Designing a reference architecture for a collaborative software production and learning environment

Juan Felipe Restrepo Naranjo
dept. de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
São Paulo, Brazil
felipe.restrepo@usp.br

Ana Claudia Rossi
Dept. de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
São Paulo, Brasil
ana.rossi@usp.br

Solange Nice Alvez-Souza
Dept. de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
São Paulo, Brasil
ssouza@usp.br

Jorge Luis Risco Becerra
Dept. de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
São Paulo, Brasil
jorge.becerra@usp.br

***Abstract*—Software engineering is both a body of knowledge and a competence, thus its teaching requires students to get involve in actual software development. For this purpose, a teaching-learning controlled software production environment can be designed. Typically, this environment has been designed as a learning software factory whit focus on the processes, which may lead to a narrowed understanding of the information systems technology required for its successful implementation, operation, and management. We present an ongoing design science research aimed to improve the design of modern Collaborative Software Production Teaching-Learning Environments (CSPLE) by using a set of models to describe not only its processes, but also the information generated by the processes, the software applications used to support the processes execution and the technological infrastructure required to run the applications. These models describing a CSPLE correspond to a set of architectural views compounding an enterprise architecture that can be used as a reference to design specific CSPLE in the context of a practical software engineering course. A preliminary result presented in this paper is a method to design an Enterprise Reference Architecture (ERA) for a CSPLE. The proposed method is also applied giving as empirical highlights about the design of an ERA for a CSPLE.**

Keywords— enterprise reference architecture, learning software factory, design science research, RM-ODP

I. INTRODUCTION

Software engineering is both a body of knowledge and a competence, so its teaching is split into theoretical and practical lessons [1]. In this context, the software factory, which emerged as an industry initiative aimed to gain productivity and quality in software [2], is also used as a controlled teaching-learning environment for practical lessons [3] [4].

For instance, a learning software factory has been used at the "Software engineering lab II" (a.k.a. PCS3853[1]), in which senior computer engineering students apply their knowledge and train technical and social skills while getting involved in actual software development. For each offering of the PCS3853 lab, the teaching team designs an updated version of the learning software factory using historical data from the lab's operations and considering current both technical (related with software production) and educational (related with teaching-learning-evaluation) requirements.

[1] https://uspdigital.usp.br/jupiterweb/obterDisciplina?sgldis=pcs3853

In "software process improvement" and "software process engineering" literature, the software factory design focuses on process definition and modeling [5]. The learning software factory used at PCS3853 lab, as well as other described in literature [2] [3] [6], has been designed with on-process-focus and thus are described by a set of process models.

However, the technical and educational requirements of a learning software factory refer not only to processes but to the information generated by its processes, to the software applications supporting the processes execution and to the technological infrastructure (hardware, software, and network) required for running the applications that supports the processes [7] [8]. For example, a modern learning software factory should be designed considering the software process automation discussed in DevOps [9], the use of cloud computing in the software development processes [10], and the latest information technology for the teaching-learning process [11] [12].

Therefore, describing a learning software factory focusing on the processes can lead to a narrowed understanding of the information systems and technology required for its successful implementation, operation, and management. A more complete understanding of the software factory can be achieved by integrating its process description with the description of its organizational structure, processes information, software applications, and technological infrastructure [7] [8] [13].

That kind of enterprise descriptions considering viewpoints beyond the process is also known as enterprise architecture [13]. When an enterprise architecture is intended to describe a generic solution for the parts of a class of enterprise, it is known as Enterprise Reference Architecture (ERA) [14] [15]. ERAs are largely used to facilitate the design of concrete architectures for an enterprise class or a domain, e.g., e-commerce [16], customer relationship management [17], banks [18] [19], telecommunication [20], financial [21] and utilities [22].

The ongoing research is twofold aimed: 1) to improve the software factory understanding, previously designed focusing on processes, by designing it as a Collaborative Software Production and Learning Environment (CSPLE) which is described by a set of architectural views composing an enterprise architecture; and 2) to model an ERA for CSPLE so its models can be used to guide and facilitate the design of concrete modern CSPLEs. A research's preliminary result registered in this paper is the design and application of a method to design an ERA for a CSPLE in the specific domain

of an under-graduation software engineering lab. The rest of this paper is organized as follows. Section II presents the theoretical background of the learning software factory and enterprise reference architecture. Section III explains how is in here applied the design science research framework proposed [23]. In section IV is proposed a method to design an ERA which is then applied in section V to design an ERA for a CSPLE used at PCS3853 lab. Finally, section VI resumes initial findings from the ongoing research.

II. THEORETICAL BACKGROUND

A. The software factory in education

A software factory is a software production environment defined from an organization's business requirements with the aim to gain productivity and quality in software development [24]. The basic condition for obtaining these gains is the existence of defined processes and information systems allowing an effective operation and control of the software production and software quality assurance [3].

The software factory is used in research as a test bed for software development methods and technologies, in industrial production as a mean to create marketable software products, and in education as a practical lessons environment in which students solve software engineering problems such as work planning or quality attributes treatment [4]. The basic format of practical lessons carried out in a learning software factory is also studied as project or problem-based learning and requires students to understand the software functioning and the process applied by engineers to develop proper solutions [25] [26].

Two core characteristics of a learning software factory are [3] [27] [28] [29]: 1) defined processes and information systems considering both technical and managerial activities, which smooths the understanding of what needs to be done in software projects and facilitates teachers to guide inexperienced students in solving problems with relevant complexity; 2) students create, share, and apply their knowledge in controlled environments, which allows the teachers to follow up each student learning progress and to reuse experience from previous projects.

B. From the learning software factory toward the CSPLE

Whether the software factory is used for research, industrial or educational purposes, its products come out of the joint effort of people with different roles. In other words, regardless the use of a software factory it is needed a joint effort to reach an objective not reachable by individuals, and this is the essence of collaboration. Hence, a software factory should be designed considering four key characteristics of any collaborative environment: 1) information exchange (communication); 2) activities coordination (planning); 3) group memory (knowledge management); and 4) group awareness (defined roles and interactions) [30] [31].

In this paper, a controlled learning software factory with defined processes and information systems allowing students to create, share, and apply their knowledge complying the aforementioned four collaboration characteristics, and also being described by a set of enterprise models, is hereinafter referred to as a Collaborative Software Production and Learning Environment (CSPLE). The CSPLE is modeled as an enterprise by follow the integrated software factory model [32] [8], through which a software production organization is described by its processes, information, applications, and technological infrastructure in an integrated manner. This model combines the strategic, tactic and operational hierarchical levels; the organization departments (specialties); and the enterprise, information, computation, engineering, and technological viewpoints suggested by the Reference Model of Open Distributed Processing (RM-ODP) [33].

Furthermore, a CSPLE should be designed in such a way that students train a set of knowledge, technical and soft skills related with both technical (software production methods and technologies) and educational (student profile and the teaching-learning-evaluation methods and technologies) requirements [27] [11, p. 23]. Moreover, the CSPLE design applies the enterprise architecture concept discussed next.

C. Enterprise architecture

An enterprise is one or more organizations sharing a defined mission, objectives, and goals stablished to offer some value by means of products or services [34]. In other words, an enterprise is a system with arranged components to perform one, some or all the functions associated with their offered products and services lifecycles [35]. Then, a CSPLE is an enterprise missioned to enabling software engineering students to learn as they collaborate developing a software product, and the CSPLE processes and technology must be designed for it.

Enterprise design involves describe it by a set of models abstracting the structure and behavior of their business processes and technology, their relationships and decompositions and detailing to the extent necessary to convey how the enterprise should operates to accomplish its mission and objectives [34]. In this context, enterprise architecture is understood as a set of representations (each relevant to a target audience) describing an enterprise so it can be produced and maintained throughout its lifespan [36] and by doing this it is possible to systematically ensure that organization strategy, processes, and technology are all aligned.

However, the enterprise architecture is not just about alignment and is becoming an engineering discipline in which the enterprise is approached as a system that can be designed and adapted in a systematic way, like in civil engineering or software engineering [21] [37, p. 3301] [20, p. 61] [13, p. 12]. A more up-to-date definition states that "*enterprise architecture is a coherent set of principles, methods, and models that are used in the design and realization of the organizational structure, business processes, information systems, and infrastructure*" [13]. So, the enterprise architecture can be interpreted either as a product or a process [15].

D. Enterprise Reference Architecture (ERA)

Enterprise architecture, interpreted as a product, is a set of work products expressing an enterprise-system architecture, a.k.a. enterprise architectural description [38] [15]. Models in an enterprise architectural description are characterized by their level of aggregation, abstraction, and realization, and describe

at least one architectural view using a notation with a given formalism level. Depending on these characteristics values, an enterprise architecture can be referred to as a reference or concrete enterprise architecture [15, p. 21] [39].

An Enterprise Reference Architecture (ERA) is a generic solution model for the parts of a class of enterprise or domain. An ERA includes principles, policies, architectural views, requirements, ontologies, standards, conceptual reference models and/or guidelines for designing enterprise concrete architectures; i.e., an ERA is an enterprise architecture with some architectural decision already made and others left open [14] [40]. In other words, an ERA generalizes solutions by abstracting and aggregating the available knowledge about a specific class of enterprise or domain in order to improve the quality and effectiveness of the architect's work [15] [21] [41].

E. Enterprise reference architecture characteristics

ERA's aggregation and abstraction levels define the detail offered by its models. Aggregation defines the number of architectural elements described while the abstraction defines the number of attributes used to describe each architectural element. ERA's models describe the main architectural elements using minimal attributes (i.e. abstracted and aggregated) so they can be reused in the design of multiple concrete architectures [42].

ERA's realization level defines whether the models focus on business or technology [15]. ERA's models describe both business and technology, nevertheless, a given focus will depend on the current phase of the organization's lifecycle and on how much the technology support the business execution for a specific enterprise class described by the ERA. ERA's architectural views define conventions for model construction, interpretation, and use. Each ERA's view frames specific concerns about the organization being described [38]. Examples of architectural views used in enterprise architecture are found in Zachman [36], TOGAF [43] and RM-ODP [33] [27] [44] [45] architectural frameworks. ERA's notation formalism level defines the breadth of concepts and syntax used to modeling. It can be used whether a set of general graphs (informal), a generic notation with standardized graphs and vocabulary (semi-formal), or a robust notation allowing models compilation and simulation (formal) [39]. Enterprise architecture descriptions generally use semi-formal notations such as Archimate [46], UML or UML4ODP [47].

The level of aggregation, abstraction, and realization, as well as the used architectural views and notation formalism are related with ERA modeling. Yet, an ERA is also characterized by its objective and usage context. ERA's objective could be whether standardize or facilitate the design of concrete architectures. Prescriptive architectures are used in the former case whereas descriptive architectures are used in the last one [15]. ERA's context defines who design it, when it is designed and who uses it; an ERA can be designed whether by a standardization organization, an independent organization, a research group, or an enterprise stakeholders group; further, an ERA can be designed to describe whether an existing or a foreseen enterprise solution; and the ERA's models can be intended to be used whether by some single or multiple organizations [42].

F. Enterprise reference architecture design

Enterprise architecture, interpreted as a process, is a set of "recipes" used by architects in enterprise design and realization [15]. ERA design "recipes" use three main "ingredients" [20]: 1) a structure reference model guiding the identification, organization, and description of the enterprise elements types, 2) one or many content reference models enlightening about an enterprise class or domain processes, information, applications, and infrastructure, and 3) methods for ERA design and use.

Two ERA design approaches are noticed in ERA design "recipes" depending on the abstraction level of the content model "ingredient". In a top-down approach, one or many abstract and aggregated content models are adapted according to an enterprise class or domain requirements, whereas in down-top approach many concrete and detailed content models are abstracted into a single generic solution [15].

Top-down ERA design approach is found in [20]. These authors used TOGAF as structure reference model and took the eTOM, SID and TAM models from Framewox [48] as content reference models and adapted them to propose a set of generic enterprise architecture solutions for the current telecommunication operator's challenges. Otherwise, the down-top ERA design approach is found in [17] for the customer management domain and in [22] for public facility providers. In both cases, the practical knowledge about an enterprise class or domain, which is available in concrete solutions, specialist professionals and the literature, was used as content reference model

Regardless the selected approach, whether it is a top-down, down-top or a hybrid of the first two, an ERA design recipe involves describing architectural views of an enterprise class or domain and transforming architectural models from an initial level of aggregation, abstraction, and realization to another desired level. An architecture method must guide this model transformation [15, p. 42]. A generic method to design an ERA for any enterprise class is proposed by [49].

III. RESEARCH METHODOLOGY

Applying the [23] Design Science Research framework in this ongoing research project is here shortened to the identification of the research context, artifact, and treatment. The research context is split into social and knowledge context. The social context is the under-graduation PCS3853 lab and the knowledge context is the theoretical background covering the structure and content models as well as the methods needed to design and use an ERA. The research artifact is an ERA for a CSPLE (hereinafter referred to as an ERA4CSPLE) and the treatment is the instantiation of the ERA4CSPLE in the design and operation of a specific CSPLE for a PCS3853 lab offering. Fig. 1 represents graphically the so-called research conceptual model and highlights with gray-background the research artifacts still under development and treated in oncoming papers.

This research is inspired by solving a real-world problem and its three main phases are: problem investigation, treatment design and treatment validation. This paper presents the results of the full first and partial second phase. Validation phase is planned to be completed by applying a technical action

research still under development in which the proposed reference architecture will be instantiated for a specific PCS3853 lab's offering.

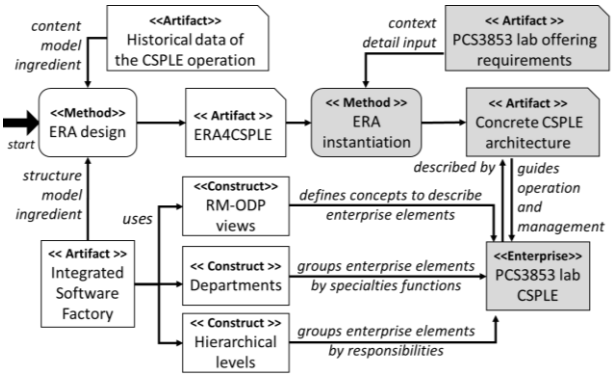

Fig. 1. Research conceptual model

According to [23], a research problem can be broken down into a set of related Analytical Knowledge Questions (AKQ), Empirical Knowledge Questions (EKQ) and Design Problems (DP). Fig. 2 presents a chronological sequence of this research AKQ, EKQ and DP while classify them by the research phase and points to its paper section discussing the answers.

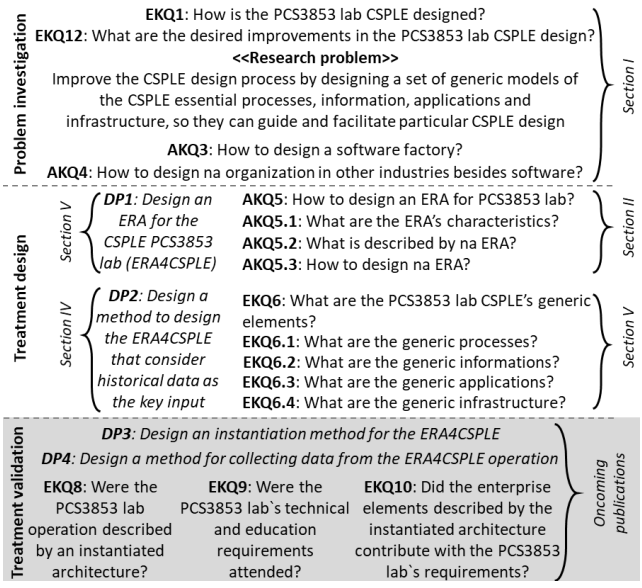

Fig. 2. Research problem broken into a set of design problems and knowledge questions

Early results of the ongoing research are presented as follows. Section IV presents a solution for the DP2: "*Design a method to design the ERA4CSPLE that consider historical data as the key input*". Since the DP2 method is analytically created from prior validated researches and isn't the main research artifact, its validity will be shortly discussed at the end of the section IV. Section V presents an ERA4CSPLE outputted from the proposed method solving the DP2. The ERA4CSPLE design method responds to the EKQ6: "*What are the CSPLE generic elements for the PCS3853 lab*?". The AKQ, EKQ and DP related with ERA4CSPLE validation will be answered in oncoming papers.

IV. ERA4CSPLE DESIGN METHOD

A. Overview of the ERA4CSPLE design method

The method phases were defined from [20]. Even an ERA design method is not stated in [20], it is presented a sequence of work products created out of the combination of conceptual ingredients (structure and content). This work products were mapped into activities and then grouped by the method phases.

Two conceptual ingredients were defined as initial constraints for the ERA4CSPLE method definition: 1) s*tructure reference model*: operational and tactical hierarchical levels, software productions and teaching-learning departments (or sub-domains as in [15] [17] [20]), and RM-ODP views, according to the integrated software factory model; and 2) c*ontent reference model*: historical data from the operation of the CSPLEs used at PCS3853 lab between 2012 and 2017. It includes process models, student-generated artifacts in software production and support-material produced and used in teaching-learning activities.

The four method phases are: 1) *Determining the architecture domain*: set of activities to determine the enterprise class of interest as well as the architecture envisioned goal and context; 2) *Determining the architecture structure*: set of activities to determine the formalism, aggregation, abstraction, and realization level of the architecture models. 3) *Abstracting the reference elements:* set of activities to abstract the roles, processes, information, applications, and infrastructure of the CSPLE of interests; e 4) *Describing the architectural views:* set of activities to describe the RM-ODP architectural views, named enterprise, information, computation, engineering, and technology views.

B. Definition of the ERA4CSPLE design method

The third phase, "abstracting the reference elements", and the forth phase, "describing the architectural views", results from mapping the description of the ERA proposed in [20] of the main characteristics of the ERA's class of enterprise of interest. This phase operation detail is complemented by the recommendations made in [42] to determine the architecture objective and context. The second method phase, "determining the architecture structure", results from mapping the architecture domains specification, architectural views selection and architectural models specifications made in [20]. Since using the RM-ODP architectural framework was an initial constrain, the related work in [20] is narrowed into explaining, according to [15] and [42] what are the levels of aggregation, abstraction, and realization of the models to be constructed.

The third phase, "abstracting the reference elements", and the forth phase, "describing the architectural views", results from mapping the description of the ERA proposed in [20]. We split this work product into two phases highlighting the required analysis on the content model to abstract the CSPLE's reference elements. The abstraction analysis uses a matrix (table I) adapted from [17] which is aimed to identify architectural elements repeated in same class system concrete architectures (historical data from CSPLE operation in our case). Repeated elements in the matrix are expected to outcome

from the instantiation of an ERA4CSPLE's reference element. Final method phase is about using the RM-ODP views to describe the already abstracted reference elements along with its relations and views correspondences. The suggested sequence for views description, according to [50], begins with enterprise view, follows with information and computation views, and ends with engineering and technology views.

C. Analytical validation of the ERA4CSPLE design method

The adopted strategy for the method analytical validation is its comparison with the generic ERA design method proposed by [49] and with the first five phases of the TOGAF ADM. Even though ADM is not explicit for ERA, it has become a "de facto" enterprise architecture framework and thus is here consider as a basic fair reference. The first phase of the proposed method achieves the definition of scope and desired capabilities of the enterprise of interest, which are goals in the "preliminary" and "architectural overview" phases of ADM. It also achieves the identification of the class of enterprise to be described by the ERA and the definitions of architecture goal, which are goals in the "project objective" phase of the [49]. The second phase of the proposed method absorbs the architectural framework selection in ADM "preliminary" phase and achieves the goals of the "modeling approach" phase in [49]. The "abstracting the reference elements" and "describing the architectural views" phases of the proposed method achieve the goal of the "reference modeling" phase in [49] which is achieved in ADM by the "business architecture", "information systems architecture", and "technology architecture" phases.

V. ERA4CSPLE

A. ERA4CSPLE domain

ERA4CSPLE's main objective is to facilitate the design of CSPLE concrete architectures. ERA4CSPLE's usage context is the PCS3853 lab theoretically described in section II. PCS3853 lab's objective is to train advanced both technical and managerial techniques to architect and develop distributed software-intensive system. Since the lab lasts 16 weeks (4 in-lab hrs/weeks), students only implement a minimal viable product. Yet, solution analysis and design are scoped in the systems context because a main educational requirement is teaching how to deal with problems like the treatment of quality attributes such as performance, availability, or security. In PCS3853 lab's operation, the students team them up into independent productive cells, each one responsible for architecting and developing parts of the system. Students are guided by scripts with recommendations on how to solve project problems and by support material explaining needed concepts.

B. ERA4CSPLE structure

ERA's aggregation and abstraction (detail) level: architectural views use only one aggregation level and two packages to group architectural elements: 1) tactical package and 2) operational package. Three sub-packages can further be used in operational package: 2.1) software production sub-package, 2.2) teaching-learning sub-package, and 2.3) collaboration sub-package. **ERA's notation formalism level:** UML4ODP [47] notation is used following the aforementioned package structure. **ERA's realization level:** processes, information and application are prioritized over technological infrastructure because PCS3853 lab should be open to use applications regardless the vendor whenever specified standards are meet.

C. ERA4CSPLE reference elements

The CSPLE roles are abstracted by RM-ODP community object composed by role objects. Table I exemplifies the use of the abstraction analysis matrix when applied for certain role. This matrix is similarly used for all the abstraction analysis. Community object is modeled in Fig. 3 of the enterprise view. The CSPLE processes are abstracted by RM-ODP process objects composed by step objects. Processes abstraction analysis was split in one matrix for each process object with the intention of abstracting also generic steps that can be used as recommendations for the CSPLE process instantiation. Process objects are modeled in Fig. 4 of the enterprise view. The CSPLE information is abstracted by RM-ODP information objects and the CSPLE applications are abstracted by computational objects. In both CSPLE information and CSPLE applications abstraction analysis, the earlier defined packages structures were used to group cohesive objects, which results in four abstraction analysis matrices for each view: 1) tactical objects, 2) operational software production objects, 3) operational teaching-learning objects and 3) operational collaboration objects. Information objects are modeled in Fig. 5 and computational objects are modeled in Fig. 6 in their respective architectural views. The CSPLE infrastructure is abstracted by RM-ODP engineering and technology objects, and its analysis is split in two matrices, one for each related view. Engineering objects are modeled in Fig. 7 and technology objects are modeled in Fig. 8 in their respective architectural views. Community object is modeled in Fig. 8 of the enterprise view.

TABLE I. ABSTRACTION ANALYSIS MATRIX FOR CSPLE ROLES

RM-ODP role object	Exists in PCS3853 20XX occurrence						Abstraction analysis results
	12	*13*	*14*	*15*	*16*	*17*	
Software engineer	**X**	**X**	**X**				Despite the name were different in ocurrences, these two roles represent the responsibility of the system analysis, design, implementation, test and deploy. Developer role is considered as the reference one becouse a software engineer has more duties related to other roles.
Developer				**X**	**X**	**X**	

D. ERA4CSPLE architectural description

Figs. 3 to 8 represent the ERA4CSPLE models and highlights with a dotted line the objects involved in the next general description of the PCS3853 lab CSPLE operation. PCS3853 lab starts by the "teacher" role executing the "general lab planning" and "CSPLE design" process objects. Next, for each lesson are executed all the processes in the "teaching-learning" package and, depending on the lesson objective, some processes from the "software production" package are invoked into "practical lesson execution" process. The "software production" package processes follow an architecture-centric approach started by the "architect" role

executing the “product engineering” process, which is aimed to define an initial system architecture.

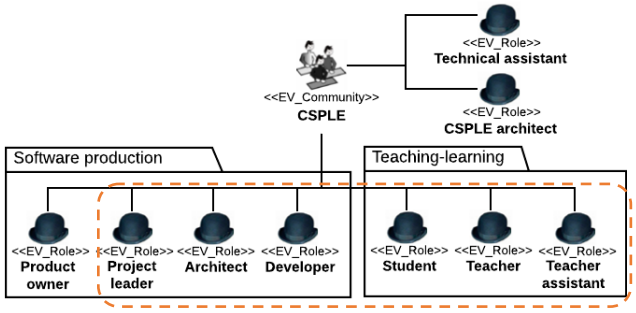

Fig. 3. Enterprise view: community model

According to the detailed educational requirements, the “architect” role can be whether partially or fully fulfilled by the person in the “student” role. Former case is more common and requires the person in the “teacher assistants” role to fulfill the “architect” role and define a system architecture’s first version that later will be detailed by the “student-architect”.

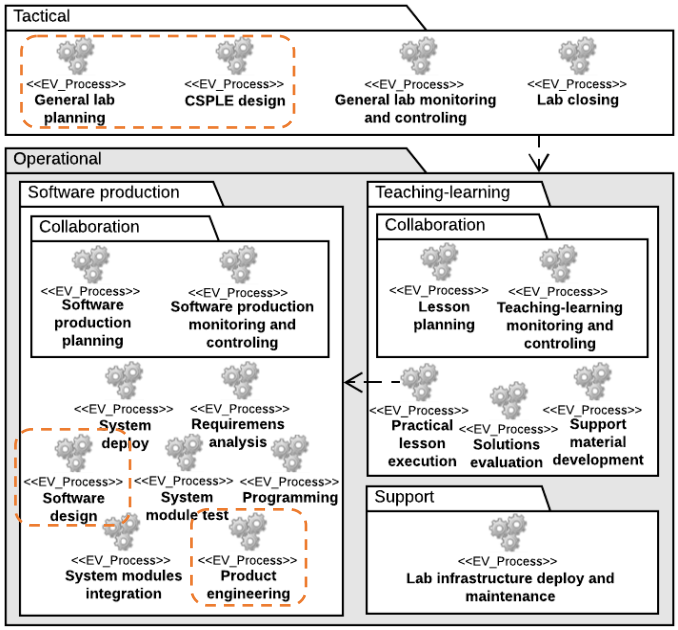

Fig. 4. Enteprise view: process objects packages

Then, the “product engineering” object process produces the “system architectural description” information object in which the main system modules/services are overviewed. Next, the “project leader” role executes the “software production planning” process object. Later, the person responsible for each system module fulfills the “developer” role and refines his/her module by executing the “software design” process object that produces the “module architectural description” information object. “Quality attributes” are treated by the “architect” role by defining scenarios and identifying “architectural tactics” addressing the solutions to be later implemented by the role “developer”. The information objects that are related with the system architecture and module architectures are managed by process objects being supported by the “UML editor”, “BPMN editor” and “Collaborative content edition” computation objects, which run on the “developer node” engineering object physically located in the “developer computer” technology object which is connected with the “ALM node”. A complete explanation of the ERA4CSPLE models along with the views correspondence is included in the full architectural description here reduced for the sake of space restrictions. To access the full ERA4CSPLE get in contact with the authors.

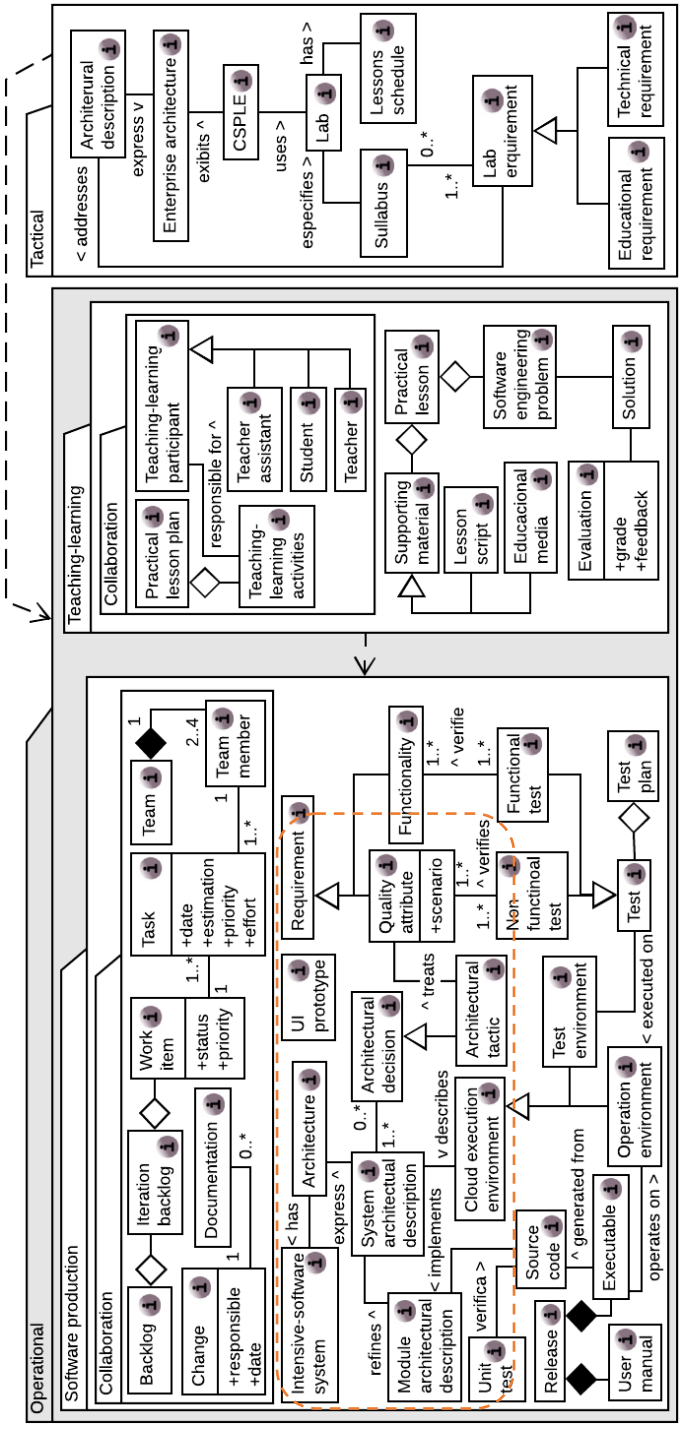

Fig. 5. Information view: information objects static schema

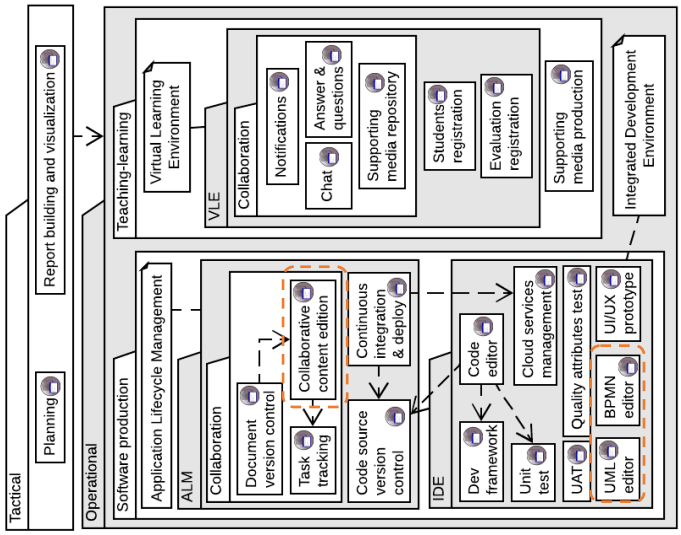

Fig. 6. Computation view: computation objects package

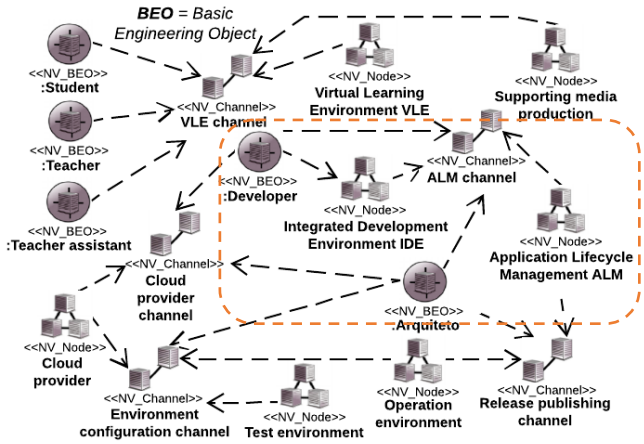

Fig. 7. Engineering view: engineering objects structure

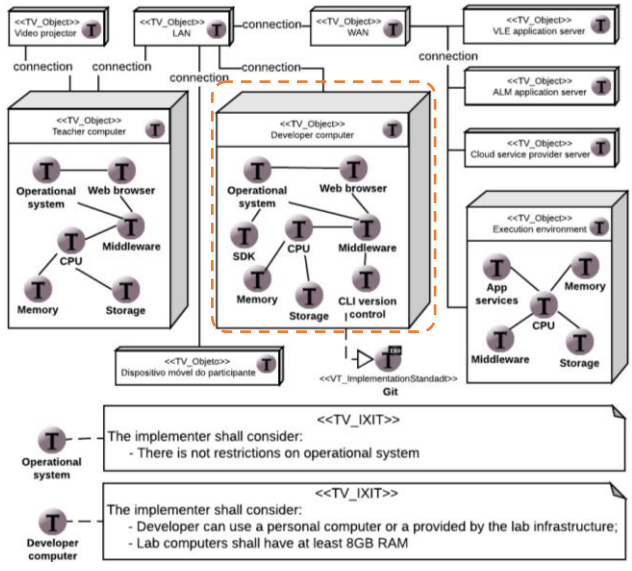

Fig. 8. Technology view: tecnology objects structure

VI. Conclusions

The ERA4CSPLE models presented here came out of four iterations between the ERA4CSPLE design method and informal revisions. The first version of the ERA4CSPLE was structured with three aggregation levels in which the models were enforced to detail and relate the architecture objects more than it was wanted and making the whole ERA4CSPLE look like a concrete architecture. Then, we heuristically realized that creating multiple aggregation levels in an ERA prompts to create over-detailed models. It was qualified as a risk in "plastering" the architecture, i.e., making it more "prescriptive-like" rather than "descriptive-like".

Thus, in a second version of the ERA4CSPLE, the one aggregation level and two packages structure here presented were determined. This structure can (and should for the sake of organization) be broken into a more specific set of consistent aggregation levels while the instantiation process. Nevertheless, the second version of the ERA4CSPLE was modeled strictly using the UML4ODP notation, which filled the architectural description models with icons unknown by the traditional UML user, making it needed to have a deeper understanding of both RM-ODP and UML4ODP. It was qualified as an "usability issue" because it is desirable that ERA4CSPLE models could be understood and applied by non-experts in RM-ODP. Thus, the information and computation views were remodeled in the third version using basic UML and letting UML4ODP icons smaller. A fourth iteration was needed to improve an usability issue remained in the engineering view in which the information systems were modeled considering three computational layers: presentation, business logic and data. Using these layers gave relevant information about the distribution of the computation objects but pollutes the model making it hard to understand so it was simplified into node relations.

The ERA4CSPLE validation, not yet detailed here because is still under development, will be fully presented in oncoming papers. Preliminary results of the validation (in the context of the PCS3853 lab executed in 2018) evidenced that ERA4CSPLE models were useful to structure the supporting material for the teaching-learning activities, and a future publication will discuss how the ERA4CSPLE models can be used to define an architecture of learning objects for a software engineering course.

References

[1] M. Jazayeri, "The education of a software engineer," em Proceedings of the 19th IEEE international conference on Automated software engineering, 2004.

[2] A. Araújo, K. Borges, S. Andrade, E. Dias and W. Pereira, "Experience and Innovation Factory: Adaptation of an Experience Factory Model for a Research and Development Laboratory," 2017.

[3] F. L. Siqueira, G. M. C. Barbarán e J. L. R. Becerra, "A Software Factory for Education in Software Engineering," em IEEE 21st Conference on Software Engineering Education and Training, 2008. CSEET'08. , 2008.

[4] P. Abrahamsson, P. Kettunen and E. Fagerholm, "The Set-Up of a Software Engineering Research Infrastructure of the 2010s," 2010.

[5] M. Kuhrmann, P. Diebold and J. Münch, "Software process improvement: a systematic mapping study on the state of the art," PeerJ Computer Science , vol. 2, nº 62, 2016.

[6] M. Pariata and N. Montaño, "Software Factory, from professional environment to academic environment proposal to build competences through authentic activities in the context of software engineering," em XL Latin American Computing Conference (CLEI), 2014.

[7] M. Pesantes, C. Lemus, H. A. Mitre and J. Mejía, "Software Process Architecture: Roadmap," em Ninth Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2012.

[8] J. Naranjo, J. L. R. Becerra, A. Rossi and F. Lopes, "Utilización de la técnica QFD en una arquitectura de procesos de software," I+i Investigación aplicada e innovación, pp. 50-59, 2016.

[9] A. Ravichandran, K. Taylor and P. Waterhouse, DevOps for Digital Leaders: Reignite Business with a Modern DevOps-Enabled Software Factory, CA, Ed., Springer , 2016.

[10] J. Cito, P. Leitner, T. Fritz and H. C. Gall, "The Making of Cloud Applications: An Empirical Study on Software Development for the Cloud," em Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, 2015.

[11] L. Kavanagh, C. Reidsema, J. Mccredden and N. Smith, Design Considerations, Singapore: Springer, 2017.

[12] D. A. Trippel, "Tools for Problem- and Project-based Learning in Sustainability Science Education: A Case Study of Two Undergraduate Classes," Arizona State University, 2013.

[13] M. Lankhorst, Enterprise Architecture at Work: Modelling, Communication and Analysis, 4 ed., Berlin Heidelberg: Springer-Verlag, 2017.

[14] A. Fattah, "Enterprise reference architecture," em In 22nd Enterprise Architecture Practitioners Conference, London, UK, 2009.

[15] P. Grefen, Business information systems architecture, Eindhoven, The Netherlands: Eindhoven University of Technology, 2016.

[16] F. Aulkemeier, M. Schramm, M. Iacob and J. Van hillegersberg, "A Service-Oriented E-commerce Reference Architecture," Journal of theoretical and applied electronic commerce research, vol. 11, nº 1, pp. 26-45, 2016.

[17] A. G. B. Cruz, An Information Systems Reference Architecture for the, Lisboa, Portugal: Técnico Lisboa, 2015.

[18] BIAN, "BIAN Standards Service Landscape 5.0," 2017. [Online]. Available: https://bian.org/servicelandscape/.

[19] Microsoft, "Microsoft Industry Reference Architecture for Banking (MIRA-B)," 2012. [Online]. Available: https://news.microsoft.com/download/presskits/msfinancial/docs/MIRAB.pdf .

[20] C. Czarnecki and C. Dietze, Reference Architecture for the Telecommunications Industry, Springer, 2017.

[21] W. T. H. Van der Beek, J. Trienekens and P. Grefen, "The Application of Enterprise Reference Architecture in the Financial Industry," em Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation, Berlin Heidelberg, Springer, 2012, pp. 93-110.

[22] F. Timm, C. Köpp, K. Sandkuhl and M. Wißotzki, "Initial Experiences in Developing a Reference Enterprise Architecture for Small and Medium-Sized Utilities," em Proceedings of Short and Doctoral Consortium Papers Presented at the 8th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling (PoEM 2015), Valencia, Spain, 2015.

[23] R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Heidelberg New York Dordrecht London: Springer, 2014.

[24] L. Dias, J. F. R. Naranjo, D. Marques and J. L. R. Becerra, "Fundamentos de uma Fábrica de Software Orientada a Objetos Processos," Augusto Guzzo Revista Acadêmica, vol. 9, pp. 53-61, 2012.

[25] R. Noël, R. Munoz, C. Becerra and R. Villarroel, "Developing competencies for software requirements analysis through project based learning," em 35th International Conference of the Chilean Computer Science Society (SCCC), 2016.

[26] K. Gary, "Project-based learning," Computer, vol. 48, nº 9, pp. 98-100, 2015.

[27] F. Fagerholm, N. Ozay and J. Munchz, "A Platform for Teaching Applied Distributed Software Development The Ongoing Journey of the Helsinki Software Factory," em CTGDSD, San Francisco, 2013.

[28] M. Galster and S. Angelov, "What makes teaching software architecture difficult?," em Proceedings of the 38th International Conference on Software Engineering Companion, 2016.

[29] A. Van deursen, M. Aniche, J. Aué, R. Slag, M. De jong, A. Nederlof and Bouwers, "A Collaborative Approach to Teaching Software Architecture," em Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, 2017.

[30] G. Vreede and R. Briggs, "Collaboration Engineering: Designing Repeatable Processes for High-Value Collaborative Tasks," em Annual Hawaii International Conference on System Sciences (HICSS), Hilton Waikoloa Village, Hawaii, USA, 2005.

[31] A. M. Magdaleno, R. Mendes and C. M. Lima W, "A roadmap to the Collaboration Maturity Model (CollabMM) evolution," em 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2011.

[32] L. Dias, "Método de instanciação de uma arquitetura de processos aplicado em fábrica de software," 2010.

[33] ITU-T , "Rec. X.911 Information technology – Open distributed processing – Reference model – Enterprise language," ITU-T , 2014.

[34] S. Karadgi, A Reference Architecture for Real-Time Performance Measurement, Springer, 2014.

[35] H. Kandjani, P. Bernus and S. Nielsen, "Enterprise Architecture Cybernetics and the Edge of Chaos: Sustaining Enterprise as Complex System in Complex Business Environments," em 46th Hawaii International Conference on System Sciences, Hawaii, 2013.

[36] J. Zachman, "Enterprise architecture: The issue of the century," Database Programming, vol. 10, p. 44–53, 1997.

[37] P. Bernus, O. Noran e A. Molina, "Enterprise architecture: twenty years of the GERAM framework," Annual Reviews in Control, pp. 83-93, 2014.

[38] ISO/IEC/IEEE, ISO/IEC/IEEE 42010 Systems and software engineering - Architecture description, IEEE, 2011.

[39] D. Garlan, "Software architecture: a travelogue," em Proceedings of the on Future of Software Engineering, 2014.

[40] F. Sanchez-puchol and J. A. Pastor-collado, "A First Literature Review On Enterprise Reference Architecture," em The 11th Mediterranean Conference on Information Systems (MCIS), Genoa, Italy, 2017.

[41] G. Muller and E. Hole, "Reference architectures; why, what and how," em White paper, 2007.

[42] S. Angelov, P. Grefen and D. Greefhorst, "A framework for analysis and design of software reference architectures," Information and Software Technology, vol. 4, nº 54, pp. 417-431, Abril 2012.

[43] The Open Group, "TOGAF Version 9.1," VanHaren Publiching, 2011.

[44] D. Hashimoto, A. Tanaka and M. Yokoyama, "Case study on RM-ODP and Enterprise Architecture," em Eleventh International IEEE EDOC Conference Workshop (EDOCW'07), 2007.

[45] L. Kutvonen, "Using the ODP reference model for Enterprise Architecture," em Eleventh International IEEE EDOC Conference Workshop, 2007.

[46] OMG, "ArchiMate® 3.0.1 Specification," 08 2017. [Online]. Available: http://pubs.opengroup.org/architecture/archimate3-doc/.

[47] ITU-T, ISO/IEC 19793 X.906 Information technology – Open distributed processing – Use of UML for ODP system specifications, 2014.

[48] TMF, "Frameworx," 2017. [Online]. Available: https://www.tmforum.org/tm-forum-frameworx/.

[49] F. Timm, K. Sandkuhl and M. Fellmann, "Towards A Method for Developing Reference Enterprise Architectures," em 13th International Conference on Wirtschaftsinformatik, St. Gallen, Suiza, 2017.

[50] P. Linington, Z. Milosevic, A. Tanaka and A. Vallecillo, Building enterprise systems with ODP: an introduction to open distributed processing, CRC Press, 2011.