing huge amounts of data in several domains, such as medical domains in which SoS is composed of systems for diagnosis, treatment, and patient management (HATA *et al.*, 2007), control domains in which the SoS is composed of aircraft and satellites systems, petrochemical (HOLLAND *et al.*, 2005), transportation (ZHENG *et al.*, 2013; MORDECAI *et al.*, 2016; BAYRAK *et al.*, 2018), and smart cities (BATTY, 2012). This substantial volume of data generated with highly velocity in different formats characterizes SoS in the big data context (BEYER; LANEY, 2012). Big data research field has as main concern making useful utilization of a collection of data and acquisition of knowledge (UDDIN *et al.*, 2014). Such knowledge can be acquired by the application of machine learning techniques, optimization, statistics, and so forth through exploratory analysis, data modeling, and processes of identifying valid, novel, useful, and understandable patterns from the datasets (FAYYAD *et al.*, 1996b).

Several approaches have supported the discovery of knowledge from databases, aiming at turning raw data into knowledge that could be useful to the application domain (UDDIN *et al.*, 2014; KAISLER *et al.*, 2013). The most consolidated one is still that proposed by Fayyad *et al.* (1996b), which is an interactive and iterative process composed of nine steps: (i) understanding of the domain; (ii) data selection; (iii) pre-processing; (iv) dimensionality reduction; (v) definition of the problem; (vi) data mining technique selection; (vii) data mining execution; (viii) data interpretation; and (ix) consolidation of knowledge in the application domain. Besides that,

focusing on the industry, [Wirth and Hipp \(2000\)](#) proposed CRISP-DM, a more dynamical process focused on the business aspects of the data discovery and composed of six steps: (i) business understanding; (ii) data understanding; (iii) data preparation; (iv) modeling; (v) evaluation; and (vi) deployment. Furthermore, some architectural approaches have been proposed to handle the big data context and knowledge discovery in an architectural level, i.e., trying to support the design of systems which enable knowledge discovery ([SENA *et al.*, 2017](#)); however, few of these approaches show mature results to support such knowledge discovery in SoS.

1.1 Problem Statement

The main problem to be addressed in the Master's project is that, currently, SoS have not taken advantages of knowledge that could be discovered from data distributed in their several constituents. Such data contains knowledge that usually belongs to a specific constituent but it is not analyzed from the SoS perspective ([MITTAL; RAINEY, 2015](#)). If identified and managed, such knowledge could improve the process of decision making by supporting stakeholders to understand the macro behavior of the SoS and react to it. As an example in the health care domain, a context in which the constituents could be systems that manage the vital state of a patient, e.g., pressure, heart rate, temperature, the analysis of the data provided by such constituents could support the prediction of events such as a heart attack and then trigger a specific SoS behavior (e.g., release of some medication into the vein) before the event occurs. Another example is in an emergence domain, where the constituents must monitor natural disasters, i.e., hurricane, earthquakes, and floods. The early prediction of critical situations could support stakeholders to evacuate regions before the disaster happens, or identify regions where help is needed. Thus, to take more advantage from this amount of distributed data, SoS must support the discovery of knowledge from data of each constituent and manage it from the global perspective.

Considering the SoS context, previous works that address the discovery of knowledge from databases are not enough to handle its characteristics, such as those proposed by [Fayyad *et al.* \(1996b\)](#), [Wirth and Hipp \(2000\)](#), and other related works ([KURGAN; MUSILEK, 2006](#); [CABENA *et al.*, 1998](#); [CIOS; KURGAN, 2005](#)). There is still a lack of a systematic approach for the knowledge discovery in SoS, considering not only the process itself (i.e., data acquisition, pre-processing, and data mining), but also the architectural perspective, facilitating data collection from constituents and integrating the system responsible for knowledge discovery.

1.2 Objectives and Contributions

Motivated by the challenges and benefits aforementioned, the goal of this project is to provide means to support knowledge discovery in SoS. To achieve this goal, we proposed KnowSoS, a Knowledge Discovery Software Architecture for Systems-of-Systems. This architecture supports

the design of a system to be integrated into SoS and to perform knowledge discovery with data provided from SoS constituents.

To design KnowSoS, we followed a method proposed by [Hofmeister *et al.* \(2007\)](#), which is composed of three activities, namely, architectural analysis, architectural synthesis, and architectural evaluation. As result, three different views are proposed (use case, process, and logical view) to support the understanding of development and operation of the systems that will be responsible for knowledge discovery.

Besides that, we present a set of guidelines that supports the understanding of the SoS, the instantiation of KnowSoS and the development of the system proposed by instantiating KnowSoS. Such guidelines were defined through the collaboration of software architects, software engineers and knowledge discovery specialists.

The present dissertation contributes to the areas of SoS, software architecture, and others related to knowledge discovery. As major contributions, we expect that the proposed architecture will serve as a model for designing SoS architectures in any domain and can support both data collection, data integration, and knowledge discovery challenges. Furthermore, we hope that this proposal can be a starting point for the constant development of more robust and granular approaches and technologies that help solving such complex problems as the discovery of knowledge in this type of system.

1.3 Dissertation Outline

In this chapter, we presented the context in which this research is inserted, the motivations related to its achievement and its main objectives. The remainder of this master dissertation is organized as follows.

In Chapter 2, we provide an overview of the background information that supports the topics investigated in this work. Initially, basic concepts related to SoS and software architectures are discussed. Then, we present the results of a systematic mapping study, discussing which approaches have been proposed to address knowledge discovery in SoS and identifying which gaps this work is filling.

Chapter 3 presents KnowSoS, a software architecture to support the development of an application that enable knowledge discovery in SoS. Besides that, we detail the method used to create it and describe each step.

In Chapter 4, we describe the evaluation of our architecture. For this, we developed a proof of concept considering two different domains: an SoS in the job applications domain, and other in the emergency management domain. Then, we developed a prototype with real data provided by constituents and extracted knowledge in an specific context to identify the benefits of this approach in both domains.

Finally, in Chapter 5 we conclude this dissertation, revisiting the achieved contributions, summarizing limitations, and presenting perspectives of future research.

Background

AT the heart of every well-engineered software system there is a good software architecture, that is, the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution ([MEDVIDOVIC; TAYLOR, 2010](#); [KRUCHTEN *et al.*, 2006](#); [BOSCH, 2004](#)). The study of software architectures has greatly changed and matured over the years, emerging as a critical component as the size and complexity of systems increase ([TAYLOR, 2009](#); [RHODES *et al.*, 2009](#)). A class of systems that is raising in complexity is called Systems-of-Systems (SoS), i.e., systems composed by different software systems called constituents, interconnected to cooperate among themselves, delivering behaviors as a composition of individual functionalities they offer ([ACHESON *et al.*, 2013](#)). Architecting this class of systems has been considered a challenging endeavor since there is a need to effectively accomplish a variety of requirements from a diversity of stakeholders, besides dealing with dynamical behaviors and evolutionary development ([FITZGERALD *et al.*, 2012](#)). Moreover, with the ever-growing data that needs to be manipulated in this dynamic SoS context, the need for a system that is capable of efficiently processing this data and extracting useful knowledge to fulfill SoS missions and support their execution becomes even more necessary.

In this chapter we provide an overview of this class of systems that underlies the research developed in this work and present the state of the art on how knowledge discovery has been addressed in SoS. Section 2.1 introduces the main concepts of SoS, including definitions, classifications, and examples. In Section 2.2, software architecture concepts and some research initiatives on software architectures for SoS are discussed. Session 2.3 presents an overview of the current methods for knowledge discovery in monolithic systems. Section 2.4 discuss the results obtained from a systematic mapping that aimed to map the state of the art on knowledge discovery in SoS, identifying challenges and trends. Finally, Section 2.5 concludes this chapter with some final remarks.

2.1 Systems-of-Systems

Even though the many colloquial meanings of the term *system*, there is a widely accepted definition characterizing it as “a collection of components organized to accomplish a specific function or a set of functions” (RADATZ *et al.*, 1990). In recent years, due to the complexity of the developed systems, the concept of *Systems-of-Systems* has emerged, representing a set of systems working together to achieve a global objective. However, there is no universal characterization of this term, implying in a diversity of different definitions that depend on the area of study (DERSIN; TRANSPORT, 2014).

From the software perspective, SoS are built through the composition of constituents systems, which are independent and interoperable, working together to accomplish a common mission that is not achievable by any single constituent in isolation (ZOPPI *et al.*, 2017). One of the most established definitions, proposed by Maier (1998a), characterizes SoS based on five principles:

Operational Independence The constituents must be able to operate independently, even when disassembled from the SoS as a whole, i.e., the constituent must have its own missions and achieve them by its own capacities;

Managerial Independence Despite collaborating with the other members of the SoS, each constituent system is developed and maintained by different organizations, with their own stakeholders and resources, i.e., the constituent maintain its existence independent from the SoS;

Geographic Distribution Initially, this characteristic implied that the constituent systems must be physically placed in different locations. However, with the evolution of systems, it is stressed that the collaborating system must not be considered quantities of a unique mass (NIELSEN *et al.*, 2015). Furthermore, to accomplish some goal, systems must communicate to each other, so it is necessary to have modules that enable such communication, as conventional monolithic distributed systems;

Evolutionary Development The existence and development of an SoS are evolutionary since it constantly evolves, i.e., constituents can be added, changed or removed based on the requirements adaptation;

Emergent Behavior The behavior that emerges through the collaboration among SoS constituents. It cannot be predict analyzing the systems individually, but as a result when the capacities of each systems are united to achieve an specific goal. This is one of the main challenges in the SoS perspective, since sometimes this behavior could be undesirable and harmful for the system (MITTAL; RAINEY, 2015). However, when it is favorable, it is interesting the development of approaches to effectively predict and react to them.

Besides that, there are several definitions addressing this class of systems focusing on different architectural aspects. Boardman and Sauser (2006) proposed a set of five principles (acronym “ABCDE”) that together differentiate SoS from traditional systems, addressing not only the constituents independence but also the merging of new constituents with existing systems to create the SoS.

Autonomy Each constituent is free to pursue its own purposes, besides to fulfill the purposes of the SoS. The idea is that the constituent must not be just a part of the whole, but independent to operate and govern by its own rules rather than by external constituent’s ones (NIELSEN *et al.*, 2015). Such principle is directly related to the managerial independence characteristics proposed by Maier (1998a);

Belonging Constituent systems choose to collaborate to the SoS to achieve a higher purpose, either by individual profit or by believing in the SoS mission. However, due to the new relationships with other autonomous systems, they have to be persuaded regarding the value of such collaboration, pondering their autonomy, the overall benefits and their efforts;

Connectivity Dynamically supplied by constituent systems through its interoperability with other constituents of the SoS, addressing issues such as connecting the system already existent in the SoS and the addition of new ones. It involves the integration and adaptation of interfaces, protocols, and standards to enable bridging between legacy and newly designed systems (NIELSEN *et al.*, 2015);

Diversity The constituents are heterogeneous self-sufficient systems open for enhancement by evolution and adaptation. The SoS should, out of necessity, be incredibly diverse in its capability as a system compared to the rather limited functionality of a constituent system, limited by design;

Emerging Through the collaboration and interaction among the constituents, a new behavior emerges, based on the capabilities of each system while trying to achieve an objective. An SoS must be rich in emergence because it may not be obvious what tactical functionality is required to achieve broad capability. Instead, an SoS has emergent capabilities designed into it by virtue of the other factors: preservation of constituent systems autonomy, choosing to belong, enriched connectivity, and commitment to diversity of SoS manifestations and behavior.

As the focus on studying this topic is mostly related to interoperability and communication among the constituents, SoS has also become focus of interest in Software Engineering Research (BOEHM; LANE, 2006), expanding its abstraction and thus, implying in new definitions such as the one proposed by ISO/IEC (2010): “A large system that delivers unique capabilities, formed by integrating independently useful systems”. Besides that, one approach to

effectively characterize SoS is by identifying the main differences between SoS and Monolithic Systems, as presented by [Oberndorf and Sledge \(2010\)](#) and described in Table 1.

Comparison Perspective	Monolithic Systems	System-of-Systems
Problems	Causes of problems and behaviors effects can be largely known.	Problems are affected by a combination of factors, some are known and some unknown.
Dependencies	System dependencies are mainly within one program context and known scope control. Changes that affect dependencies within a system can be controlled.	SoS dependencies are outside the scope of control and the context may not be completely known. SoS changing dependencies may be only managed.
System focus	The focus is to optimize their capabilities.	The focus is achieving the collective emergent capabilities.
Negotiations and decisions	Negotiations and decisions are performed within the program scope.	The boundaries are not clear. There is more dependence on collaboration and influence in negotiations and decisions.

Table 1 – Systems-of-Systems vs. Monolithic systems. Extracted from [Oberndorf and Sledge \(2010\)](#).

Finally, there are two more important characteristics that must be highlighted regarding the concept of SoS, considered as essential features for such systems:

Mission A goal, functionality, or more specifically, a set of required capabilities of constituents systems and the interactions among these systems that lead to emergent behaviors towards the accomplishment of the global goals of the SoS ([SILVA et al., 2017b](#)). Within the SoS perspective, each constituents systems accomplishes its own individual missions and is able to collaborate and contribute to the accomplishment of the global mission.

Middleware A software component inherited from distributed systems which enables the software systems developed in one technology the capability to communicate with other software systems developed in a distinct technology, providing external representation of data, network support for communication, and abstraction of hardware ([COULOURIS et al., 2011](#)). Middlewares are commonly addressed as part of the constituent system since they are tightly integrated.

2.1.1 Classification of Systems-of-Systems

SoS have been largely distinguished by their level of managerial control ([DAHMANN et al., 2008](#)), and by their central authority, and collaboration level ([BRYANS et al., 2013](#); [PÉREZ et](#)

al., 2013). This classification establishes four types of SoS (DAHMANN *et al.*, 2009; MAIER, 1998a; PÉREZ *et al.*, 2013): directed, collaborative, acknowledge and virtual. According to Maier's taxonomy (MAIER, 1998a), these are the essential types of SoS and its respective characteristics (MAIER, 1998a; DAHMANN *et al.*, 2009; LANE, 2013):

- **Directed:** SoS in which the integration is built and managed to fulfill specific purposes. It is centrally managed by an entity that receives requests for any new constituent system which may wish to collaborate and orchestrates such systems for the fulfillment of global missions. Constituent systems maintain their autonomy but their normal operational mode is subordinated to the central entity managed purpose. For example, most integrated air defense networks are centrally managed to defend a region against enemy systems, although its constituents retain the ability to operate independently, and do so when needed under the stress of combat. In short, constituents systems operate subordinated to the central managed purpose (PÉREZ *et al.*, 2013);
- **Acknowledged:** These SoS have capability objectives, management, and resources to support them, but are comprised of existing systems along with new developments. This type of SoS has recognized objectives and a designated entity manager to fulfill its purposes. However, the constituent systems retain their independent ownership, objectives, funding and development and sustainment approaches. Changes in the systems are based on collaboration between the SoS and the system (DAHMANN *et al.*, 2008). In short, there is a considerable independence of the constituent systems, but there are specific objectives, management team, and resources addressed;
- **Collaborative:** Collaborative SoS are distinct from directed systems in that the central management software entity does not exist. Constituent systems must, more or less, voluntarily collaborate to fulfill the agreed upon central purposes. The Internet is a collaborative system. While standards are defined to it, there is no an absolute power to enforce them. Agreements among the central players on service provision and rejection provide what enforcement mechanism there is to maintain standards. The Internet began as a directed system, controlled by the US Advanced Research Projects Agency, to share computer resources. Over time it has evolved from central control through unplanned collaborative mechanisms (MAIER, 1998a);
- **Virtual:** Virtual SoS lack both a central management authority and centrally agreed upon purposes. Large-scale behavior emerges, and may be desirable, but the whole system must rely upon relatively invisible mechanisms to maintain it. Constituents are not subjected to a central management authority and there is not a clear SoS purpose. Constituent systems are not necessarily known and the mechanisms to maintain the SoS are not evident. Some examples are the current form of the World Wide Web, and national economies. Both systems are distributed physically and managerially. The World Wide Web is even more

distributed than the Internet since no one plays a central control, except at the earliest stages (MAIER, 1998a; LANE, 2013).

Figure 1 depicts a visual conception of how constituent systems interact with each one regarding to types of SoS discussed (LANE, 2013). The arrows indicate authority, feature request for SoS and data request. Under this perspective, and aligned to the discussion made, in Directed SoS, a central entity has a bidirectional communication with constituents, where those systems query this software entity to discover other constituents available and services provided, and the central authority delivers a response, enabling constituents communicate among themselves. In Acknowledged type, a software entity establishes a unilateral via of communication, organizing constituents to collaborate, accomplishing a mission as a result. Collaborative type, as mentioned before, lacks a central control, and cooperation among constituents arises spontaneously, but with a predicted intention of interaction within a predefined and embedded purpose. Finally, the most *ad-hoc* type of SoS, the Virtual, where constituents can interoperate supported by some type of mechanism (as a controlled environment, middleware configuration, an others), and this interaction is totally *ad-hoc*, without a predefined intention of interaction, making it totally spontaneous.

Figure 1 – SoS types and its inherent arrangement.

Source: Adapted from Lane (2013).

After presenting such SoS inherent features, it is possible to exhibit a representative conceptual model which compiles all these concepts and relations among them, proposed by

Gonçalves *et al.* (2014) and presented in Figure 2. In their context, they use the term Software-Intensive SoS (SiSoS) as an SoS which exists to accomplish global missions, and it is composed of constituent systems, which contribute to the accomplishment of such missions. Constituents can have managerial independence and their own purposes. Besides that, they have operational independence and present the ability of connecting to each other, thus enabling their cooperation to provide emergent functionalities, called emergent behavior.

Moreover, both the SiSoS and its constituent systems have their respective stakeholders, who have interests in their individual or global missions. Considering the software-intensive systems perspective, a SiSoS is an SoS in which software is the dominant factor (i.e., it exhibits a software dominance) in its composition. SiSoS are typically complex, large-scale systems whose functionalities and purposes can change and dynamically evolve, i.e., they present an evolutionary development. As a consequence, the architecture exhibited by a SiSoS is dynamic and reactive since it must evolve according to the dynamic collaboration of constituent systems that are also changing. Besides that, the representation and simulation of these architectures allow the understanding of possible emergent behaviors, even the not desired ones. Some of the concepts are determinant to categorize a SiSoS, such as: (i) the central authority of the SiSoS that determines how the constituent systems cooperate, thus resulting in four basic categories of SiSoS, namely virtual, collaborative, acknowledged, and directed; and (ii) the awareness level, which stands for the degree in which the constituent systems are aware of their participation in the SiSoS as whole. In virtual SiSoS, the constituent systems are completely unaware of their participation in the system. For the other categories of SiSoS (collaborative, acknowledged, and directed), the constituent systems are aware of such a participation.

Figure 2 – A conceptual model for SoS.

Source: Gonçalves *et al.* (2014).

2.2 Systems-of-Systems Software Architectures

Due to the increase in size and complexity of software systems since the late 1980's, software architecture research emerged as the principled study of the design, specification and overall structures of these systems (GARLAN, 2001; SHAW; CLEMENTS, 2006). Structural challenges addressed by such area include the organization of a system as a composition of components, protocols for communication, synchronization, the assignment of functionalities to design elements, physical distribution, scaling and performance, evolution, and so on (GARLAN, 2001).

From its roots in qualitative descriptions and its advances in offering guidance for complex software design and development, practitioners have come to realize that having a good architectural design, which includes codification of architectural principles, standards for documentation, reviewing and implementation of architectures, is a critical success factor for complex system development (GARLAN, 2001; SHAW; CLEMENTS, 2006).

2.2.1 Software Architectures

The concept of software architecture has been evolving over the years. Although several definitions have been proposed, they share common characteristics and can describe the core of the concept. Medvidovic and Taylor (2010) defined software architecture as “the set of principal design decisions made during systems development and any subsequent evolution”. Considering a technical perspective, ISO (2011) presented software architecture as “the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evaluation”. Besides that, as one of the most established definitions in the literature, Garlan (2001) and Bass *et al.* (2013) presented software architecture as “a set of structures needed to reason about the system, which comprise software elements, relation among them, and properties of both”.

Software architecture typically plays a key role as a bridge between requirements and code, impacting in at least six aspects of software development (GARLAN, 2001): (i) understanding: it simplifies the ability to comprehend large systems by presenting them at a level of abstraction at which a system's design can be easily understood (GARLAN; SHAW, 1993; MEDVIDOVIC; TAYLOR, 2000); (ii) reuse: it supports the reuse of large components and also of the frameworks into which components can be integrated (BUSCHMANN *et al.*, 2007); (iii) construction: it provides a partial view for development by indicating the major software components and dependencies between them; (iv) evolution: software architectures can expose the dimension along which a system is expected to evolve and provide descriptions that permits easily change mechanisms; (v) analysis: it provides descriptions for system consistency checking, conformance to constraints and conformance to quality attributes (CLEMENTS *et al.*, 1995); (vi) management: it is a key milestone in a software development process, in which the critical evaluation leads to a much clearer understanding of requirements, implementation strategies

and potential risks (BURGE *et al.*, 2008) ; and (vii) communication: it supports communication among stakeholders.

2.2.2 SoS Software Architectures

As mentioned above, software architectures have been considered the backbone for any successful software system and have played a fundamental role in determining the system quality (KRUCHTEN *et al.*, 2006; SHAW; CLEMENTS, 2006). Decisions made at the architectural level directly enable, facilitate, or interfere with the achievement of business goals as well as functional and quality requirements (NAKAGAWA *et al.*, 2013). In the SoS perspective, architectures represent the technical framework for the systems that compose the SoS designating how such systems will be employed by the user in an operational setting, the internal and external relationships and dependencies among them, their functions and details related to data flow (BOARD, 2017). Hence, software architectures have been noticed as an essential element to promote SoS inherent characteristics, such as emergent behavior, geographic distribution, and evolutionary development, without losing the sight of the quality requirements (BRØNDUM; ZHU, 2010; SAGE, 2011).

Due to the complexity of building SoS, the development of such architectures faces several challenges, mainly because the constituent elements can be software, hardware, and humans, and the architecture needs to represent as well as to specify the relationships and the composition of the individual systems to fulfill a new integrated mission (BATISTA, 2013; NAKAGAWA *et al.*, 2013). Besides that, a key factor that impacts SoS architectures is the fact the constituents systems of an SoS may not have been designed, developed, and employed with regard to their role in the SoS (BOARD, 2017). Moreover, different constituents collaborate to fulfill one of the SoS missions. However, the mission can be changed, and the architecture must be dynamic to change over time in ways that were not predicted when they were originally composed (BATISTA, 2013)

During the architecting process of an SoS, the constituents that collaborate to fulfill SoS missions are typically in place when the SoS is established and the SoS engineer needs to consider their current state and individual plans as factors in developing an architecture for the SoS (BOARD, 2017). Considering it along with the fact that constituent systems may be complex systems in their own right, leads to a set of challenges to architecting SoS (NAKAGAWA *et al.*, 2013; BOARD, 2017). In this context, Batista (2013) and Nakagawa *et al.* (2013) investigated the state of the art of SoS architectures and presented several aspects that must be considered in their development, as follows.

Representation Two levels may be considered in an SoS representation: the SoS-level and the constituents level. The elements of both levels can be architecturally represented by components, which should explicitly declare their interfaces, functional and nonfunctional

properties, and constraints, to provide means for reliable reasoning about the individual properties and their impact on the SoS whole property. Besides that, to capture the goal of the SoS and of each individual constituent, the architectural representation needs to go beyond the structural description of components, but it may involve the behavioral specification of both the SoS and constituent components. The behavioral description differs in both levels since the component that represents the SoS has goals that the individual constituents are not able to achieve independently.

Interaction The interaction among the constituent systems of an SoS plays an essential role in the architecture specification as the overall architecture is defined by integrating the different constituents parts. Although architectural connectors are flexible mechanisms to accommodate different interaction protocols (ALLEN; GARLAN, 1997), some specific issues of the SoS perspective deserve a special attention: (i) the level of complexity of the interactions can be high and severe architectural mismatches (GARLAN *et al.*, 1995) may occur; (ii) the number of interactions between the SoS elements can be explosive in a global distributed large-scale SoS; (iii) sometimes it would be impossible to foresee all the connections in advance, at design time, as the emergent behavior can impact on existing connections and also demand new connections. In order to support these SoS specificities, architecture description languages may provide composition mechanisms able to express and reason about complex interactions, to handle the architectural mismatches, to deal with the complexity of the interactions, and to allow dynamic connections.

Dynamism Due to changes in the SoS mission at runtime, and the consequent evolution of the architecture to fulfill it, the connections among the constituents are not stable, and therefore they are added and removed dynamically. In this highly dynamic scenario, a self-organizing support is of outstanding importance to avoid the need of an explicit management of the architecture, which is a very difficult task in open and distributed systems as SoS. Self organized architectures must preserve the architectural properties despite the insertion or removal of components and connections. The characteristics of each constituent including its interfaces and communication rules may help to guarantee the correctness of the SoS evolution as a rigorous documentation for ensuring the compatibility of the new and old constituent systems.

Quality Quality attributes represent a key issue for the successful fulfillment of the SoS missions (BIANCHI *et al.*, 2015). All systems involved in an SoS must satisfy quality attributes, such as availability, performance, reliability, to name a few. Several problems, such as inconsistencies and ambiguities, can arise when the support for quality attributes is neglected in SoS architecture description (modeling and analysis). Thus, the architecture may support the achievement of the SoS targeted quality attributes to early mitigate the potential hazards to the SoS. The specificities of each constituent must describe functional and non-functional properties to achieve such quality requirements.

Evolution The vast majority of SoS evolve by nature. Simply ignoring the need for controlling their evolution is not certainly the most adequate way. Hence, it is firstly important to distinguish a stand-alone system's evolution from SoS evolution and to treat SoS evolution as part of the whole development process supported by proper methodologies and techniques. Moreover, it is important to understand why, how, when, and where this evolution occurs and which characteristics and decisions (e.g., external interfaces of the SoS constituents) must be taken when proposing SoS architectures, to avoid future architecture degradation. If it is intended to keep quality in SoS along their evolution, new processes, methods, and techniques that systematize SoS evolution must be investigated, proposed, and widely adopted.

2.2.3 Quality Attributes

In the SoS perspective, several quality attributes are critical, such as interoperability, performance, reliability and security, and, therefore, should be carefully considered to satisfy the SoS mission (ACKERMANN *et al.*, 2009). However, since the constituents systems are developed and maintained by different organizations, with their own stakeholders, development teams, processes and resources, the achievement of these quality requirements is a challenge (SANTOS *et al.*, 2014). Despite that, due to their main characteristics, such systems must be prepared to support a dynamic and evolving development to make it possible the incorporation of new features and even the reorganization of the existent constituents at runtime (BOARDMAN; SAUSER, 2006).

On the other hand, a well-accepted way to support quality control is to adopt software quality models that have been over the last decades an widely researched area in Software Engineering (WAGNER, 2013). Several quality models have been proposed in the literature to evaluate the quality of modern computer-intensive systems (AZIZIAN *et al.*, 2011). However, in the SoS perspective, such challenge has not yet been overcome by the classical software perspective (WALDEN *et al.*, 2015).

Facing this challenge, Bianchi *et al.* (2015) performed a systematic mapping study (SMS), which aimed at investigate what are the most common quality attributes for SoS architectures. As a result of analyzing 52 primary studies, they extracted a set of 56 quality attributes addressed in the literature in the SoS perspective. Figure 3 presents the quality attributes and the number of primary studies which addressed them.

Considering the quality attributes identified in the SMS, Schneider *et al.* (2011) argued that it is very difficult to assure quality properties like safety, reliability, performance, and security for the whole SoS since it is hard to guarantee such properties for all constituent systems. For this reason, dependencies, trade-offs and relationships among the quality attributes become more complex (BIANCHI *et al.*, 2015). Besides that, some quality attributes, such as reliability and evolvability, are defined in a way that cannot be directly mapped to the SoS perspective (GARRO; TUNDIS, 2015). Such divergences are related mainly to the SoS failures, which are

Figure 3 – Quality attributes for SoS.

Source: Bianchi *et al.* (2015).

not considered when one of the constituents fails, but when the performance of the whole system achieves an unacceptable level (BIANCHI *et al.*, 2015).

Regarding interoperability, one of the most relevant quality attributes for SoS, it can be handled as a cross-cutting concern that has a unique and crucial coordination role relative to the other quality concerns (ROTHENBERG, 2008). However, Bianchi *et al.* (2015) emphasized that an SoS is usually conceived without considering the interoperability of its constituent systems on the early stages of development (MADNI; SIEVERS, 2014) due to the fact that the systems could exist even before the goals of the SoS were defined. In addition, the metrics established by the current quality models to measure interoperability cannot directly deal with the SoS characteristics (GUARINIELLO; DELAURENTIS, 2014). This is explained by the emergent behavior of SoS that makes difficult to capture and evaluate interoperability at the level of constituent systems (MEILICH, 2006).

From that, it is important to consider the interdependence among quality attributes in scenarios that cascading failures and bottlenecks could result in a complete SoS blackout (CHIPRIANOV *et al.*, 2014). To do that, the quality attribute need to be measured and controlled for each constituent system to address the impact on SoS quality (CHIPRIANOV *et al.*, 2014; BOARDMAN; SAUSER, 2006; WALLER; CRADDOCK, 2011).

2.2.4 Examples

SoS principles have been applied in a wide range of application domains, such as healthcare, transportation, emergency response, defence, and infrastructure (NIELSEN *et al.*, 2015). As an example discussed by Nielsen *et al.* (2015), the successful treatment of a patient in an emergency results from the integration of several separately owned and managed systems including telephony, ambulance assignment, information sharing, communications and hospital management. The agility required of an emergency response SoS, and the often unique circumstances of each substantial emergency, lead to astronomically large models. In examining emergency management SoS in China, an hierarchically structured SoS with local, regional and national SoS provide services at the level of jurisdiction required.

In the same context, Dayananda *et al.* (2017) presented an SoS architecture for a Unmanned Aerial Vehicle (UAV). Such architecture is composed of twelve systems that work in collaboration to ensure that the vehicle can achieve safe flight and provide medical aid, supporting an acute medical emergency. Figure 4 presents the systems of this architecture and highlights the structure of the Image Recognition System, responsible for obtain visual feedback of the area through several steps: (i) recognize the image; (ii) convert the image into pixels; (iii) compare the image with a trained database; (iv) identify location for defibrillation; (v) analyze the data; and (vi) send the data to the actuator system, which plays a central entity role. Besides that, different examples also identified by Nielsen *et al.* (2015) are presented as follows

Figure 4 – An unmanned aerial vehicle architecture

Source: [Dayananda et al. \(2017\)](#).

Transportation Transport networks are often composed of independently owned and operated elements, and by definition geographically distributed. They display emergent behaviours and evolve over multiple time bands. [DeLaurentis \(2007\)](#) maps SoS traits, derived initially from the aerospace domain, onto facets of the transportation problem domain more generally, emphasizing the role of networks in defining connectivity between constituent systems, and the heterogeneity of those systems. [Tongur and Sundelin \(2016\)](#) proposed the evolution of a system for electric road into an SoS, managed by several central operators, and in which no transport operation could be considered typical, addressing not only the geographical distribution characteristics of SoS but also the emergent behavior, through the simultaneous collaboration of the operators and systems.

Figure 5 – An electric road SoS

Source: [Tongur and Sundelin \(2016\)](#).

Healthcare Healthcare SoS borrows in the context of an increasing integration of electronic health records and network enabled access to patient information. They introduce the idea of a Healthcare SoS, the constituent systems of which include physicians, pharmacists, patients, clinics, regulating agencies and other stakeholders. The World Health Information Grid is proposed as an example, identifying a global SoS property of ensuring freshness and consistency of information delivered to decision makers no matter where they are located (i.e., avoiding information asymmetry).

2.3 Knowledge Discovery

Data, information, and knowledge are three often-encountered words that belong closely together, seem to have slightly different meanings, yet are often used interchangeably as synonyms, and thus lead to continuing confusion ([ZINS, 2007](#)). Aiming at mitigating such confusion, [Schreiber \(2000\)](#) provided a consolidation about the topics that in which there is a consensus in the literature, as follows.

Data Data are the uninterpreted signals that reach our senses every minute. A red, green, or yellow light at an intersection is one example. Computers are full of data: signals consisting of strings of numbers, characters, and other symbols that are blindly and mechanically handled in large quantities.

Information Information is data equipped with meaning. For a human car driver, a red traffic light is not just a signal of some colored object, rather, it is interpreted as an indication

to stop. In contrast, an alien being who had just landed on Earth from outer space, and happened to find itself on a discovery tour during the Friday evening rush hour, will probably not attach the same meaning to a red light. The data are the same, but the information is not.

Knowledge Knowledge is the whole body of data and information that people bring to bear to practical use in action, to carry out tasks and create new information. Knowledge adds two distinct aspects: first, a sense of purpose, since knowledge is the “intellectual machinery” used to achieve a goal; second, a generative capability, because one of the major functions of knowledge is to produce new information. It is not accidental, therefore, that knowledge is proclaimed to be a new “factor of production.”

Considering these definitions, a hierarchy comparing meaning and value was presented by Chaffey and White (2010), as presented in Figure 6. Such hierarchy indicates that data have no value and no meaning if analyzed outside a context that will provide the information for the acquisition of knowledge. Thus, information can already produce some kind of value, but only knowledge can actually show what is possible to conclude from the original dataset.

Figure 6 – Data-information-knowledge hierarchy

Source: Chaffey and White (2010).

Nowadays in a wide variety of fields, data are being collected and accumulated at a continuous flow. There is an urgent need for a new generation of computational techniques and tools to assist humans in extracting useful information from the rapidly growing volumes of digital data (FAYYAD *et al.*, 1996a). These techniques are the subject of an still emerging field of knowledge discovery in databases (KDD), also known as knowledge extraction, information

discovery, information harvesting, and data mining. Although data mining is defined as the computational exploration of patterns in a specific dataset, some definitions consider data mining as the complete process of knowledge discovery.

Knowledge discovery refers to a set of activities designed to extract new knowledge from complex datasets by automating the process of analyzing it through the establishment of techniques, methodologies and tools (BEGOLI; HOREY, 2012). Such process is composed of several steps ranging from the data in its raw state until the obtainment of new knowledge (OLIVEIRA, 2000).

Because computers have enabled humans to gather more data than it can digest, it is natural to turn to computational techniques to help us unearth meaningful patterns and structures from the massive volumes of dataset (FAYYAD *et al.*, 1996a). In this context, KDD has evolved, and continues to evolve, focusing on the overall process of knowledge discovery but intersecting several research fields, such as artificial intelligence, statistics, databases, machine learning and high-performance computing (FAYYAD *et al.*, 1996b).

The evolution of KDD and data mining in recent years implied in the emergence of several process models for the extraction of knowledge (OLIVEIRA, 2017). Process models refers to the set of steps to be followed, aiming at the development of an outcome, a product, which arises from a sequence of conceptual tasks (MARBÁN *et al.*, 2009).

According to GOLDSCHMIDT and PASSOS (2005), the knowledge discovery process is composed of three broad steps: i) pre-processing: the step that must deal with what is preliminary in relation to the desired out, from the access to the collection of data to the organization of them, through steps, such as subset selections, dimensionalities reductions or fine adjustments; ii) data mining: it refers to the activity of machine learning algorithms, obtaining patterns, rules, models, among other output formats; and iii) post-processing: it is composed of the evaluation tasks, with human inspections to verify the quality of the knowledge extracted in relation to the initial objectives, so that the results can be used by the end users. Although well established, several processes have been adapted to knowledge discovery. Following, we present two of the most consolidated processes, considering academia and industry (CIOS; KURGAN, 2005).

2.3.1 Fayyad Process

Fayyad *et al.* (1996b) proposed an interactive and iterative process, composed of nine steps that require decision making in the choice of algorithms and techniques for the development of a methodology. This process is most commonly used in the literature and is presented in Figure 7.

The first stage focuses on the understanding of the data domain, and on obtaining knowledge for the execution of the following steps. In addition, in this step it is necessary to define the goals of the extraction based on the end user needs. Then, in the second step, the subset of the whole data is defined. The subset selected must contain only the data that is important to

Figure 7 – Fayyad process.

Source: Fayyad *et al.* (1996b).

achieve the established goals.

Cleaning and pre-processing of data occurs in the third step. The subset of data must be refined by eliminating noise, standardizing the data types, eliminating repeated data and completing the gaps. In the fourth step, the data are submitted to dimensionality reduction, that is, the removal of attributes that are not relevant to the objectives.

The fifth step is based on the combination of the KDD goals defined in the first step, with an analysis of the problem class that must be solved, (e.g., summarizing, classification, clustering, and regression). Then, in the sixth step, considering the class of problem that is intended to solve, the data mining algorithm is selected. Such selection is performed through an exploratory analysis of the available algorithms considering parameters, architectures and other configurations.

In the seventh step data mining is performed, that is, the algorithms are processed and provide a significant result, which can be clusters, prediction models, decision trees or classification rules. By analyzing the outputs, it is possible to find the patterns and characteristics that may be useful for the domain. Considering the results obtained, the eighth stage has the objective of interpreting them. Generally, data visualization techniques are used because they facilitate the analysis for the experts.

Finally, in the last step, knowledge is consolidated, incorporating it into the application for future actions. The knowledge obtained can also be documented and used in larger processes. In addition, in this stage the verification and resolution of conflicts that may exist between the results and the hypothetically defined knowledge is done in this step.

2.3.2 CRISP-DM Process

CRISP-DM (CROSS-Industry Standard Process for Data Mining) is a process based on the main steps of the KDD theory and also well recognized by the industry (CIOS; KURGAN, 2005). Such model was developed by the joint work of four companies: Integral Solutions Ltd. (data mining solutions provider), NCR (database provider), DaimlerChrysler (car manufacturer) and OHRA (insurance company). This model is composed of six steps, as presented in Figure 8. Besides that, it is dynamic and does not have a strict sequence of steps.

Figure 8 – CRISP-DM process.

Source: Wirth and Hipp (2000).

Business understanding, in the first stage, aims to understand the requirements and objectives of the discovery, considering the business perspective. In this step, it is necessary to define the problem and the objectives of the data mining, in addition to defining a preliminary plan to execute it.

The second step involves data understanding, collecting data that will be processed for the verification of their quality. Moreover, an exploratory analysis of the data is needed for the formation of ideas and possible hypotheses about the implicit and discoverable information.

Data preparation is performed in the third stage, and covers the activities related to pre-processing, as well as others for the preparation of the final data set, so that they are ready to be processed by data mining techniques. In the fourth stage the modeling is performed. In this step, several techniques are selected and applied on the data, to adjust the specific parameters

of each algorithm. In addition, such step is necessary because each technique requires different forms of data, and this among other factors can impact the choice of the algorithm.

In the fifth stage the evaluation of the model is the main concern. It is necessary to ensure that the model meets the objectives of the discovery, determining whether the objectives defined in the first step have been met. Based on this analysis, it is defined if the process needs to go through another process iteration. The last step is deployment, in which the concern is having an effective use of the results obtained, applying it in the context of the domain. In industry, end-users use this data in explicit form, reporting or implementation plans for business decision making.

2.4 Knowledge Discovery in SoS: A Systematic Mapping

Aiming at identifying the state of the art about how knowledge discovery has been addressed in SoS, we performed a Systematic Mapping Study (SMS) (SENA *et al.*, 2017). SMS can provide a detailed overview of a research area by categorizing and investigating studies related to the topics of interest (PETERSEN *et al.*, 2015). For this, we followed the guidelines recommended by Kitchenham and Charters (2007) and Petersen *et al.* (2015), which has three steps: (i) planning: in which the research objectives and the systematic review protocol are defined; (ii) conduction: in which primary studies are identified, selected, and evaluated according to the inclusion and exclusion criteria previously established in the protocol; and (iii) synthesis: in which information extracted from the studies are synthesized in a report. Planning phase is detailed in Appendix A. The rest of this section details conducting and reporting phases.

In the conduction phase, we adapted the search string, established during planning phase (see Appendix A), to each database. During conducting the search, time limits were not placed, and filters on title, abstract, or keywords were not used. On the completion of this process, we obtained 407 primary studies. The title and abstract of each study were inspected and the selection criteria were applied. In total 39 studies were selected for detailed inspection. The full text of each one of these studies was read and the selection criteria were again applied. As a result, 17 primary studies were selected to be included and thus to data extraction. The complete list of primary studies selected is presented in Table 12.

2.4.1 Results

As observed in Table 12 (column “Year”), studies published in this topic have remained stable over 13 the years, since the first one published in 2005 until the recent one in 2018. Contributions already made in this field are still low, showing that there are several paths of research to address the current limitations.

Regarding the publication venues, 14 studies (S1, S2, S3, S4, S5, S6, S8, S9, S10, S11, S12, S13, S15, and S17) were published in conference or workshops, while only 3 (S07, S14,

and S16) were published in periodicals (journals), as presented in the left side of Figure 9.

Such results may be related to the fact that most studies are not consolidated enough to be published in a periodical, which may also suggest that the topic is emerging. Considering the venues, most of the studies were published in three general conferences or workshops: International Conference on Information Systems for Crisis Response and Management (ISCRAM), International Conference on Internet Computing for Science and Engineering (ICICSE), and the IEEE Systems-of-Systems Engineering Conference (SoSE).

Figure 9 – Number of studies per type of publications, years of publication, and authors

Source: Research data.

Considering the authorship of the studies, most of them were written by academic researchers, and only two were written by industry practitioners. Figure 9 (right side) presents the study distribution in these different combinations of authors. Three of the studies result from partnership between industry and academy. In general, as there are still several open challenges, there is no guarantee that the current solutions are not suitable enough to have more intense adherence to the industry.

Figure 10 – Wordcloud of the studies.

Regarding the results extracted to understand the state of the art, none of the selected studies proposes or uses an enough robust approach that encompasses all aspects of the discovery of knowledge, i.e., data collection, data integration, and in fact, knowledge discovery. In general, each study focuses on a specific aspect and assumes that others have already been resolved. Furthermore, most studies are not possible to be compared directly, since they present different perspectives; some studies present architectural solutions, others present technological solutions, modeling, methods, or just discussions about trends and limitations. In this way, these studies can complement each other and together can support the construction of a more complete approach. The following sections present details of knowledge discovery in SoS in different aspects.

Approaches to Data Collection

Most of the studies presented a means to collect data from SoS constituents; more importantly, all of them addressed architectural solutions, indicating that attention on software architecture seems to be interesting. From a wider viewpoint, there are two distinct perspectives to collect data in SoS. The first one requires that each constituent provide their data voluntarily on a platform specified by them. In this approach, data becomes available and is only collected or used by other constituents. In this sense, 3 of the 9 studies (S04, S06, S11) suit this perspective. Regarding communication, the platform where data is available can be shared among constituents, if there is an agreement between them, or it may be individual. In this last one, it is important that there are protocols of communication and control among constituents, to know which platforms the data can be collected and in which way these data can be used. Otherwise, in the second perspective, constituents do not provide their data automatically, but only when the data is requested by an external entity through communication interfaces. The request can be made to fulfill a mission or other reasons. Constituents must agree to provide data to the end as necessary, and have an external interface (or services) so that the data can be requested and provided in a systematic way. We found 6 studies (S01, S02, S03, S04, S14, and S15) that address this perspective. The main difference between these two approaches lies in the quality attribute they try to achieve. In the first approach, the focus is on data availability, as data must be constantly available to be consulted and used. On the contrary, when data needs to be requested for constituents, security must be a element of first class for SoS because only authorized data will be shared. In both perspectives, contracts should be well defined to not influence the business operation of such systems.

Considering quality requirements, several of them are mentioned in the studies as important for architecting data collection in SoS. As presented in Figure 11, 13 requirements were found. The figure shows the number of studies among the 9 selected that presented the requirements. Availability is the most recurrent attribute (S1, S3, S4, S11, S14, and S15) showing that the data required for KD must always be available for consult/request by a central entity and other constituents. Security (S1, S3, S4, S11, and S14) and interoperability (S3, S4, S6, S11, and S15) have been presented to ensure the integrity and authenticity of the data, as well as providing means for the data to be communicated among various constituents, and modularity (S1, S3, S4, S11, and S16) to ensure that the module responsible for data collection and data request interfaces are independent of different constituents. Scalability (S1, S3, S11, and S16) and performance (S1 and S16), which also includes latency and response time, are also critical attributes for data collection scenario, as they can ensure that data will be collected no matter the amount of data generated. The other attributes, although important, were defined by reasons of specific characteristics of application domains. It is interesting to highlight that, besides the generic attributes, application domains require different ones, at least for this set of studies already published. For instance, real-time operation is presented as a specific attribute in a crisis context, where the SoS must make decisions in a reaction to the current state; and Reliability is

necessary in health domains where a failure could cause the death of patients.

Figure 11 – Number of studies that addressed each quality requirement.

Source: Research data.

In this same perspective, several architectural patterns were found to the fulfillment of quality requirements. Among the main patterns we can highlight: Service-Oriented Architecture (SOA), orchestrator, producer/consumer, shared repository, data-as-a-service, layers; and monitoring ([SENA *et al.*, 2017](#); [INGRAM *et al.*, 2015](#); [AVGERIOU; ZDUN, 2005](#)). Shared repository and data-as-a-service patterns are useful in the approach where data is sent to a central platform and made available to other constituents. Orchestration, producer/consumer, and monitoring are management patterns (more used when the central entity must request data from constituents), which control what data needs to be collected and also keeps monitoring the constituents so that whenever a new data is generated, it can be considered for analysis and sent to consumers. SOA and layers are patterns that allow the modularization and organization of the architecture so that the tasks are modularized and then achieve weak coupling. In the SOA scenario, constituents act as services with their independent functionalities, and their communication interface is the only data input and output port for the SoS.

Approaches to Data Integration

Data integration is the most challenging task of the entire KD process in SoS. This is because of the dynamism of the SoS, i.e., constituents and missions change at runtime, also

changing the data that is generated. In this way, new ones emerge and are often not treated to integrate with the other data. Since we are dealing with a big data context, the volume of data generated directly impacts the variety of them, which requires very complex processes and techniques for the integration to be performed correctly. Only six studies (S1, S2, S6, S11, S14, and S15) presented initiatives for integrating data provided by SoS constituents.

In the same analysis of data collection, there are also two main perspectives for data integration, namely: (i) continuous flow integration; or (ii) integration by information queries. In the first one, data is already sent to a central platform in the format in which it will be stored, which needs to be defined by operations contracts previously between the SoS constituents and the central entity. This transformation can occur either in the constituent itself (which is not common because constituents use data in different formats of those of the central entity) or in the communication interface between the constituent and the central entity, which will actually store this data. In fact, the important point of this perspective is that data is sent continuously and already transformed into a specified format. In the central entity, this data can be stored in several different database architectures, be it data lake, relational databases, non-relational databases, and so forth. Storage challenges are minimized by the fact that the data is already standardized.

In the second perspective, data is stored in its original format/structure in both constituent and shared repository, and will only be processed for format change when an operation requires for some specific knowledge. This type of integration increases query processing but prevents unused data from being preprocessed. As an example, in S06, the authors state that data should be integrated in an SoS only when necessary, because a previous data integration may require formatting and processing efforts and could not be aligned to the further use of such integrated data; otherwise, performance could be better achieved when data are already integrated.

In general, in this scenario of collection and integration, formats must be defined in advance in the operating contracts so that the communication interfaces between the constituents and the central entity know the data format they need to request and receive. In addition, formats can be changed at runtime, but without prior configuration of the collection interface, communication errors will be thrown and the data from that constituent will probably not be collected to support knowledge discovery.

Among the studies that presented an approach in which data is sent to a central platform, two of them (S2 and S11) proposed the use of a module especially destined to receive data from different constituents and adapting them to the use of the SoS. S02 proposed this module as a mediator responsible for exchanging formats and keeping the information interoperability among constituents, while S11 proposed an adapter service to perform data format transformation between two different formats from the domain connected to the Enterprise Server Business module, a central controller of their specific domain that controls the communication among different constituents.

As previously mentioned, S14 presented a four-tier architecture, and the second tier is

responsible for the communication among constituents, using protocols and communication technologies in networks, such as Wi-Fi and Ethernet. S01 presented an approach composed of two main components, one for collecting data and one responsible for merging data collected. The authors highlighted that this structure should be scalable to support various types of data generated in an SoS (Aloisio 2005).

Finally, S15 proposed an approach in which the SoS stakeholders established a shared terminology to keep the knowledge in the same format. In short, this study defines that each constituent must store data in a specific format to be integrated. Therefore, it is important to define an approach to define whether the integration will be carried out at SoS level, establishing a specific format and being controlled by a central entity, or whether the integration will be individual, performed by each constituent in a collaborative perspective.

Regarding the relevant architectural requirements addressed in the data integration process, the same ones discussed in data collection section were found; however, different patterns were addressed. The patterns and the reason why they were used are summarized in Table 2.

Pattern	Reason
Data-as-a-Service	In this pattern, all modules need to be constructed as a service to be consumed, including data collection and integration.
Pipe and Filters	This pattern creates a flow (pipe) composed of several steps (filters) in which data must pass through, including integration.
Broker	This pattern is used to structure several constituents and guide the flow of data, focusing on data translation among the modules that request data in different formats.
Orchestration	An orchestration pattern proposes the use of an entity that manages the flow of data during SoS, identifying what data should be collected (when) and to which constituent such data should be sent.
Layered	In a layered architecture, an intermediate layer of integration between the collection layer and the knowledge extraction layer is proposed; so the implementation is modularized and used according to the requirements of the domain.
Monitoring	This pattern proposes the use of a monitoring entity, which performs data integration whenever it is necessary for the accomplishment of the mission.

Table 2 – Patterns used for integration

Approaches to Knowledge Discovery

Considering the studies found in our SMS, the main reason why the final number of studies selected decreased from the initial set of studies is because we focused on studies in which there was the discovery of knowledge in the SoS, not studies that used SoS to improve knowledge discovery. In this sense, 15 studies (S02, S03, S04, S05, S07, S08, S09, S10, S11, S12, S13, S14, S15, S16, and S17) addressed approaches for KD in SoS. From these studies, two

categories can be identified according to the objective that guided the extraction of knowledge. Eight studies (S02, S05, S07, S08, S09, S11, S12, and S15) focused on extracting information that is useful for a given application domain. Examples of knowledge domain which were addressed in the studies are: (i) classify messages from social networks (S11); (ii) create early warnings for tsunamis (S09); (iii) improve healthcare data analysis (S12); or (iv) forecasting the vacancy risk of neighborhoods and predicting costs (S07). Five of them focused on KD for SoS improvements (S10, S13, S14, S16, and S17). In S16, for instance, authors turned the systems of the city into a smart city. Besides that, S14 optimized an energy management system based on a model to forecast receive solar power.

Most of the 15 studies proposed architectural solutions, except S05, which proposed a logical model, and S08, S10, and S11, which presented each a system to specific applications (prediction, regression, classification). Regarding the architectural solutions, S03 presented an architectural pattern to extract knowledge from the constituents. Such pattern is composed of three main entities: (i) the services (i.e., the constituents); (ii) the messenger and the router, in which the messenger is responsible for data transmission between services and the router responsible for intercepting data and reroutes them; and (iii) service/data connectors, which hide the internal structure of the connected services and connect only the ones necessary to provide and expose data. Moreover, S09 reported the development of an architecture for a collaborative, complex, and critical decision making SoS, which enables constituents distributed in heterogeneous environments to communicate through message exchange (MOSSGRABER *et al.*, 2013).

Addressing the layered architectural pattern presented in data integration section, S16 presented an architecture containing a specific layer to event/decision management. This architecture provides real-time intelligent decision making, and scalability in the data manipulation with the clusters. Each technology is used to facilitate processing a huge amount of data and to improve the performance of the application. Another layered architecture was presented in S14, which included a layer with the analytical system responsible for data processing. Moreover, S07 presented a system that can support predictive analytics; for this, there is a generic method for data analysis and dissemination of knowledge. S08 proposed an architecture containing two modules: (i) one involves the knowledge acquisition from historical data; and (ii) another uses the knowledge to the automatic risk identification (in this case, vessels in risk among all in the shipping traffic). S17 introduced an approach responsible for selecting the SoS constituents based on a set of five key performance attributes (i.e., performance of the prediction, performance of the decision, affordability, scalability, and robustness) to improve intelligent predictions and decisions in a time-series environment.

One of the critical domains in which KD has been explored is healthcare. In S13, the authors proposed a general architecture with not only several services and subsystems, but also technical support units, such as computing facilities for data storage and management,

data distribution, networks, processing, and analysis. In addition, S02 presented an architecture with several modules for managing data/information and a module for knowledge extraction. S12 proposed an architecture considering cutting-edge technologies of cloud computing and big data. Such architectures are composed of layers: data resources, file storage, data storage, programming model, data analysis, and platform management. In particular, the data analysis layer is responsible for data analysis, data processing, analysis models, business value, and also knowledge extraction.

Aiming at supporting the operation and decision making in SoS, S05 proposed a knowledge-driven logical model to support such operation composed of the following steps: data ingestion, image archive (for constituents that provide satellite images), browsing engine, image features extraction, inventory, query engine, classification and knowledge acquisition, and interactive learning. Facing the same context, S04 presented a data-shared architecture with four components: (i) web search, in which users search for and retrieve data; (ii) catalog node, which holds the metadata catalog and forward data requests of users to data repositories or metadata index server; (iii) data repository, which provides metadata information to discover, locate, and describe data; and (iv) metadata index server, which stores different domains metadata information from other catalog nodes distributed in geographical areas. The SoS architecture discussed in S11 focuses on the loose coupling of the services in a centralized service management catalog. This architecture proposes: (i) a module for query processing to access the data needed for KD; (ii) an enterprise server business module for interaction and communication among services; and (iii) an API management server, which manages the external interfaces. Finally, in S10, the authors proposed a system that could be integrated as a constituent in SoS. This system implements two big data techniques, namely: (i) extract relevant information to derive knowledge; and (ii) apply such knowledge.

Regarding the use of knowledge, four studies (S08, S14, S15, and S16) detailed how the knowledge was used to support the SoS operation. S08 and S15 proposed an external interface in which SoS stakeholders must evaluate the results before any action, while S14 and S16 proposed a specific layer to receive the knowledge and send it to controller entities (orchestrator and monitor) or SoS stakeholders. Further details of how this knowledge is used could be provided to show how decision-making can be supported by constituent data. However, even those studies that present this idea show the impact of the analysis very superficially.

Finally, in relation to quality requirements and architectural standards, the same ones presented in data collection and integration sections can be considered for the discovery of knowledge. In general, no specific attribute falls to this part except for the performance and efficiency required for the execution of the models. Moreover, scalability and availability are still critical in this step and must be considered in order to keep the quality of the entire data flow and to support missions fulfillment.

2.4.2 Discussions

In general, in a process of KD in SoS, data are collected from several constituents and then integrated, storing them in repositories shared by SoS. After this, the extraction of knowledge is performed and the decision-making process is initiated. At this point, the controlling entity may send control commands to the constituents so that such execution can support SoS missions fulfillment. The main problems of this process are within these modules, which are still poorly defined and present open challenges in the research area.

Figure 12 – General process of KDD in SoS.

Source: Elaborated by the author.

In general, the total number of contributions already made in this field is still low, indicating that there are further gaps to be researched so that the state of the art can move forward. Despite of several evidences of the need for a robust approach that supports KD in SoS (ALUISIO *et al.*, 2005; MCDERMOTT *et al.*, 2005; DEMCHAK; KRÜGER, 2008; MOSSGRABER *et al.*, 2013), many challenges must be addressed to consolidate such an approach. In general, the existence of these limitations can be justified by the lack of research advances in this sense. In fact, we performed some evaluations to verify the quality of the studies found. We analyzed if there was a clear justification about the methods used during the study; only 58.8% of the studies detailed the choices and the development of their approaches. Besides that, 47.1% of the studies performed some kind of evaluation. Finally, analyzing if there was a subsection discussions about the credibility and limitations of the studies, 47.1% of the studies knew how to insert themselves in the state of the art, showing related works and justifying the limitations of their

approach. These numbers indicate that there is still a lack of maturity in the development of these work, but it is important to emphasize that the first steps have been taken.

A challenge that can be drawn from the lack of maturity of studies is related to the generalization of approaches. First, because they do not present well-defined empirical evaluation, it is difficult to know whether they actually bring benefits to solve the presented problems. In addition, these approaches were developed for specific domains, which bring certain limitations of quality requirements, architectural standards and implementation decisions. In this way, one approach that has worked for one domain would not necessarily work in another. Finally, it is necessary that more abstract solutions could be proposed so that the challenges of diverse SoS are solved in an efficient and systematic way.

Considering the types of SoS, 14 studies (S01, S02, S03, S04, S05, S07, S08, S09, S11, S13, S14, S15, S16, and S17) dealt with collaborative SoS, i.e., SoS in which there is no central entity to control the constituents, which choose to be part of an SoS to assist in the fulfillment of the SoS missions. Besides, two studies (S06 and S12) approached acknowledged SoS, in which there is a central entity to guide the operation of the constituents, even though they retain their independent ownership, objectives, funding, and development. Moreover, one study (S10) addressed directed SoS, in which there is a central entity, but the level of authority of the central entity differs since the constituents operation mode is subordinated to it. No study addressed virtual SoS, in which there is no central authority, nor clear SoS purposes. The different types of SoS directly impact KD from their constituents. Systems that have a central entity with authority over the constituents can request the data needed for the extraction of knowledge, while systems without the central entity require that the constituents send the data of their own accord. In addition, the existence or not of centrality impacts on aspects such as storage, security, and management of missions.

Therefore, it is important to also consider that for each type of SoS, the approach will be different. For SoS that have a central entity, all collection, integration, and discovery can be performed within this entity. In contrast, when the SoS does not have a central entity, a new constituent system must be integrated with the SoS and connected to all constituents to perform these tasks. Regarding stakeholder support, few approaches mentioned the need for a module to visualize results and make decisions, which is essential to support the SoS in missions fulfillment.

Another interesting point to be highlighted in the studies found is that because they did not detail the proposed modules, few references were made to recent technologies and frameworks related to machine learning and big data. These techniques are essential both for dealing with large amounts of data and for extracting knowledge from them, whether in tasks such as classification, prediction, or clustering. In fact, some of the studies presented general

technologies from the KD context, such as Hadoop¹, HBase², and Hive³, together with NoSQL, as a solution to achieve a scalable, high performance storage. Even so, these presentations are still superficial and do little to describe how solutions can be realized.

Perspectives of Future Works

Considering the results of our SMS, the total number of contributions already made in this field is still low, indicating that there are further gaps to be researched so that the state of the art can move forward. Despite some evidence of the need for a robust approach that supports KD in SoS (Aloisio et al. 2005; McDermoth et al. 2005; Demchak et al. 2008; and Moßgraber et al 2013), many challenges must be addressed to consolidate such an approach. The existence of these limitations can be justified by the lack of research advances in this sense. As perspectives of future lines of research, we highlight the items presented as follows.

Data Collection :

- develop more complete approaches that address data collection in different types of SoS, i.e., how to deal with the protocols of each of the constituents to collect the data for the collection of knowledge, or establish protocols that can monitor the data and send it to a repository in which it can be accessed;
- proposal of architectural patterns that support the collection of data from different sources such as sensors, internet, different systems, and that can adapt to the security settings, query and request forms of each source;
- integration of SoS infrastructure and streaming technologies that support real-time data collection without creating bottlenecks.

Data Integration :

- develop approaches that address the need to integrate data efficiently and securely, without losing the context of information and that are adaptable to different types of data;
- develop approaches that support the integration of different data formats that can be changed at runtime.

Knowledge Discovery :

- develop systematic methods that allow the extraction of knowledge in an SoS, detailing the interior of each of the modules, considering the tasks of pre-processing, analysis and post-processing;

¹ A framework that allows for the distributed processing of large data sets across clusters of computers. More information in: <https://hadoop.apache.org>

² A distributed, scalable, big data database: <https://hbase.apache.org>

³ A data warehouse: <https://hive.apache.org>

- support the dissemination of knowledge in the SoS as a whole, i.e., to apply the knowledge to fulfill some mission of the system;
- many of the proposals have been made to architect an SoS that is not yet in execution, i.e., the SoS is in the design phase, but it is necessary approaches that enable the SoS to adapt when receiving new missions. When knowledge is needed the modules need to be connected to the constituents.

2.5 Final Remarks

In this chapter, we introduced key concepts related to the main topics that encompasses this project. Such concepts include definitions, characteristics, and examples. Besides that, we also presented the results of a SMS which aimed at mapping the state of the art on how knowledge discovery has been addressed in SoS. As main results, we identified that there were few initiatives in this area of research, and many open challenges that can be addressed to achieve this goal. Thus, in the next chapter we will present KnowSoS, our first approach to address knowledge discovery in SoS. This proposal addresses some of the topics discussed in the perspectives of future trends, creating a general architecture distributed in several modules to support knowledge discovery in SoS.

KnowSoS: A Software Architecture for Knowledge Discovery in Systems-of-Systems

KNOWLEDGE discovery has been a present challenge in several areas of our society, due to the intelligence and automation that brings to the domains. One of the classes of systems that benefits from the knowledge discovery process are Systems-of-Systems, which can use the data provided by constituent systems to achieve missions not achievable by any of them individually. As discussed in the previous chapter, little progress has been made in this area to address the challenges of such discovery. Thus, in this chapter, we present KnowSoS, a first proposal towards the establishment of a complete approach to perform knowledge discovery in SoS. KnowSoS is a software architecture that supports the design of a system that can be inserted into the SoS to achieve knowledge discovery.

In this chapter, Section 3.1 presents the method used to develop our architecture. Section 3.2 presents the architecture, detailing its main requirements and the selected views for its representation, namely, use case view, process view and logical view. In Section 3.3 is presented guidelines that support the use of the architecture and the entire process of establishing knowledge discovery from its definition to actually the design of architecture. Finally, section 3.4 presents the final remarks.

3.1 Method

To design the architecture for supporting knowledge discovery in SoS, we followed a general model proposed by Hofmeister et al ([HOFMEISTER *et al.*, 2007](#)). To create this model, they selected five different methods for architectural design and compared the main activities and architectural artifacts addressed by each one, collecting the concepts that are shared by all of them, and analyzing those that are different, so that the final model is more efficient and addresses the essential characteristics of architectural design. The five selected methods are:

Attribute-Driven Design Method (BASS *et al.*, 2013), 4 Views method (HOFMEISTER *et al.*, 2000), the Rational Unified Process 4+1 (KRUCHTEN, 1995), Business Architecture Process and Organization (AMERICA *et al.*, 2003), and Architectural Separation of Concerns (RAN *et al.*, 2000). Such general model is composed of three main activities, namely, architectural analysis, architectural synthesis, and architectural evaluation, as presented in Figure 13. These activities are described as follows.

Figure 13 – Architectural Design Activities.

Source: Hofmeister *et al.* (2007).

Architectural Analysis Architectural analysis serves to define the problems the architecture must solve. This activity examines, filters, and/or reformulates architectural concerns and system context in order to come up with a set of architectural significant requirements (HOFMEISTER *et al.*, 2007). The input of this activity are the architectural concerns, and the architectural context. The output is the set of architectural requirements. Following, both artifacts are detailed.

- Architectural Concerns:** According to IEEE 1471 (HILLIARD, 2000), architectural concerns are those interests which pertain to the system’s development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns may include quality aspects such as performance, reliability, security, distribution, and evolvability (ISO/IEC-24765, 2010). Most concerns are expressed as requirements of the system, but they can also include mandated design decisions (i.e., use of specific architectural patterns) (HOFMEISTER *et al.*, 2007).
- System Context:** The IEEE 1471 (HILLIARD, 2000) defines that the system’s context “determines the setting and circumstances of developmental, operational, political, and other influences upon the system”. Such context includes concepts related to business goals, characteristics of the organization, and state of the technologies (HOFMEISTER *et al.*, 2007).

- **Architectural Significant Requirements:** An Architectural Significant Requirement is “a requirement upon a software system which influences its architecture” (OBBINK *et al.*, 2002). Two points must be considered to define them (HOFMEISTER *et al.*, 2007): (i) not all of the system’s requirements will be relevant to the architecture; and (ii) not all requirements will have originally been expressed as requirements, i.e., they may arise from different concerns or from the system context.

Architectural Synthesis Architectural synthesis is the core of architecture design. This activity proposes architecture solutions to address the set of architectural requirements, thus it moves from the problem to a group of possible solutions (HOFMEISTER *et al.*, 2007). Such solutions could be defined as a set of viewpoints and views, or different prototypes to simulate the operation of the system. This activity receives as input the architectural requirements, and generates as output the candidate architectural solutions described below.

- **Candidate Architectural Solutions:** Such solutions may be alternative (complete considering one aspect) or partial (i.e., fragments of the whole architecture). They reflect design decisions about the software structure and must include information related to the design rationale, i.e., detailing why decisions were made, what decisions were considered and rejected, and traceability of requirements (HOFMEISTER *et al.*, 2007).

Architectural Evaluation Architectural evaluation ensures that the architectural design decisions made are the right ones. The candidate architectural solutions are measured against the architectural requirements. Although repeated evaluations of different architectural solutions are expected, the eventual result of architectural evaluation is the validated architecture (HOFMEISTER *et al.*, 2007). This activity also receives as input the architectural requirements, but generates as output the final validated architectures.

- **Validated Architectures:** The validated architectures consist of those candidate architectural solutions that are consistent with the architectural requirements. Those architectures must also include information about the design rationale (HOFMEISTER *et al.*, 2007).

3.2 KnowSoS

3.2.1 Architectural Analysis

3.2.1.1 Context

KnowSoS is proposed as a software architecture with the objective of enabling the extraction of knowledge in an SoS. Ideally, the architecture must not describe only how to apply data min-

ing/machine learning techniques to the collected data, but rather must address the entire process of knowledge creation and application, which includes: (i) data collection from constituents; (ii) pre-processing of such data, which may include integration, cleaning and storage; (iii) the development of knowledge discovery techniques; and (iv) the analysis of knowledge according to the missions of the system, so that it can facilitate the decision-making process of the system's stakeholders, or even the application of this knowledge in the system in a self-adaptive way.

The need for this architecture arises in a scenario where the amount of data generated by any simple system is huge, in addition to the fact that such data are generated in a velocity that makes both processing and analysis difficult. In this sense, SoS are usually built in critical domains such as health, avionics, automotive, and emergency management. For each domain, this class of system has several constituents that, although independent, can be simple or complex. In both cases, they generate a large amount of data that can be useful both for their operation and for the SoS as a whole.

The importance of this data is related to their impact in the SoS missions fulfillment. In a health domain, for example, there are several systems that measure the vital signs of the patient, systems that control medicines, administrative systems, among others. If one of the missions of this SoS is *"To prevent the patient from having a high blood pressure crisis"*, it is important that all data provided by the constituents that measure the patient's vital signs are analyzed and processed in real time, and if any change is identified or even predicted the system should trigger an specific medication to be inserted into the patient or send alerts to the administrative systems (i.e., doctor's or nurser's interfaces) advising that the patient needs attention. This scenario exemplifies data collection, integration, knowledge discovery, and even the application of this knowledge back into the system.

In another scenario, in the emergency management domain, there are sensor-linked systems that measure temperature, water river volumes, wind speed, as well as those that collect geographic information from the population, which can be linked to social networks. This SoS may have as its mission *"Monitor the region, and if a flood is detected, alert the emergency services"*. As in the previous scenario, the data must to be collected and analyzed in real time, and if any event is detected, alerts should be sent to the relevant services.

The problem identified is that often the constituents of SoS are not designed to participate in a larger system, since they have their independent functionalities. In this light, because there was no concern about the construction of intelligent SoS, these data collection and processing functions were not developed. In this scenario, KnowSoS serves as an aid to the development of a system that can be inserted as an SoS constituent and will manage the entire process of data management and knowledge extraction required to fulfill the SoS mission.

3.2.1.2 Architectural Concerns

Regarding the data, KnowSoS should not be responsible for managing the storage and management of data generated and used by a single constituent. Instead, it should only concern with the data that is sent to the central entity and which may be useful for the operation of the SoS as a whole.

KnowSoS could be applied to three types of SoS: directed, acknowledged, and collaborative. In the first two, a central entity already exists, so the developed system could be coupled to that entity that already has a means of integration with the other constituents. However, in the last one, KnowSoS must also provide a solution of communication with the constituent systems and must play a central, non-controlling, but managerial role. The architecture should not provide a solution to deal with virtual SoS, since its definitions are still abstract and little is known about the communication protocols among the constituents. Besides that, although designed to be included as a constituent, KnowSoS must also support the design of a system that is being planned to behave like an SoS since its development.

Considering data collection, KnowSoS should provide information on how the interfaces should be implemented in each constituent. These interfaces are responsible for providing the data belonging to each constituent so that the knowledge entity can collect them when necessary without interfering with the operation of the constituent. In addition, the data should be collected only when necessary for the accomplishment of some mission, that is, the sending of data to the entity should not be continuous if they are not necessary for the fulfilling of currently missions.

Thus, the architecture should consider a business view that details how missions can be mapped into data belonging to each constituent, and then an architecture module is responsible for coordinating the missions that are being performed, and which constituents should provide the data in specific execution scenarios.

Facing technical concerns, KnowSoS must provide scalability, that is, it should be developed to increase and support different amounts of data, processing them uniformly, allocating resources without impacting costs or efficiency. Besides that, it must be able to process large streams of data in a short period of time, thus returning the results to users as efficiently as possible. In general, due to its big data characteristics, it should enable distributed services and allow all processes to run efficiently.

Related to its design, KnowSoS must provide several views to support the development of the system, and these views must be independent in order to be implemented in modular ways. Essential views should cover: (i) data collection; (ii) integration and storage; (iii) knowledge processing and discovery; and (iv) decision making. Moreover, it should describe how these modules will be connected, and finally, provides a business view, describing how the system is implemented, from the stage of defining the missions to analyzing the results to meet them.

It is also important to highlight that this architecture is a more general approach, showing

the design of the knowledge discovery architecture as a whole. Expert details of the modules such as technologies used in each one will be addressed in future works.

3.2.1.3 Requirements

Based on the results of the SMS that addressed big data software architectures (SENA *et al.*, 2017), and the results of the SMS that focused on knowledge discovery in SoS, besides the context and the architectural concerns, we identified a general list of requirements for this specific domain that should guide the architectural solution. This list is presented in Figure 14, where the Y axis identifies the architectural requirement and the X axis shows the number of studies (related to the two mappings) that identified such requirement as an essential concern.

Figure 14 – Domain Requirements

Source: Research data.

We present a brief context of each one in this specific domain, as follows.

Performance In terms of performance, it should also encompass efficiency and efficacy, i.e., the data flow must be processed with a fast response time, both for data collection and for extracting knowledge from data. In addition, this process should be performed without breaking the available resources. In another perspective, the performance should also consider that the data processing is correct and that it generates adequate results for the context.

Scalability By approaching scalability, data flow may change according to the missions being performed, that is, some missions may require large amounts of data provided by various

constituents, and some not. In this way, the system and resources must be adapted to meet demand and to avoid bottlenecks. Moreover, the real-time operation is included as a scalability sub-feature. In several SoS, data are provided with sensors and this information needs to be analyzed at the same time as it is generated. For this, in addition to being scalable, the architecture needs to ensure efficient communication protocols.

Modularity and Loosely Coupling The main functionalities of the system must be implemented independently and distributed in modules. In addition, the execution of each module must have zero or little dependence on the execution of any other module. This requirement also makes the system maintainability and adaptability increase because if one module fails the others can continue running, and different modules with different technologies can be included and removed as needed.

Security Security issues involve authenticity, authorization, and privacy. They are essential when dealing with diverse sets of data that are provided by various constituents. In addition, it is important to consider that although there are communication contracts between the central entity and constituents for data to be collected and used, these contracts do not always exist among the constituents, i.e., data can be used but not propagated by the whole system. To address these issues, contracts need to be well defined, including authorizations to use data and collectors' authenticity.

Availability This essential requirement indicates that the data need to be available when it needs to be used, ie, the data needed to fulfill the mission that is running must be available to be collected from the constituents. In addition, the knowledge stored in the central entity must also be constantly made available for the decision-making of the stakeholders of the system related to the adaptation of the SoS.

Interoperability According to ISO/IEC 25010 ([ISO/IEC-24765, 2010](#)), interoperability is the degree to which two or more systems, products or components can exchange information and use the information that has been exchanged. In the context of SoS, considering the diversity of systems which are interacting to each other, besides the amount of data which is exchanged, interoperability should be a core concern, making sure the constituents that need knowledge are receiving it, and the information provided by the ones that are sending remains reliable. Data interoperability should also be addressed, this includes an specific module for translating data from various formats (meteorological, sensors, images, or audios) to one that can be processed by the central entity.

Consistency The consistency is mainly related to the data that is sent to the central entity, that is, the data cannot be modified to maintain the actual context of the application. In addition, if they are stored, any change in the constituents must also remain consistent with the central entity. In this context, storage/databases architectures that provide consistency must be chosen.

Facing such concerns, a set of architectural and functional requirements are considered in the KnowSoS design, as presented in Table 3.

Domain Requirements	
FR1	The architecture must enable the development of a module that allows the extraction of knowledge in SoS based on data generated from different data sources.
FR2	The architecture must enable the development of a module that communicates with SoS constituents and collect data according to the SoS missions.
FR3	The architecture must enable the development of a module responsible for storing the data provided by the constituents according to the SoS needs to fulfill the domain missions.
FR3.1	The architecture must enable the development of interfaces to be inserted in each constituent so that the data can be collected.
FR4	The architecture must enable the development of a module responsible for supporting the decision making with SoS stakeholders.
FR5	The architecture must enable the development of a module that monitors the system states.
Performance Requirements	
PR1	The architecture must enable the development of systems that perform knowledge discovery with low response time, according to the domain specifications.
PR2	The architecture must enable the development of systems that process data using domain available resources.
Scalability Requirements	
SR1	The architecture must enable the development of a systems that avoid bottlenecks in constituent systems communications.
SR2	The architecture must enable the development of a system that collects and processes the constituent data in real time according to the domain missions.
Modularity Requirements	
MR1	The architecture must enable the development of a system composed of independent modules with specific functionalities.
MR2	The architecture must allow the modules to be implemented with different technologies.
MR3	The architecture must allow the modules to be inserted or removed according to the domain's missions.
Security Requirements	
SR1	The architecture must enable the development of a central constituent that protect data provided by distributed constituents from unauthorized access.

SR2	Distributed constituents must provide contracts to the central entity detailing the permissions to collect and use their data.
Availability Requirements	
AR1	The communication interface with the constituents must be developed in such a way that the data is always available according to the needs of the missions.
AR2	The data of the storage module should be available for stakeholder analysis and for decision making whenever necessary.
Interoperability Requirements	
IR1	The storage module must support data interoperability among SoS constituents.
IR3	The architecture must enable the development of translation modules of various types of data provided by different constituents.
Consistency Requirements	
CR1	The data storage in the central entity must remain consistent with changes in the distributed constituents.
CR2	The choice of database architecture should be directed to those that provide data consistency and integrity.

Table 3 – KnowSoS Requirements

3.2.2 Architectural Synthesis

Systems developed by using KnowSoS will be included in different SoS as a new constituent responsible for extracting knowledge to support in mission fulfillment. This architecture could be used by software architects in conjunction with SoS stakeholders to help define system data flow and knowledge discovery. As any constituent, this one is inserted or removed at any time in the SoS according to the needs of the whole system.

Regarding the architectural difference between knowledge discovery process in SoS and monolithic systems, we highlight the dynamism in the input and output of the constituents at runtime, caused by the fact that for different missions the required data of each constituent is different. Thus, such characteristic must be supported by the architectural design. Moreover, in SoS, the challenges of knowledge discovery go far beyond the application of machine learning techniques and artificial intelligence, but involve data collection, integration, storage, and especially the dissemination of knowledge throughout the system.

Three views have been defined for the design of KnowSoS, aiming to clarify different perspectives on the architecture and how it should be used. The selected views are: (i) use case view; (ii) process view; and (iii) logical view. The use case view shows what the main features of the system are and how they connect with SoS authors. The process view shows the activities performed by the system to achieve the knowledge discovery goals, detailing data flow and system operation. Finally, the logical view shows how the modules of the system designed by

using KnowSoS are structured and how they connect with external constituents. Details of these views are presented as follows.

3.2.2.1 Use Case View

In Figure 15, we present an use case view addressing KnowSoS functional (domain) requirements, which were listed in Table 3. In general, the main functionality required by stakeholders and the constituent SoS is the fulfillment of the SoS missions. For this, there is a direct relationship *«include»* between this use case and the four main functionalities defined in the requirements previously: (i) data collection; (ii) data integration and storage; (iii) knowledge discovery; and (iv) provide data to support decision making.

Figure 15 – Usecase View

Source: Elaborated by the author.

3.2.2.2 Process View

In Figure 16, we present the process view composed of the main activities involved in knowledge discovery. The first activity can be started in two ways: through an external entity that selects a

mission to accomplish or through a scheduler programmed to select the mission periodically. With this, the orchestrator initiates three consecutive activities: (i) select the mission performed; (ii) refer to mission business rules, which should define which constituents should support the fulfillment of the mission that has been selected, i.e., which constituents will provide data and what kind of data will be collected; and (iii) check the state of the system.

The system monitor then performs system's state analysis to identify which constituents are active and available to provide the data. Finally, the monitor sends the system state report back to the orchestrator, which will initiate the collection requests.

With the business side concluded, the collector is responsible for collecting constituent data and sending such data to be stored in the central storage entity of SoS. The data store entity stores this data and waits until all the information defined in that mission's business rules is ready for knowledge discovery to begin.

Finally, the knowledge extractor applies machine learning techniques to support the fulfillment of the missions that are being fulfilled, and sends this data to the visualizator, which generates the reports and sends them to the orchestrator and SoS stakeholders. In this activity, the decision-making process is carried out together with the system stakeholders, who evaluate the results and define the next data flow. Finally, the results are disseminated through the constituents, and the orchestrator defines which mission has been accomplished, ending the process cycle that can be repeated.

Figure 16 – Process View

Source: Elaborated by the author.

3.2.2.3 Logical View

The overview of the architectural solutions for KnowSoS is presented in Figure 17, as a logical view. KnowSoS is a layered software architecture, which indicates that each layer represents a set of features that can be developed and managed independently of the other layers. In addition, the use of this architectural standard also improves the maintainability and modularity of the architecture, dividing the main concerns according to the requirements.

Figure 17 – Architecture Overview

Source: Elaborated by the author.

Aiming at achieving the main concerns and requirements, KnowSoS is divided in five layers, namely, collection layer, storage layer, discovery layer, decision making layer, and orchestrator layer.

The first layer is responsible for collecting the data of the various constituents and sending such data to be stored. The control of what data will be collected and which constituents will provide them is done by the orchestration layer controller. The storage layer receives the collected data from the constituents and transforms them to be stored as needed in the centralized repository. These data need to be available to be queried by the knowledge layer whenever necessary.

As the data is collected by the storage layer, the knowledge discovery layer is able to pre-process it to apply the standards according to the missions that are being fulfilled by SoS. In addition, this data is also used to query the system state. The application of the models and the control of the knowledge that is being discovered is done by the orchestrator's controller, which knows what data is needed for the accomplishment of the missions.

The knowledge is sent to the decision-making layer where system stakeholders (or systems that consume SoS) may make decisions based on the current state of the system. In addition, decision-making can also send commands to the orchestrator according to the SoS assignments so that the system can adapt and the constituents can be configured. Finally, the orchestration layer serves to control the entire flow of data throughout the application of knowledge discovery.

In addition, there are two external entities that communicate with the layers of architecture: the SoS missions and stakeholders.

- **SoS Stakeholders:** SoS stakeholders are the entities interested in fulfilling the system's missions and the knowledge that will be extracted. Stakeholders can define the structure of the SoS and use the states of the system to make adaptive decisions.
- **SoS Missions:** The SoS mission artifact represents all missions mapped so they can be considered during system operation. The missions need to represent not only the objective but also which constituents should be consulted to achieve them and different metrics of knowledge that can be used for decision making and system adaptation.

Details of each of the layers are presented in the following sections.

3.2.2.3.1 Collection Layer

Figure 18 details the structure of the data collection layer. This layer is usually executed through requests from the orchestrator layer, which manages the data flow. This orchestrator sends the request to the data collection controller, a module that knows the specific characteristics of each constituent and its interfaces, and is therefore responsible for making the data requests specific to each constituent. The structure of each constituent is not part of the scope of this architecture. Once the constituents provide the data, they are received by a bridge that stores such data until the collection is complete and the data can be sent to the storage layer.

Figure 18 – Data Collection Layer

Source: Elaborated by the author.

The main difference between this module in SoS and in monolithic systems is that different interfaces need to be developed to communicate with each constituent. In addition, a bridge must be developed to standardize data types in a format that can be sent to the storage layer.

3.2.2.3.2 Storage Layer

Figure 19 presents the structure of the storage layer, responsible for the storage of data needed to extract knowledge from the system. This layer takes the data from the collection layer, and transforms it into a format that can be understood and processed by the SoS. For this, a transformation module divides the data into a raw data set, and a meta data set, which stores the structure of the stored data (such structure depends on how data are being stored, i.e., by columns, by row, and so forth). This data is accessed by the knowledge layer through query tools, which access the metadata database to find the data requested, and then query it in the raw database.

Figure 19 – Data Storage Layer

Source: Elaborated by the author.

3.2.2.3.3 Knowledge Discovery Layer

Figure 20 shows the structure of the knowledge discovery layer. This layer is divided into a pipeline composed of three steps: (i) data: which accesses the SoS storage layer and preprocesses this data, including data transformation, data cleaning, data reduction and data integration; (ii) processing: responsible for actually applying machine learning models to data, i.e. classification, regression, clustering or reinforcement models (among others); and (iii) results: which is responsible for sending the result of the techniques to the decision-making layer, where the next system flow will be defined. This complete flow is controlled and required by the orchestrator layer, which knows the constituents and business rules of the system.

Figure 20 – Discovery Layer

Source: Elaborated by the author.

3.2.2.3.4 Decision Making Layer

Figure 21 details the structure of the decision making layer, responsible for supporting the stakeholders to operate the system. The decision-making layer receives the data generated by the knowledge discovery layer, and generates key visualizations for stakeholders to access and define which flows to execute along with system missions. Such decisions must be recorded so that SoS can learn from them and automate them in similar system scenarios in the future. Decisions are then sent to the orchestrator layer, which will be responsible for future streams.

Figure 21 – Decision Making Layer

Source: Elaborated by the author.

3.2.2.3.5 Orchestrator Layer

Figure 22 presents the structure of the orchestrator layer, responsible for controlling the data flow. This is the main layer of the system as it is central and controls the entire data flow. It is responsible for storing system business rules defined using missions. These rules are established to help define which constituents should assist with each mission and what data they should provide for each mission. Thus, system monitoring should periodically analyze SoS to identify where the data is at any given time. Knowing the state of the system and the business rules, the controller sends requests to all other layers requesting their execution.

Figure 22 – Orchestrator Layer

Source: Elaborated by the author.

3.3 Guidelines for using KnowSoS

So far in this chapter, we presented a set of views that encompass an architecture that aims to design a system responsible for performing knowledge discovery in SoS. However, the

knowledge discovery process is more comprehensive and involves the study of constituents, the analysis of context, requirements, and also the dissemination of knowledge. Thus, in this section some guidelines are presented, which can serve as a guide to instantiate KnowSoS to an specific domain and conduct a knowledge discovery project in a running SoS or the initial architecture structure in an SoS that is in the design phase.

The establishment of such guidelines was supported by consulting experts in software engineering, software architecture, SoSs, and machine learning. Firstly, experts in machine learning provided knowledge and techniques to acquire and manage information from SoS. With their support, we defined the following data flow to be considered in the guidelines: (i) data acquisition from constituent systems; (ii) data processing through the application of learning techniques to extract knowledge; (iii) extracted knowledge dissemination to constituents; and (iv) knowledge application by stakeholders for decision making purposes. Then, experts in software engineering, software architecture, and SoSs were consulted to structure in a systematic way the flow of information identified in the previous step. In addition, we considered information about the requirements from software architectures in the big data context, such as scalability and interoperability, which were extracted from a systematic mapping study performed by our research group (SENA *et al.*, 2017).

The resulting guidelines is composed of five phases, as presented in Figure 23 and detailed as follows. Specific stakeholders for each phase were identified, requiring their participation during the conduction of the process. As shown in Figure 23, the flow between phases are bidirectional, due to the fact that the decisions made in each phase may imply changes in the previous steps for improving the results, e.g., change of the selected technique, reduction of the scope, or adaptation of the architecture.

3.3.1 Phase 1: Identification of the SoS constituents

The first phase aims to characterize the SoS by gathering information related to SoS characteristics and architecture: (i) type of SoS, namely, acknowledged, collaborative, directed and virtual; (ii) information provided by constituent system and how it is communicated; (iii) when possible, interactions between constituents and the SoS; (iv) constituents interfaces, through which the service developed can receive and send data; and (v) technology used by all systems. In details, such characterization includes two steps: (i) identification of the constituents in which will be identified interfaces and domain information; (ii) establishment of the SoS architecture, identifying the connection among the constituents. In general, the goal is to understand how the SoS works, what constituents exist and how they communicate. This analysis is essential to understand in which architecture the system developed when instantiating KnowSoS will be inserted.

Both steps must be supported by software architects, who can identify architectural patterns and the best way to connect the service to the architecture, and domain experts, who

Figure 23 – Guidelines for using KnowSoS.

Source: Elaborated by the author.

can provide information related to each constituent, their interfaces and individual missions. As results, an architecture for the SoS will be established.

3.3.2 Phase 2: Identification of the SoS Missions

This phase aims to identify the scope of the domain problem that will be solved by extracting the knowledge from the SoS. Thus, in this step we define the missions that the system developed when instantiating KnowSoS will support to fulfill. This phase cannot be generalized because each SoS belongs to its own domain and manages different information. For each SoS, several scopes could be identified.

The identification of such scope is performed in four steps: (i) definition of the problem to be solved; (ii) identification of the stakeholders and constituents that belong to such scope, i.e., the constituents that will provide data, and the constituents and stakeholders that will use the knowledge extracted from such data; (iii) establishment of the interfaces for data collection; and (iv) consolidation of missions in a formal language to be identified by the system. Besides that, this phase must be supported by domain experts, who can provide information regarding the scope in which knowledge should be acquired, besides identifying the data sources from the constituents.

3.3.3 Phase 3: Selection of the Knowledge Discovery Approach

This phase aims to translate the scope defined in phase 2 into a technical solution perspective (machine learning techniques), i.e., identifying how the scope could be computationally solved using such techniques. Such analysis is performed in the following two steps: (i) identification of the problem class, i.e., if the problem addressed is concerned with classification, clustering, or information prediction; and (ii) selection of the most adequate technique for the problem. Moreover, both steps must be supported by machine learning experts, who can provide information regarding to techniques more used and more efficient for a certain problem. Rationale on alternative techniques that can be used to solve the problem must be documented. As result, the technique to be implemented is identified.

3.3.4 Phase 4: Application of KnowSoS

At this stage, the understanding of SoS is already complete, that is, (i) the constituents of the system have already been identified; (ii) the communication interfaces between the constituents have already been mapped; and (iii) the mission to be accomplished has already been selected. In addition, the study of knowledge discovery techniques (machine learning) has already been conducted and the techniques were selected. Therefore, the main objective of this phase is to design and develop the system that will perform the knowledge discovery and will be inserted in the SoS architecture.

To support in the design, KnowSoS was proposed and should be used. Thus, the instantiation must guarantee the existence of all the proposed components (collection, storage, integration, discovery, orchestration, and decision making) and the knowledge of the SoS architecture acquired in the first three phases. Moreover, to the development, this phase is divided in five steps: (i) identification of system domain requirements, which must address functional requirements specific of the domain, such as *the service must keep the knowledge database updated, in order to provide historical information to future analysis*, and non-functional requirements, i.e., quality attributes, such as scalability and interoperability; (ii) design a class diagram, in which the implementation will be modelled; (iii) selection of technologies that better fit to the capacities of the developers; (iv) development of the system including data processing and the machine learning technique; and (v) testing the system to verify if it is meeting the requirement.

Besides that, this phase must be supported by software engineers in order to verify the correctness of the diagrams; software architects, to ensure that the quality attributes are met; and developers, responsible for the implementation. As result, this phase provides the system designed and implemented.

3.3.5 Phase 5: Validation of the Knowledge System

This last phase aims to validate the knowledge discovery system developed in phase 4. Techniques frequently used to evaluate applications based on machine learning techniques are bootstrap and cross-validation that assess whether the technique really learned from data to create a model which can be generalized and applied for different datasets (KOHAVI *et al.*, 1995). Bootstrap works by sampling with replacement the original data several times, and calculating the average score as estimation for the model performance, and cross-validation splits the original datasets in two different parts, one for training and another one for testing (KOHAVI *et al.*, 1995; KOTSIANTIS *et al.*, 2007).

Furthermore, metrics, such as accuracy, precision and recall, can be used to investigate how precise the implementation is to acquire useful knowledge to the stakeholders. This phase must be supported by machine learning experts, who are responsible for the evaluation, and by domain experts who are responsible to analyze data utility. Based on the results of such analysis, the machine learning technique can be improved.

3.4 Final Remarks

This chapter introduced KnowSoS, a software architecture for the discovery of knowledge in SoS. We discussed its use as a support mechanism to support SoS architects to develop a system that can be inserted into a SoS and is responsible for data collection, storage and discovering the knowledge of that data, in order to assist in the fulfillment of SoS missions.

KnowSoS differs from other approaches of knowledge discovery because it considers specific characteristics of SoS, such as the challenges of dealing with different constituents, the orchestrator built based on business rules, and responsible for controlling the data, and the necessity of such control to accomplish different missions. In addition, it must be efficient to support the dynamism of the entry and exit of SoS constituents.

To the best of our knowledge, no approach has been proposed to achieve knowledge discovery in this class of systems. In fact, as discussed in the SMS presented in Chapter 2, few initiatives have been proposed to deal with data in SoS, so KnowSoS is a first step forward in this direction aiming to bridge this gap.

In the next chapter, we present the evaluation of KnowSoS, which includes a proof of concept in two different domains (i.e., human resources domain and emergency management domain), detailing how KnowSoS can support the SoS in the knowledge discovery process and bring benefits to the system's context.

KnowSoS Evaluation

According to [Oxford \(2014\)](#), a proof-of-concept is a noun phrase attributable to “evidence (usually deriving from an experiment or pilot project) demonstrating that a design concept, business idea, or other product is feasible” ([OXFORD, 2014](#)). In general, it describes research in the beginning stages, at the cutting edge of new applications or technologies, and used to mark out scientific research as potentially extendable and scalable ([KENDIG, 2016](#)). The aim of conducting such method is to prove that a casual connection hypothesized, structure proposed, function suggested, or methodological approach taken in the research is valid in at least one actual case ([KENDIG, 2016](#)).

Regarding an architectural perspective, the purpose of the proof of concept is to determine whether there exists, or is likely to exist, a solution that satisfies the architectural requirements ([RCP, 2001](#)). Therefore, in this case, a proof of concept may take many forms to represent a solution, such as, a simulation, an executable prototype, list of known technologies, sketch of a concept model, and so forth ([RCP, 2001](#); [KENDIG, 2016](#)). It brings results to help determine the feasibility of the project, assess the technical risks attaching to its development, and formulate and refine the architecturally-significant requirements ([RCP, 2001](#)).

As there are several open challenges regarding data manipulation in SoS, and especially regarding the availability of this data and architectures, it is still a challenge to conduct a robust experiment to verify the feasibility of a proposal. Thus, in this project, we conducted a proof of concept in two different domains to show that a system designed and developed using the proposed architecture, when inserted in a SoS context, can bring benefits to the domain and automate data flow.

To develop both systems, we followed the process presented in the previous chapter, i.e., the guidelines for conducting knowledge discovery in SoS. In the first one, we chose an SoS in the Human Resources domain (HR-SoS), focused on job applications, in which there are thousands of dynamic constituents that enter and leave the system providing a huge variety of

data, which could be used to recommend jobs and bring benefits for both the constituents who provide the job data and for the stakeholders who consume that content. In the second domain, we use KnowSoS and the guidelines to develop a prototype in the Emergency Management domain (EM-SoS) and support emergency stakeholders with alerts when a natural disaster is happening.

In Section 4.1, we present the design and implementation of the prototype, the context of knowledge, and the main results of the system in the Human Resources domain. Then, in Section 4.2 we present the results of applying the same process in the Emergency Management domain. Finally, in Section 4.3 we present some general discussions with the final remarks.

4.1 Human Resources Domain

4.1.1 Phase 1: Identification of the SoS constituents

As presented in the previous section, we are analyzing a SoS in the human resources domain (HR-SoS). The main context in which this SoS operates is in the announcement of job positions for users on various platforms, and in the application of users in each of the positions according to their profile and preferences. Thus, there is a central entity that collects job positions from various systems provided by companies that wish to offer their vacancy in this service.

HR-SoS has two main classes of constituents, social networks and companies systems. Social networks can provide data related to the professional and personal profiles of each user, including curriculum and preferences, as well as being a channel through which several companies advertise their job openings, such as LinkedIn¹. Companies systems, on the other hand, are independent because they manage their own business and manage their data internally, but they also serve as a feeding channel for SoS, since vacancies can be collected and distributed through the system. To complete the architecture, these companies also send, constantly, their positions to a central entity, which provides this data in an integrated manner to system users who are looking for jobs.

When a user is interested in a job opening by the central entity, he is referred to the responsible company's system, which follows the selection process independently. In addition, data collected from social networks can also serve as an input to assess the profile of candidates and can be consumed by companies. Thus, the relationship of the central entity with all the constituents of the system is always in two directions, in relation to the collection of data from the constituents and provision of information for them.

¹ www.linkedin.com

4.1.2 Phase 2: Identification of the SoS missions

Personalized systems such as the recommendation ones have attracted the attention of researchers and industry professionals for their ability to provide customized customer service in various segments of society, such as movies, music and hotels, thereby increasing their satisfaction. (MALINOWSKI *et al.*, 2006; HONG *et al.*, 2013). One of the scenarios that has been greatly benefited from such systems is the search for jobs. With the evolution of technology and the rise of online social interactivity, many job seekers spread their personal information and interests on the Internet. In addition, companies have constantly used these media to publicize job offers to potential employees. (YANG *et al.*, 2017; KENTHAPADI *et al.*, 2017).

The recruiting process has been extensively studied in human resources field, giving increasing attention to the recruitment's based on web information, especially gathered from networks such as LinkedIn², Facebook³, Twitter⁴, and Xing⁵ (DOMENICONI *et al.*, 2016). However, the amount of job information and online candidates is overloaded, which makes it difficult to select which candidates should be hired for jobs and which jobs should be recommended for them (MISHRA; REDDY, 2016; AL-OTAIBI; YKHLEF, 2012).

In the HR-SoS scenario, an immense amount of job openings data is generated by several segments of companies (e.g., tech companies, healthcare companies, and domestic service companies), in addition to the curriculum and profile of candidates on social networks, and from the job application data themselves. However, this data is used only for operational purposes and sent from one system to the other without any knowledge being acquired. In this way, it would be interesting for the data to be used not only as a flow, but to support decision making both by the company that needs to select the best candidates for their positions, and by the candidates that need to choose the best job opening for their profiles. Therefore, the main mission of this SoS is to recommend the right job openings for candidates, and the right candidates for companies.

For this HR-SoS system, we will consider nine different constituents⁶, namely, (i) a system of tech company positions; (ii) administrative assistant positions; (iii) housekeeper positions; (iv) managers positions; (v) teaching positions; (vi) sales positions; (vii) one system that has positions in several different segments; (viii) a system that contains historical and recent data from several candidates; and (ix) a system that used to be a central entity and has data on candidates' applications.

Therefore, in this system, we will consider the same context in three perspectives on data. The first one is regarding **users**, in which the constituent provides relevant information of each user, such as: city, state, zip code, degree type, major, graduation date, work history count, total years of experience, whether the user is currently employed, and if the user has already

² <https://www.linkedin.com>

³ <https://www.facebook.com>

⁴ <https://www.twitter.com>

⁵ <https://www.xing.com>

⁶ Data were provided by CareerBuilders organization and is available at <https://www.careerbuilder.com/>.

managed others. In total, this system provides information of 389708 unique users. In the second perspective, each of the company systems provides information related to the **job openings**, i.e., contains all information about different jobs, including title description, requirements, city, state, country, and zip code. Considering all the companies, we have 1091923 job positions in the dataset. Finally, the last perspective is regarding **applications**, in which the system provides information of all applications performed by each user in a specific job vacancy. Its features are: application date, job id and user id. 1603111 historical applications were performed during the study period.

4.1.3 Phase 3: Selection of the Discovery Approach

In this system, the approach used to discover knowledge in SoS will be a recommendation technique. The methodology for performing the recommendations will be presented in this phase of the process, and then it will be implemented in the knowledge discovery module of the architecture in the next phase. In summary, for the recommendation, three strategies will be considered, one that recommends based only on similarity among jobs, one based on similarity among users, and the last one that recommends based on the intersection of users' applications in different jobs. We present details of each strategy, as follows.

4.1.3.1 Strategy 1: Similarity among jobs

In the first strategy, jobs are recommended using only similarity between jobs features, without considering any characteristics of users. The workflow designed to recommend jobs using this strategy is presented in Figure 24.

In general, for each user u_i in the dataset, we split the n jobs in which u_i has applied into a training set containing $n - 1$ jobs and a testing set with the remaining application. The main idea is to recommend jobs that are similar to the job in the testing set. For this, we generate a latent vector⁷, using simple mean, which represents the main characteristics of the jobs that the user has applied (the training set), and calculate the similarity of such latent vector to all the jobs in the dataset. Finally, the k most similar jobs to this latent vector are recommended to the user, and then this recommendation is evaluated.

4.1.3.2 Strategy 2: Similarity among Users

In the second strategy, recommendations are made considering only the similarity between users, ignoring the historical applications. The flow of this strategy is presented in Figure 25. In this strategy, the testing set is created in the same way as in the previous strategy, considering the most recent job the user has applied. We calculate the similarity between user u_i and all other users in the database, and then indicate which are the closest m users to the one we are recommending jobs.

⁷ a vector of data not directly observed

Figure 24 – Workflow of Strategy 1

Source: Elaborated by the author.

At this point, it is recommended to the user u_i the jobs in which u_i nearest m neighbors have applied, but he had not. Finally, this recommendation is evaluated.

Figure 25 – Workflow of Strategy 2

Source: Elaborated by the author.

In some cases, considering the nearest neighbors, more jobs than necessary are recommended for each user, as this value varies according to the amount of jobs in which u_i neighbors

have applied. In this scenario, we mix this strategy with the first one, and recommend the k jobs where neighbors have applied that are most similar to the u_i training set.

4.1.3.3 Strategy 3: Intersection among applications

In this last strategy, we do not consider intrinsic characteristics of either jobs or users, but only the history of applications made by each user. For this, given a user u_i , we calculate the number of common applications that this user and all others in the dataset applied. We then select m users with the most job application intersections, and recommend to the user u_i the jobs that these m users applied that are different from those u_i applied. The same issue of the number of jobs discussed in strategy 2 may occur in this case, and to solve this problem, we apply a mix with strategy 1. Figure 3 shows an example of strategy 3.

Figure 26 – Descriptive of jobs and users

Source: Elaborated by the author.

In this example, we have the u_i user who applied to the yellow, green, pink, and white jobs (the white one is hidden in the testing set). After that, the applications of u_i is compared with the three other users of the base, where the third one has the most intersections with u_i (2 similar applications: yellow and pink jobs). Therefore, the gray job is recommended for the user u_i , as it is a job that someone very similar to u_i has applied for and u_i did not. Finally, this gray job is compared to the job in the testing dataset.

4.1.4 Phase 4: Application of KnowSoS

Considering the constituents identified in Phase 1, the SoS missions selected in Phase 2, and the machine learning techniques discussed in Phase 3, we have an overall understanding of

the HR-SoS architecture. Therefore, in this phase, we present the design and development of a system that will be integrated in this SoS, collect data from the constituents identified and use the techniques selected to support in the fulfillment of the established mission.

The design of the knowledge discovery system architecture was performed using Know-SoS, presented in the previous chapter. Thus, in addition to the constituents already discussed, the layers and modules were included to support data flow and facilitate knowledge discovery. The results found in the decision-making layer will be sent to the users of the system, who is consuming the knowledge discovered.

In Figure 27 we present the final architecture of the knowledge discovery system that is integrated into HR-SoS (Human Resources SoS). In relation to the classification proposed by Maier (1998a), HR-SoS is an Acknowledged SoS, i.e., a system with recognized objectives and a designated entity manager to fulfill its purposes. Besides that, each constituent retains their independent objectives, funding and development approaches.

As the system architecture is defined, we have instantiated it and develop the system. In this evaluation, the goal is to develop a system that shows the main functionalities that the knowledge discovery system could have. Therefore, we do not focus here on architectural quality attributes of that system.

In relation to development, the Python⁸ programming language was used, which facilitates the use of machine learning and validation techniques. The Pandas⁹ library was used for efficient data manipulation. To simulate the execution of each module, they were implemented as independent classes with methods that represent the communication between the modules and the data flow.

To the knowledge discovery, we used data provided by the nine constituents. For the execution of the evaluation methods, all of them must be numerical and in some cases normalized. Thus, several techniques were applied in the job and user bases to transform the data into the same format, as described below. When standardized, such data is stored in a relational database.

To represent each job, we will use the following features: title, description, requirements and city, as these are the features that really matter in choosing a job. The city was chosen because the distance from the user's city to the city of employment is a factor that may also impact the application or not for that vacancy. We concatenate these features as a single feature, and then, the following techniques were applied.

Non-textual items removal: for each job, we removed items such as: HTML tags, spaces, special characters, emoticons, hyperlinks, and line breaks. We also turned all letters to lowercase.

⁸ <https://www.python.org/>

⁹ <https://pandas.pydata.org/>

Figure 27 – Knowledge Discovery System Architecture proposed for Human Resources SoS

Source: Elaborated by the author.

Stop words: Stop words are the extremely common words in a vocabulary which have little value in helping to select documents, analyze contexts or to build a classifier. We used Natural Language Toolkit¹⁰ to remove the stop words of each job, aiming at keeping only the intrinsic information about an specific job.

¹⁰ <https://www.nltk.org/>

4.1.5 Phase 5: Validation of the Knowledge System

4.1.5.1 Descriptive Analysis

Before the development of the models, it is important to analyze how the data is distributed. In Figure 29, we present the boxplots that show the frequency of user applications per job, and the frequency of jobs by each user. In fact, the distributions are very close with approximately 75% of the frequencies below 5 (in average, five applicants for each job and five jobs for each applicant), but we have a lot of jobs and users with frequencies higher than 5, which in this case cannot be considered as noise, as they do not represent a data error, but just a distinct distribution.

Figure 29 – Descriptive of jobs and users

Source: Research data.

Regarding the application time, our data is concentrated in a few months window, and each vacancy is open on average for 30 days. Thus, since they represent a regular pattern, it is not a factor that can positively or negatively impact results.

4.1.5.2 Evaluation Method

The evaluation metric used in this evaluation is the cosine similarity, which is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them (LI; HAN, 2013). Such metric is derived from the Euclidean dot product formula, as presented in Equation 4.2, which generates the formula presented in Equation 4.3.

$$u \cdot v = \|u\| \|v\| \cos\theta \quad (4.2)$$

$$\cos\theta = \frac{u \cdot v}{\|u\| \|v\|} \quad (4.3)$$

Cosine similarity has a special property that makes it suitable for metric learning: the resulting similarity measure is always within the range of -1 (when the angle between the vectors is maximum) and 1 (when the angle between the vectors is minimum, indicating similarity). This property allows the objective function to be simple and effective (NGUYEN; BAI, 2010). Thus, our goal is to maximize the cosine distance between the enrolled jobs and recommended jobs, which indicates that the system is indeed recommending similar jobs to those that the user is interested in.

4.1.5.3 Results

In this section we present the results of applying the strategies in the database. To define the best parameters of our strategies, i.e, the ideal number of nearest neighbors and the ideal number of jobs to be recommended, a third dataset (namely validation dataset) was defined using a random job of training data that differed from the the one in testing set. After several executions, we set out to consider the 10 most similar users and recommend the first 10 jobs on the recommendation list.

To apply the strategies, we use the complete dataset, instead of using data provided by only one constituent at a time. Before recommending jobs, we analyzed the similarity among all job offers and all users in the dataset to assess whether there is homogeneity in the data and to set the expectations of the recommendations. As a result, we got a similarity of 0.30 among job offers, and a similarity of 0.61 among users.

Thus, although we want to maximize the similarity in the recommendation, it is important that the resulting metric should be greater than the average similarity, which indicates that the approach would be better than a random recommendation. In strategy 1, we achieved an average similarity of 0.50, which is higher than the global average. However, the precision of hitting the exactly testing set job is low. This is caused by the number of jobs to be compared in the database, which is huge, making it difficult to find the specific one. Even so, the recommended jobs are very similar to those we want to predict. An example of this fact can be seen in Table 4, where the hidden job that should be recommended was a Therapist spot, and in the recommendations we have several jobs related to therapies, youth care and human resources positions.

Considering the second strategy, we achieved a similarity of 0.55, which is better than the first, and indicates that similarity among users may be more efficient at recommending jobs than similarity between job vacancies. One hypothesis that can be built on this strategy is that each user's educational background, job history, and personal data can actually define which jobs the user would look for. In Table 5 we present an example of the results using this strategy. In this case, although we are not recommending exactly the correct job, the recommended ones

Testing	
1699	Therapist
Recommended	
4408	Care Manager
184119082	Human Resources Specialist
23952	Youth Care Worker-Group Home
28245	Recruiter
27599	Care Coordinator
28129	High Needs Case Manager
28419	Reintegration Worker
29675	Team Leader
29947	Student Service Representative
29948	Student Affairs Representative

Table 4 – Recommendations using Strategy 1

are very close from it in terms of description and requirements, indicating a consistent result.

Testing	
7491	Customer Service Payment Rep Job
Recommended	
12859	Warehouse
20380	Client Care Specialist
14163	Account Manager
16625	Packaging
16946	Experienced Sales Representative
17443	Administrative Assistant
18343	Staff Accountant
20287	Customer Service
20652	Secretary
22718	Health and Wellness Sales Professional

Table 5 – Recommendations using Strategy 2

Finally, in strategy 3, where we considered only the similarity in job applications by each user, the average similarity obtained was 0.51, which is higher than similarity considering only jobs, and worse than similarity considering users. An example of the result of this approach can be seen in Table 6, where we can see that the recommended assistant and reception jobs are very similar to the job in which the user actually applied.

In general, despite achieving satisfactory results, strategies were expected to recommend jobs more similar to the jobs we are looking for. We believe one reason that can cause this is that we have a huge amount of job offers, which are distributed in very different areas of knowledge. To verify this hypothesis, we created 6 clusters of distinct areas based on word frequency. The defined clusters were: i) system related offers; ii) sales offers; iii) housekeeper and related offers;

Testing	
7280	Receptionist / Administrative Assistant
Recommended	
361	Data Entry Clerk/Warehouse Associate
806	Leasing Consultant
908	Office Assistant
2155	Sales
3692	Administrative Assistant
3711	Receptionist
4512	Administrative Assistant
6093	Entry Position
7081	Executive Assistant
7168	Incoming Mailroom Clerk

Table 6 – Recommendations using Strategy 3

iv) manager offers; v) assistant offers; and vi) professor offers. Clusters were defined based on the initial system which provided the data. After filtering out the offers in each cluster, all the pre-processing and the three strategies were applied to each one. Table 7 presents the main information considering the complete dataset and each of the previously defined clusters. As we might expect, in all clusters the similarity between job openings is greater than when we consider the full dataset. Similarly, in most clusters, the similarity between users is also higher. In the last row, we can observe the average of the similarities of the clusters, and when compared with the complete dataset, we can highlight that the average stands out in relation to the global results.

Input set	Dimension	Average Similarities	
		Jobs	Users
General	1091923	0.30	0.61
Sales	146073	0.46	0.61
Systems	17827	0.43	0.65
Housekeeper	613	0.36	0.57
Manager	132001	0.39	0.60
Assistant	47669	0.34	0.67
Professor	332	0.40	0.60
Cluster average	57420	0.41	0.61

Table 7 – Statistics of each dataset

Moreover, in this table, we show the results of two different ways to recommend jobs: (i) using random recommendation (column Rand), i.e., recommend a random job from the jobs list; and (ii) using popular recommendation (column Pop), i.e., recommend to each user the most popular jobs in the dataset. As expected, in the specific clusters, the random and popular approaches are not much better than strategies, since even with a random approach, the job is still part of the same speciality. However, when we consider the complete dataset, the strategies stand out because they actually learn information about users, reinforcing the need for such systems.

Input set	Rand	Pop	ST1	ST2	ST3
General	0.40	0.43	0.50	0.55	0.51
Sales	0.61	0.61	0.54	0.67	0.68
Systems	0.57	0.59	0.59	0.64	0.65
Housekeeper	0.50	0.51	0.53	0.57	0.58
Manager	0.51	0.47	0.46	0.60	0.64
Assistant	0.47	0.51	0.52	0.60	0.49
Professor	0.52	0.63	0.56	0.69	0.66
Cluster average	0.55	0.54	0.51	0.63	0.64

Table 8 – Results in the total dataset and in each cluster

An important point to be highlighted after obtaining the results of the three techniques is that they can be integrated to be used at different moments in a selection process. It is obvious that approach 2 should be used when a new user enters the system, as we compare only its intrinsic characteristics with all others since we do not have information about the jobs the user is interested. Approach 1 can be used with a new job in the system, which can avoid the cold start problem in recommendations. Finally, approach 3 can be used when we have well-established historical data. Although this evaluation presents a comparison, it is interesting to note that in the end, the three approaches can be used together in order to maximize similarity and serve as a recommendation to users at various stages of system development and evolution.

Analyzing the results in a perspective superior to the numbers, we can consider that when combining the approaches and even considering others, the use of a system that extracts knowledge for this SoS brings several benefits in the domain and can increase the intelligence in the operation of the system.

4.2 Emergency Management Domain

As presented in the last section in a HR-SoS, we continue to evaluate our proposal through the conduction of a proof of concept. The goal of the study in the emergency management domain (EM-SoS) is to investigate whether the proposal is valid to guide the knowledge extraction in a real SoS, whose mission is to detect critical situations during natural disasters, and support the emergency services to identify affected regions. Many of the decisions made in the development of this module are similar to those made in the HR-SoS. This way, these details will not be explicated so that they are not repetitive.

4.2.1 Phase 1: Identification of the SoS constituents

In the first phase, we characterized the SoS from which the knowledge must be extracted. EM-SoS is considered as an acknowledged SoS, which has a central entity to manage the fulfillment of the recognized objectives, but the constituents systems keep their independent ownership,

objectives, funding, development and sustained approaches (MAIER, 1998b). The following constituent systems of EM-SoS were identified: (i) **social networks**: they represent a strong information source during emergency situations, due to the several users that shared information to warn their friends regarding the occurrence of an event; (ii) **monitoring sensors**: they are responsible for monitoring several natural situations in the cities and to identify the occurrence of small situations such as flood. (iii) **volunteered geographic information**: it is used to create, assemble, and disseminate geographic data provided voluntarily by individuals (GOODCHILD, 2007); and (iv) **emergency services**: they are consumers of the information processed in the SoS, and are the responsible for decision making in crisis situations.

4.2.2 Phase 2: Identification of the SoS missions

Regarding to the emergency management problems, hurricanes have been a recurring source of natural disasters in recent years, affecting not only the environment, but the people's life. In 2017, the hurricane season exceed the index of previous years, implying in an extensive damage in people property, several deaths in the community and also a huge impact to governments (ERDMAN, 2017; ASTOR, 2017). Therefore, the ability to detect the occurrence of a hurricane could improve the emergency management, preparing the concerned services, such as firefighters, police, and hospitals, to better assist the population. Besides that, due to the diversity of events that occur during a hurricane, such as landfalls and floods, the issue of alerts with information about the places that need more help during the disasters could also improve victims assistance.

During crisis situation, people turn to social media platforms for exchanging information about the incident, or to express their views about, for instance, relief works and the government role in the response and recovery efforts (PALEN *et al.*, 2010). Often, messages exchanged through the social media contain valuable information that can contribute to a situational awareness during an incident if analyzed in a timely and efficient manner (STARBIRD *et al.*, 2010; LATONERO; SHKLOVSKI, 2010; CHOWDHURY *et al.*, 2013). For this purpose, social networks such as *Twitter* have become a major channel for communication during disasters and a strong data source for emergency responders (ZAREIE; SHEIKHAHMADI, 2017). Therefore, the social networks constituents are selected as the main data source to support the knowledge extraction that will be performed to solve the domain problems. Besides that, the emergency responders constituents, i.e., firefighters, police department, and hospitals, will also be considered as the stakeholders using the knowledge to make decisions.

In this system, considering the huge impact that Hurricane Irma had recently in the Caribbean Sea¹¹, data related to such disaster will be considered. To collect the data, we used Tweepy¹² API, a complete set of twitter libraries for Python. As results, we collected 124874 twitter messages from the Miami region in the period of September 07 to September 15. Such

¹¹ <<https://www.theguardian.com/world/2017/sep/11/hurricane-irma-storm-numbers>>

¹² <http://www.tweepy.org>

region is presented in Figure 30.

Figure 30 – Data collection window.

Source: Elaborated by the author.

4.2.3 Phase 3: Selection of the Discovery Approach

It was considered that the problem identified in last phase can be solved using classification techniques of machine learning. Specifically, we intend that inputs offered by constituent systems are classified in a pre-defined set of categories.

Two classifications were proposed. The first one aims to detect **occurrence of disaster events, e.g., floods, landfalls, or hurricanes.**; hence, two categories were considered for this classification: (i) it is an occurrence; (ii) it is not an occurrence. To support the occurrence classification it was created a taxonomy of terms usually identified in a disaster event. Such terms were extracted from news published by recognized journals, i.e., BBC News, The Daily Telegraph, The Guardian, The New York Times, The Sun, and The Wall Street Journal. Selected news were related to reports about the hurricane Irma passing on the Caribbean sea. One-hundred and four terms were extracted from these news. The terms blackout, calamity, and inundation, compose the created taxonomy. The taxonomy was validated with experts in disasters management.

The second classification proposed was related with the **impact level of a disaster event**. Two categories of impact level were considered: **(i) low impact:** events that do not present risks to people's lives nor damage to city infrastructures but are related to preparedness, storage and protection; and **(ii) high impact:** events that represent situations on moderate or high risk on people's lives and destruction of the city infrastructure. High impact events are related to evacuations, damages, despair, and calls for help, involving a large number of affected people.

To address these approaches, we analyzed different classification problems using machine learning techniques applied in tweets recently (LI *et al.*, 2015; DELANY *et al.*, 2005; CORMACK *et al.*, 2007; CARAGEA *et al.*, 2011; NEPPALLI *et al.*, 2017). Among such techniques, we selected the Naive Bayes, because it assumes a strong independence among the features (i.e., the words that compose a message), and it could be adapted to our context since we are using an independent terminology resulted from the information sources.

4.2.4 Phase 4: Application of KnowSoS

The architecture of the system that will be integrated in EM-SoS to perform knowledge discovery is presented in Figure 31. As proposed by KnowSOS, such architecture is represented in different layers to guide the information flow among constituents, and the knowledge acquisition to support decision making.

Regarding to the collection layer, data are collected from each constituent by the collectors. Such mediators must be scalable to deal with different amounts of data generated by constituents in real time. The data collected is sent to the data bridge module, which must integrate the structured and nonstructured data, and sent it to the storage and discovery layers, responsible for the knowledge extraction. Moreover, depending on the domain and the data collected, the storage layer must implement translators that convert the data to a unified type and facilitate the extraction process.

Then, we used the presented architecture to develop the system responsible for extracting knowledge from the EM-SoS. Several preprocessing techniques were performed such as removal of stop words, i.e., the most common words in a language, besides removing non-English messages, and non-textual elements such as punctuation and figures. Details of each technique were presented in the evaluation proposed in the human resources domain.

To guide the implementation, the system was modeled in a class diagram, describing the controller layer presented in the system inserted into EM-SoS architecture. We selected web service technologies to implement our system, due to its ease of integration and adaptation to EM-SoS architecture. In detail, we use JAVA EE specifications¹³: JAVA-RS to ensure communication protocols and JPA to database persistence.

¹³ <http://www.oracle.com/technetwork/java/javaee/overview/index.html>

Figure 31 – Knowledge Discovery System Architecture proposed for Emergency Management SoS

Source: Elaborated by the author.

4.2.5 Phase 5: Validation of the Knowledge System

In this evaluation, we are addressing two different problem scopes, namely, the occurrence of a disaster event, and the classification of the impact level of a disaster event in a certain region. Hence, both classification problems must be evaluated to verify whether the system is viable and supports the stakeholders of EM-SoS in the knowledge extraction process. To validate the system, data collected in Phase 2, which corresponds to the Hurricane Irma passing on Caribbean region during the period from September 7th to September 15th, 2017, were analyzed.

4.2.5.1 Detection

Regarding the detection of the hurricane, we used the semantic models to classify the 124874 collected messages into two categories: (i) it is an occurrence of a disaster; (ii) it is not an occurrence of a disaster. The goal of such evaluation is to identify whether the detection can generalize a semantic model for each class, and identify if a disaster is occurring with the highest accuracy possible. To analyze such results we calculated the classification accuracy. This metric assesses the overall effectiveness of the classifier by analyzing the probability of the true value of the class label (SOKOLOVA *et al.*, 2006).

Figure 32 presents the distribution of the classified messages in the period analyzed. The x-axis indicates the dates when the messages were posted, the y-axis indicates the number of messages that were classified as “is the occurrence of an event” on the respective day. According to the information sources selected, the major impact of Hurricane Irma happened in the selected region from September 8 to September 11. Based on this information and analyzing the Figure, we can suggest that our approach seems to work well in the task of detect the Hurricane, i.e., the peak of message happened in September 9 and 10. which were the strongest days before the hurricane move away to a different region. In addition, messages related to this disaster increases before the day in which the hurricane hit, indicating the observation alerts, i.e., when people are getting prepared to the disaster. In the days that follow, messages are still generated relating to people who need help to recover from some kind of damage. However, such messages are gradually decreasing as the days go by, with the tendency to disappear soon.

Besides that, we compared the results of such classification with a manual classification, to identify whether the system really learned from the semantic of the messages or if was just guessing the classification. For this, a subset of messages was manually classified into one of the classes already defined based on its semantic. This analysis is important because sometimes words have more than one meaning and looking just at the words which were used may not represent the global idea of the message. An example of such divergence can be the message “*can someone rescue my XBOX? it did not survive the irma :(*”. This user used strong words as “rescue” and “survive” but the global idea does not represent a situation in which a high emergency alert must be issued. As results, we obtained an accuracy of 76%, which indicates that the model is satisfactory at identifying the occurrence of disasters.

4.2.5.2 Classification

To achieve the second goal, which aims at classifying messages according to their impact levels, we used the model to train the Naive Bayes technique in the full dataset. Naive Bayes classified messages in low impact and high impact. Then, we performed a 10-fold cross validation to test the precision and accuracy of such classification, which divides the data set into 10 pieces, and then in each interaction uses nine of these pieces to train the algorithm and the last one to test it (KOHAVI *et al.*, 1995). However, accuracy is a measure that does not achieve high results

Figure 32 – Amount of messages detected through the days.

Source: Research data.

in cases where the classes are unbalanced, which is the case of information related to natural disasters, i.e., it is not expected to have the same number of low and high impact messages. High impact messages should be rarer. Therefore, for this situation, we considered only the *precision* of the technique, which indicates the fraction of the true positive detected that are actually correct for each class (LANDGREBE *et al.*, 2006), i.e., of all the alerts that were labeled as high impact, how many actually were high.

We applied 10-fold cross validation three times. Tables 9, 10, and 11 presents the results for each of the three interactions.

	Low Impact Precision	High Impact Precision	Average Precision	Accuracy
1	0.58	0.56	0.57	0.58
2	0.54	0.44	0.49	0.54
3	0.55	0.44	0.50	0.55
4	0.55	0.78	0.66	0.55
5	0.59	0.44	0.52	0.59
6	0.52	0.67	0.59	0.52
7	0.54	0.56	0.55	0.54
8	0.52	0.67	0.60	0.53
9	0.53	0.44	0.49	0.53
10	0.57	0.89	0.73	0.58
Total Average	0.55	0.59	0.57	0.55

Table 9 – 10-fold cross validation: 1st Execution

The first and second columns present the classification precision for the low impact and

	Low Impact Average	High Impact Average	Prediction Average	Accuracy
1	0.56	0.56	0.56	0.55
2	0.59	0.78	0.68	0.56
3	0.52	0.78	0.65	0.55
4	0.54	0.67	0.60	0.55
5	0.56	0.56	0.56	0.56
6	0.56	0.56	0.56	0.49
7	0.56	0.67	0.61	0.55
8	0.54	0.67	0.60	0.57
9	0.56	0.44	0.50	0.54
10	0.54	0.78	0.66	0.56
Total Average	0.55	0.64	0.60	0.55

Table 10 – 10-fold cross validation: 2nd Execution

	Low Impact Precision	High Impact Precision	Average Precision	Accuracy
1	0.55	0.56	0.55	0.56
2	0.56	0.67	0.61	0.59
3	0.55	0.67	0.61	0.52
4	0.55	0.56	0.55	0.54
5	0.55	0.67	0.61	0.56
6	0.55	0.56	0.55	0.56
7	0.55	0.44	0.50	0.55
8	0.57	1.00	0.78	0.54
9	0.54	0.67	0.60	0.56
10	0.55	0.89	0.72	0.55
Total Average	0.55	0.67	0.61	0.55

Table 11 – 10-fold cross validation: 3rd Execution

high impact classes, respectively. The third column presents the average precision, while the last one presents the classification accuracy. Each row presents a classification alternating nine sub sets of data for training and one for testing (KOHAVI *et al.*, 1995).

In all of these executions, the tweets in data sets were divided differently, then the fact that the precision of the technique tends to the same number may indicate a convergence. Results indicate that the precision of the low impact class was 55%, while the high impact class was 64%. However, the overall accuracy tends to the low impact class because it is the class with more observations, which is expected since high impact messages should be rare (i.e, an emergency event should not happen all the time). Thus, as an overall result, the system classified correctly 55% of the messages.

Moreover, we also investigated the classification of high impact messages, which should

be a critical focus of attention in a real application, because it deals with lives at risk. The distribution of messages is presented in Figure 33.

Figure 33 – Amount of “high impact” messages through the days.

Source: Research data.

These messages also had a peak of growth that reached the maximum on days 9 and 10, which is consistent according to the dates that the hurricane hit the region. In addition, the decrease is more direct because few messages of this level are expected after the hurricane passes through the region. However, due to the number of accidents and situations that occur, few messages are still issued a few days after, which may not necessarily be connected to it. However, the Figure 33 suggests that most high-impact messages were also identified correctly.

4.3 Final Remarks

In this chapter, we presented the main results of two proofs of concept conducted in two different domains, namely, human resources (HR-SoS) and emergency management (EM-SoS). The main objective of this evaluation is to verify whether the use of KnowSoS and the guidelines supported in the design and development of a system that enabled knowledge discovery in SoS and, consequently, supported in the fulfillment of specific missions given by SoS stakeholders. The idea of conducting two different proofs of concept in different domains is to show that this approach is horizontal, that is, it can provide support regardless of which domain is applied.

In general, as we developed a prototype, some of the functionalities and modules defined in the proposal's conceptualization were not used. However, we try to bring the essence of the proposal, developing the collection, integration and discovery modules. In addition, we detail the discussion that maps the system's mission into a machine learning problem, which generates business rules and enables automation of knowledge discovery.

Considering the results, they were quite satisfactory in both domains. In human resources, we have developed a system that recommends job openings more accurately than a random approach, and above that, with more intelligence than just showing the open positions to users. Regarding the emergency management domain, the system demonstrated good precision to generate emergency alerts in moments of natural disasters, facilitating the domain's stakeholders to prioritize places where help is needed.

Thus, we can conclude that the approach is satisfactory to meet the proposed objectives, and can be considered a first proposal consolidated towards the long journey that is the discovery of knowledge in SoS. In this evaluation, we do not consider technical aspects of the approach, but focus on showing that it is really useful to support SoS.

In the next chapter, we will present the main conclusions of this work, including the contributions of this approach to research in software architecture and Systems-of-Systems, the limitations that were not addressed in this project, and therefore, identifying the future works.

Conclusions

Knowledge discovery has been the focus of research in several fields of research in industry and academia (L'HEUREUX *et al.*, 2017; PAPARRIZOS *et al.*, 2011). Nowadays, systems generate an immense amount of data at runtime, and using that data only in an operational manner is a waste of information for any domain. The use of technologies related to artificial intelligence and machine learning has grown steadily (UDDIN *et al.*, 2014), mainly in systems that deal with big data, i.e., a large volume of data in different formats generated at high velocity (UDDIN *et al.*, 2014; HORITA, 2017).

One of the classes of systems that represents this scenario is Systems of Systems. With the increasing interoperability of systems in different domains, it is necessary that this infrastructure also brings knowledge and intelligence to improve its execution. Thus, in this work, we propose KnowSoS, a software architecture to support the design and the development of systems that enable knowledge discovery in SoS.

KnowSoS was built following a process proposed by Hofmeister *et al.* (2007), and is composed of modules that communicate and allow the correct flow of data. In addition, its design has different views that support different aspects of its instantiation. To support the use of KnowSoS, we also propose a set of guidelines that can be used to guide the complete knowledge discovery process, from SoS understanding to the dissemination of knowledge through it.

To validate this proposal, we conducted two proofs of concept in different domains, which generated results that indicate that KnowSoS can support knowledge discovery and bring great benefits to the SoS domains. In the context of human resources, KnowSoS supported the development of a system that recommends job openings according to the profile of users and their preferences. In the field of emergency management, KnowSoS supported the development of a system that emits alarms in the occurrence of natural disasters to the emergence stakeholders.

5.1 Research Contributions

The main contributions of this Master's research are the following:

- **Characterization of the state-of-the-art on Knowledge Discovery in Systems-of-Systems:** we conducted a systematic mapping study to characterize the current scenario of how knowledge discovery has been conducted in Systems-of-Systems. This SMS provided an overview (a map) of the approaches that have been proposed in this context. Particularly, the results showed a lack of research initiatives on the use of knowledge discovery techniques to support SoS missions fulfilment. Our work was a step forward in this direction aiming to bridge this gap.
- **Establishment of an architecture for knowledge discovery in systems-of-systems:** we propose an architecture based on a data flow, which is composed of five main modules that communicate to enable the discovery of knowledge in SoS. To design this architecture, three views were built, detailing the use case view, the process view and the logical view of each of these modules.
- **Establishment of guidelines to support knowledge discovery in systems-of-systems:** to support the use of the architecture, we propose a set of guidelines that guide knowledge discovery in a broader view than system design, including the definition of requirements, identification of system missions, the mapping of SoS missions to data provided by constituents and the dissemination of knowledge through the system.
- **Development of two prototypes:** we evaluated KnowSoS and the guidelines by conducting two proofs of concept in different domains, developing prototypes that could be used to bring benefits to the domains.

5.2 Research Limitations

The main limitations related to the work undertaken in this Master's research are summarized as follows:

- **Technological details:** In this project, we identified modules and interfaces. However, we do not detail the technologies that could have been used to implement each of these modules and how they could help meet the requirements.
- **Experimental studies:** Although we applied the architecture in different domains, we had validations only for the application of the architecture, but few results on the quality of the architecture and the guidelines.

- **KnowSoS application:** Although we developed two prototypes, KnowSoS was not applied in a real-world context. There is a need to apply this architecture in a real context to identify if there is a lack of modules and interfaces in each layer.
- **Parallelism of the missions:** The architecture was developed to fulfill one mission at a time. Competitive techniques and more robust technological infrastructure are necessary for the architecture to be able to accomplish more missions at the same time.

5.3 Future Works

Based on the limitations previously discussed, we can identify several possibilities of continuity of the work undertaken in this Master's research and future directions for research. Following we briefly summarize some of them:

- **Evolution of KnowSoS:** Considering the constant evolution of technologies, it is necessary for KnowSoS to evolve over time. As a first approach, our focus on this project was the architectural structuring and the identification of the main modules. With this base defined, it is important to detail each of these modules in a technological level, indicating how the interfaces and connections should be developed. Besides that, identifying how the data flow can maintain architectural requirements such as scalability, latency and security.
- **Conduction of more evaluations:** In addition to the evaluation with the prototypes developed in this work, it is important to understand the impact of such systems in real scenarios, and opinions of experts regarding technologies that could support the architecture. To bridge this gap, case studies and surveys could be conducted for a qualitative analysis. Furthermore, it is necessary to evaluate the quality aspects of the approach, i.e., quality requirements such as scalability, performance, interoperability and security.
- **Address the dynamic characteristics of SoS:** Identify how the architecture could be more dynamic, adapting itself to real time changes in the environment, and thus including different knowledge discovery techniques to predict the occurrence of behaviors that were not expected in the system.

5.4 Resulting Publications

The main publications resulting from the activities conducted during the Master's research are summarized as follows:

- GARCES, L. ; SENA, BRUNO ; NAKAGAWA, Elisa Yumi . Towards an Architectural Patterns Language for Systems-of-Systems. In: 26th Conference on Pattern Languages of

Programs (PLoP 2019), 2019, Ottawa. 26th Conference on Pattern Languages of Programs (PLoP 2019), 2019. p. 1-24.

- **SENA, B.**; NAKAGAWA, E. Y. . Classifying Social Network Messages to Support Emergency Responders in Natural Disasters. In: Latin American Contest of Final Term Papers (CLATG 2018) @ XLIV Latin American Computing Conference (CLEI 2018), 2018, São Paulo. Proceedings of the Latin American Contest of Final Term Papers (CLATG 2018) @ XLIV Latin American Computing Conference (CLEI 2018), 2018.
- **SENA, B.**; GARCÉS, LINA ; ALLIAN, ANA P. ; NAKAGAWA, E. Y. . Investigating the Applicability of Architectural Patterns in Big Data Systems. In: 25th Conference on Pattern Languages of Programs (PLoP 2018), 2018, Portland. Proceedings of the 25th Conference on Pattern Languages of Programs, 2018.
- **SENA, BRUNO**; ALLIAN, ANA PAULA ; NAKAGAWA, ELISA YUMI . Characterizing big data software architectures. In: the 11th Brazilian Symposium, 2017, Fortaleza. Proceedings of the 11th Brazilian Symposium on Software Components, Architectures, and Reuse - SBCARS '17. New York: ACM Press, 2017. p. 1.

Furthermore, other works indirectly related to this Master's work have been published:

- FIORAVANTI, MARIA LYDIA ; **SENA, BRUNO** ; PASCHOAL, LEO NATAN ; SILVA, LAÍZA R. ; ALLIAN, ANA P. ; NAKAGAWA, ELISA Y. ; SOUZA, SIMONE R.S. ; ISOTANI, SEIJI ; BARBOSA, ELLEN F. . Integrating Project Based Learning and Project Management for Software Engineering Teaching. In: the 49th ACM Technical Symposium, 2018, Baltimore. Proceedings of the 49th ACM Technical Symposium on Computer Science Education - SIGCSE '18, 2018. p. 806-6.
- ALLIAN, ANA ; **SENA, BRUNO** ; NAKAGAWA, ELISA YUMI . Evaluating variability at the software architecture level: An Overview. In: ACM/SIGAPP Symposium On Applied Computing (SAC 19), 2018, Limassol. Proceedings of The 34th ACM/SIGAPP Symposium On Applied Computing, 2018.
- ROCHA, J. ; MELO, H. ; COELHO, R. ; **SENA, B.** . A Catalogue of Java Exception Handling Bad Smells and Refactorings. In: Pattern Languages of Programs (PLoP), 2018, Portland, EUA. 25th Conference on Pattern Languages of Programs (PLoP'18), 2018.

References

- ACHESON, P.; DAGLI, C.; KILICAY-ERGIN, N. Model based systems engineering for system of systems using agent-based modeling. **Procedia Computer Science**, Elsevier, v. 16, p. 11–19, 2013. Citation on page 27.
- ACKERMANN, C.; LINDVALL, M.; CLEAVELAND, R. Towards behavioral reflexion models. In: IEEE. **20th International Symposium on Software Reliability Engineering**. 2009. p. 175–184. Citation on page 37.
- AL-OTAIBI, S. T.; YKHLEF, M. A survey of job recommender systems. **International Journal of Physical Sciences**, Academic Journals, v. 7, n. 29, p. 5127–5142, 2012. Citation on page 85.
- ALLEN, R.; GARLAN, D. A formal basis for architectural connection. **ACM Transactions on Software Engineering and Methodology (TOSEM)**, ACM, v. 6, n. 3, p. 213–249, 1997. Citation on page 36.
- ALOISIO, G.; CAFARO, M.; FIORE, S.; QUARTA, G. A grid-based architecture for earth observation data access. In: **Symposium of Applied Computing (SAC)**. 2005. p. 701–705. Citations on pages 55 and 126.
- AMERICA, P.; ROMMES, E.; OBBINK, H. Multi-view variation modeling for scenario analysis. In: SPRINGER. **International Workshop on Software Product-Family Engineering**. 2003. p. 44–65. Citation on page 60.
- APPEL, S. U.; BOTTLI, D.; JAMISON, J.; PLANT, L.; SHYR, J. Y.; VARSHNEY, L. R. Predictive analytics can facilitate proactive property vacancy policies for cities. **Technological Forecasting and Social Change**, Elsevier, p. 161–173, 2014. Citation on page 126.
- ASTOR, M. **The 2017 Hurricane Season Really Is More Intense Than Normal**. 2017. <<https://www.nytimes.com/2017/09/19/us/hurricanes-irma-harvey-maria.html>>. Accessed: 2017-11-11. Citation on page 97.
- AVGERIOU, P.; ZDUN, U. Architectural patterns revisited-a pattern language. 2005. Citation on page 50.
- AZIZIAN, N.; MAZZUCHI, T.; SARKANI, S.; RICO, D. F. A framework for evaluating technology readiness, system quality, and program performance of us dod acquisitions. **Systems Engineering**, Wiley Online Library, v. 14, n. 4, p. 410–426, 2011. Citation on page 37.
- BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**. : Addison-Wesley Pearson Education, Boston, 2013. Citations on pages 22, 34, and 60.
- BATISTA, T. Challenges for sos architecture description. In: ACM. **Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems**. 2013. p. 35–37. Citation on page 35.
- BATTY, M. Smart cities, big data. In: . : SAGE Publications Sage UK: London, England, 2012. p. 191–193. Citation on page 22.

BAYRAK, A. E.; EGILMEZ, M. M.; KUANG, H.; LI, X.; PARK, J. M.; UMPFENBACH, E.; ANDERSON, E.; GORSICH, D.; HU, J.; PAPALAMBROS, P. Y. *et al.* A system-of-systems approach to the strategic feasibility of modular vehicle fleets. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, IEEE, 2018. Citations on pages 21 and 22.

BEGOLI, E.; HOREY, J. Design principles for effective knowledge discovery from big data. In: **6th Joint Working Conference on Software Architecture and European Conference on Software Architecture, (WICSA/ECSA)**. Helsinki, Finland: IEEE, 2012. p. 215 – 218. Citation on page 43.

BEYER, M. A.; LANEY, D. The importance of ‘big data’: a definition. **Stamford, CT: Gartner**, p. 2014–2018, 2012. Citation on page 22.

BIANCHI, T.; SANTOS, D. S.; FELIZARDO, K. R. Quality attributes of systems-of-systems: a systematic literature review. In: IEEE. **IEEE/ACM 3rd International Workshop on Software Engineering for Systems-of-Systems (SESoS)**. 2015. p. 23–30. Citations on pages 21, 22, 36, 37, 38, and 39.

BOARD, B. E. **The Guide to the Systems Engineering Body of Knowledge (SEBoK) v. 1.9**. 2017. Available: <[http://www.sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_\(SEBoK\)](http://www.sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK))>. Citation on page 35.

BOARDMAN, J.; SAUSER, B. System of systems-the meaning of of. In: IEEE. **IEEE/SMC International Conference on System of Systems Engineering**. 2006. p. 6–pp. Citations on pages 29, 37, and 39.

BOEHM, B.; LANE, J. A. 21st Century Processes for Acquiring 21st Century Software-Intensive Systems of Systems. **Crosstalk: Journal of Defence Software Engineering**, v. 19, n. 5, p. 4–9, 2006. Citation on page 29.

BONDAVALLI, A.; CECCARELLI, A.; LOLLINI, P.; MONTECCHI, L.; MORI, M. System-of-systems to support mobile safety critical applications: Open challenges and viable solutions. **IEEE Systems Journal**, IEEE, 2016. Citation on page 21.

BOSCH, J. Software architecture: The next step. In: SPRINGER. **European Workshop on Software Architecture**. 2004. p. 194–199. Citation on page 27.

BRØNDUM, J.; ZHU, L. Towards an architectural viewpoint for systems of software intensive systems. In: ACM. **Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge**. 2010. p. 60–63. Citation on page 35.

BRYANS, J.; PAYNE, R.; HOLT, J.; PERRY, S. Semi-formal and formal interface specification for system of systems architecture. In: **Systems Conference**. Orlando, FL, USA: , 2013. p. 612–619. Citations on pages 30 and 31.

BURGE, J. E.; CARROLL, J. M.; MCCALL, R.; MISTRIK, I. **Rationale-based software engineering**. : Springer, 2008. Citation on page 35.

BUSCHMANN, F.; HENNEY, K.; SCHIMDT, D. **Pattern-oriented Software Architecture: on patterns and pattern language**. : John wiley & sons, 2007. Citation on page 34.

CABENA, P.; HADJINIAN, P.; STADLER, R.; VERHEES, J.; ZANASI, A. **Discovering data mining: from concept to implementation**. : Prentice-Hall, Inc., 1998. Citation on page 23.

- CARAGEA, C.; MCNEESE, N.; JAISWAL, A.; TRAYLOR, G.; KIM, H.-W.; MITRA, P.; WU, D.; TAPIA, A. H.; GILES, L.; JANSEN, B. J. *et al.* Classifying text messages for the haiti earthquake. In: CITESEER. **International Conference on Information Systems for Crisis Response and Management**. 2011. Citation on page 99.
- CHAFFEY, D.; WHITE, G. **Business information management: improving performance using information systems**. : Pearson Education, 2010. Citation on page 42.
- CHIPRIANOV, V.; FALKNER, K.; GALLON, L.; MUNIER, M. Towards modelling and analysing non-functional properties of systems of systems. In: IEEE. **9th International Conference on System of Systems Engineering (SOSE)**. 2014. p. 289–294. Citation on page 39.
- CHOWDHURY, S. R.; IMRAN, M.; ASGHAR, M. R.; AMER-YAHIA, S.; CASTILLO, C. Tweet4act: Using incident-specific profiles for classifying crisis-related messages. In: **Information Systems for Crisis Response and Management (ISCRAM)**. 2013. Citation on page 97.
- CIOS, K. J.; KURGAN, L. A. Trends in data mining and knowledge discovery. In: **Advanced techniques in knowledge discovery and data mining**. : Springer, 2005. p. 1–26. Citations on pages 23, 43, and 45.
- CLEMENTS, P.; BASS, L.; KAZMAN, R.; ABOWD, G. Predicting software quality by architecture-level evaluation. In: **Proceedings of the Fifth International Conference on Software Quality**. 1995. v. 5, p. 485–497. Citation on page 34.
- CORMACK, G. V.; HIDALGO, J. M. G.; SÁNZ, E. P. Spam filtering for short messages. In: ACM. **6th ACM conference on Conference on information and knowledge management**. 2007. p. 313–320. Citation on page 99.
- COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T.; BLAIR, G. **Distributed Systems: Concepts and Design**. 5th. ed. USA: Addison-Wesley Publishing Company, 2011. Citation on page 30.
- CURRY, D. M.; BEAVER, W. W.; DAGLI, C. H. A system-of-systems approach to improving intelligent predictions and decisions in a time-series environment. In: IEEE. **Conference of Systems-of-Systems Engineering (SoSE)**. 2018. p. 98–105. Citation on page 126.
- CURRY, E. System of systems information interoperability using a linked dataspace. In: IEEE. **System of Systems Engineering (SoSE), 2012 7th International Conference on**. 2012. p. 101–106. Citation on page 126.
- DAHMANN, J.; BALDWIN, K. J.; REBOVICH, G. Systems of systems and net-centric enterprise systems. In: **7th Annual Conference on Systems Engineering Research**. 2009. Citation on page 31.
- DAHMANN, J. S.; JR., G. R.; LANE, J. A. Systems engineering for capabilities. **CrossTalk Journal - The Journal of Defense Software Engineering**, v. 21, n. 11, p. 4–9, November 2008. Citations on pages 30 and 31.
- DAYANANDA, K. R.; GOMES, R.; STRAUB, J. An interconnected architecture for an emergency medical response unmanned aerial system. In: **2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)**. 2017. p. 1–6. Citations on pages 39 and 40.

DELANY, S. J.; CUNNINGHAM, P.; COYLE, L. An assessment of case-based reasoning for spam filtering. **Artificial Intelligence Review**, Springer, v. 24, n. 3, p. 359–378, 2005. Citation on page 99.

DELAURENTIS, D. Understanding transportation as a system-of-systems design problem. In: AIAA RENO, NV. NEW YORK. **43rd AIAA Aerospace Sciences Meeting and Exhibit**. 2005. v. 1. Citation on page 21.

DELAURENTIS, D. System of systems definition and vocabulary. **School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN**, 2007. Citation on page 40.

DEMCHAK, B.; KRÜGER, I. H. Rich feeds for rescue. In: **Information Systems for Crisis Response and Management (ISCRAM)**. 2008. Citations on pages 55 and 126.

DERSIN, P.; TRANSPORT, A. Ieee reliability society. technical committee on systems of systems. 2014. Citation on page 28.

DERSIN PIERRE E TRANSPORT, A. Ieee reliability society. technical committee on systems of systems. 2014. Citation on page 22.

DOMENICONI, G.; MORO, G.; PAGLIARANI, A.; PASINI, K.; PASOLINI, R. Job recommendation from semantic similarity of linkedin users' skills. In: **International Conference on Pattern Recognition Applications and Methods**. 2016. p. 270–277. Citation on page 85.

ERDMAN, J. **2017 Atlantic Hurricane Season Among Top 10 Most Active in History**. 2017. <<https://weather.com/storms/hurricane/news/2017-atlantic-hurricane-season-one-of-busiest-september>>. Accessed: 2017-11-11. Citation on page 97.

FAYYAD, U.; PIATETSKY-SHAPIO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI magazine**, v. 17, n. 3, p. 37, 1996. Citations on pages 42 and 43.

FAYYAD, U.; PIATETSKY-SHAPIO, G.; SMYTH, P. The kdd process for extracting useful knowledge from volumes of data. **Communications of the ACM**, ACM, v. 39, n. 11, p. 27–34, 1996. Citations on pages 22, 23, 43, and 44.

FITZGERALD, J.; BRYANS, J.; PAYNE, R. A formal model-based approach to engineering systems-of-systems. In: SPRINGER. **Working Conference on Virtual Enterprises**. 2012. p. 53–62. Citation on page 27.

GARLAN, D. **Software Architecture**. : Wiley Encyclopedia of Software Engineering, John Wiley & Sons, 2001. Citation on page 34.

GARLAN, D.; ALLEN, R.; OCKERBLOOM, J. Architectural mismatch or why it's hard to build systems out of existing parts. In: IEEE. **17th International Conference on Software Engineering (ICSE 95)**. 1995. p. 179–179. Citation on page 36.

GARLAN, D.; SHAW, M. An introduction to software architecture. In: **Advances in software engineering and knowledge engineering**. : World Scientific, 1993. p. 1–39. Citation on page 34.

GARRO, A.; TUNDIS, A. On the reliability analysis of systems and sos: The ramsas method and related extensions. **IEEE Systems Journal**, IEEE, v. 9, n. 1, p. 232–241, 2015. Citation on page 37.

GOLDSCHMIDT, R.; PASSOS, E. Data mining: um guia prático, conceitos, técnicas, ferramentas, orientações e aplicações. **Rio de Janeiro: Campus**, v. 1, 2005. Citation on page 43.

GONÇALVES, M. B.; CAVALCANTE, E.; BATISTA, T.; OQUENDO, F.; NAKAGAWA, E. Y. Towards a conceptual model for software-intensive system-of-systems. In: IEEE. **IEEE International Conference on Systems, Man and Cybernetics (SMC)**. 2014. p. 1605–1610. Citation on page 33.

GOODCHILD, M. F. Citizens as sensors: the world of volunteered geography. **GeoJournal**, Springer, v. 69, n. 4, p. 211–221, 2007. Citation on page 97.

GUARINIELLO, C.; DELAURENTIS, D. Integrated analysis of functional and developmental interdependencies to quantify and trade-off ilities for system-of-systems design, architecture, and evolution. **Procedia Computer Science**, Elsevier, v. 28, p. 728–735, 2014. Citation on page 39.

HASSAN, M. M.; HUH, E.-N. An efficient grid based metadata processing and sharing architecture for geoss. In: IEEE. **ICACT**. 2008. p. 2071–2075. Citation on page 126.

HATA, Y.; KAMOZAKI, Y.; SAWAYAMA, T.; TANIGUCHI, K.; NAKAJIMA, H. A heart pulse monitoring system by air pressure and ultrasonic sensor systems. In: IEEE. **IEEE International Conference on System of Systems Engineering (SoSE'2007)**. 2007. p. 1–5. Citation on page 22.

HILLIARD, R. Ieee std 1471 2000 recommended practice for architectural description of software-intensive systems. **IEEE**, <http://standards.ieee.org>, v. 12, n. 16-20, p. 2000, 2000. Citation on page 60.

HOFMEISTER, C.; KRUCHTEN, P.; NORD, R. L.; OBBINK, H.; RAN, A.; AMERICA, P. A general model of software architecture design derived from five industrial approaches. **Journal of Systems and Software**, Elsevier, v. 80, n. 1, p. 106–126, 2007. Citations on pages 24, 59, 60, 61, and 107.

HOFMEISTER, C.; NORD, R.; SONI, D. **Applied software architecture**. : Addison-Wesley Professional, 2000. Citation on page 60.

HOLLAND, C. P.; SHAW, D. R.; KAWALEK, P. BP's multi-enterprise asset management system. **Information and Software Technology**, Elsevier, v. 47, n. 15, p. 999–1007, 2005. Citation on page 22.

HONG, W.; ZHENG, S.; WANG, H. Dynamic user profile-based job recommender system. In: IEEE. **2013 8th International Conference on Computer Science & Education**. 2013. p. 1499–1503. Citation on page 85.

HORITA, F. E. A. **An approach for improving decision-making with heterogeneous geospatial big data: an application using spatial decision support systems and volunteered geographic information to disaster management**. Phd Thesis (PhD Thesis) — Universidade de São Paulo, 2017. Citation on page 107.

IDIRI, B.; NAPOLI, A. The automatic identification system of maritime accident risk using rule-based reasoning. In: **Conference on Systems-of-Systems Engineering**. 2012. p. 125–130. Citation on page 126.

INGRAM, C.; PAYNE, R.; FITZGERALD, J. Architectural modelling patterns for systems of systems. In: WILEY ONLINE LIBRARY. **INCOSE International Symposium**. 2015. v. 25, n. 1, p. 1177–1192. Citations on pages [22](#) and [50](#).

ISO. **ISO/IEC/IEEE 42010:2011 Systems and software engineering—architecture description**. 2011. Citation on page [34](#).

ISO/IEC. **ISO/IEC 25010 System and software quality models**. 2010. Citation on page [29](#).

ISO/IEC-24765. Systems and software engineering – vocabulary. **Technical Report, International Organization for Standardization and International Electrotechnical Commission**, 2010. Citations on pages [21](#), [22](#), [60](#), and [65](#).

JAMSHIDI, M. **Systems of systems engineering: principles and applications**. : CRC press, 2008. Citation on page [21](#).

JAMSHIDI, M. **System of systems engineering: innovations for the twenty-first century**. : John Wiley & Sons, 2011. Citation on page [21](#).

KAISLER, S.; ARMOUR, F.; ESPINOSA, J. A.; MONEY, W. Big data: Issues and challenges moving forward. In: IEEE. **46th Hawaii International Conference on System Sciences (HICSS'2013)**. 2013. p. 995–1004. Citation on page [22](#).

KENDIG, C. E. What is proof of concept research and how does it generate epistemic and ethical categories for future scientific practice? **Science and Engineering Ethics**, Springer, v. 22, n. 3, p. 735–753, 2016. Citation on page [83](#).

KENTHAPADI, K.; LE, B.; VENKATARAMAN, G. Personalized job recommendation system at linkedin: Practical challenges and lessons learned. In: ACM. **Proceedings of the Eleventh ACM Conference on Recommender Systems**. 2017. p. 346–347. Citation on page [85](#).

KING, R.; YOUNAN, N.; DATCU, M.; NEDELCU, I. Innovative data mining techniques in support of geoss: A workshop's findings. In: IEEE. **International Conference on Software Testing**. 2011. p. 1–4. Citation on page [126](#).

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007. 57 p. Citations on pages [46](#), [123](#), [124](#), and [125](#).

KOHAVI, R. *et al.* A study of cross-validation and bootstrap for accuracy estimation and model selection. In: STANFORD, CA. **Ijcai**. 1995. v. 14, n. 2, p. 1137–1145. Citations on pages [81](#), [101](#), and [103](#).

KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. Supervised machine learning: A review of classification techniques. **Emerging artificial intelligence applications in computer engineering**, v. 160, p. 3–24, 2007. Citation on page [81](#).

KRUCHTEN, P.; OBBINK, H.; STAFFORD, J. The past, present, and future for software architecture. **IEEE Software**, v. 23, n. 2, p. 22–30, March 2006. ISSN 0740-7459. Citations on pages [27](#) and [35](#).

KRUCHTEN, P. B. The 4+ 1 view model of architecture. **IEEE software**, IEEE, v. 12, n. 6, p. 42–50, 1995. Citation on page [60](#).

KURGAN, L. A.; MUSILEK, P. A survey of knowledge discovery and data mining process models. **The Knowledge Engineering Review**, Cambridge University Press, v. 21, n. 1, p. 1–24, 2006. Citation on page [23](#).

LANDGREBE, T. C.; PACLIK, P.; DUIN, R. P. Precision-recall operating characteristic (p-roc) curves in imprecise environments. In: IEEE. **Pattern Recognition, 2006. ICPR 2006. 18th International Conference on**. 2006. v. 4, p. 123–127. Citation on page [102](#).

LANE, J. A. **What is a System of Systems and Why Should I Care?** 2013. Citations on pages [31](#) and [32](#).

LATONERO, M.; SHKLOVSKI, I. “respectfully yours in safety and service emergency:” management & social media evangelism. 2010. Citation on page [97](#).

LI, B.; HAN, L. Distance weighted cosine similarity measure for text classification. In: SPRINGER. **International Conference on Intelligent Data Engineering and Automated Learning**. 2013. p. 611–618. Citation on page [92](#).

LI, H.; GUEVARA, N.; HERNDON, N.; CARAGEA, D.; NEPPALLI, K.; CARAGEA, C.; SQUICCIARINI, A. C.; TAPIA, A. H. Twitter mining for disaster response: A domain adaptation approach. In: **Information Systems for Crisis Response and Management**. 2015. Citation on page [99](#).

L’HEUREUX, A.; GROLINGER, K.; ELYAMANY, H. F.; CAPRETZ, M. A. Machine learning with big data: Challenges and approaches. **IEEE Access**, IEEE, v. 5, p. 7776–7797, 2017. Citation on page [107](#).

MADNI, A. M.; SIEVERS, M. System of systems integration: key considerations and challenges. **Systems Engineering**, Wiley Online Library, v. 17, n. 3, p. 330–347, 2014. Citation on page [39](#).

MAIER, M. W. Architecting principles for systems-of-systems. **Systems Engineering**, v. 1, n. 4, p. 267–284, 1998. Citations on pages [21](#), [22](#), [28](#), [29](#), [31](#), [32](#), and [89](#).

MAIER, M. W. Architecting principles for systems-of-systems. **Systems Engineering**, John Wiley & Sons, v. 1, n. 4, p. 267–284, 1998. Citation on page [97](#).

MAJD, S.; MARIE-HÉLÈNE, A.; VÉRONIQUE, M.; CLAUDE, M.; DAVID, V. Integration of brainstorming platform in a system of information systems. In: ACM. **MEDES**. 2016. p. 166–173. Citation on page [126](#).

MALINOWSKI, J.; KEIM, T.; WENDT, O.; WEITZEL, T. Matching people and jobs: A bilateral recommendation approach. In: IEEE. **Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS’06)**. 2006. v. 6, p. 137c–137c. Citation on page [85](#).

MARBÁN, Ó.; MARISCAL, G.; SEGOVIA, J. A data mining & knowledge discovery process model. In: **Data Mining and Knowledge Discovery in Real Life Applications**. : InTech, 2009. Citation on page [43](#).

MCDERMOTT, W. J.; ROBINSON, P.; DUNCAVAGE, D. P. Knowledge mining application in ishm testbed. In: **AeroConf**. 2005. p. 1–9. Citations on pages [55](#) and [126](#).

- MEDVIDOVIC, N.; TAYLOR, R. N. A classification and comparison framework for software architecture description languages. **IEEE Transactions on software engineering**, IEEE, v. 26, n. 1, p. 70–93, 2000. Citation on page 34.
- MEDVIDOVIC, N.; TAYLOR, R. N. In: IEEE. **Software architecture: foundations, theory, and practice**: Software engineering, 2010 acm/ieee 32nd international conference on. 2010. v. 2, p. 471–472. Citations on pages 27 and 34.
- MEILICH, A. System of systems (sos) engineering & architecture challenges in a net centric environment. In: IEEE. **IEEE/SMC International Conference on System of Systems Engineering**. 2006. p. 5–pp. Citation on page 39.
- MICHAEL, J.; RIEHLE, R.; SHING, M.-T. The verification and validation of software architecture for systems of systems. In: **Systems-of-Systems Engineering 2009**. 2009. p. 1–6. Citation on page 22.
- MISHRA, S. K.; REDDY, M. A bottom-up approach to job recommendation system. In: ACM. **Proceedings of the Recommender Systems Challenge**. 2016. p. 4. Citation on page 85.
- MITTAL, S.; RAINEY, L. Harnessing emergence: The control and design of emergent behavior in system of systems engineering. In: SOCIETY FOR COMPUTER SIMULATION INTERNATIONAL. **Proceedings of the Conference on Summer Computer Simulation**. 2015. p. 1–10. Citations on pages 21, 23, and 28.
- MORDECAI, Y.; ORHOF, O.; DORI, D. Model-based interoperability engineering in systems-of-systems and civil aviation. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, IEEE, 2016. Citations on pages 21 and 22.
- MOSSGRABER, J.; CHAVES, F.; MIDDLETON, S. E.; ZLATEV, Z.; TAO, R. The seven main challenges of an early warning system architecture. In: **Information Systems for Crisis Response and Management**. 2013. p. 322–331. Citations on pages 53, 55, and 126.
- NAKAGAWA, E. Y.; GONÇALVES, M.; GUESSI, M.; OLIVEIRA, L. B.; OQUENDO, F. The state of the art and future perspectives in systems of systems software architectures. In: ACM. **Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems**. 2013. p. 13–20. Citation on page 35.
- NAKAGAWA, E. Y.; OQUENDO, F.; AVGERIOU, P.; CUESTA, C. E.; DRIRA, K.; MALDONADO, J. C.; ZISMAN, A. Foreword: Towards reference architectures for systems-of-systems. In: **2015 IEEE/ACM 3rd International Workshop on Software Engineering for Systems-of-Systems**. 2015. p. 1–4. Citation on page 22.
- NEPPALLI, V. K.; CARAGEA, C.; SQUICCIARINI, A.; TAPIA, A.; STEHLE, S. Sentiment analysis during hurricane sandy in emergency response. **International Journal of Disaster Risk Reduction**, Elsevier, v. 21, p. 213–222, 2017. Citation on page 99.
- NGUYEN, H. V.; BAI, L. Cosine similarity metric learning for face verification. In: SPRINGER. **Asian conference on computer vision**. 2010. p. 709–720. Citation on page 93.
- NI, J.; CHEN, Y.; SHA, J.; ZHANG, M. Anatomy of functional components of healthcare information system. In: IEEE. **International Conference on Internet Computing for Science and Engineering**. 2015. p. 172–180. Citation on page 126.

- NI, J.; CHEN, Y.; SHA, J.; ZHANG, M. Sose architecture of data intensive computing for healthcare information system. In: **International Conference on Internet Computing for Science and Engineering**. 2015. p. 159–167. Citation on page [126](#).
- NICHOLS, C.; DOVE, R. 7.1. 3 architectural patterns for self-organizing systems-of-systems. In: WILEY ONLINE LIBRARY. **INCOSE International Symposium**. 2011. v. 21, n. 1, p. 856–867. Citation on page [22](#).
- NIELSEN, C. B.; LARSEN, P. G.; FITZGERALD, J.; WOODCOCK, J.; PELESKA, J. Systems of systems engineering: Basic concepts, model-based techniques, and research directions. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 48, n. 2, p. 18:1–18:41, Sep. 2015. Citations on pages [21](#), [22](#), [28](#), [29](#), and [39](#).
- OBBINK, H.; KRUTCHEN, P.; KOZACZYNSKI, W.; HILLARD, R.; RAN, A.; POSTEMA, H.; LUTZ, D.; KAZMAN, R.; TRACZ, W.; KAHANE, E. **Report on Software Architecture Review and Assessment (SARA), Version 1.0**. 2002. <https://pkruchten.files.wordpress.com/2011/09/sarav1.pdf>. Accessed: 2019-02-27. Citation on page [61](#).
- OBERNDORF, P.; SLEDGE, C. Evolution of a software engineer in a sos system engineering world. In: **Annual IEEE Systems Conference**. 2010. p. 91–96. Citations on pages [17](#) and [30](#).
- OLIVEIRA, B. R. d. N. **A quality model for critical embedded systems**. Phd Thesis (PhD Thesis) — Universidade de São Paulo, 2017. Citation on page [43](#).
- OLIVEIRA, R. B. T. d. **O processo de extração de conhecimento de base de dados apoiado por agentes de software**. Phd Thesis (PhD Thesis) — Universidade de São Paulo, 2000. Citation on page [43](#).
- OXFORD, D. **Oxford References Online**. : Oxford: Oxford University Press, 2014. Citation on page [83](#).
- PALEN, L.; ANDERSON, K. M.; MARK, G.; MARTIN, J.; SICKER, D.; PALMER, M.; GRUNWALD, D. A vision for technology-mediated support for public participation & assistance in mass emergencies & disasters. In: BRITISH COMPUTER SOCIETY. **Proceedings of the 2010 ACM-BCS visions of computer science conference**. 2010. p. 8. Citation on page [97](#).
- PAPARRIZOS, I.; CAMBAZOGLU, B. B.; GIONIS, A. Machine learned job recommendation. In: ACM. **Proceedings of the fifth ACM Conference on Recommender Systems**. 2011. p. 325–328. Citation on page [107](#).
- PÉREZ, J.; DÍAZ, J.; GARBAJOSA, J.; YAGÜE, A.; GONZALEZ, E.; LOPEZ-PEREA, M. Large-scale Smart Grids As System of Systems. In: **The First International Workshop on Software Engineering for Systems-of-Systems**. New York, NY, USA: ACM, 2013. p. 38–42. Citations on pages [30](#) and [31](#).
- PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information & Software Technology**, Butterworth-Heinemann, v. 64, n. C, p. 1–18, 2015. Citations on pages [46](#), [123](#), and [125](#).
- RADATZ, J.; GERACI, A.; KATKI, F. **IEEE standard glossary of software engineering terminology**. 1990. v. 610121990, n. 121990, 3 p. Citation on page [28](#).

- RAMOS, J. *et al.* Using tf-idf to determine word relevance in document queries. In: PISCAT-AWAY, NJ. **Proceedings of the first instructional conference on machine learning**. 2003. v. 242, p. 133–142. Citation on page 91.
- RAN, A.; JAZAYERI, M.; LINDEN, F. van der. Ares conceptual framework for software architecture. **Software Architecture for Product Families Principles and Practice**, Addison-Wesley, p. 1–29, 2000. Citation on page 60.
- RATHORE, M. M.; AHMAD, A.; PAUL, A.; RHO, S. Urban planning and building smart cities based on the internet of things using big data analytics. **Computer Networks**, Elsevier, p. 63–80, 2016. Citation on page 126.
- RCP. **Artifact: Architectural Proof-of-Concept**. 2001. Available: <https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/process/artifact/ar_archpoc.htm>. Citation on page 83.
- RHODES, D. H.; ROSS, A. M.; NIGHTINGALE, D. J. Architecting the system of systems enterprise: Enabling constructs and methods from the field of engineering systems. In: **IEEE Systems Conference, 2009 3rd Annual IEEE**. 2009. p. 190–195. Citation on page 27.
- ROTHENBERG, J. Interoperability as a semantic cross-cutting concern. **Interoperabiliteit: Eerlijk zullen we alles delen**, Forum Standaardisatie Den Haag, 2008. Citation on page 39.
- SAGE, A. P. **System of systems engineering: innovations for the 21st century**. : John Wiley & Sons, 2011. Citation on page 35.
- SANTOS, D. S.; OLIVEIRA, B.; GUESSI, M.; OQUENDO, F.; DELAMARO, M.; NAKAGAWA, E. Y. Towards the evaluation of system-of-systems software architectures. In: **Proceedings of Workshop on Distributed Development, Software Ecosystems and Systems of Systems, ser. WDES. SBC, Maceió**. 2014. p. 53–57. Citation on page 37.
- SCHNEIDER, D.; BECKER, M.; TRAPP, M. Approaching runtime trust assurance in open adaptive systems. In: **ACM. Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems**. 2011. p. 196–201. Citation on page 37.
- SCHREIBER, G. **Knowledge engineering and management: the CommonKADS methodology**. : MIT press, 2000. Citation on page 41.
- SENA, B.; ALLIAN, A. P.; NAKAGAWA, E. Y. Characterizing big data software architectures: a systematic mapping study. In: **ACM. Proceedings of the 11th Brazilian Symposium on Software Components, Architectures, and Reuse**. 2017. p. 9. Citations on pages 23, 46, 50, 64, and 78.
- SHAW, M.; CLEMENTS, P. The golden age of software architecture. **IEEE software**, IEEE, v. 23, n. 2, p. 31–39, 2006. Citations on pages 34 and 35.
- SILVA, B. N.; KHAN, M.; HAN, K. Integration of big data analytics embedded smart city architecture with restful web of things for efficient service provision and energy management. **Future Generation Computer Systems**, Elsevier, p. 1–13, 2017. Citation on page 126.
- SILVA, E.; CAVALCANTE, E.; BATISTA, T. Refining missions to architectures in software-intensive systems-of-systems. In: **5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems**. 2017. p. 2–8. Citation on page 30.

- SOKOLOVA, M.; JAPKOWICZ, N.; SZPAKOWICZ, S. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In: **Australian conference on artificial intelligence**. 2006. v. 4304, p. 1015–1021. Citation on page [101](#).
- STARBIRD, K.; PALEN, L.; HUGHES, A. L.; VIEWEG, S. Chatter on the red: what hazards threat reveals about the social life of microblogged information. In: **ACM. Proceedings of the 2010 ACM conference on Computer supported cooperative work**. 2010. p. 241–250. Citation on page [97](#).
- TANNAHILL, B. K.; JAMSHIDI, M. Big data analytic paradigms-from pca to deep learning. In: **AAAI**. 2014. p. 84–90. Citation on page [126](#).
- TAYLOR, R. N. Software architecture: Many faces, many places, yet a central discipline. In: **Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering**. New York, NY, USA: ACM, 2009. (ESEC/FSE '09), p. 303–304. Citation on page [27](#).
- TONGUR, S.; SUNDELIN, H. The electric road system transition from a system to a system-of-systems. In: **2016 Asian Conference on Energy, Power and Transportation Electrification (ACEPT)**. 2016. p. 1–8. Citations on pages [40](#) and [41](#).
- UDDIN, M. F.; GUPTA, N. *et al.* Seven v's of big data understanding big data to extract value. In: IEEE. **American Society for Engineering Education (ASEE Zone 1), 2014 Zone 1 Conference of the**. 2014. p. 1–5. Citations on pages [22](#) and [107](#).
- VERÍSSIMO, P. E. Travelling through wormholes: A new look at distributed systems models. **SIGACT News**, ACM, New York, NY, USA, v. 37, n. 1, p. 66–81, Mar. 2006. ISSN 0163-5700. Citation on page [21](#).
- WAGNER, S. **Software product quality control**. : Springer, 2013. Citation on page [37](#).
- WALDEN, D. D.; ROEDLER, G. J.; FORSBERG, K.; HAMELIN, R. D.; SHORTELL, T. M. **Systems engineering handbook: A guide for system life cycle processes and activities**. : John Wiley & Sons, 2015. Citation on page [37](#).
- WALLER, A.; CRADDOCK, R. Managing runtime re-engineering of a system-of-systems for cyber security. In: IEEE. **6th International Conference on System of Systems Engineering (SoSE)**. 2011. p. 13–18. Citation on page [39](#).
- WILLETT, P. The porter stemming algorithm: then and now. **Program Emerald Group Publishing Limited**, v. 40, n. 3, p. 219–223, 2006. Citation on page [91](#).
- WIRTH, R.; HIPPE, J. Crisp-dm: Towards a standard process model for data mining. In: CITE-SEER. **Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining**. 2000. p. 29–39. Citations on pages [23](#) and [45](#).
- WOJCIK, L. A.; HOFFMAN, K. C. Systems of systems engineering in the enterprise context: a unifying framework for dynamics. In: IEEE. **IEEE/SMC International Conference on System of Systems Engineering**. 2006. p. 8–pp. Citation on page [21](#).
- WONG, R. K.; CHI, C. H.; YU, Z.; ZHAO, Y. A system of systems service design for social media analytics. In: **International Conference on Services Computing**. 2014. p. 789–796. Citation on page [126](#).

YANG, S.; KORAYEM, M.; ALJADDA, K.; GRAINGER, T.; NATARAJAN, S. Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach. **Knowledge-Based Systems**, Elsevier, v. 136, p. 37–45, 2017. Citation on page [85](#).

ZAREIE, A.; SHEIKHAHMADI, A. A hierarchical approach for influential node ranking in complex social networks. **Expert Systems with Applications**, Elsevier, 2017. Citation on page [97](#).

ZHENG, Y.; LIU, F.; HSIEH, H.-P. U-air: When urban air quality inference meets big data. In: ACM. **Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining**. 2013. p. 1436–1444. Citation on page [22](#).

ZINS, C. Conceptual approaches for defining data, information, and knowledge. **Journal of the Association for Information Science and Technology**, Wiley Online Library, v. 58, n. 4, p. 479–493, 2007. Citation on page [41](#).

ZOPPI, T.; CECCARELLI, A.; BONDAVALLI, A. Exploring anomaly detection in systems of systems. In: **The Symposium on Applied Computing**. New York, NY, USA: ACM, 2017. (SAC '17), p. 1139–1146. Citations on pages [21](#), [22](#), and [28](#).

Knowledge Discovery in Systems-of-Systems: A Systematic Mapping Protocol

In this appendix, we discuss the planning and execution of the SMS. The research method adopted in this study follows the guidelines proposed by [Kitchenham and Charters \(2007\)](#) and [Petersen et al. \(2015\)](#). In short, this process presents three main phases: (i) planning: in this phase, the research objectives and the systematic review protocol are defined. The protocol constitutes a pre-determined plan that describes research questions and how the SMS will be conducted; (ii) conduction: during this phase, primary studies are identified, selected, and evaluated according to the inclusion and exclusion criteria. For each selected study, data are extracted and synthesized; and (iii) reporting: in this phase, a final report is organized and presented. As the results of this SMS were already reported in Chapter 3, this appendix describes the first and second phases in more details.

A.1 Planning

To conduct our SMS, we followed well-known guidelines ([PETERSEN et al., 2015](#); [KITCHENHAM; CHARTERS, 2007](#)), that encompass the following elements: (i) research objectives and questions; (ii) search strategy; (iii) selection criteria; (iv) selection of studies; and (v) analysis of results.

A.1.1 Research Objectives and Research Questions

The goal of this SMS is to investigate how the knowledge contained in the data distributed in the SoS constituents has been used to discover the overall knowledge for the SoS. To achieve this research objectives, three research questions (RQs) were addressed: **RQ1:** *Which approaches have been proposed to collect data from SoS constituents?*; **RQ2:** *Which approaches have been proposed to integrate data from SoS constituents?*; and **RQ3:** *Which approaches have been*

proposed to extract knowledge from SoS constituents?.

A.1.2 Search Strategy

We firstly defined a search string including terms from the research objective and from our RQs and thus focusing on terms related to SoS and KD and their main synonyms. Hence, our search string is: (“*systems of systems*” OR *systems-of-systems*) AND (“*knowledge discovery*” OR “*knowledge engineering*” OR “*knowledge acquisition*” OR “*knowledge extraction*” OR “*big data*” OR “*data mining*”). To perform the search of studies, we selected electronic search engines and digital libraries also recommended for the software engineering area (KITCHENHAM; CHARTERS, 2007): ACM Digital Library, EI Compendex, IEEE Digital Library, ISI Web of Science, Science@Direct, Scopus, and Springer Link.

A.1.3 Selection Criteria

We established selection criteria (i.e., inclusion criteria (IC) and exclusion criteria (EC)) to assess each study: [IC1] The study addresses collection or integration of data, or extraction of knowledge from the SoS constituents; and [IC2] The study addresses decisions-making from knowledge of SoS. Besides that, the EC are: [EC1] The study is not related to SoS; [EC2] The study does not address collecting or integrating data, extracting knowledge, or supporting decision-making from the SoS constituents; [EC3] The study is an extended abstract, table of contents, foreword, tutorial, editorial, or summary of conference; [EC4] The study is a previous version of a more complete study about the same research; and [EC5] The study does not have an abstract or the full text is not available.

A.1.4 Selection of Studies

Conducted from April to November 2018, our SMS identified, selected, and evaluated primary studies using the search strategy and selection criteria. For each selected study, data were extracted and synthesized¹, as shown in Figure 34.

- **Recovery from Databases:** The search string was applied to the selected databases and 407 studies were found (in which 147 were duplicated), resulting in 260 unique studies;
- **First Selection:** We read the title, abstract, and keywords of each one of the 260 studies; also we read the introduction and the conclusion subsections when the abstract did not provide us enough information. Applying the selection criteria, we selected 39 potentially relevant studies;
- **Second Selection:** We performed a full reading of the 39 studies and again applied the selection criteria. As a result, 17 studies were included (listed in Table 12); and

¹ We used Parsifal (<https://parsif.al/>), a reference management tool that stores information on the primary studies (e.g., title, authors, venue, and abstract), as well as sets selection criteria applied in each primary study.

Figure 34 – Search conduction results.

- **Data Extraction:** To extract data from such studies, we used a form and collected the study title, authors, bibliographic information, details on the approach proposed (technologies, architectural patterns, architectural requirement, and so on), reason to extract the knowledge, and type of SoS (i.e., directed, acknowledge, collaborative, and virtual).

A.2 Threats to Validity

To increase the trustworthiness of our study and minimize biases, we identified potential threats to validity and discuss below the actions we put in place to mitigate them.

- **Missing of relevant primary studies:** A threat to the validity is that important studies were not found through the systematic search in databases. To minimize this threat, the result of the searches was validated against a control group of studies, which were initially recommended by an expert in the area of SoS and KD. All studies in this group were included in the result of our search; hence, we believe the most relevant studies are included in our SMS. Furthermore, we systematically followed our predefined protocol, as recommended in (KITCHENHAM; CHARTERS, 2007) (PETERSEN *et al.*, 2015), to avoid bias in the study search. As stated before, we considered the seven most important digital databases in the software engineering area aiming to find all studies available about SoS and KD.
- **Study selection:** The studies selection process may have been influenced by the knowledge and opinion of the researchers involved in this SMS. To mitigate this threat, each study was independently evaluated by the first two authors. At the end, results of both researchers were compared and divergences were discussed. To ensure the reliability of our SMS and obtain

Table 12 – Final list of primary studies

ID	Title	Author	Year
S1	A Grid-Based Architecture for Earth Observation Data Access	Aloisio <i>et al.</i> (2005)	2005
S2	Knowledge Mining Application in ISHM Testbed	McDermott <i>et al.</i> (2005)	2005
S3	Rich Feeds for RESCUE	Demchak and Krüger (2008)	2008
S4	An Efficient Grid Based Metadata Processing And Sharing Architecture For GEOSS	Hassan and Huh (2008)	2008
S5	Innovative Data Mining Techniques in Support of GEOSS	<i>et al.</i> King <i>et al.</i> (2011)	2011
S6	System of Systems Information Interoperability using a Linked Dataspace	Curry (2012)	2012
S7	Predictive analytics can facilitate proactive property vacancy policies for cities	Appel <i>et al.</i> (2014)	2013
S8	The automatic identification system of maritime accident risk using rule-based reasoning	Idiri and Napoli (2012)	2013
S9	The Seven Main Challenges of an Early Warning System Architecture	Moßgraber <i>et al.</i> (2013)	2013
S10	Big Data Analytic Paradigms -From PCA to Deep Learning	Tannahill and Jamshidi (2014)	2014
S11	A System of Systems Service Design for Social Media Analytics	Wong <i>et al.</i> (2014)	2014
S12	SoSE Architecture of Data Intensive Computing for Healthcare Information System	Ni <i>et al.</i> (2015b)	2015
S13	Anatomy of Functional Components of Healthcare Information System	Ni <i>et al.</i> (2015a)	2015
S14	Urban planning and building smart cities based on the Internet of Things using Big Data analytics	Rathore <i>et al.</i> (2016)	2016
S15	Integration of Brainstorming Platform in a System of Information Systems	Majd <i>et al.</i> (2016)	2016
S16	Integration of Big Data analytics embedded smart city architecture with RESTful web of things for efficient service provision and energy management	Silva <i>et al.</i> (2017a)	2017
S17	A System-of-Systems Approach to Improving Intelligent Predictions and Decisions in a Time-series Environment	Curry <i>et al.</i> (2018)	2018

an unbiased selection process, our SMS planning, including the research question and the selection criteria, was carefully set up in advance and was accurately followed.

- **Data Extraction:** During data extraction, we created a data sheet to fill and save all answers from each study. Since not all information was explicitly available in some studies, some data had to be interpreted (e.g., the use of an architectural pattern is not explicit in the text, but it is

presented in the architectural design). To minimize this threat, discussions among the authors were conducted.