Representing Interoperability Between Software Systems by Using Pre-Conceptual Schemas

Diana M. Torres Ricaurte[1], Mónica K. Villavicencio Cabezas[2], Carlos M. Zapata Jaramillo[3]

[1.3]Computer and Decision Science Department, Universidad Nacional de Colombia, Medellín, Colombia [2]Faculty of Electrical and Computer Engineering, Escuela Superior Politécnica del Litoral, ESPOL, Guayaquil, Ecuador
dimtorresri@unal.edu.co, mvillavi@espol.edu.ec, cmzapata@unal.edu.co

Abstract: Interoperability is a software property for exchanging and using information between software systems. Some interoperability proposals are intended to use metamodels, models, and frameworks, in which visual representations are included. However, such proposals describe interoperability according to their point of view, focus of analysis, solution techniques, and specific level of interoperability; in addition, some of the visual representations use no standards, which lead to subjective interpretations by the reader. In this paper we propose a representation formalizing interoperability which includes essential elements and structural relationships among them. After a review of previous work, a summary of the analyzed literature is performed by using linguistic analysis in order to recognize the relationships among interoperability essential elements and mapping them to controlled language expressions. Finally, the mapping process is used to represent the interoperability essential elements and their relationships by using pre-conceptual schemas linked to the definition of each essential element. As a result, a pre-conceptual schema of the software system interoperability is proposed. Such a pre-conceptual schema is also used for explaining an interoperability lab study adapted from the literature. The proposed pre-conceptual schema explains the interoperability between two software systems and allows for characterizing interoperability problems and solutions.

Keywords: Interoperability, Formal representation, Interoperability essential elements, Pre-conceptual schema.

1. Introduction

Interoperability is defined as a software system property and a communication situation happening between two software systems [1], [2], [3], [4]. In general, interoperability is related to the cooperation among systems for exchanging meaningful information [1], [5]. Therefore, the interoperability implies the ability for exchanging data (technical and syntactic interoperability) and for understanding and interpreting information (semantic and organizational interoperability).

Interoperability metamodels, models, and frameworks are proposed for addressing aspects, levels, and issues of interoperability [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26],[27], [28], [29]. Some visual representations of interoperability are developed in such proposals in order to simplify, re-express, and summarize the understanding of interoperability. Interoperability representations can be a source of comparison and discussion about interoperability elements, relationships, and requirements. In addition, visual representations are useful to recognize and perceive data related to interoperability issues [30].

Interoperability is described in previous work according to different points of view—*i.e.*, federated, unified, integrated—, their focus of analysis—*e.g.*, model based-system engineering, systemic view, property value statement model, SOA, etc.—, and their techniques of solution—*e.g.*, middleware, software service, metamodel, standardization, and templates, etc.—[4], [6], [7], [8], [9], [10], [11], [12], [13], [25], [29]. The interoperability is analyzed in a

Received: January 26th, 2021. Accepted: March 15th, 2022
DOI: 10.15676/ijeei.2022.14.1.7

general way in previous work. Some problems are related to lack of a unified semantic interoperability theory [23], lack of identification of essential elements for describing interoperability [19], [31], [37], use of a disunified terminology about interoperability by the authors [21], [32], partial view of interoperability [23], [24], and lack of a definition of concepts, relations, and properties for describing enterprise architecture [13]. For this reason, interoperability elements seem to be different in each proposal; hence, interoperability phenomenon seems to depend on a specific technique rather than to follow an invariant behavior. Consequently, a variety of concepts can emerge from the interoperability visual representations, some of them developed outside standardized semantics and syntax. Therefore, the correctness of the representation is compromised and its interpretation becomes subjective [30]. For all previous reasons, the interoperability proposals remain difficult for comparing, evaluating, and measuring.

In this paper we analyze interoperability as a phenomenon of the software systems, reaching a high level of abstraction for embracing some points of view, focuses, techniques, and levels of interoperability. As a result of such analysis, a visual representation of interoperability is proposed by using pre-conceptual schemas—*i.e.*, graphical models for representing structural and dynamic views of a given domain. In the pre-conceptual schemas, interoperability essential elements are represented as concepts and their associations are represented by using structural relationships—*i.e.*, "IS", "HAS". Seven essential elements of interoperability—*i.e.*, source software system, target software system, information, language, symbol, interface, and context—are identified by using a systematic literature review and a terminology unification process [31], [32].

Our schema is a view of interoperability between software systems integrating seven essential elements. Such a schema represents how interoperability happens when the essential elements interact, regardless the way to be implemented. In addition, different levels of interoperability—*i.e.*, syntactic, organizational, semantic, and technical—are included. Our pre-conceptual schema about interoperability provides a generic way to describe interoperability proposals, allowing comparisons of existing proposals based on the interoperability essential elements.

We instantiate each concept in the pre-conceptual schema by using an example case reported in the literature for the sake of exemplification and characterization of the essential elements. Characterization is necessary to identify interoperability problems and compare different interoperability solutions and proposals. The automation of our model is out of the scope of this paper, but it is a possible future work, which can be useful for predicting processes and making decisions regarding software interoperability.

The remainder of this paper is organized as follows: in Section II we present the theoretical framework; Section III contains related work about the visual representation of interoperability; in Section IV we explain our pre-conceptual schema about interoperability; in Section V we propose a lab study; in Section VI we discuss the results, and finally, in Section VI we conclude the paper and propose the expected future work.

2. Theoretical Framework

In this Section, we present the fundamentals of interoperability, the essential elements of interoperability, the definition of some commonly used visual representations to describe and formalize interoperability, and the pre-conceptual schemas for graphically representing a domain.

A. Interoperability Fundamentals

In a software quality model [1], interoperability is defined as an inherent property of the software systems characterized by name, type, level, methods, and measures. Specifically, interoperability is related to the degree in which systems cooperatively operate [1] or exchange meaningful information (in a context) via an interface [5]. In other definitions, interoperability is defined as ability, *e.g.*, ability of systems for exchanging information in heterogeneous

platforms [2]; ability of enterprises/entities for mutual communication and interaction [3]; ability of the manufacturing software systems for sharing and exchanging information with a functional objective by using a common interface [33]. In a theoretical point of view, interoperability is a paradigm where incompatible systems are put in relation [4] or the satisfaction of a communication needed among actors [10].
Such definitions include a variety of elements in interoperability. Some of them are applied in a specific scenario of use (*e.g.*, middleware, ontology, and services, among others); others seem to be enduring notions about interoperability points of view (*e.g.*, systems, information, and interface, among others). Also, some concepts are missing in the literature about interoperability. For these reasons, the comparison of interoperability proposals is difficult, since they avoid commonalities, even though they refer to the same topic.

B. Interoperability Essential Elements

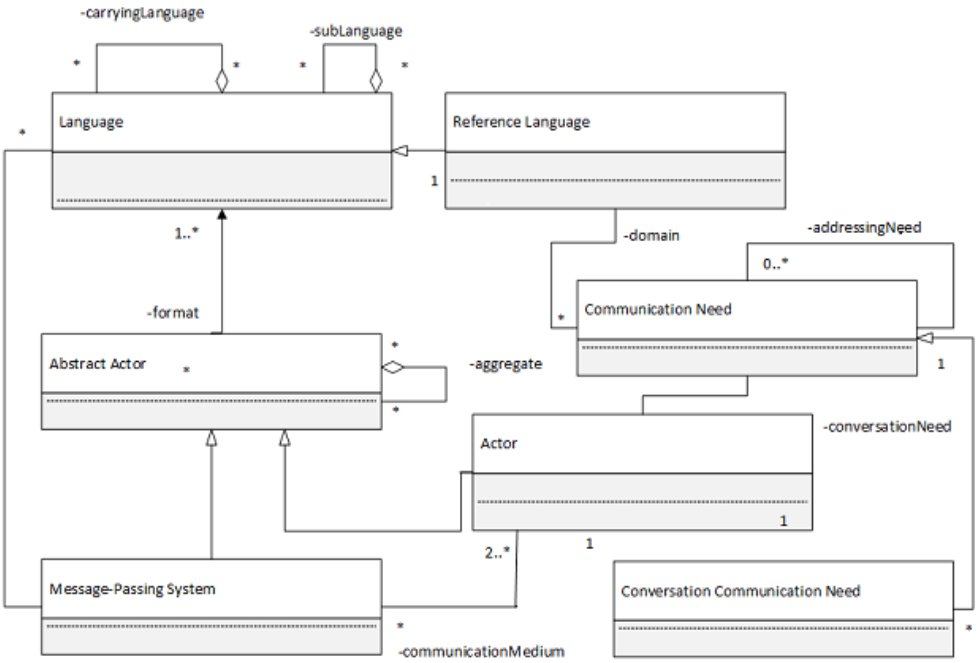
In previous work [32], a systematic literature review and a unification process are performed for identifying seven interoperability essential elements—*i.e.*, source software system, target software system, information, interface, language, context, and symbol—which resemble the communication process elements—*i.e.*, transmitter, receiver, message, channel, code, context, and signal. The essential elements and their definitions are presented in Tab. 1.

Table 1. Interoperability essential elements [32]

Concept	Definition
Software System (referred to source and target software system)	"A system solely consisting of software and possibly the computer equipment on which the software operates." "A system made up of software, hardware, and data that provides its primary value by the execution of the software."
Information	"The meaning assigned to data by known conventions. The meaning that humans assign to data by means of known conventions that are applied to the data."
Symbol	"A representation of something by reason of relationship, association, or convention."
Language	"A system consisting of: 1) a well-defined, usually finite, set of characters 2) rules for combining characters with one another to form words or other expressions 3) a specific assignment of meaning to some of the words or expressions, usually for communicating information or data among a group of people, machines, etc."
Context	"Any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."
Interface	"(i) A common boundary between a considered system and another system, or between parts of a system, through which information is conveyed. (ii) A hardware or software component that connects two or more other components for the purpose of passing information from one to the other."

C. Visual Representations of Interoperability

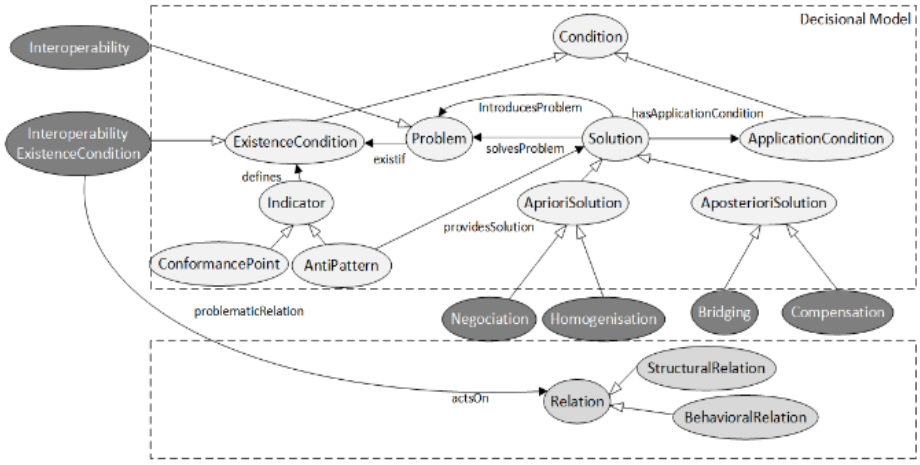
Knowledge representation is a discipline used for analyzing, classifying, and reasoning about existing knowledge from a domain [44]. Such a discipline is complemented with constructors in other fields of knowledge like logic, ontology, and computing. In addition, some formalisms are aimed to represent knowledge, including object-oriented representation, decision trees and tables, and description logics, among others.



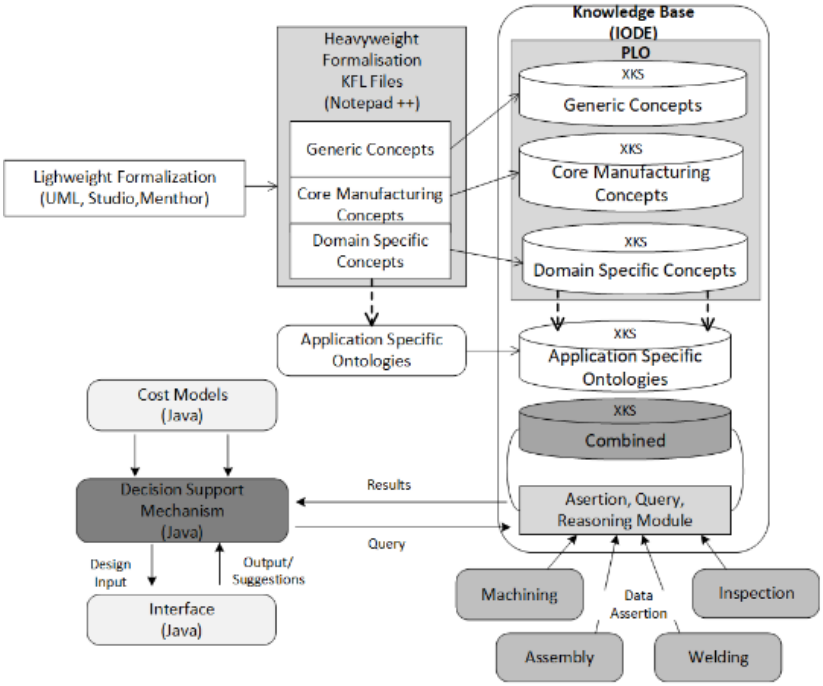
a. Class diagram

Characteristic·Group¤	Description·(#·characteristics)¤
Conformance·with¶ requirements¤	Related·to·the·conformity·with·the·requirements,·both· systems·and·organizations.¤
Share·common¶ concepts¤	Understanding·related·to·concepts,·terms,·processes,·and· policies·relevant·to·the·interacting·systems.¤
Architecture¶ definition¤	Architecture·approach·and·style,·platforms·and·other· properties.¤
Use·of·services·¤	Concerns·like·service·discovery·are·specific·to·the·use·of· services·as·a·solution·for·composition.¤
Protocols¶ definition¤	Select·the·suitable·protocol·for·each·interaction·level.¤

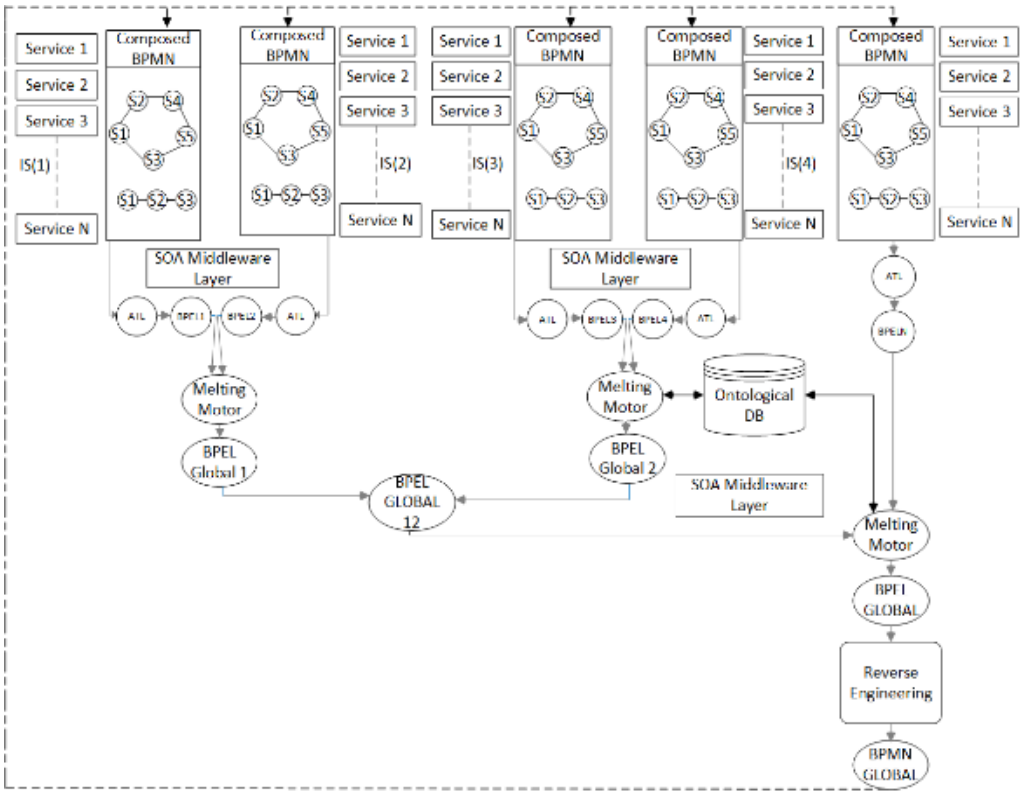
b. Tables



c. Conceptual model



d. Block diagram



e. Architectural model

Figure 1. Visual representations of interoperability (Based on [4], [6], [7], [9], and [10])

Also, visual representations are considered useful for understanding and perceiving data [30]. Some of the visual representations used in formalizations of interoperability include:

- Class diagram for describing the static view of an application focused on the class specification [34], [35]. The main elements of a class diagram are classes, their attributes, operations, and relationships (see Figure 1a).
- Tables for presenting information in an organized and summarized way [36]; they should be self-contained and easy to understand (see Figure 1b).
- Conceptual models for graphically representing a domain (see Figure 1 c). Conceptual models can be used to describe in a precise and complete way the knowledge about some domain [37], [38]. Conceptual models can take the shape of ontologies related to taxonomies, typologies, and component systems [39].
- Block diagrams for comprising a set of blocks and their connections based on data flow notion [40]. Such diagrams are frequently used in the electronic area for representing logical/functional relationship among system components [41] (see Figure 1d).
- Architectural model for representing different structures of a system, called views which embrace some kinds of models—*e.g.*, component diagrams, deployment diagram, and client-service models, among others—[5] (See Figure 1e).

D. Pre-conceptual Schemas

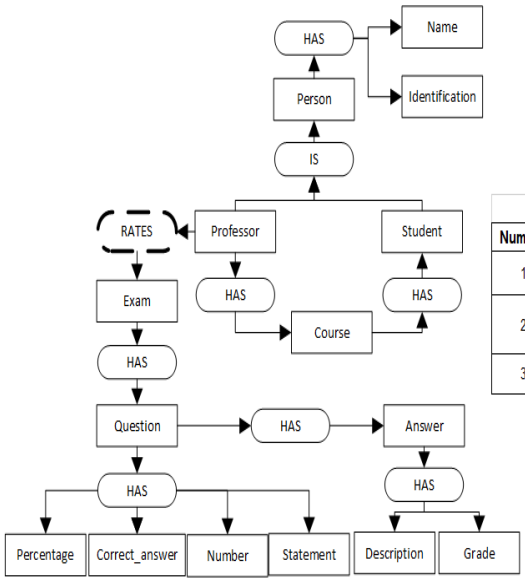
Pre-conceptual schemas (PCS) are graphical models used for representing knowledge and summarizing the main characteristics of a problem domain [42]. PCS involve structural and dynamic elements of a domain by using a language close to natural language. The structural view involves concepts and their relationships, whereas the dynamic view involves operations happening into the system. The main elements of a pre-conceptual schema are concepts, connections, structural relationships, and dynamic relationships, among others. These elements are described in tabel 2.

Table 2. Pre-conceptual schema elements (Based on [43])

Element Name	Description	Graphical representation
Concept	Represents nouns and noun phrases. Concept-class representing real-world things and leaf concepts representing attributes of class-concepts are included in this symbol	CONCEPT
Structural relationship	Represents the verbs "to be" and "to have" for producing permanent relationships among concepts (IS and HAS)	STRUCTURAL RELATIONSHIP
Dynamic relationship	Represents verbs of activity or operation	DYNAMIC RELATIONSHIP
Connection	Represents links between concepts	CONNECTION
Annotation	Represents possible values of concepts	ANNOTATIONS
Frame	Represent a set of elements of the pre-conceptual schema	FRAME

An example of a graphical representation of a domain by using PCS is presented in Figure. 2. The concepts professor and person are connected by using a structural relationship is—denoting an inheritance relationship. The concept person (concept-class) relates to name and identification by using structural relationship has—denoting attributes of a person or leaf concepts. The concepts professor and exam are connected by using a dynamic relationship rates—denoting a temporal action executed by a professor. The direction of the connections

indicates the right way for reading the schema. In addition, a table associated with the class concept question is presented where some concepts are instantiated to test the functionality of the represented domain. Exemplifying the schema is a practical way to clarify the correspondence between the represented concepts and the problem domain—the common ground for communication and discussion on a domain of interest.



Question

Number	Percentage	Statement	Correct_answer	Description	Grade
1	20	¿What UML does mean?	Unified Modeling Language	A logical model	0
2	25	How a dynamic relationship is represented?	A dynamic relationship is represented with an action verb	Action verb	25
3	20	What NAT does mean?	Network address translation	Network address translation	25

Figure 2. Example of a pre-conceptual schema (Adapted from [43])

3. Related Work

Conceptual and visual representations are used to express knowledge about interoperability. In general, previous interoperability work refers to perspectives and levels of interoperability, different visual representations, proposed usability, and software system states (see Tab. 3). In such representations a variety of concepts and elements emerge to explain what interoperability is, how it works, and what relationships exist between the interoperability elements. Despite such effort, some identified problems persist: lack of an agreed theory of interoperability [23], lack of a unified vocabulary [21], [32], variety of essential elements of interoperability [31], [14], [19], and lack of an overview about interoperability [23], [24], among others.

In the case of interoperability perspectives, proposals include some of the three points of view about interoperability—*i.e.*, unified, integrated, and federated—(see the column point of view in Tab.3). In addition, some of such proposals have different focus for analyzing the software interoperability such as Model Driven Engineering MDE, Model Driven Architecture MDA, Model Driven Service Engineering MDSE, and Service Oriented Architecture SOA, among others [6], [12] (see the column focus in Tab. 3). Such proposals are intended to face the heterogeneity of the business logic, business process models, the implemented enterprise software systems or platform providers, the enterprise policies, vocabulary, and terminology of the domains, etc. [4], [11], [12], [13], [33], [25], [29]. Finally, the proposals involve some techniques for solving interoperability issues: middleware layer, mediators, BPMN languages, formal ontologies, templates, etc. (see the column technique in Tab. 3).

Four levels of interoperability (organizational, technical, semantic, and syntactic) are related to the proposals in Tab. 3. Organizational interoperability is represented by modeling the integration of physical resources—*e.g.*, servers, databases, etc.—, information technology resources—*e.g.*, enterprise software, modeling languages, simulation strategies, etc.—, and human resources for proving services among companies [12], [13], [25], [29]. Technical interoperability is represented for showing the interaction and connectivity among diverse

physical resources—sensors, computer, tablets, etc.—and modeling software architecture [9], [13], [21], [26]. The semantic interoperability is represented for achieving the understanding of the domains involved in an interoperability relationship [7], [8], [12], [21], [22], [33], [25], [27], [28]. Syntactic interoperability is commonly represented inside the other three levels as an intermediate process—*e.g.*, data processing, categorization, and standardization, etc.

Visual representations (see the columns 7–10 in Tab. 3) frequently show a structural view, leaving aside the interactions of the interoperating systems. Other processes—taking, transforming, translating, and interpreting the information to achieve the desired interoperability—are left aside. Furthermore, lack of exemplification of some proposals keeps them at a conceptual level, making their application difficult. For example, a table with a characterization of interoperability in context-aware software system CASS is presented [9]. However, the relationships among interoperability features are misidentified and interoperability is unexplained. Also, they lack examples of the suggested visual representation.

Something similar happens where the main interoperability concepts are presented from the point of view of general systems theory [4]. An explanation of how interoperability occurs is missing. Even though an example of the model usage is provided, it is complex and difficult to understand.

The use of the proposed usability (see the column proposal usability in Tab. 3) shows a trend towards the domain knowledge and the interoperability understanding. However, the interoperability understanding and the terminology unification are limited to make an analysis of each proposal. For example, in standard 11354 [11] enterprise interoperability is explained in terms of four dimensions—*i.e.*, business, process, service, and data. Some of such concepts are untraceable into other interoperability formalizations. In contrast, interoperability is defined as “the satisfaction of the need for communication between two or more actors” [10]; then, interoperability is presented by using particular concepts such as message passing systems, actor-software system and constructions-elements.

Finally, the essential elements of interoperability are outlined in few proposals. For this reason, the phenomenon of interoperability lacks a description with a unified terminology. Therefore, the characterization of interoperability is partially achieved and a comparison between the different proposals—even referring to the same point of view and technique of interoperability—is difficult.

Something similar happens with the interoperability assessment. The evaluation is carried out by using criteria associated with a large number of dimensions, features, profiles, templates, etc. [9], [21], [22], [24], [33].

Also, most of the proposals are useful to identify/correct interoperability problems when software systems are in operation (see the column state of software system in Tab. 3), while others are intended to predict and design interoperability in early stages of software system development.

Table 3. Findings of the literature review about interoperability

Study		Interoperability level				Visual representation				Proposed usability							Interoperability perspective			Software system
Authors	Proposal contribution	Semantic	Syntactic	Technical	Organizational	Visual representation type	Exemplification	Structural view	Dynamic view	Interoperability evaluation	Terminology unification	Essential domain knowledge	Interoperation	Simulation	Interoperability understanding		Point of view	Focus	Technique	State of software system
Amr *et al.*, 2017 [6]	Model				X	Architectural model			X					X			Integrated	MDA, SOA service-oriented architecture, Intermediate	Middleware	Operational
Bazoun *et al.*, 2016 [12]	Common framework				X	Block diagram		X					X		X		Integrated	MDSEA	Simulation	Operational
Saha *et al.*, 2017 [7]	Common framework	X				Block diagram		X					X				Integrated	Data assertion	Formal ontology	During all states
Epple *et al.*, 2017 [8]	Model	X				Class diagram		X					X				Unified	Property value statement model	Standardization	Operational
Motta *et al.*, 2016 [9]	Systematic literature review					Table		X			X				X		N/A	CASS	Interoperability characterization	Operational
Ulberg *et al.*, 2012 [10]	Metamodel			X		Class diagram		X		X					X		Unified	Architecture models	N/A	Architecture selected
Naudet *et al.*, 2010 [4]	Metamodel				X	Conceptual model	X	X				X			X		Integrated, unified, federated	General system theory	Negotiation, homogenization, bridging, compensation	Operational
ISO 11354, 2011 [11]	Standard				X	Conceptual model	X	X		X	X				X		Integrated, unified, federated	Manufacturing companies	N/A	During all states
Chen *et al.*, 2008 [13]	Systematic literature review				X	Table		X			X				X		N/A	Architectures for enterprises	N/A	Architecture selected
Serapião *et al.*, 2019 [20]	Metamodel				X	Class diagram	X	X		X	X				X		Integrated, unified,	Model-based system	N/A	Operational

Study		Interoperability level				Visual representation				Proposed usability							Interoperability perspective			Software system
																	federated	engineering		
Motta *et al.*, 2019 [21]	Systematic literature review			X		Conceptual model	X	X				X	X				N/A	CASS	N/A	During all states
Liu *et al.*, 2020 [22]	Systematic literature review				X	Table		X		X			X				N/A	Business processes	N//A	Operational
ISO 16100, 2009 [33]	Common framework			X		Conceptual model		X		X	X		X				Integrated	Manufacturing software	Template	During all states
Haile and Altmann, 2018 [25]	Model	X			X	Block diagram		X								X	Unified	Software services platforms	Standardization	Operational
Jepsen *et al.*, 2021 [26]	Model	X	X	X		Architectural model		X						X			Unified	Intelligent networks of machines	Middleware	Operational
Turk, 2020 [27]	Interoperability study	X				Conceptual model		X								X	Unified, integrated, federated	Construction integration technology	Standardization	During all states
Burzlaff *et al.*, 2019 [28]	Systematic literature review	X				Architectural model		X				X				X	N/A	Software architecture	N/A	During all states
Zacharewicz *et al.*, 2020 [29]	Common framework				X	Block diagram		X						X			Integrated	MDSEA	Simulation	Operational

4. Software Interoperability Representation

Some interoperability perspectives and levels are considered in previous work, as depicted in the last section. Such work present interoperability solutions and models by using visual representations. According to Tab. 3, such work is limited by the interoperability level, focus, and technique, likewise the terminology is dependent to the analysis scope and the particular essential elements of interoperability. In this Section we propose a high-level abstraction for analyzing the software interoperability by reaching an independent explanation of any focus, level, and technique of interoperability. We use pre-conceptual schemas for representing the seven essential elements of interoperability.

The main research question motivating the literature review in [32] are: what is the main set of concepts describing interoperability? This question is derived from the lack of unification regarding the terminology for explaining the interoperability phenomenon (as we can also see in Tab. 3). We identify seven concepts as essential for describing interoperability after performing the terminology unification process: source software system, target software system, information, context, symbol, language, and interface. In consequence, underlying research questions (RQ) and challenges (CH) arise from the identification of essential elements:

RQ1—How the essential elements are related to interoperability?
RQ2—Can we describe all the levels of interoperability in terms of its essential elements?
CH1—Achieving a classification of interoperability solutions and problems based on the interoperability essential element for supporting decision making from design to operation of software systems.
CH2—Defining interoperability evaluation criteria based on categorization of the interoperability knowledge.

We propose to relate the seven interoperability essential elements [31, 32] by using a PCS. PCS use only two types of structural relationships (IS and HAS) making it intuitive and easy to understand. Structural relationships are important for understanding the taxonomic relationships is-a and whole-part in a given domain, *e.g.*, interoperability. Another advantage of PCS is related to obtaining source code and collecting data of every concept-class in tables allowing to evaluate the consistency with the modeled domain. Such advantages make PCS suitable for representing interoperability phenomenon.

Our PCS about interoperability is developed in two phases: i) mapping: for extracting information about the interoperability essential elements from the literature and translating it into a controlled language; and ii) development: for relating the essential elements, representing their relationships, and defining each essential element.

A. Mapping Phase

We perform a systematic literature review for recognizing relationships among the interoperability essential elements. We use as a guide some elements of Tab. 3, *e.g.*, explanations of interoperability from different points of view, focuses, techniques, and levels; proposals considering interoperability in different states of a software system—*i.e.*, from the architecture state to the operational state; and proposals with visual representations and models (see Tab. 3). We aim to complete a description of interoperability embracing each perspective. We identify the seven essential elements of interoperability are present in all the interoperability definitions—different names are used but the notion remains. Based on such premise, we search again the literature for finding how such elements are associated with each other, what is the role of each element—the kind of relationship they exhibit—during the interoperability, what are their dependencies, and what attributes are relevant in the context of interoperability.

We adapt the methodology for mapping discourses from business-based technical documents [44] in order to perform our search. In our case, we just focus on the semantic analysis—lexical-semantic analysis and phrasal expression analysis—and the extraction of information (a manual process). The analyzed documents belong to the previous work of this paper (see Section 3) and other work [32]. So, we collect a corpus related to the software interoperability area. The stages covering the complete process are: (i) structural and functional analysis (ii) semantic processing, and (iii) discourse mapping. In the first stage, we locate each one of the seven elements for: getting the given definition, if any, by the authors; drawing the concept into a context close to the interoperability definition and explanation; looking for the concept relationships indicated in visual representation; and identifying attributes associated to the element. In the second stage, we execute lexical semantic analysis and phrasal-expression analysis for simplifying expressions based on the parse-tree technique and understanding the structure and relationships of the words in sentences; next, we make context analysis looking for phrasal expressions used for tenure and inheritance relationships. In the third phase, we rule the structure of common phrasal expression in the interoperability corpus by using some rules [44] as common grammatical structures for technical documents written in English; moreover, we group some verbs based on WordNet—a semantic resource for grouping senses of the same word—, mapping them into a controlled language near to the PCS—*i.e.*, expressing IS/HAS relationships.

The resulting process for searching the literature and mapping interoperability elements into a controlled language is summarized in Tab. 4. First column contains the interoperability

element. The text defining/including the interoperability element is described in second column. The third column comprises the mapping rule used for translating the text—the centered rules are applied for more than one expression—into a controlled-language expression included in the last column.

Table 4. Mapping process to controlled language (1/3)

	Relationships of concept	Mapping rule	Expression in CL (ECL)
INFORMATION	...ability of the models to **exchange** information… [14]	***If Obj1 + [exchange| provide| share| communicate| carry] + Obj2 -> Obj1 + has + Obj2*** **exchange** S(v) give or receive from one another **provide** S(v) give something useful or necessary **share** S(v) have in common **communicate** S(v) transmitting information **give** S (v) transfer possession of something concrete or abstract to somebody **carry** S(V) have with oneself; have on one's person	Model ***has*** information
	… source and receiver can **exchange** information… [15]		Source ***has*** information Receiver ***has*** information
	...ability two or more systems or components to **exchange** information… [4]		System/components ***has*** information
	…ability of two systems to **exchange**, and correctly **interpret** and **process** information… [16]		System ***has*** information
	...organization **sharing** information and **cooperating**… [45]		Organization ***has*** information
	...software systems capability to **communicate** and exchange information… [17]		Software system ***has*** information
	…in the context of information **exchange** among software …[27]		Software ***has*** information
	… interoperating for a system means to **provide** a mutual flow of desired outputs(information, products) to another system… [3]		System ***has*** outputs System ***has*** information
		If Obj1 + [: | "(" | i.e.| e.g.] + Obj2+ "," + Obj3+ ","+.. + [(")")]-> Obj2|Obj3|.. + is + Obj1	Information ***is*** output
	… systems or components use exchanged information…[4]	***If [Adj| VBN]+Obj-> Obj + has + attribute [value: Adj | VBN]***	Information ***has*** an attribute [value: exchanged]
	...Information **is** understood to be data in context…[14]	***If [Adj| VBN]+Obj-> Obj + has + attribute [value: Adj | VBN]*** ***If Obj1 + [is|(a| an)] + Obj2 -> Obj1 + is + Obj2***	Information ***is*** data in context
	…information **is** the meaning assigned to data by known conventions ... [2]		data ***has*** attribute [value: meaning assigned] Information ***is*** data
	...A Conversation **consists of** a set of Constructs…[10]	***If Obj1 + [consist of| contain] + Obj2 -> Obj1 + has + Obj2***	Conversation ***has*** constructs
	(VR) Constructs is the **content** of Conversation [10]		
SOURCE SOFTWARE SYSTEM (SSS) TARGET SOFTWARE SYSTEM (TSS)	…System **is made by** subsystems… [3]	***If Obj1 + [is made (up| by)] + Obj2 -> Obj1 + has + Obj2***	System ***has*** subsystems
	...A system **is a** bounded set of inter-connected elements… [14]	***If Obj1 + [is|(a| an)] + Obj2 -> Obj1 + is + Obj2***	System ***has*** elements
		If [Adj| VBN]+Obj-> Obj + has + attribute [value: Adj | VBN]	Element ***has*** attribute [value: inter-connected]

	Relationships of concept	Mapping rule	Expression in CL (ECL)
	…interoperability is a relation between systems…[10]	***If Obj1 + [is (a) (kind| type) of]*** + ***Obj2 -> Obj1 + is + Obj2***	Interoperability ***is*** relation between systems
	...Semantic interoperability means that sources and receivers can **exchange** information in a meaningful manner… [15]	***If Obj1 + [exchange| provide| share| communicate| carry]*** + ***Obj2 -> Obj1 + has + Obj2***	Source ***has*** information Receiver ***has*** information
	… a system can be real or abstract and can **belong to** any **part of** another system: a software component, as well as a hardware component, a person, a business rule, etc., can all **be** elements **of a** system…[4]	***If Obj2 [belong to] Obj1 -> Obj1 has Obj2***	System ***has*** another system
		If Obj1 + [: | "(" | i.e.| e.g.] + ***Obj2+ "," + Obj3+ ","+ .. + [(")")]-> Obj2|Obj3|.. + is + Obj1***	Software ***is*** system Hardware ***is*** system, etc.
		If Obj1 + [of the| of each| of a)] + ***Obj2 -> Obj2 + has + Obj1***	System ***has*** elements
	...system **comprising of** a heterogeneous mix of software and physical components…[15]	***If Obj1 + [consist| contain (of)|comprise (of|for)]*** + ***Obj2 -> Obj1 + has + Obj2***	System ***has*** software component; System ***has*** component
	…Software system involves hardware, software, and data associated with the software execution…[46]	***If Obj1 +[include| involve]*** + ***Obj2 -> Obj1 + has + Obj2***	Software system ***has*** software Software system ***has*** hardware Software system ***has*** data
	...Actor (the source actor) knows the Address of another Actor (the target actor)…[10]	***If Obj1 + [: | "(" | i.e.| e.g.]*** + ***Obj2+ "," + Obj3+ ","+ .. + [(")")]-> Obj2|Obj3|. + is + Obj1***	Source actor ***is*** actor Target actor ***is*** actor
	(VR) Actor **is** the **communicator** of the conversation [10]	***If Obj1 + [exchange| provide| share| communicate| carry]*** + ***Obj2 -> Obj1 + has + Obj2***	Actor ***has*** conversation

Table 4. Mapping process to controlled language (2/3)

	Relationships of concept	Mapping rule	Expression in CL (ECL)
SSS TSS	(VR) Engineered system inherits from System [20]	***If Obj1 + [is|(a|an)]+ Obj2 -> Obj1 + is + Obj2***	Engineered system ***is*** system
	…A system **can be** an abstraction for many organizational and institutional entities in the construction industry such as engineering software and on-line engineering service…[27]	***If Obj1 + [: | "(" | i.e.| e.g.| such as]*** + ***Obj2+ "," + Obj3 + ","+ .. + [(")")]-> Obj2|Obj3|. + is + Obj1***	Engineering software ***is*** system On-line engineering service ***is*** system
	…An interoperable IT service platform allows for the creation of a system of compatible suppliers…[25]	***If Obj1 + [of]*** + ***Obj2 -> Obj2 + has + Obj1***	Compatible suppliers ***has*** System
LANGUAGE	...Language are used to encode and **transmit** information…[10]	***If Obj1 + [exchange| provide| share| communicate| carry]*** + ***Obj2 -> Obj1 + has + Obj2***	Language ***has*** information
	...Language might **consist of** several sub-languages…[10]	***If Obj1 + [consist (of)|contain| comprise (of|for)]*** + ***Obj2 -> Obj1 + has + Obj2***	Language ***has*** sublanguages
	… Actors **share** a Language ….[10]	***if Obj1 + [exchange| provide| share| communicate| carry]*** + ***Obj2 -> Obj1 + has + Obj2***	Actor ***has*** language
	(VR) Language **is made up** of sublanguage [10]	***If Obj1 + [is made (up| by)]*** + ***Obj2 -> Obj1 + has + Obj2***	Language ***has*** sublanguage
	(VR) Language **carrying** language [10]	***if Obj1 + [exchange| provide| share| communicate| carry]*** +	Language ***has*** language

	Relationships of concept	Mapping rule	Expression in CL (ECL)
		Obj2 -> Obj1 + has + Obj2	
	… Business Process Execution Language (BPEL) …When the transformation of BPMN process to BPEL **of** each interconnected information system is completed…[6]	***If Obj1 + [of the| of each| of a)] + Obj2 -> Obj2 + has + Obj1***	Information system ***has*** BPEL
	...There are different approaches to interoperability and they **are** all **equivalent** in that they result in a common language that is either prescribed or emerges…[14]	***If Obj1 + [to be + (assimilate| equivalent) (to)] + Obj2 -> Obj1 + has + Obj2***	Interoperability ***has*** common language
	...Communication may not be achieved because of mismatches (heterogeneity),*i.e.*, the languages **of** the two protocols are different, although semantically equivalent…[18]	***If Obj1 + [of the| of each| of a)] + Obj2 -> Obj2 + has + Obj1***	Protocol ***has*** language
SYMBOL	… language **is a** set of structured symbols. A unified language helps achieving homogenization….[14]	***If [Adj| VBN] + Obj-> Obj + has + attribute [value: Adj | VBN]***	Symbol ***has*** attribute [value: structured]
		If Obj1 + [is (a| an)]+ Obj2 -> Obj1 + is + Obj2	Language ***is*** symbol
	… Language is a system **consisting** of a well-defined, usually finite, set of characters…[2]	***If Obj1 + [consist | contain (of)|comprise (of|for)] + Obj2 -> Obj1 + has + Obj2***	Language ***has*** character
	...A Language can be described in detail by **its** containing Constructs ….[10]	**If [PAdj] + Obj2-> X + has + Obj2**	X ***has*** construct X=Language
	...The context of a language is the set of constructs associated with the symbols **of** the language…[15]	***If Obj1 + [of the| of each| of a)] + Obj2 -> Obj2 + has + Obj1***	Language ***has*** symbol
	...A symbol **is a** physical object (*e.g.* a character string) that is used to designate meaning…[15]	***If Obj1 + [is|(a| an)] + Obj2 -> Obj1 + is + Obj2***	Symbol ***is*** physical object (*e.g.* a character string)
INTERFACE	(VR) interface **has** design input…. [7]	→	Interface ***has*** input
	(VR) interface **has** output/suggestions…. [7]	→	Interface ***has*** output interface ***has*** suggestion
	… systems **exchange** information through a well-defined interface… [14]	***if Obj1 + [exchange| provide| share| communicate| carry] + Obj2 -> Obj1 + has + Obj2***	Interface ***has*** information
	(VR) system **has** interface [4]	→	System ***has*** interface
	..behavior of a protocol from an external point of view thus **consisting** of the sequence of input and output actions at interface level…[18]	***If Obj1 + [consist | contain (of)|comprise (of|for)] + Obj2 -> Obj1 + has + Obj2***	Protocol ***has*** action Protocol ***has*** interface level
		If [Adj| VBN] + Obj-> Obj + has + attribute [value: Adj | VBN]	Action ***has*** attribute [value= input] /[value=output)
	...denote the data exchange interfaces **required** to **exchange** or externally store product information in a standardized format… [19]	***If Obj1 + [exchange| provide| share| communicate| carry] + Obj2 -> Obj1 + has + Obj2***	Interface ***has*** information
	…the system behavior (its running actions and transformations) is mostly reflected through the interface, which we **assimilate to** the set of inputs/outputs characterizing	***If Obj1 + [to be + (assimilate| equivalent) (to)] + Obj2 -> Obj1 + has + Obj2***	Interface ***has*** input/output

	Relationships of concept	Mapping rule	Expression in CL (ECL)
	quasiisolated systems…[4]		
	…interface is a common boundary between a considered system and another system, or between parts of a system, through which information **is conveyed**…[2]	***if Obj1 + [convey| pass| transfer] + Obj2-> Obj1 + has + Obj2*** ***convey*** *S(V) (of information) make known; pass on* ***transfer*** *S(V) (v) move from one place to another, send from one person or place to another*	Interface ***has*** information
	…interface is a hardware or software component that connects two or more other components for the purpose of **passing** information from one to the other…[2]		
	...the "symbiosis channel" that is a set of elements and interfaces that **provides the exchange** of elements (information, parts of any kind) useful to the interoperation…[3]	***If Obj1 + [exchange| provide| share| communicate| carry] + Obj2 -> Obj1 + has + Obj2***	Interface ***has*** element
		If Obj1 + [: | "(" | i.e.| e.g.| such as] + Obj2+ "," + Obj3 + ","+ .. + [(")")]-> Obj2|Obj3|. + is + Obj1	Information ***is*** element
	… the interfaces of a software component **are described by** interface types, the distinction between required an provided functionality… [28]	***If Obj1 + [of the| of each| of a)] + Obj2 -> Obj2 + has + Obj1***	Software component ***has*** interfaces
		If Obj1 + [describe by] + Obj2 -> Obj1 + has + Obj2 **describe** S(V) to point out the qualities of an individual	Interface ***has*** type Interface ***has*** functionality
		if [Adj| VBN] + Obj-> Obj + has + attribute [value: Adj | VBN]	Functionality ***has*** attribute [value= required] /[value=provided)
	…The standardized interfaces **can comprise** those for business tasks, administration tasks, execution task, and specific data formats…[25]	***If Obj1 + [consist| contain (of)|comprise (of|for)] + Obj2 -> Obj1 + has + Obj2***	Interfaces ***has*** business tasks Interfaces ***has*** administration tasks Interfaces ***has*** execution task
	…These characteristics are the interfaces (I/O) requested… [29]	***If Obj1 + [: | "(" | i.e.| e.g.| such as] + Obj2+ "," + Obj3 + ","+ .. + [(")")]-> Obj2|Obj3|. + is + Obj1***	Input interface ***is*** interface Output interface ***is*** interface

Table 4. Mapping process to controlled language (3/3)

	Relationships of concept	Mapping rule	Expression in CL (ECL)
CONTEXT	…Reference Language **is** a Language in which all the concepts relevant to the Communication Need **can be** unambiguously **defined**…[10]	***If Obj1 + [is (a| an)]+ Obj2 -> Obj1 + is + Obj2***	Reference language ***is*** language
	(VR) reference language **is the domain of the** communication need [10]	***If Obj1 + [is (a| an)]+ Obj2 -> Obj1 + is + Obj2***	Reference language ***is*** domain
		If Obj1 + [of the| of each| of a)] + Obj2 -> Obj2 + has + Obj1	Communication need ***has*** domain
	….Useful models **will be** those identifying and describing the elements that **are involved** in the interoperation…[4]	***If Obj1 + [to be + VBN] + by + Obj2 -> Obj2 + VB + Obj1*** ***If Obj1 +[include| involve] + Obj2 -> Obj1 + has + Obj2***	Interoperation ***has*** element
	(VR) model **has** representation [4]	→	Model ***has***

	Relationships of concept	Mapping rule	Expression in CL (ECL)
			representation
	(VR) Context **characterizes** level [21]	***If Obj1 + [characterize] + Obj2 -> Obj2 + has + Obj1 characterize*** *S(V) be characteristic of*	Level ***has*** context
	...From the analyzed data, context information (business information, extrinsic environment information, interaction time, intrinsic environment in- formation, and user knowledge) does not prevent the interaction from happening… [21]	***If Obj1 + [: | "(" | i.e.| e.g.| such as] + Obj2 + "," + Obj3+ ","+.. + [(")")] -> Obj2|Obj3|.. + is + Obj1***	Extrinsic environment information ***is*** context information
	(VR) Business processes supported by the digital systems in **their** individual contexts can be aggregated to achieve the overall intended purpose [22]	**If [PAdj] + Obj2-> X + has + Obj2**	X ***has*** context X=Digital systems
	...Ensuring semantic interoperability is difficult when sources and receivers **have** different contexts…[18]	⟶	Source ***has*** context, receiver ***has*** context
	...context **defined** by Bunge **as** C = (S, P, D). D, the domain of the context, **is a** set of individuals. There are no other individuals in C other than those in D. P **is a** set of predicates that are meaningful in C, with well-defined arity (*i.e.* number of arguments in its argument list)... S **is a** set of statements or propositions which contain only predicates from P and arguments from D...[18]	***If Obj1 + [is (a| an)]+ Obj2 -> Obj1 + is + Obj2***	Context ***is*** C=(S,P,D) S ***is*** proposition P ***is*** predicate D ***is*** individual
	...semantic interoperability not only entails the data being universally accessible and reusable but addresses the lack of common understanding caused by use of different semantic representations, **such as** dissimilar contexts…[22]	***If Obj1 + [: | "(" | i.e.| e.g.| such as] + Obj2+ "," + Obj3 + ","+.. + [(")")]-> Obj2|Obj3|.. + is + Obj1***	Dissimilar context ***is*** different semantic representation
	...right information is exchanged which by definition implies some type of context **sharing between** the systems, *i.e.* there is a common understanding of the meaning and context of use of the information... [14], [45]	***If Obj1 + [to be + VBN] +by+ Obj2 -> Obj2 + VB + Obj1***	Context ***is shared*** by systems
		If Obj1 + [exchange| provide| share| communicate| carry] + Obj2 -> Obj1 + has + Obj2	Systems ***has*** sharing context
	...Metadata registries, while useful in storing information, do not have the structure necessary for a formal semantic alignment simply because they do not maintain the context **of** information…[14]	***If Obj1 + [of] + Obj2 -> Obj2 + has + Obj1***	Information ***has*** context
	...A formal context **is a** triple (G, M, I) is called a formal context, if G and M are sets and I Í G X M is a binary relation between G and M. the elements of G **are** usually **called** objects and the elements of M attributes…[16]	***If Obj1 + [is (a| an)]+ Obj2 -> Obj1 + is + Obj2***	Context ***is*** triple(G,M,I) G ***is*** elements M ***is*** attribute I ***is*** relationship
	...Many problems are caused by	***If Obj1 + [of] + Obj2 -> Obj2 +***	Data ***has*** context

	Relationships of concept	Mapping rule	Expression in CL (ECL)
	systems' heterogeneity on data representation, meaning or context… [17]	***has + Obj1***	
	...The broader engineering context **would include** this common core along with additional concepts (objects, relationships, attributes, etc.)…[19]	***If Obj1 +[include| involve] + Obj2 -> Obj1 + has + Obj2***	Context ***has*** common core; context ***has*** element; context ***has*** attribute; context ***has*** relationship
	…Ontologies are a means to create a shared metadata model, describing knowledge through concepts, relations, and context… [26]	***If Obj1 + [describe (by|through)] + Obj2 -> Obj1 + has + Obj2*** **describe** S(V) to point out the qualities of an individual	Metadata model ***has*** context Metadata model ***has*** concepts Metadata model ***has*** relations

CONVENTIONS: Adj=Adjective**; VB=**Verb Base form**; VBN=** Verb Past Participle**; PAdj=** Possessive Adjective; **(VR)**= from visual representation; ABC: gray underlined words correspond with concepts mapped in the original concepts

B. Development Phase

We develop the PCS of Figure. 3 by using the expressions in controlled language from Tab. 4. In the next subsections we discuss the explanation about how the relationships of the elements are obtained based on the results of the mapping process and the definition of each element.

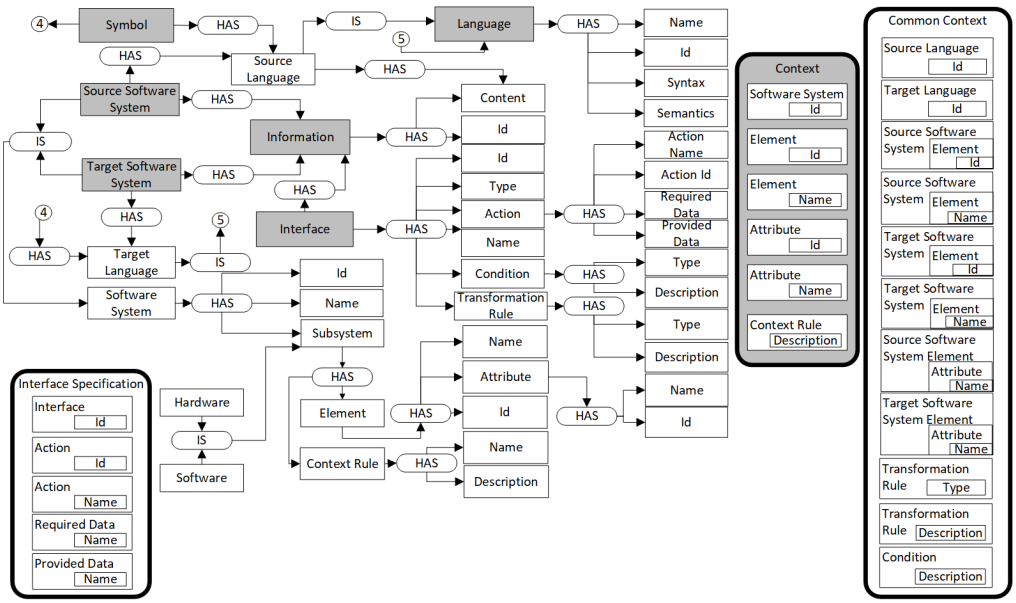


Figure 3. Interoperability pre-conceptual schema

1) Source Software System and Target Software System

According to Tab. 4, interoperability is explained as a relation—*ECL: interoperability* ***is*** *relation between systems*—happening between systems. The system has subsystems and elements—*ECL: system* ***has*** *elements, subsystems, another system*. At the same time, a software system is itself a system—*ECL: engineered system* ***is*** *system, software* ***is*** *system; hardware* ***is*** *system*—therefore, it inherits all the properties of a system. In addition, the exchange of information during interoperability is related to a source and a target system—

*ECL: source **has** information, receiver **has** information*—so source software system and target software system appear as specialized concepts from software system by using an "IS" relationship. In the event that interoperability occurs, from the target software system to the source software system—*e.g.*, in a mutual flow deriving into an answer—the roles of the systems are switched for describing a new interoperability situation.
Definition: interoperability happens at least between two software systems. Source software system is the system intended to create the information to be exchanged, and target software system is intended to use the exchanged information. Source and target software systems have an interoperability context—*i.e.*, information describing a set of characteristics of the software system.

2) Information
Interoperability is the ability of systems—source software system and target software system—for exchanging information; such information is something belonging to the systems—*ECL: system **has** information, software system **has** information, organization **has** information, etc*. Specifically, source software system has the information for producing the exchange and target software system is interested in accessing this information—*ECL: source **has** information, target **has** information*. For this reason, information in our PCS is associated with the two software systems presenting a structural relationship "HAS," meaning the same information belongs to both software systems.
Information can represent a set of some outputs of a software system—*ECL: information **is** output*—, which comprises data organized in some way—*ECL: information **is** data, information **is** data in context*. In this sense, we encompassed several types of structured data under the information concept—*e.g.*, in a manufacturing context, information is a type of structured data called capability profile representing a set of key features of a manufactured equipment.
Definition: Information is the main resource exchanged between source and target software systems during interoperability. Information comprises organized data. In addition, a condition for interoperability is the use of information; therefore, interoperability implies the correct interpretation and processing of the exchanged information.

3) Context
Context element represents the needed information for interpreting the exchanged information—*ECL: information **has** context, data **has** context, business information **is** context information, etc*. Each software system has its own context for creating and interpreting information—*ECL: source **has** context, receiver **has** context*. However, a shared context between the software systems is needed for the information exchange—*ECL: context **is** shared by systems, system **has** sharing context, dissimilar context **is** different semantic representation*. In addition, definition of context are related to different elements—*ECL: Context **is** triple(G,M,I), Context **is** C=(S,P,D)*. For these reasons, context is represented as a collection of atomic concepts of the software system rather than a singular element. Common context is also a collection of elements needed for interpreting the exchanged information.
Definition: Context is related to the interpretation of exchanged information during interoperability. The context involves a set of software system elements used for matching and transforming the exchanged information.

4) Interface
Interface is used for exchanging information between software systems. Interface has information at some level—*ECL: interface **has** information.* Hence, a structural relationship "HAS" links interface to information and interface is an independent element of the two software systems. Moreover, interface is related to the processing of inputs and outputs—*ECL: interface **has** input/output, interface **has** output, interface **has** suggestion*—concerned to the

manipulation and transformation of the exchanged information. Finally, interface is key in the achievement of the common context between source and target software systems.
Definition: Interface is the means for exchanging information between software systems. The interface involves transformation rules for establishing several relationships among the elements of a source and target software systems. Each transformation rule also involves conditions about the elements.

5) Language
Language is used for encoding information to be exchanged—*ECL: language* ***has*** *information*. A language includes syntax and semantic rules represented with structural relationship "HAS" in PCS. Source and target software systems have their own language for writing information related to software interoperability—*ECL: actor* ***has*** *language, protocol* ***has*** *language, information system* ***has*** *BPEL language*. Both source and target language are specializations of the language concept represented by an "IS" relationship.
Definition: Language is the code in which information is written. A language includes structures and rules for defining well-formed messages. In the case of the source software system, an exchange language is used for writing the information to be exchanged. A transformation could be needed for understanding the exchanged information in target software system.

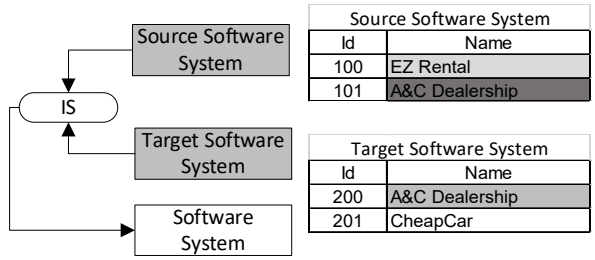
6) Symbol
Symbol and language are connected by using a "HAS" relationship because a language is made up by symbols—*ECL: language* ***has*** *symbols, language* ***is*** *symbol, language* ***has*** *character, language* ***has*** *construct, etc*. Languages encompass symbols in more complex structures named words. Words are the basic structures of a language which have a meaning.
Definition: Symbol is one of the individual elements conforming the content of the information. Symbols are representations with a name inside a language. Symbols can belong to several languages and have different applications.

5. Lab study—Example of Application

In this section, we instantiate the proposed PCS (see Figure. 3) by using a lab study. Such a study is a description of an interoperability practical situation with sufficient data for describing interoperability elements. Our lab study allows for recognizing interoperability elements regardless interoperability level, focus, and solution technique. Our outcomes are sets of tables—one for each interoperability element and its attributes—representing instances of the interoperability elements at a given time.

We use the example case proposed by Tolk et al. [47] related to the interoperability among three software systems—the car renting company EZRental, the dealership company A&C, and the manufacturing company CheapCar. The purpose of the interoperability between EZRental (playing the role of the source software system) and A&C Dealership (playing the role of the target software system) is the exchange of information about cars. Meanwhile, the purpose of the interoperability between A&C Dealership (source software system) and CheapCar (target software system) is the exchange of information about car parts when some parts should be replaced.

Following the interoperability PCS (see Figure. 3), interoperability has two software systems participating in the example at some time, source and target software systems take values depending on the described interoperability situation—*i.e.*, interoperability between EZRental and ChepCar or interoperability between EZRental and A&C Dealership. According to such a PCS, the attributes associated with the software systems are id and name; for this reason, source software system table and target software system table have two fields Id and Name. In Figure. 4, we present again the portion of the PCS being instantiated and color the rows that correspond to every software system in order to differentiate them in future tables.



Source Software System

Id	Name
100	EZ Rental
101	A&C Dealership

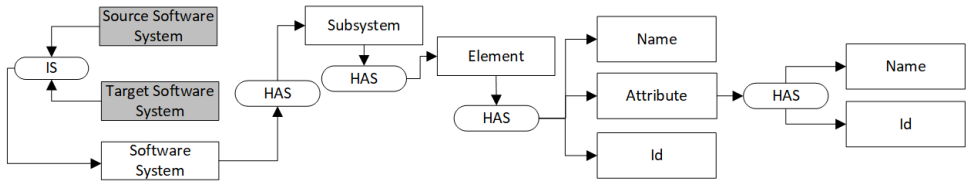
Target Software System

Id	Name
200	A&C Dealership
201	CheapCar

Figure 4. Source and target software system instantiated tables (The authors)

Then, we can observe the elements of EZRental software system mainly comprise cars, which are rented by customers. Some relevant attributes of a car are id, type, manufacturer, and mileage (attributes in light gray color in Figure. 5). Regarding the system A&C Dealership, the relevant elements when interoperability happens are car, part, and manufacturer (elements in gray color in Figure. 5). Finally, for CheapCar the most important elements are customer and part (elements in white color in Figure. 5). We use as foreign keys one of the leaf concepts belonging to the concept before a "HAS" relationship. For example, in Figure. 5 Software System Id field—in Source Software System Element table—indicates this element belongs to a particular software system.

Both software systems have a set of languages for exchanging information which are a portion of a complete language. For this reason, tables source language and target language (see Figure. 6) have references to a language, a symbol, and a software system (source and target respectively). Such elements are associated with the usage and interpretation of symbols in every software system. In the lab study, the exchange language is XML; hence, source and target software systems are intended to adopt such a language. However, the resulting syntax (format) and semantics (meaning) for exchanging information are particular to each software system. Syntax and semantic concepts are out of the scope of this paper.



Source Software System Element

Id	Name	Software System Id
EIS001	Car	100
EIS002	Car	101
EIS003	Customer	101
EIS004	Part	101

Source Software System Attribute

Id	Name	Element Id
AtrS001	Manufacturer	EIS001
AtrS002	Car_Name	EIS002
AtrS003	Serial Number	EIS002
AtrS004	Model	EIS002
AtrS005	Id	EIS001
AtrS006	Type	EIS001
AtrS007	Mileage	EIS001
AtrS008	Customer_Name	EIS003
AtrS009	Location	EIS003
AtrS010	Type	EIS003
AtrS011	Policy Number	EIS003
AtrS012	Part_Name	EIS004
AtrS013	P_Num	EIS004
AtrS014	Type_Part	EIS004
AtrS015	Make	EIS004
AtrS016	Model	EIS004

Target Software System Element

Id	Name	Software System Id
EIT001	Car	200
EIT002	Part	200
EIT005	Manufacturer	200
EIT003	Customer	201
EIT004	Part	201

Target Software System Attribute

Id	Name	Element Id
AtrT001	Car_Name	EIT001
AtrT003	Mark	EIT001
AtrT004	Model	EIT001
AtrT002	Part_Name	EIT002
AtrT005	Serial Number	EIT002
AtrT006	Model	EIT002
AtrT007	Type	EIT002
AtrT008	VIN_Number	EIT001
AtrT009	Customer_Name	EIT003
AtrT010	Location	EIT003
AtrT011	Part_Name	EIT004
AtrT012	Number_serial	EIT004
AtrT013	Tipo	EIT004
AtrT014	Car_Model	EIT004
AtrT015	Id_Manufacturer	EIT005
AtrT016	Name_Manufacturer	EIT005

Figure 5. Elements and attributes of source and target software system instantiated tables (The authors)

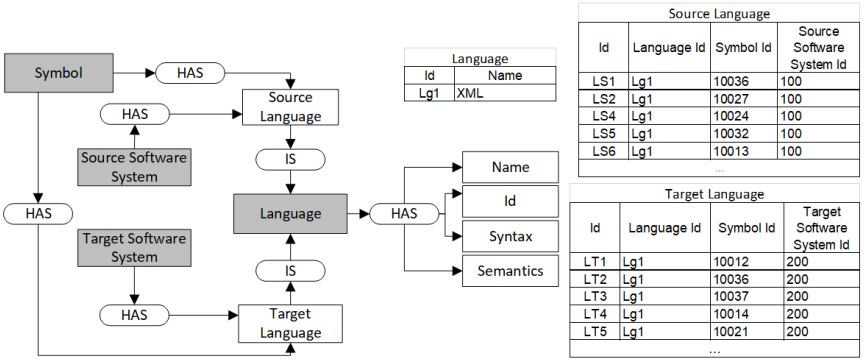


Figure 6. Source and target language instantiated tables (The authors)

Information is the core object of interoperability; the content of the information involves structures coded by using an exchanged language (Source Language Id column in table Content). In addition, the information content is specified with an order (Order column in table Content), which is necessary for rebuilding the original information in the target software system. The values related to the tables of information are presented in Figure. 7, dotted box exemplify a portion of information.

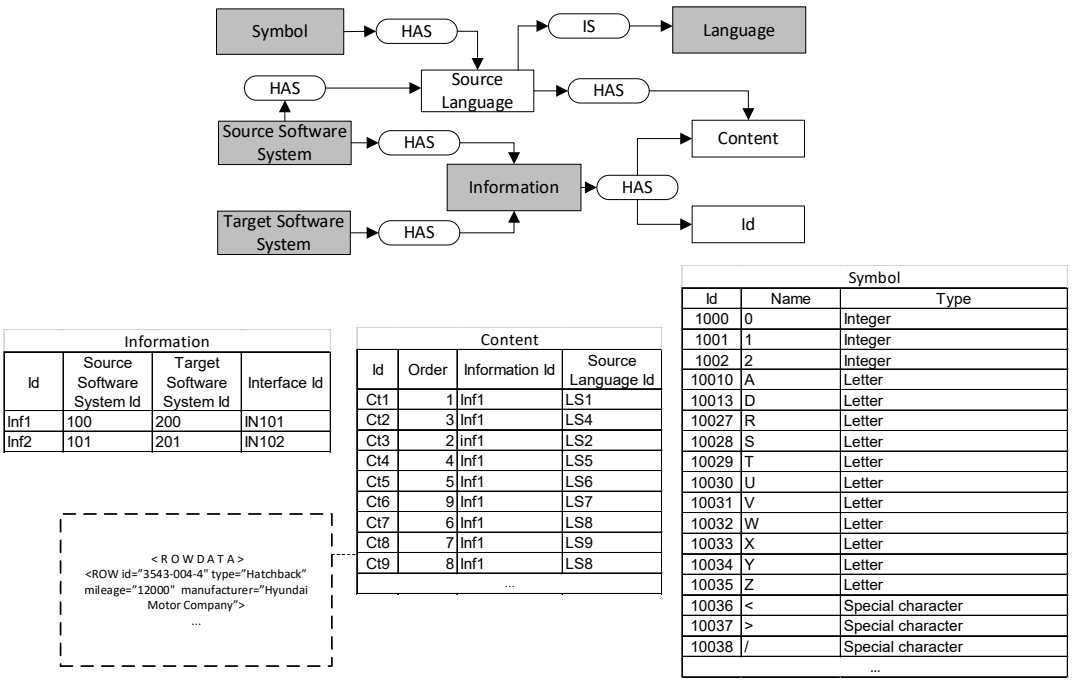
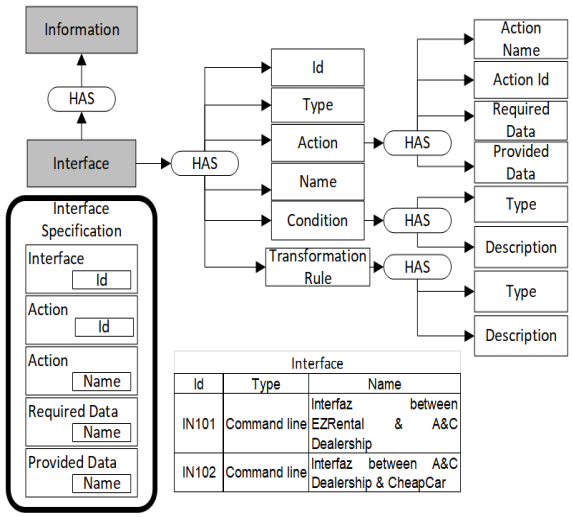


Figure 7. Information content and symbol instantiated tables (The authors)

Interface is an element related to the exchanged information. Suppose two interfaces, IN101 and IN102 presented in the tables of Figure. 8. Generally speaking, the actions intended to establish the interoperation between two software systems are related to the mapping of elements and their attributes (see Interface Specification table in Figure. 8). Such actions are developed by using APIs, web services, databases links, etc., which implies an agreement—implicit or explicit—between both software systems interested to interoperate.



Interface Specification

Interface Id	Action Id	Action Name	Required Data	Provided Data
IN101	FN1	Maping elements	XSD EZ Rental, XSD A&C Dealership	XSD A&C Dealership
IN101	FN2	Maping attributes	XSD EZ Rental, XSD A&C Dealership	XSD A&C Dealership
IN101	FN3	Insert attributes	XSD EZ Rental, XSD A&C Dealership	XSD A&C Dealership
IN102	FN4	Mapping elements	XSD A&C Dealership, CheapCar	XSD CheapCar
IN102	FN5	Mapping attributes	XSD A&C Dealership, CheapCar	XSD CheapCar
IN102	FN6	Adding attributes	XSD A&C Dealership, CheapCar	XSD CheapCar

Figure 8. Interface and Interface specification instantiated tables (The authors)

The context is represented as a report for collecting information about elements, attributes, and context rules (describing the relations between elements) in our PCS. Context is related to each software system; meanwhile, the common context is related to both software systems. In the common context, the relationships between elements from the software systems contexts are established by using transformation rules. Conditions are constraints for transforming and interpreting the exchanged information. We respectively map the representation of context and common context in our PCS in Figure. 9 and Figure. 10 and we show a table where such elements are instantiated.

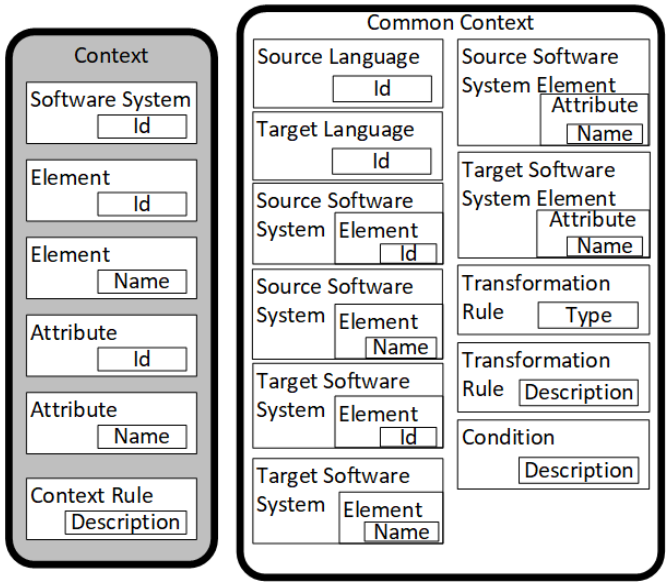


Figure 9. Context and Common context representation (The authors)

Common Context										
Source Language Id	Target Language Id	Source Software System Element Id	Source Software System Element Name	Target Software System Element Id	Target Software System Element Name	Source Software System Element Attribute Name	Target Software System Element Attribute Name	Transformation Rule Type	Transformation Rule Description	Condition Description
Lg1	Lg1	EIS001	Car	EIT001	Car	Car_Id	Car_VIN_Number	Equivalence Attribute x Attribute	Car_Id is equivalent to Car_VIN_Number	If Car_Id !="" Then ERROR
Lg1	Lg1	EIS001	Car	EIT001	Car	Car_Mileage	Car_Mileage	Agregation	Car_Mileage is equivalent to Car_Mileage	If Car_Mileage != null Then Create Car_Mileage
Lg1	Lg1	EIS001	Car	EIT001	Car	Car_Manufacturer	Car_Make	Transformation	Car_Manufacturer is equivalent to Car_Make	
Lg1	Lg1	EIS001	Car	EIT001	Car	Car_Manufacturer	Car_Model	Transformation	Car_Manufacturer is equivalent to Car_Model	
Lg1	Lg1	EIS001	Car	EIT005	Manufactur	Car_Manufacturer	Manufacturer_Id	Equivalence Attribute x Element	Car_Manufacturer is equivalent to Manufacturer_Id	
Lg1	Lg1	EIS003	Customer	EIT003	Customer	Customer_Name	Customer_Name	Equivalence Attribute x Attribute	Customer_Name is equivalent to Customer_Name	Si Customer_Name entonces Customer_Name= Customer_Name
Lg1	Lg1	EIS003	Customer	EIT003	Customer	Customer_Location	Customer_Location	Equivalence Attribute x Attribute	Customer_Location is equivalent to Customer_Location	
Lg1	Lg1	EIS003	Customer	EIT003	Customer	Customer_VIN_Number	Customer_VIN	Equivalence Attribute x Attribute	Customer_VIN_Number is equivalent to Customer_VIN	
Lg1	Lg1	EIS003	Customer	EIT003	Customer	Customer_P_Num	Customer_Number_serial	Equivalence Attribute x Attribute	Customer_P_Num is equivalent to Customer_Number_serial	
Lg1	Lg1	EIS004	Part	EIT004	Part	Part_P_Num	Part_Number_serial	Equivalence Attribute x Attribute	Part_P_Num is equivalent to Part_Number_serial	
Lg1	Lg1	EIS004	Part	EIT004	Part	Part_Type_Part	Part_Type	Equivalence Attribute x Attribute	Part_Type_Part is equivalent to Part_Type	
Lg1	Lg1	EIS004	Part	EIT004	Part	Part_Make	Part_Model	Transformation	Part_Make is equivalent to Part_Model	
Lg1	Lg1	EIS004	Part	EIT004	Part	Part_Model	Part_Model	Transformation	Part_Model is equivalent to Part_Model	

Figure 10. Common context instantiated tables (The authors)

6. Discussion

Software interoperability characterization in terms of its essential elements lead to identify how such elements are related to each other when interoperability happens. These seven elements resulting from a process of terminology unification remain present in visual representations of interoperability. While at a quick glance the visual representation made some elements visible, a linguistic analysis is necessary to abstract elements of interoperability present in different points of view, focuses, and techniques. The proposed level of abstraction allows for understanding how different concepts in the literature can be mapped to one of the seven essential elements and explained as roles participating in the interoperability process.

For these reasons, it is possible to compare the behavior of the essential elements of interoperability with the definition of invariant entities (*i.e.*, entities unalterable by particular transformations); therefore, essential elements of interoperability remain invariant across different interoperability definitions despite the names used in the analyzed perspectives. Additionally, the proposed PCS is a result of processes made with an interoperability level-independent view: terminology unification and mapping process for obtaining the relationships among interoperability essential elements.

In the context of software measurements, three properties of correctness are defined, such as: objectivity, impartiality, and sufficient precision [48]. Objectivity refers to the need for factual results and data, impartiality refers to the unbiased for the results, and precision refers to the selection of the material used as basis. Adapting the previous definition to the proposed visual representation, we can assess the correctness of our PCS of interoperability, thus: (i) objectivity is achieved by using the linguistic analysis, because such a process is applied on the original discourse by using linguistic rules for arriving to the results; (ii) impartiality is achieved from the criteria for obtaining the results, which depend on the linguistic rules rather than personal judgment; and (iii) sufficient precision is achieved because previous work is gathered looking for a variety of points of view, focuses, and techniques of interoperability, different visual representations of interoperability, and recognized databases as sources.

A classification of the available interoperability proposals of solutions is an alternative for informed decision making concerned to the software interoperability elements. Such classification also allows for understanding the problems of the specialized literature and the advance of solutions. For example, some software-system-oriented points of view are focused on characterizing aspects of capacity of components (hardware and software)—*e.g.*, the standard ISO 16100 for describing the manufactured equipment profiles and the proposal for

characterizing interoperability in a context-aware software systems—heading towards the possibility of reaching a software system interoperability profile. Some information-oriented points of view are evolved as an indissoluble unity including syntax and semantic aspects of data—e.g., the model of property type and value statement. Other aspects concerning the definition of an exchange language some of them in widespread use—*e.g.*, XML, JSON, BPMN, BPE, etc.—are masked as an information modeling decision. Something similar happens with the context and common context agreement which are under the notions of the information semantic modeling—*e.g.*, MDA, MDE, formal ontologies, etc. Finally, interface is a neglected subject linked to oblique decisions as architectural models, development frameworks, and organizational policies, among others.

7. Conclusions

The proposed pre-conceptual schema represents an interoperability structural view showing the relationships among interoperability essential elements—*i.e.*, source software system, target software system, information, interface, context, language, and symbol—in a visual way. Such interoperability relationships are collected with a linguistic analysis of the interoperability corpus based on interoperability literature. The corpus is selected regardless of points of view, focuses, techniques, and interoperability levels. In this way, we can identify the nature of each essential element (represented in PCS as a structural relationship "IS") and the role played in the achievement of interoperability (represented as a structural relationship "HAS"). A definition of the essential elements is given linking them to the interoperability explanation. In addition, we exemplify a practical interoperability situation selected from literature. Exemplification is useful for collecting features both implicitly described and explicitly reported. We provide tables for each interoperability essential element. Also, the exemplification allows for illustrating the suitability of the representation for explaining interoperability in terms of its essential elements. Finally, the proposed characterization is useful for comparing points of view and solutions of interoperability and supporting decision-making process.

As future work, we suggest using our proposed representation for characterizing actual interoperability cases with different levels of interoperability and different levels of abstraction. A formalization of our representation would be useful for testing its accuracy. A deep categorization of the interoperability issues and solutions is needed for comparing different focuses, points of view, and techniques. Furthermore, a dynamic extension to the interoperability PCS could lead a software prototype for instantiating the interoperability elements from early stages of software systems and detecting weaknesses of our model. A first step towards a dynamic representation of interoperability in IoT, for example, it should introduce the use of services (offered and consumed by Things) triggered by system events. Finally, interoperability assessment criteria, measures, and maturity models need to be collected for classifying them according to the essential elements of interoperability.

8. References

[1]. ISO/IEC 25010:2008: Software engineering-Software product Quality Requirements and Evaluation (SQuaRE) Quality model, 2008.

[2]. IEEE 100:2000: The Authoritative Dictionary of IEEE Standards Terms, 2000.

[3]. M. Dassisti and D. Chen, "Interoperability analysis: general concepts for an axiomatic approach", Lecture Notes in Computer Science, New York: Springer, pp 49-60, October 2011 [the Move to Meaningful Internet Systems: OTM 2011 Workshops].

[4]. Y. Naudet, T. Latour, W. Guedria and D. Chen, "Towards a systemic formalisation of interoperability", Computers in Industry, Vol. 61, pp. 176–185, 2010.

[5]. L. Bass, P. Clements and R. Kazman, Software Architecture in Practice: Software Architect Practice, Massachusetts: Adisson-Wesley, 2015.

[6]. M.F. Amr, M. Qbadou, K. Mansouri and B. Riyami, “Towards a model of adaptation and interfacing based on a middleware layer SOA for interoperability of several different information systems”, Intelligent Systems and Computer Vision (ISCV), Fez, April 2017.
[7]. S. Saha, Z. Usman, S. Jones, R. Kshirsagar and W. Li, “Towards a formal ontology to support interoperability across multiple product lifecycle domains”, 11th IEEE Conference on Semantic Computing (ICSC), San Diego, January-February 2017.
[8]. U. Epple, M. Mertens, F. Palm and M. Azarmipour, “Using properties as semantic base for interoperability”, *IEEE Transactions on Industrial Informatics*, Vol. 13, pp. 3411-3419, 2017.
[9]. R.C. Motta, K.M. De Oliveira and G.H. Travassos, “Characterizing interoperability in context-aware software systems”, VI Brazilian Symposium on Computing Systems Engineering (SBESC), João Pessoa, May 2016.
[10]. J. Ullberg, P. Johnson and M. Buschle, “A language for interoperability modeling and prediction”, Computers in Industry, Vol. 632, pp. 766–774, 2012.
[11]. ISO 11354:2011: Advanced automation technologies and their applications -- Requirements for establishing manufacturing enterprise process interoperability -- Part 1: Framework for enterprise interoperability, 2011.
[12]. H. Bazoun, J. Ribault, G. Zacharewicz, Y. Ducq and H. Boyé, “SLMTOOLBOX: enterprise service process modeling and simulation by coupling devs and services workflow”, Journal of Simulation and Process Modelling, Vol. 11 (6), pp. 453-467, 2016.
[13]. Chen, G. Doumeingts and F. Vernadat, “Architectures for enterprise integration and interoperability: Past, present and future”, Computers in Industry, Vol. 59 (7), pp. 647-659, 2008.
[14]. S. Diallo, Towards a formal theory of interoperability [Doctoral dissertation], Norfolk: Old Dominion University, 2010.
[15]. J. Lee, Integrating information from disparate contexts: a theory of semantic interoperability [Doctoral dissertation], Cambridge: Massachusetts Institute of Technology, 1996.
[16]. G. Yang and J. Feng, “Database Semantic Interoperability based on Information Flow Theory and Formal Concept Analysis”, *Information Technology and Computer Science*, Vol. 4 (7), pp. 33–42, 2012.
[17]. H. Abukwaik, D. Taibi and D. Rombach, “Interoperability-Related Architectural Problems and Solutions in Information Systems: A Scoping Study”, Switzerland: Springer International Publishing, 2014.
[18]. R. Spalazzese, A Theory of Mediating Connectors to achieve Interoperability [Doctoral dissertation], L’Aquila: Università degli studi de l’Aquila, 2011.
[19]. S. Szykman, S. Fenves and W. Keirouz and S. Shooter, “A foundation for interoperability in next-generation product development systems”, *Computer-Aided Design*, Vol. 33 (7), pp. 545–559, 2001.
[20]. G. Serapião, W. Guédria and H. Panetto, “An ontology for interoperability assessment: A systemic approach”, Computer in Industry, Vol. 16, pp. 1- 13, 2019.
[21]. R. Motta, M. de Oliveira and G. Travassos, “A conceptual perspective on interoperability in context-aware software systems”, *Information and Software Technology*, Vol. 114, pp. 231-257, 2019.
[22]. L. Liu, W. Li, N. Aljohanic, M. Lytrasc, S. Hassand and R. Nawaza, “A framework to evaluate the interoperability of information systems – Measuring the maturity of the business process alignment”, *International Journal of Information Management*, Vol. 54, pp. 1-13, 2020.
[23]. H. Panetto, M. Zdravkovic, R. Jardim-Goncalves, J. Cecil and I. Mezgár, “New perspectives for the future interoperable enterprise systems”, Computers in Industry, Vol. 79, pp. 47–63, 2016.
[24]. R. Rezaei, T. Chiew, S. Lee and Z. Shams Aliee, “Interoperability evaluation models: A systematic review”, Computer in Industry, Vol. 65, pp. 1–23, 2014.

[25]. N. Haile and J. Altmann, “Evaluating investments in portability and interoperability between software service platforms”, Future Generation Computer Systems, Vol. 78, pp. 224-241, 2018.
[26]. S.C. Jepsen, T. Worm, T. I. Mork and J. Hviid, “Industry 4.0 Middleware Software Architecture Interoperability Analysis”, 3rd International Workshop on Sotware Engineering Research and Practices for the IoT (SERP4IoT), Madrid, Jun 2021.
[27]. Z. Turk, “Interoperability in construction – Mission impossible?”, Developments in the Built Environment, Vol. 4, 2020.
[28]. F. Burzlaff, N. Wilken, C. Bartelt and H. Stuckenschmidt, “Semantic Interoperability Methods for Smart Service Systems: A Survey”, *IEEE Transactions on engineering management*, 2019.
[29]. G. Zacharewicz, N. Daclin, Doumeingts G. and H. Haidar, “Model Driven Interoperability for SystemEngineering”, Modelling, Vol. 1, pp. 94-121, 2020.
[30]. Görg, M. Pohl, E. Qeli and K. Xu, “Visual representations”, in Human-Centered Visualization Environments, A. Kerrens, A. Ebert and J. Meyer, Eds. Heidelberg: Springer Berlin, 2007, pp. 163–230.
[31]. Torres, Formulating a theory about interoperability among heterogeneous software systems, based on the Semat kernel [Doctoral dissertation], Medellin: Universidad Nacional de Colombia, 2019.
[32]. D. Torres, M. Villavicencio and C. Zapata, “Towards a Terminology Unification in Software Interoperability”, 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Prague, August 2018.
[33]. ISO 16100:2009: Industrial automation systems and integration -- Manufacturing software capability profiling for interoperability -- Part 1, 2009.
[34]. J. Rumbaugh, I. Jacobson and G. Booch, The unified modeling language reference manual, New Jersey: Addison-Wesley, 2005.
[35]. H. Purchase, L. Colpoys, M. McGill, D. Carrington and C. Britton, “UML class diagram syntax: an empirical study of comprehension”, Australian Symposium on Information Visualization, Sydney, December 2018.
[36]. S.M. Lynch, Presenting results of statistical analysis, in Using Statistics in Social Research, New York: Springer, 2005, pp. 175–184.
[37]. Shanks, E. Tansley and R. Weber, “Using ontology to validate conceptual models”, Communications of the ACM, Vol. 46, pp. 85-89, 2003.
[38]. I.M. Greca and M.A. Moreira, “Mental models, conceptual models, and modelling”, International Journal of Science Education, Vol. 22, pp. 1–11, 2000.
[39]. Paquette, Graphical ontology modeling language for learning environments, vol 5, Technical report, Cognition and Learning, 2007.
[40]. Y. Orlarey, D. Fober and S. Letz, An algebraic approach to block diagram constructions, Technical report, Centre National de Création Musicale, 2002.
[41]. W. Wang, J.M. Loman, R.G. Arno, P. Vassiliou, E.R. Furlong and D. Ogden, “Reliability block diagram simulation techniques applied to the IEEE std. 493 standard network”, IEEE Transactions on Industry Applications, Vol. 40, pp. 887–895, 2004.
[42]. A. Manrique and C.M. Zapata, “Transformación de lenguaje natural a controlado en la educción de requisitos: una síntesis conceptual basada en esquemas preconceptuales”, Revista Facultad de Ingeniería, Vol. 70, pp. 132-145, 2014.
[43]. C.M. Zapata, G.L. Giraldo and S. Londoño, “Executable pre-conceptual schemas”, Revista Avances en Sistemas e Informática, Vol. 8, pp.15-24, 2011.
[44]. A. Manrique, A formalization for mapping discourses from business-based technical documents into controlled language texts for requirements elicitation [Doctoral dissertation], Medellin: Universidad Nacional de Colombia, 2014.
[45]. S. Diallo, H. Herencia-Zapana and J. Padilla, “Understanding interoperability”, Emerging M&S Applications in Industry and Academia Symposium, San Diego, April 2011.

[46]. Jacobson, P. Ng, P. McMahon, I. Spence and S. Lidman, “The Essence of Software Engineering: Applying the Semat Kernel”, New Jersey: Addison-Wesley, 2014.
[47]. A. Tolk, S. Diallo, R. King and C. Turnitsa, “A layered approach to composition and interoperation in complex systems”, in Complex Systems in Knowledge-based Environments: Theory, Models and Applications SE, A. Tolk and L. Jain, Eds. Berlin: Springer, 2009, pp. 41–74.
[48]. ISO/IEC 9126-2:2006: Software engineering- Product Quality – Part 2: External metrics, 2006.

Diana M. Torres Ricaurte received her Ph.D. from Universidad Nacional de Colombia. She is member of the computational language research group. Her research interests are software engineering, software quality and interoperability issues related to context alignment, data mapping, and interfaces definition languages.

Mónica K. Villavicencio Cabezas is a full professor at the Electrical and Computer Engineering department at ESPOL University. She received her Ph.D. from the École de technologie supérieure, Université du Québec in Montreal, Canada. She teaches Software Engineering courses for undergraduate and graduate programs at ESPOL. Her research interests are software process improvement, software quality, software engineering education, and applied software engineering.

Carlos M. Zapata Jaramillo is a full Professor (tenured) at the Computer and Decision Science Department—Universidad Nacional de Colombia, Medellin campus. He received his Ph.D from Universidad Nacional de Colombia. He is also chairman of the Executive Committee of the Latin American Chapter of Semat (Software Engineering Method and Theory). His research interests are software engineering, requirements engineering, computational linguistics, and didactical strategies for teaching engineering.