

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/328471895>

Where Enterprise Architecture and Early Software Engineering Meet: An approach to use cases definition

Conference Paper · October 2018

DOI: 10.1145/3275245.3275271

CITATIONS

0

READS

53

4 authors, including:



Gabriel M Miranda

Universidade Federal do Espírito Santo

9 PUBLICATIONS 1 CITATION

[SEE PROFILE](#)



Monalessa Perini Barcellos

Universidade Federal do Espírito Santo

76 PUBLICATIONS 295 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Creating a method to support defining and monitoring indicators and strategies to IT Services [View project](#)



Análises das avaliações dos tutores a distância em um curso de Pós-graduação por meio de Mineração de Dados [View project](#)

Where Enterprise Architecture and Early Software Engineering Meet

An approach to use cases definition

Gabriel M. Miranda

Federal University of Espírito Santo (UFES)
Vitória, ES
gabrielmartinsmiranda@gmail.com

Lucas A. Santos

Federal University of Espírito Santo (UFES)
Vitória, ES
lucasaugsantos@gmail.com

César H. Bernabé

Federal University of Espírito Santo (UFES)
Vitória, ES
cesar.hber@gmail.com

Monalessa P. Barcellos

Federal University of Espírito Santo (UFES)
Vitória, ES
monalessa@inf.ufes.br

ABSTRACT

Software development involves the resolution of technical problems related to a certain domain. However, in order to provide a suitable technical solution, it is necessary to take the organizational environment related to the software into account. Use cases have been often used to elicit requirements and represent functionalities that the software must provide to its users. However, use cases are not expressive enough to represent the organizational environment. Moreover, this is not the purpose of use cases. In this context, Enterprise Architecture (EA) emerges as a way to describe the organization's domain. EA provides architectural descriptions that support the alignment between information technology (IT) and organizational processes and, thus, helps developers to properly understand the requirements the software must meet. In this paper, we propose an approach that uses EA models as a basis to define use cases, named CEA (use Cases definition oriented by Enterprise Architecture modeling). To demonstrate the proposal use, we applied it in a project in the Public Security domain. Additionally, CEA was evaluated in an experimental study. The results indicate that EA models helped requirements engineers to define use cases.

CCS CONCEPTS

• **Software and its engineering** → **Requirements analysis**;

KEYWORDS

Enterprise Architecture, Use Case, Requirements Engineering

ACM Reference Format:

Gabriel M. Miranda, César H. Bernabé, Lucas A. Santos, and Monalessa P. Barcellos. 2018. Where Enterprise Architecture and Early Software Engineering Meet : An approach to use cases definition. In *17th Brazilian Symposium*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SBQS, October 17–19, 2018, Curitiba, Brazil

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6565-9/18/10...\$15.00

<https://doi.org/10.1145/3275245.3275271>

on Software Quality (SBQS), October 17–19, 2018, Curitiba, Brazil. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3275245.3275271>

1 INTRODUCTION

Use cases are a key means to elicit software requirements from the point of view of the users of a system [22]. Requirements elicitation via use cases supports iterative software development and increases user participation. However, software development is a complex task that not only require the resolution of technical problems. The organizational environment related to the proposed software has also to be taken into account [5, 7, 9]. In business contexts, the application domain consists of the organization where the software will be deployed. Therefore, a good knowledge of the application domain is critical to succeed in requirements elicitation. If not properly addressed, requirements can hinder business/IT alignment and produce a software that does not meet organizational needs [8].

To analyze and understand an organization, the requirements engineer should describe the organization's knowledge domains (business processes, information technology and infrastructure), and the relations between such domains. In this regard, while textual representations are easy to produce and edit, they are not as strong a communication tool as graphical models. A visual representation of such textual information is helpful during the process of knowledge acquisition [22]. In this context, the Enterprise Architecture (EA) discipline emerges as a way to manage the knowledge domains of an organization and describe them by using architectural descriptions (or models) [12, 16]. Architectural descriptions provide an effective and less costly alignment between information technology and organizational processes, services, actors and other business components [15, 16]. Therefore, they help developers to properly understand the organizational environment and the requirements the software must meet [8].

According to Padua et. al. [18] and Sinha and Paradkar [22], the use of EA models in the requirements elicitation phase, particularly on use cases development, is a valuable practice, since such models deal with information about several aspects of the organization and are, thus, a rich font of information for acquiring knowledge about

the proposed software.

The use of EA models in requirements elicitation provides several benefits, such as [8, 18]: (i) the requirements tend to reflect the business needs; (ii) low number of redundancies in requirements, and (iii) the software development is conducted by the business needs. According to Monsalve et al. [17] the software development depends on the quality of activities related to requirements elicitation. Adopting EA models in knowledge acquisition can subside the software development, avoiding (i) incomplete requirements that are not related to business needs; (ii) rework, and (iii) leading the project to fail.

In view of the above, in this paper, we present an approach that employs EA models as artifacts to requirements elicitation trajectory, named CEA (use Cases definition oriented by Enterprise Architecture modeling). While these models may play a role in knowledge acquisition in general, here we focus on their role in supporting the definition of use cases.

The main feature of CEA is the use of the semantics of the elements represented in enterprise architectures to support use cases definition. These elements are represented by constructs of the modeling language used to create the EA models. The modeling language considered in CEA is the one used in the ArchiMate EA framework [1]. The constructs addressed in CEA are related to the Business and Application layers of EA models represented by using the ArchiMate language.

In this paper, we present CEA and its use in a real-world e-Government project involving systems related to the Public Security domain. We also present an experimental study in which we investigated if, in contrast with textual representations, architectural descriptions provide a better support to requirements engineers in the definition of use cases reflecting the business needs.

The paper is structured as follows. Section 2 provides the background for the paper, presenting information about the ArchiMate EA framework, software requirements engineering and use cases. Section 3 describes CEA. Section 4 presents the use of CEA in a real-world project. Section 5 addresses an experimental study to evaluate CEA. Section 6 discusses related works. Finally, Section 7 presents our final considerations and perspectives of future works.

2 BACKGROUND

In this section we present the main theoretical foundations relevant to this paper.

2.1 ArchiMate

ArchiMate comprises an EA modeling framework and a homonymous EA modeling language [1, 16]. Its main purpose is to promote the integration of the various viewpoints of the organizational architecture, promoting communication between stakeholders and analysis of various aspects of the organization [16].

Table 1: EA elements description and ArchiMate notation [1]

Description	Notation
Business actor - is a business entity that is capable of performing behavior. A business actor may be assigned to one or more business roles. It can then perform the behavior to which these business roles are assigned.	
Business role - is the responsibility for performing specific behavior, to which an actor can be assigned, or the part an actor plays in a particular action or event.	
Business process - represents a sequence of business behaviors that achieves a specific outcome such as a defined set of products or business services.	
Application Component - represents an encapsulation of application functionality aligned to implementation structure.	
Application process - represents a sequence of application behaviors that achieves a specific outcome.	

The central feature of the ArchiMate is the incorporation of the concepts of the service orientation paradigm. A service is defined as a unit of functionality that an entity (e.g., an organization, a department, a system) provides to another. This service orientation allows for a layered view of the architectural models, in which the service concept is one of main links between the different layers.

The main graphical elements provided by ArchiMate are disposed in three architectural layers: (i) the business layer - which concerns the products and services produced by business processes executed by actors or roles; (ii) the application layer - which concerns the systems that support the business layer; and (iii) the technology layer - which concerns infrastructural elements.

In this paper, we focus on the elements of the business and application layers. Table 1 presents the description and notation of the EA modeling elements used in this paper. In the table, business elements are represented in yellow (the first three elements in the table) whereas the application elements are presented in blue (the two last elements in the table).

Concerning relationship elements, in this paper we focus on the ones showed in Table 2.

2.2 Requirements Engineering

Requirements play a central role in software development. They are determinant to a project succeed. One of the key measures to evaluate the success of a software is the degree to which the software meets the established requirements [19].

Table 2: EA relationships description and ArchiMate notation[1]

Description	Notation
Composition - indicates that an element consists of the behavior of one or more other elements.	
Aggregation - indicates that an element groups a number of other elements.	
Assignment - expresses the allocation of responsibility, performance of behavior, or execution.	
Used by - models that an element provides its functionality to another element.	

A well-known classification of software requirements categorizes them as functional requirements and non-functional requirements. Functional requirements describe functionalities and services that the software must provide. Non-functional requirements, in turn, describe constraints on the functionalities and services provided by the software. Therefore, in summary, requirements define what the software must provide and the circumstances under which it must operate [23].

Requirements Engineering provides the appropriate mechanisms to understand the customer needs, assess feasibility, negotiate a reasonable solution, specify the solution unambiguously, validate the specification and manage the requirements that specify how the software will be implemented [24].

The Requirements Engineering process is responsible for eliciting, analyzing, documenting and managing requirements. It consists of the following activities [23]: (i) requirements elicitation, when the requirements to be met by the software are established; (ii) requirements analysis, when models are created to represent a conceptual view of the software and to serve as a basis to the subsequent phases of the development process; (iii) requirements documentation, when the requirements are recorded in documents or CASE tools; (iv) requirements verification and validation, when the requirements are evaluated; and (v) requirements management; when changes in the requirements are controlled.

2.3 UML Use Cases

Since the 90's, when Jacobson [14] proposed an approach that defines interactions between an actor (someone using system functionalities) and the functionality itself, use case diagrams have been one of the main artifacts produced in requirements elicitation and analysis. Use case diagrams provide a view of the functionalities that a software must provide and the actors that interact with them.

The following use case elements are relevant in this paper: actors, use cases, and association, include and extend relations.

An actor is as a type of role played by an entity that interacts with the software (e.g., by exchanging signals and data), but which is external to the software (in the sense that an instance of an actor is not a part of the instance of its corresponding software). Actors may represent roles played by human users, external hardware, or other subjects. It is represented by a “stick man” icon with the name of the actor in the vicinity [20].

Use cases describe the outwardly visible requirements of a software. They are used to create and validate a proposed design and to ensure that the software meets all requirements, contributing to the whole software process development [20].

Associations are lines between actors and use cases or between use cases. An association between an actor and a use case means that the actor is responsible for performing the functionality represented by the use case. Associations between use cases can be extend or include relations.

An extend relation specifies that the behavior of a use case may be extended by the behavior of another (usually supplementary) use case. The extension takes place at one or more specific extension points defined in the extended use case. However, the extended use case is defined independently of the extending use case and it is meaningful independently of the extending use case. On the other hand, the extending use case typically defines behavior that may not necessarily be meaningful by itself. Instead, the extending use case defines a set of modular behavior increments that augment an execution of the extended use case under specific conditions [20].

An include relation is a relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. The included use case is not optional and it is always required for the including use case to execute correctly. Therefore, an included use case cannot exist without its including and if an including use case is excluded, all included use cases are also excluded [20].

The notation used to represent use case diagram elements are presented in Table 3.

3 CEA APPROACH

The complete understanding of an enterprise environment favors software development [18]. As previously argued, EA models can represent the enterprise environment by structuring it in layers (business, information technology and infrastructure). Therefore, in this section we discuss how EA models providing an overview of the enterprise environment can be used to support requirements engineers in use cases definition.

The CEA approach consists in applying the following set of mapping rules (from EA to use case elements): (M1) map application processes to use cases, (M2) map composition and aggregation relations between application processes to include and exclude relations (respectively) between use cases, and (M3) map business

Table 3: Use Case Diagram elements from UML (Adapted from [14])

Description	Notation
Actor	
Use Case	
Association	
Include	
Extend	



Figure 1: First Mapping Rule

actors and roles to use case actors.

In M1, the requirements engineer must analyze the application processes in the EA model. Application processes represent application behaviors to achieve a specific outcome. Use cases, in turn, represent functionalities that must be performed to meet established requirements. Thus, application processes can be used to identify use cases. The application processes represented in an EA model should be represented as use cases in the use case diagram. However, the mapping between application processes and use cases are not necessarily 1 to 1, i.e., one application process can derive two or more use cases, as well as, two or more application processes can derive a single use case. The use cases name does not need to be equal to the application processes name. Figure 1 illustrates the mapping between application process and use case.

In M2 the requirements engineer should analyze the composition and aggregation relations between application processes. As explained in Section 2, on the ArchiMate side, the composition relationship indicates that the behavior of an element consists of the behavior of one or more other elements. On the use cases side, the include relation indicates that the behavior of the included use case is inserted into the behavior of the including (the base) use case. Thus, both relations represent a behavior that includes other behaviors. Therefore, composition relations in EA models are mapped to include relations in use case diagrams.

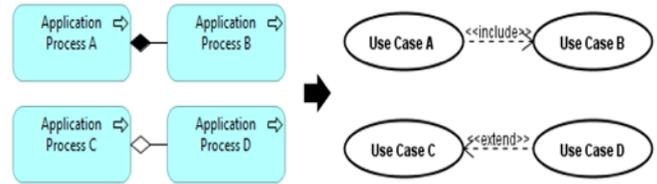


Figure 2: Second Mapping Rule

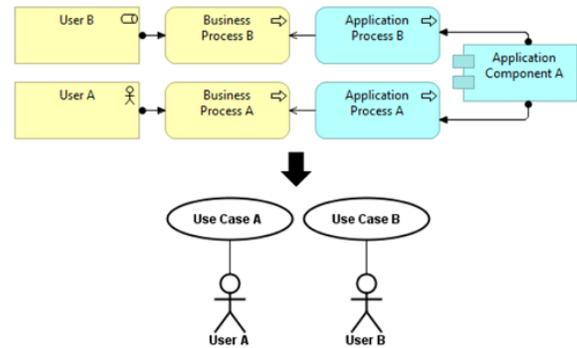


Figure 3: Third Mapping Rule

The aggregation relationship in ArchiMate was inspired by the aggregation relationship in UML class diagrams. In contrast to the composition relationship, the part can exist independent of the whole. In use case diagrams, when there is an extend relation between two use cases, the extended use case is meaningful on its own, i.e., it is independent of the extending use case. Thus, in both relations, an independent part adds some behavior to the whole. Therefore, aggregation relations in EA models are mapped to extend relations in use case diagrams. Figure 2 illustrates these mapping rules.

In M3, the requirements engineer must analyze the role and actor business elements and the process and component application elements in the EA models. Actors and roles represent the entities responsible (represented by the assignment relation) for performing business processes. Usually, business processes are supported by softwares (application components). Therefore, if in the EA model a business process is supported by an application process through the used by relation, we can infer that the business actor or role responsible for the business process is a potential user of the functionality (application process) of a software (application component). Thus, actors and roles in EA models are mapped to actors in use case diagrams. Figure 3 illustrates this mapping rule.

4 APPLYING CEA IN THE PUBLIC SECURITY DOMAIN

According to the Brazilian Health Ministry, between 1996 and 2010 there were almost 1.9 million violent deaths in Brazil, including 710 thousand homicides; and 174 thousand deaths whose basic cause could not be determined by the State. That is, violent death incidents of undetermined cause represent 9.6% of all violent events. In

developed countries, this proportion is a residue with less than 1% of all violent cases [6]. Moreover, recent data of the 2017's Violence Atlas [6] shows that Brazil has the highest murder rate in the world.

As Cerqueira [6] noted, the lack of consistent and qualified information on crimes and violent deaths in Brazil is caused in part by deficiencies concerning the sharing and dissemination of information among public administration (PA) agencies. Although these agencies have a large amount of information in their information systems, these systems function in isolated silos, failing to support overall decision making.

Problems in decision making bring consequences regarding the planning of actions for preventing new tragedies that could save uncountable lives in future. After all, without updated and quality information there is no way to establish accurate diagnoses about events, much less verify whether policies and implemented programs have had a positive effect or not.

In a research project in the Public Security domain, particularly in the Violent Crimes against Life subdomain, we have studied PA agencies involved in solving crimes in the state of Espírito Santo, Brazil, and the information systems used by these PA agencies. Solving crimes is a complex process, involving a large amount of information which permeates several PA agencies (e.g., Public Security Secretary and Court of Justice). To support this process, agencies use several applications, which are not integrated. The lack of data integration leads to data inconsistency and impacts on decision-making.

Our goal in the research project is to create solutions to provide the necessary data to the PA agencies to make well-informed decisions. These solutions involve systems reengineer and systems integration, among others.

One of the information systems is the PC4SEG [2], which is used to provide statistics about crime (e.g., total of violent crimes by location or period). The Public Security Secretary (SESP) is responsible for imputing data into PC4SEG and maintaining this information system. In order to understand the SESP domain we developed EA models. Figure 4 presents the “as-is” EA model referring to the current statistics development process performed by the SESP.

As presented in 4, the statistics development process is composed of six activities (performed by a SESP agent):

Collect Data about Police Report: the agent collects data about a police record (i.e., data about an incident - e.g., a dead body), based on a request of police incident answered by a police officer, which aggregates several information, such as, location, time, possible victim, possible offender, among others.

Collect Data about Crime Investigation: the agent collects data about a police investigation, based on a police inquest developed by a police chief officer, which aggregates information about the

incident, more refined than police record.

Collect Data about Victims: the agent collects data about possible victims. These data can be a physical description, if the victim has been no identified, or a civil registry, if the victim has been identified. Moreover, in case of a fatal victim, the Scientific Police performs an autopsy, to determinate the death cause, and produces a death report.

Collect Data about Offenders: the agent collects data about possible offenders. Similar to data about victims, data about an offender can be a physical description or a civil registry. Moreover, whether the offender has been identified, his criminal history is annexed with the data.

Complete Technical Report: the agent describes all data collected on the previous activities in a Technical Report.

Present Statistics: based on the technical report developed on the previous activity, the agent presents statistics to the government high-level executives.

The EA models helped us to identify problems and suggest solutions about the statistic development process and the information systems that support it. For example, the PC4SEG system provides support only to activities of technical report management (“Complete Technical Report” and “Present Statistics” in 4) on the statistics development process, i.e., the PC4SEG stores only monthly reports containing crime statistics focused on violent crimes using data refined throughout the process by other applications. To collect and record data about Police Reports, Crime Investigations, Victims and Offenders, the agent uses a Spreadsheet Editor (e.g., Excell or Calc). The agent collects data from other information systems and records them in a spreadsheet. He/she evaluates data integrity and accuracy and, then, inputs them into the PC4SEG. The need to manually collect data from other systems, organize them in spreadsheets and input into PS4SEG makes the statistic development difficult and effort-demanding.

In the Figure 4, the “SISP” application is a system that maintains data about each person in the country, collected from the general registry (in Brazil is the RG). In order to identify a victim or an offender, the agent searches the civil registry in SISP. Once the registry is found, a pdf file is presented to the agent and he/she needs to manually copy each piece of information to a spreadsheet.

It is worth mentioning that the PC4SEG, the SISP and other information systems that store information about crimes are all part of a network of systems of the SESP and share a common database. Therefore, the PC4SEG could access data about crime to develop the statistics, if such functionalities data were developed. This all reveals that, despite the PC4SEG to be considered the system to develop statistics at SESP, the lack of functionalities to help the agent on statics development reduces PC4SEG basically to a database to store statistic data.

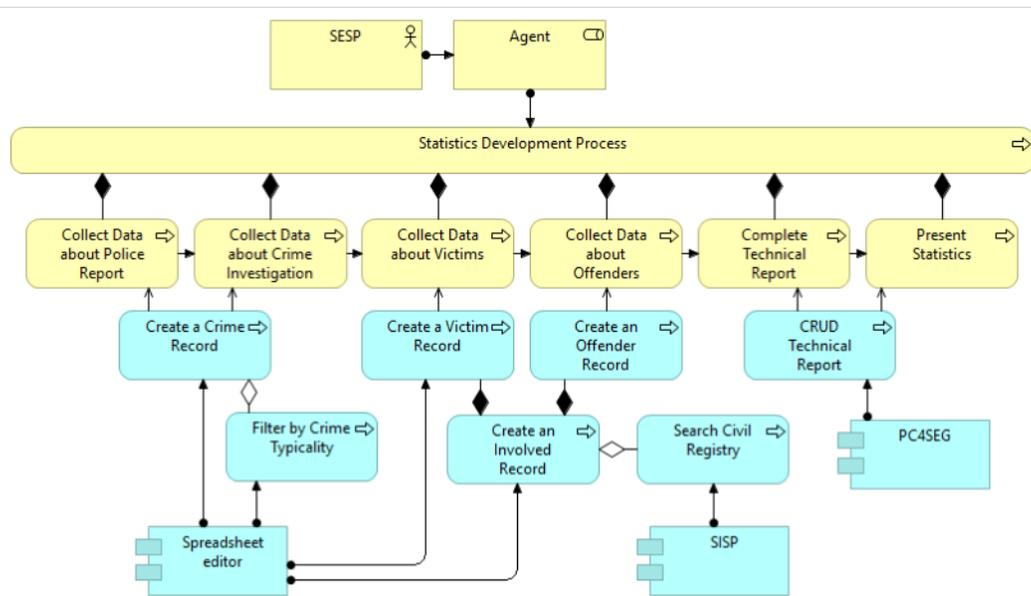


Figure 4: “AS-IS” EA Model of Statistic Development Process

After developing the “as-is” EA model of the statistic development process and identifying problems and possible improvements in the PC4SEG, we developed a “to-be” EA model to represent an architecture that could improve the current scenario. Figure 5 presents such model.

As the figure shows, ideally, the PC4SEG information system should provide support to all activities of the statistics development process, i.e., each business process that composes the statistic development process should be supported by some application process related to the PC4SEG. In other words, if the application processes are implemented as PC4SEG functionalities, it will be able to support all activities of the statistics development process, avoiding the need of agents to deal with a lot of different systems and spreadsheets.

Once structured the “to-be” EA model, we were able to define a use case diagram following the CEA approach. The result is shown in Figure 6, which presents the performed mappings:

- The business role “Agent” was represented as a use case actor (M3).
- Each application process was represented as a use case (M1).
- The composition relation between the application process “Create an Involved Record” and “Create a Victim Record”, and between “Create an Involved Record” and “Create an Offender Record”, were represented as include relations (M2).
- The aggregation relation between the application process “Create a Crime Record” and “Filter by Crime Typicality”, between “Create an Involved Record” and “Search Civil Registry”, and between “CRUD Technical Report” and “Filter by Period”, was represented as extend relations (M2).

The resulting use case diagram provides a systemic view of the functionalities PC4SEG should provide. This model has been

used by SESP software engineers to improve the PC4SEG aiming at providing a new version able to address the agents needs and provide a better support to the statistics development process.

5 EXPERIMENTAL STUDY

In the project described in Section 4, CEA was used by its own authors. Aiming to evaluate the use of this approach by other people, we carried out an experimental study. Experimental studies have been used in Software Engineering to find evidence and improve the use of techniques in software projects [25].

5.1 Study Planning

The *goal* of the study was to evaluate the support provided by the use of CEA in use cases modeling. According to the GQM approach [4], this goal is formalized as follows: *analyze CEA with the purpose of evaluating the support provided to define use cases with respect to CEA usefulness, CEA ease of use and adequacy of use case diagrams produced by using CEA from the point of view of requirement engineers in the context of software development.*

The following *indicators* were defined to analyze the results: (i) percentage of participants that would use CEA in future projects; (ii) percentage of participants who found easier to construct use case diagrams using CEA than the traditional approach; (iii) adequacy score in the use case diagrams.

The *instrumentation* used in the study consisted of two forms and two documents: (i) a form to characterize the participants’ profile, including questions about the participants’ knowledge of UML use case diagrams and Enterprise Architecture modeling using ArchiMate; (ii) a document with a brief description of the ArchiMate

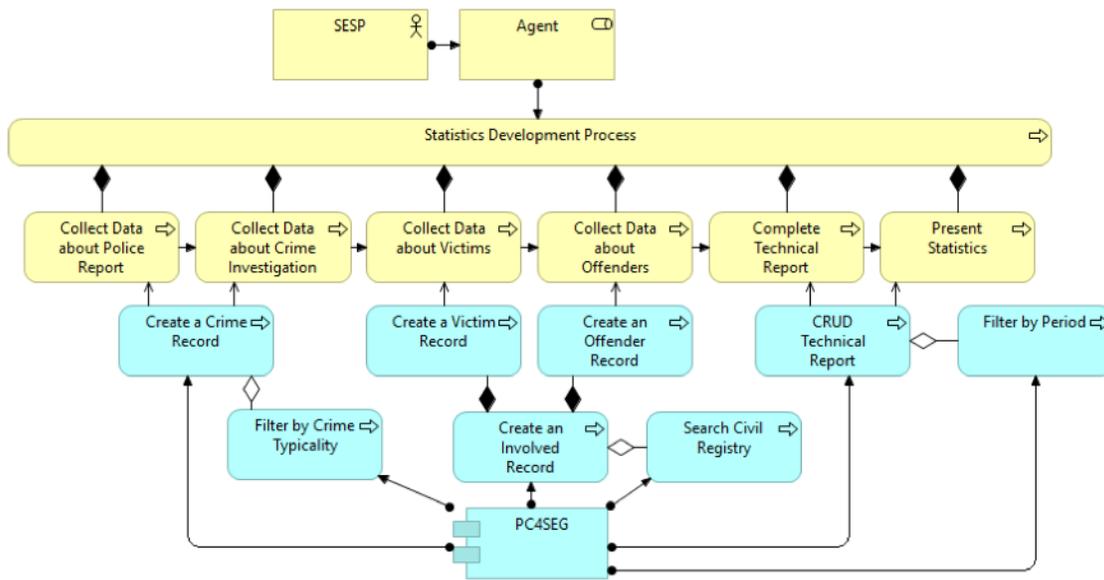


Figure 5: “TO-BE” EA Model of Statistic Development Process

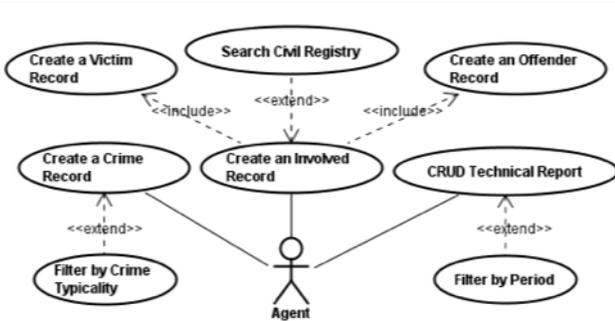


Figure 6: Use Case Diagram of PC4SEG

and use case elements considered in the study, and the CEA specification; (iii) a document presenting the scenarios to be considered in the study and instructions on how participants should proceed; (iv) a form with questions that allowed participants to record their perceptions after using CEA.

The *procedure* adopted in the study comprised three phases. In the first phase, the authors gave a brief presentation about UML use case diagrams and EA modeling using ArchiMate, and then they presented CEA. In the second phase, the participants received the forms and documents cited above and performed the activity described in (iii). The activity consisted in the elaboration of use case diagrams for hypothetical softwares in two different domains. In one case, the participants elaborated use case diagrams based on a textual description of the domain. In the other case, the participants used CEA to elaborate a use case diagram based on EA models. The participants were divided into two groups, A and B. Participants in group A received a textual description of domain 1

(bandwidth enterprise) and an EA model of domain 2 (games rental). Participants in group B received an EA model of domain 1 and a textual description of domain 2. Participants had about an hour and a half to elaborate the use case diagram for each domain. In the third phase of the study, the participants answered the feedback questionnaire (iv).

The *questionnaire* contained questions about the understandability of the domain description (both textual and visual) and difficulties of the usage of CEA. Moreover, the questionnaire contained questions about which approach (domain description or EA models) participants considered easier to use and which one(s) they would use in future projects.

The *participants* in the study were 21 undergraduate students in Computer Engineering and 12 graduate students in Computer Science. To obtain information about the participants’ profile, they answered questions about their knowledge level (low, medium, high) of both EA modeling and UML use case diagram modeling. Only one participant declared (medium) experience in EA modeling. 18 participants stated that they had low knowledge of UML (less than one year), while six participants stated they had an average knowledge of UML (one to two years) and six declared a high level of knowledge of UML (more than two years).

5.2 Data Synthesis and Analysis

In order to evaluate CEA usefulness, we asked participants to indicate if they would use CEA or the traditional approach (using textual domain descriptions) to define use cases in future projects. 13 participants (39.1%) said that would use CEA, one participant (3.1%) would use domain description, 18 participants (54.8%), would use both, domain description and CEA, and one participant (3%)

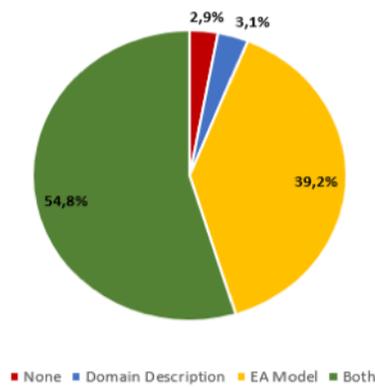


Figure 7: Approach that participants would use in future projects

would not use any of the approaches. Thus, 93.9% of the participants would use CEA in future projects. Figure 7 illustrates these results.

CEA usefulness is also influenced by the quality of the use case diagrams produced by using the approach. To evaluate the use case diagrams adequacy, the diagrams elaborated by the participants were analyzed against a proof template. Each use case equivalent to a use case in the proof template counted one correct point. The absence of a use case considered mandatory did not count any point. Each unnecessary or incorrect use case counted one point of error. The ratio between the number of correct points and the number of points of error provided the adequacy score.

The adequacy scores were combined in different ways to allow for different comparisons between the adequacy score of use case diagrams elaborated using domain description and use case diagrams elaborated using CEA. Figure 8 shows two different views of adequacy scores (see x-axis): (i) Average Score refers to the average score reached by the participants in each domain, and (ii) Completely Adequate refers to the rate of participants who produced diagrams without errors. With respect to (i), as the figure shows, the adequacy of the diagrams produced by using CEA (91.8% and 92.4%) is higher than the adequacy of the diagrams produced based on the domain description (57.7% and 64.1%). As for (ii) the rate of participants who were able to produce a completely correct diagram is higher in the cases in which CEA was used (81.3% and 76.5%) than in the cases where the traditional approach was used (12.5% and 17.7%).

For evaluating CEA ease of use, we asked the participants to inform if they found easier to develop the use case diagrams based on the domain description or using CEA. 28 participants (85%) found easier to develop use case diagrams using CEA. Five participants (15%) found easier to use the traditional approach.

For each question, participants were asked to justify their answers. We have got some perceptions from these justifications. We noticed that most participants who answered that they would still continue to use the traditional approach, would do so because they are familiar with this approach. The same type of justification was

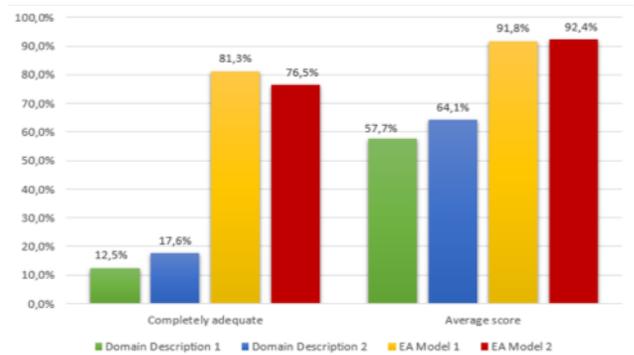


Figure 8: Adequacy Scores

presented by participants who said it was easier to use the traditional approach than CEA.

The participants also raised some advantages of using EA models instead of the traditional approach. Most of participants stated that EA models can facilitate the communication between client and developers. Since EA models are a representation with visual resources, they allow fast and clear communication, minimizing ambiguities. Additionally, many participants said that EA models are less wordy, thereby avoiding unnecessary information and also facilitating the differentiation between business-related processes and application processes.

Considering the indicators used to analyze collected data and the participant’s feedback, we can conclude that, CEA is useful, ease to use and the use case diagrams produced by using CEA tend to be adequate.

5.3 Validity Threats

Every study presents threats to the validity of its results. Threats should be addressed as much as possible and they should be considered together with the results obtained in the study. Next, we discuss some threats related to this study, following the classification defined by [26].

Internal validity: it is defined as the ability of a new study to repeat the behavior of the current study with the same participants and objects [3]. The main threat to internal validity is communication and sharing of information among the study participants. The study was applied within a classroom, where the participants were in the same physical space. Thus, there is a possibility of information exchanging between the participants. To address this threat, the authors informed the participants that communication was not allowed and were attentive to any form of communication between participants in order to prevent them from exchanging information.

External validity: this threat is related to the ability to repeat the same behavior with different participants [3]. The main threat is the homogeneous profile of the participants. It is possible that participants with different profile (e.g., experienced requirements

engineer professionals) produce different results.

Construct validity: refers to the relationship between the instruments and the participants of the study and the theory being tested [3]. It was observed that participants could give incorrect answers because of nervousness or pressure by imagining that they would be evaluated. In order to minimize this threat, participants were informed that the study does not represent any type of personal assessment. The anonymity of the responses was also ensured. Another threat refers to the fact that the same actor who was responsible for developing the EA models used in the study also participated in the validation of the models that were developed by the participants and, hence, a validation directed to the example could occur. Moreover, another threat concerns the documentation made available to the participants. We sought to produce a simple and small documentation aiming not to require much time from the participants. However, in doing so, information that could have an impact on the results may not have been included not be sufficiently clear. Moreover, documents were printed in grayscale. The ArchiMate language uses colors to create semantic differences between the elements. This fact may have confused the participants during the mappings between EA and use case elements, since it was not possible to use all the resources to differentiate the elements. For minimizing this threat, the participants were informed about the color of each element. Finally, there is also a threat related to the domains considered in the study. Due to the limited time available to perform the study, we used not complex domains (in fact, we consider a portions of the domains), so that the participants would be able to build the use case diagrams in the available time. Studies involving more complex domains may produce different results.

Conclusion Validity: it measures the relationship between the treatments and the results and affects the ability of the study to generate conclusions [3]. The main threats to conclusion validity are (i) the study participants were mostly undergraduate students, (ii) the study was performed in an academic environment, and (iii) low complexity of the domains considered in the study.

Considering the threats to the study results, they are not conclusive. Therefore, they should be considered as preliminary results and cannot be generalized.

6 RELATED WORKS

There are some works in the literature proposing the use of organizational models to define software requirements. Next, we present some of them.

Santander and Castro [21] define guidelines that can help the development of UML use cases based on organization business models described by using the iStar (i*) framework [13]. These guidelines support the identification of actors from organizational models and discovery of scenarios to possible use cases.

Dias et al. [11] present an approach that allows transforming business process models, using a free tool named RAPIDS (Rules

And Process for the Development of Information Systems) into requirements models represented as UML use cases diagrams and class diagrams.

Demirörs et al. [10] carried out an experimental study in a real case scenario to investigate a requirements elicitation approach based on business process models, particularly on goals of business process models. The authors concluded that process-oriented approaches help organizations and stakeholders to see the big picture of the system context, hence improving the accuracy of elicited requirements.

Monsalve et al. [17] use BPM (Business Process Modeling) to support the requirements engineering process. The authors stress the importance of using process models to ensure homogeneous understanding by all stakeholders. The authors carried out a research in a Canadian organization about the use of BPM at multiple levels of abstraction to explore the usefulness of the strategic level.

Cardoso et al. [5] report a study in which business process was used as a starting point to derive alternative sets of requirements for a process-oriented system. They present a case study involving the development of a real system to manage processes in a Human Resources Department of a large organization. They compared the use of “conventional” requirements engineering and the requirements specifications derived from the business process model and it was noticed the approach has led to a more complete, correct and traceable requirements specification.

It can be noticed that in all cited related works the authors emphasized the importance of using business process models to produce more precise requirements models. In our study, we go beyond, focusing not only on business process, but on the organizational environment as a whole. Different from the cited works, we propose the use of EA models, which provide a more comprehensive view of the organizational environment, structuring in in different layers and explicitly connecting IT and infrastructure resources to business processes. EA models are more aligned with the reality of stakeholders. Thus they can aid in reducing the risk of multiple interpretations of software requirements.

7 CONCLUSION AND FUTURE WORKS

Requirements Engineering is a crucial process in software development. In order to properly identify the requirements a software must meet, requirements engineers should understand the organizational environment where the software will be used. Enterprise architectural models can help in this matter. EA models provide a general view of an organization by structuring it in layers. Understand how different layers connect help understand the needs the software must meet.

In this paper, we presented an approach to define use cases based on EA models, named CEA (use Cases definition oriented by Enterprise Architecture modeling). CEA focuses on the analysis of application and business processes, proving three mapping rules to develop a use cases diagram based on information about the

organizational environment. Since CEA takes into account business needs, the use cases defined tend to reflect these business needs and thus, improve the software development. The main contributions of this paper are (i) the use of EA models as a means to understand the application domain and aid in use case definition and (ii) CEA.

CEA was applied in a real-world e-Government project involving information systems related to the Public Security. By using EA models, we produced use cases representing the functionalities the information systems should provide to properly support the organization needs.

CEA was also used in an experimental study in which we investigated if, in contrast with textual representations, architectural descriptions represented as EA models provide a better support to requirements engineers in the definition of use cases reflecting the business needs. The main purpose of the experimental study was to evaluate CEA usefulness, ease of use and efficacy (by means of the adequacy of use case diagrams produced by using CEA).

The obtained results showed that CEA facilitated the development of use case diagrams. Moreover, the correctness rate was higher in cases where CEA was used, revealing that use case diagrams produced using the approach were more correct and similar (large deviations in accuracy were visible only in diagrams produced using textual descriptions). As main advantage of using CEA, the participants pointed out ease and speed in communication, since the visual description is more direct and easily understood. The study participants had knowledge of use cases, but not of EA models. Even so, the results were favorable, showing that even less experienced requirements engineers could benefit from using CEA.

It worth to mentioned that a more complete study would present only a textual description to the participants, who would be responsible to develop the EA model. However, we did not have participants with the needed experience to develop such models (among all participants, only one stated experience with EA). Therefore, since the study do not includes the elaboration of EA models, we have been performing complementary validations in real projects, concerning from the elaboration of EA models to their adoption in use cases definition, as presented in the case study of section 4.

Regarding future works, our intention is to improve CEA, identifying new mapping rules involving other business layers and elements. Moreover, we see an opportunity to link the ArchiMate motivation and business views. Analyzing goals together with business could provide new software requirements. Lastly, we have a proposal to implement a tool to perform automatic transformations of EA models to use case diagrams based on the mapping rules defined in CEA. Such tool could expedite the application of CEA in requirements engineering process.

ACKNOWLEDGMENTS

This research is funded by the Brazilian Research Funding Agency CNPq (Process 407235/2017-5), CAPES (Process 23038.028816/2016-41), and FAPES (Process 69382549/2014).

REFERENCES

- [1] [n. d.]. Archimate 3.0 Specification. <http://www.opengroup.org/subjectareas/enterprise/archimate/>. Accessed: 2018-06-29.
- [2] [n. d.]. Programa de Cadastro e Consulta de Crimes Contra a Vida da Seguradora Pública. <http://pc4seg.sisp.es.gov.br/homicidio-war/xhtml/estatisticaHomicidios.jsf,urldate={2010-06-29}>
- [3] MO Barros, CML Werner, and GH Travassos. 2002. Um Estudo experimental sobre a Utilização de Modelagem e Simulação no Apoio à Gerência de Projetos de Software. In *Proceedings of the XVI Brazilian Conference of Software Engineering*.
- [4] Victor Basili, Gianluigi Caldiera, and H Dieter Rombach. 1994. *Encyclopedia of Software Engineering*. (1994).
- [5] Evellin Cristine Souza Cardoso, João Paulo A Almeida, and Giancarlo Guizzardi. 2009. Requirements engineering based on business process models: A case study. In *Enterprise Distributed Object Computing Conference Workshops, 2009. EDOCW 2009. 13th. IEEE*, 320–327.
- [6] Lima R. S. Bueno S. Valencia L. I. Hanashiro O. Machado P. H. G. Lima A. S. Cerqueira, D. 2017. Atlas da Violência. http://www.ipea.gov.br/portal/images/170602_atlas_da_violencia_2017.pdf
- [7] Jose Luis de la Vara and Juan Sánchez. 2008. Improving requirements analysis through business process modelling: A participative approach. In *International Conference on Business Information Systems*. Springer, 165–176.
- [8] Jose Luis de la Vara, Juan Sánchez, and Oscar Pastor. 2008. Business process modelling and purpose analysis for requirements analysis of information systems. In *International Conference on Advanced Information Systems Engineering*. Springer, 213–227.
- [9] Jose Luis De La Vara, Krzysztof Wnuk, Richard Berntsson-Svensson, Juan Sánchez, and Björn Regnell. 2011. An Empirical Study on the Importance of Quality Requirements in Industry. In *SEKE*. 438–443.
- [10] Onur Demirörs, Çigdem Gencel, and Ayça Tarhan. 2003. Utilizing Business Process Models for Requirements Elicitation. In *EUROMICRO*. 409–412.
- [11] Felipe Dias, Gisele Morgado, Pedro Oscar, Denis Silva da Silveira, Antonio Juarez Alencar, Priscila Lima, and Eber A Schmitz. 2006. Uma Abordagem para a Transformação Automática do Modelo de Negócio em Modelo de Requisitos.. In *WER*. 51–60.
- [12] CIO DoD. 2012. DoD Architecture Framework Version 2.02. *DoD Deputy Chief Information Officer, Available online at http://dodcio.defense.gov/dodaf20/dodaf20_pes.aspx, accessed Nov (2012)*.
- [13] S Yu Eric. 2009. Social Modeling and i*, Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos.
- [14] Ivar Jacobson. 1993. *Object-oriented software engineering: a use case driven approach*. Pearson Education India.
- [15] Henk Jonkers, Marc Lankhorst, Rene Van Buuren, Stijn Hoppenbrouwers, Marcello Bonsangue, and Leendert Van Der Torre. 2004. Concepts for modeling enterprise architectures. *International Journal of Cooperative Information Systems* 13, 03 (2004), 257–287.
- [16] Marc Lankhorst. 2009. *Enterprise architecture at work: Modelling, communication and analysis*. Springer.
- [17] Carlos Monsalve, Alain April, and Alain Abran. 2011. Requirements elicitation using BPM notations: focusing on the strategic level representation. *ACACOS 11 (2011)*, 235–241.
- [18] SID de PÁDUA, Edson Walmir Cazarini, and Ricardo Yassushi Inamasu. 2004. Modelagem organizacional: captura dos requisitos organizacionais no desenvolvimento de sistemas de informação. *Gestão & Produção* 11, 2 (2004), 197–209.
- [19] Shari Lawrence Pfleeger and Joanne M Atlee. 1998. *Software engineering: theory and practice*. Pearson Education India.
- [20] James Rumbaugh, Ivar Jacobson, and Grady Booch. 2004. *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education.
- [21] Victor FA Santander and Jaelson FB Castro. 2002. Deriving use cases from organizational modeling. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*. IEEE, 32–39.
- [22] Avik Sinha and Amit Paradkar. 2010. Use cases to process specifications in business process modeling notation. In *Web Services (ICWS), 2010 IEEE International Conference on*. IEEE, 473–480.
- [23] Ian Sommerville. 2008. Construction by configuration: Challenges for software engineering research and practice. In *Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on*. IEEE, 3–12.
- [24] Richard H Thayer, Sidney C Bailin, and M Dorfman. 1997. *Software requirements engineerings*. IEEE Computer Society Press.
- [25] Guilherme Horta Travassos, Dmytro Gurov, and EAGG Amaral. 2002. *Introdução à engenharia de software experimental*. UFRJ.
- [26] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.