Computers in Industry 135 (2021) 103569

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Improving information system design: Using UML and axiomatic design☆

Luís Cavique[a,*], Mariana Cavique[b], Armando Mendes[c], Miguel Cavique[d]

[a] Universidade Aberta and LASIGE-FCUL, Portugal
[b] Universidade Europeia and ISCTE-UL, Portugal
[c] Universidade Açores and LIACC, Portugal
[d] Escola Naval, Portugal

ARTICLE INFO

Article history:
Received 11 June 2021
Received in revised form 26 September 2021
Accepted 28 October 2021
Available online xxxx

Keywords:
Enterprise architecture
Organization alignment
UML
CRUD matrix
Axiomatic Design

ABSTRACT

A unified view of the Information System (IS) design is essential for dealing with complexity. However, the literature proposes many denominations, depending on the layer, methodology, framework, or tool. This multitude of approaches does not allow a holistic view of the system. Besides, in Information Systems, the search for good practices in design is still a relevant issue. A subset of essential Unified Modeling Language (UML) diagrams is chosen to create a broad view of the IS. CRUD matrix is one of the preferred approaches to articulate the sub-systems of applications and data. Axiomatic Design (AD) provides rules for the improvement of the IS design. This work presents a method to create object-oriented elements based on the CRUD matrix aligned with the business strategy. An integrated student-based case study on logistics is provided. In the discussion, a new IS architect role is proposed supported by the CRUD/AD method.

1. Introduction

Information Systems (IS) are an input/output mechanism that collects data, store it, and distribute information. However, new legislation, new entities, and new interactions create challenges in information management. The alignment of business strategy with technology is crucial. The alignment asks of the information technology (IT) process to achieve business objectives in any business organization.

Software development is far from having a genuine impact on organizations. The Standish Group International (2010)) says 64% of software features are rarely or never used. Besides, the cost of poor software quality in 2018 in the USA is two trillion dollars (Krasner, 2021). This value includes unsuccessful software projects, poor quality in legacy systems, and operational software failures. Therefore, a better-quality software design is needed.

Despite the high number of methodologies (Van-Bon and Verheijen, 2006) in Information Systems, the search for good practices in design is still a relevant issue. What a good design is, is still a subject of controversy in IS.

UML (Unified Modeling Language) (Fowler, 2003) is a powerful toolbox that facilitates systems analysis and is an essential vehicle for design quality improvement. The use of UML iteratively in analysis and design allows the fulfillment of the system requirements with object-oriented design and relational databases models.

The bibliography on UML tools is vast, and it is presented at different levels and formats.

Despite that, the UML bibliography gives a disconnected view of the systems, as each chapter refers to use-case diagrams, class diagrams, activity diagrams, state diagrams, sequence diagrams, and physical diagrams. UML joined a set of diagrams of different authors without merging them. Each UML diagram corresponds to a partial view of the system. As a result, UML is usually presented in fragments in a poorly unified vision.

Most design methodologies are too detailed and allow inconsistency in process descriptions instead of providing a global overview. Enterprise Architecture (EA) (Lankhorst, 2013) promises an integrated approach to deal with complexity. EA goes beyond the symbolic models, such as UML diagrams, and achieves more coherent and meaningful tools, the so-called semantic models.

Returning to the origins, BSP (Business Systems Planning) (IBM, 1978) presents four essential elements for Information Systems Planning, coined as the Iron Cross: actors, applications, data, and technological systems (Rocha and Freixo, 2015). This work demonstrates that they can create a unified IS view using a subset of essential UML diagrams.

☆ The authors gratefully acknowledge the support of WorldCIST 2021 conference.
* Corresponding author.
E-mail addresses: luis.cavique@uab.pt (L. Cavique), mariana.cavique@universidadeeuropeia.pt (M. Cavique), armando.b.mendes@uac.pt (A. Mendes), cavique.santos@marinha.pt (M. Cavique).

https://doi.org/10.1016/j.compind.2021.103569
0166-3615/© 2021 Elsevier B.V. All rights reserved.

This document defines an Enterprise Architecture perspective by creating vertical and horizontal alignments to ensure consistency. In search of a unified view of the system, UML tools links with the CRUD matrix, the acronym of < create, read, update, delete > (Martin, 1983).

Axiomatic Design (AD) is a theoretical structure to achieve a good design. AD represents the design using matrices from customer needs to process variables. The output of Axiomatic Design is a system composed of a set of loosely coupled modules. For a thorough explanation see Suh (1990, 1995, 2001, 2005). AD is a comprehensive, concise, and consistent theory that provides insight into the IS design.

1.1. Objective

EA aims for a high-level view of the system. Some methodologies use the intricate concepts of views, and new languages that evolve into more specific fields, increasing the lexical complexity and losing the necessary simplicity of the architecture. However, most scientific literature proposes solutions with many synonyms depending on the layer, methodology, framework, or tool to answer these challenges. The objective is to create a holistic view of the EA, mainly by improving the Information System design using the minimum jargon.

1.2. Contribution

This document extends the work that integrates UML diagrams and CRUD matrix to align information systems (Cavique et al., 2021). It merges another work on Axiomatic Design to improve the design in object-oriented diagrams (Cavique and Cavique, 2021). The contribution of this work is the proposal of the CRUD/AD method. In this work, by CRUD/AD matrix, we mean a CRUD matrix that ensures the axioms of AD. The novelty of this document is the articulation of UML diagrams, CRUD matrix, and Axiomatic Design to find what the good practices in Information System design are.

1.3. Organization

The remaining of the paper is organized as follows. Section 2 reports the background information. Section 3 presents the proposed model with a running student-based case study on the logistics of Covid-19 vaccines. Section 4 reports the discussion, and finally, in Section 5, conclusions are drawn.

2. Background information

This section presents the concept of Enterprise Architecture. Previous works of the authors integrated the CRUD matrix with UML. Moreover, this section shows an approach on how to align architectures in EA. The Axiomatic Design theory is introduced to establish the rules for making good designs. In the final sub-section, Axiomatic Design is applied to redesign the CRUD matrix using an object-oriented approach.

2.1. Enterprise architecture

EA is the current approach to long-term Planning for Information Systems. EA provides an appropriate context to business goals, integration between information systems, and better utilization of information technology.

John Zachman introduces the first enterprise architecture framework (Zachman, 1987). The framework has a two-dimension matrix that the enterprise architects must fill. The lines refer to the levels, roles in the organization, and columns to essential resources such as data, function, and people.

Spewak and Hill (1995)) propose the Enterprise Architecture Planning, a data-oriented approach to provide data quality, data interoperability, and data sharing. It allows the intervention in changing environments with controlled costs.

TOGAF, The Open Group Architecture Framework, (TOGAF, 2011) is a more recent generic framework for developing technical architectures that evolved into an enterprise architecture framework. TOGAF has a set of four components, where the Architecture Development Method (ADM) stands out. The ADM is the core of TOGAF, consisting of a cyclic approach for developing the enterprise architecture. TOGAF's ADM comprises eight iterative steps, which constitute a framework for enterprise architects.

EA models usually have three layers: business/process architecture, information system (data and applications) architecture, and technological architecture (Ward and Peppard, 2002; TOGAF, 2011; Lankhorst, 2013).

Enterprise Architecture (Lankhorst, 2013) uses the word 'architecture' from building and construction, referring to a holistic view. ArchiMate is one of the most famous enterprise modeling languages, created to be a meta-model of UML or BPMN (Business Process Model and Notation) tools. However, the proposed ArchiMate language evolves into specific fields, increasing the linguistic complexity and losing architecture simplicity.

Hinkelmann et al. (2016)) anticipate a next-generation enterprise information systems paradigm based on an Agile approach. The proposed meta-modeling framework supports enterprise ontologies.

In the book of Desfray and Raymond (2014)) proposed the EAP (Enterprise Architecture Profile) language, extending the UML concepts to represent all TOGAF objects. In addition, the authors promised a practical guide using UML and BPMN in the paper sub-title.

Barros et al. (2000)) modeled business processes, business entities, business roles, and business events using UML language. However, they dealt only with the business layer, not addressing the IS and technological layers.

Silingas and Butleris (2009) proposed an approach to customizing UML tools for domain-specific modeling needs. The authors reused the generic Zachman framework to answer the 6 Wh (what, how, why, who, when, where) questions in the Business, System, and Technology models. However, they do not provide a straightforward procedure to apply the approach.

Perez-Castillo et al. (2019) conducted a systematic mapping study on enterprise architecture. The authors conclude that the process is costly and subject to errors, discouraging enterprises from adopting EA.

2.2. CRUD matrix integrated with UML

In Enterprise Architecture, many synonyms depend on the layer, methodology, framework, or tool used. It makes essential to find a synthesis with a reduced number of entities. In this work, the four elementary elements of the Iron Cross were chosen: the actors, the activities, the data, and the infrastructure. Some of the synonyms are as follows:

(i) actors are a synonym of lines-of-responsibility;
(ii) activities are a synonym of applications, tasks, uses-cases, or operational processes;
(iii) data is a synonym of classes or informational entities;
(iv) infrastructure is a synonym of service.

In the following examples, actors are (α, β), activities (A, B, C), and data (X, Y, Z, W). The proposed version of Enterprise Architecture has three layers:

- Business / Processes Architecture: where processes are made up of activities (A, B, C) and managed by human actors (α, β);
- Information System Architecture: with two different software groups, the data (X, Y, Z, W) and the applications (A, B, C);
- Technological Infrastructure Architecture: consists of hardware components and essential software (operating systems and database management systems).

CRUD matrix was popularized by James Martin (1983)) in his book Managing the Data-base Environment. CRUD matrix crosses information between applications and data classes.

Once filled, the CRUD matrix goes through exchanging rows and columns to find clusters where each application performs Create, Update and Delete (CUD) operators. In the matrix, multiple applications can use the Read operator with no restrictions.

We add the CRUD counters to the matrix to validate inconsistencies in the CRUD matrix (Cavique, 2020). In the CRUD matrix, many reads, R, are allowed. On the other hand, only one CUD is advisable to maintain consistency. For example, Fig. 1 shows the CRUD counter with value 1211, indicating 1 Create, 2 Reads, 1 Update, and 1 Delete. Preferably, CRUD counters should be 1N11, i.e., a unique Create, Update and Delete and multiple Read operators, in order to uncouple the system.

The CRUD counters help identify areas that should be studied and support questioning about the duplication of operators. Many conflicts happen if the counters are different from 1N11. For example, when a CRUD counter returns 2222, it conflicts with CUD. One way to solve it is to disaggregate the involved elements to achieve a loosely coupled global system.

We combine CRUD with UML in this work since UML diagrams are popular in the software industry. Moreover, UML has a broad community of professionals.

In the Business and Information Architectures layers, three of the four elements of the Iron Cross are used: the actors, the use-cases/applications, and the data classes. Fig. 2 shows that only three UML diagrams (use-case, class diagram, and sequence diagram) are enough to represent a system. Although the CRUD matrix is not a UML tool, it can complete a system view. The fourth element regards the Technological Architecture layer, where the deployment diagram is used.

It is possible to address a system view using one element, a pair of elements, and a trio with the three elementary vectors (actors, applications/use-cases, data/classes). However, the only possible diagram with a single element is the class diagram. When using the pair (actors, applications/use-cases), the use-case diagram is represented. The CRUD matrix represents the pair (applications/use-cases, data/classes). Finally, the sequence diagrams illustrate the trio of (applications/use-cases, actors, data/classes).

applications vs data	X	Y	Z	W
A	CRUD			R
B		CRUD	R	
C	R		CRUD	CRUD
CRUD counters	1211	1111	1211	1211

Fig. 1. CRUD matrix with counters.

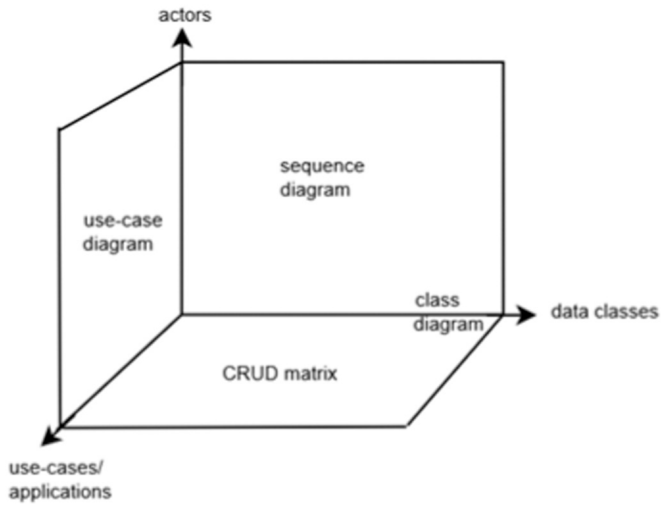


Fig. 2. The three essential elements supported by UML diagrams and the CRUD matrix.

2.3. Enterprise architecture alignment

The challenge of EA is to create a vertical alignment that allows communication among the business team, IS team, and IT team, merging the three layers into a single architecture. Organization alignment is essential in companies that foster unity between layers (vertical alignment) and within the same layer (horizontal alignment).

Valorinta (2011) shows how external and internal organizational management impacts IT alignment, given the increasing importance of aligning IT and business functions.

Pereira and Sousa (2005) align architectures using a set of heuristic rules grouped in three clusters. This work presents an overview of alignment in organizations, supported by rules that alert to poor designs.

Based on the work of Pereira and Sousa (2005), this work proposes an approach that reuses the CRUD matrix and assures vertical and horizontal alignment. Three heuristic rules support the alignment of the information system. The heuristic rules, shown in Fig. 3, can be summarized as follows:

#1 The data items must support the processes activities of the business processes;.

#2 Each processing activity of the business process is automated by a single application;.

#3 Each data item is managed (CUD) by a single application in the CRUD matrix.

Rules #1 and #2 guarantee the vertical alignment, and rule #3 the horizontal alignment. Rules #1 and #2 are closely related to business processes, and they are matched by rule #3. Note that rule #3 uses the CRUD matrix and is equivalent to the CRUD counters.

2.4. Axiomatic design

To obtain a self-contained document, in this section, the most relevant aspects of Axiomatic Design (AD) (Suh, 1990, 2001, 2005) are presented. AD has similarities with the CRUD matrix regarding the Design Matrix.

The design process maps along with four domains: customer domain, functional domain, physical domain, and process domain, as

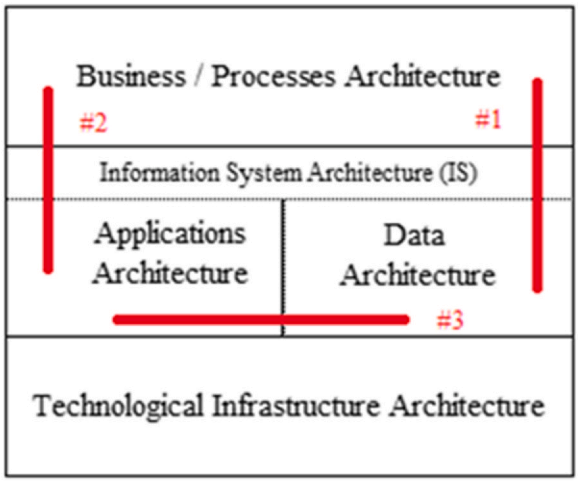

Business / Processes Architecture
#2
#1
Information System Architecture (IS)
Applications Architecture
Data Architecture
#3
Technological Infrastructure Architecture

Fig. 3. Heuristic rules for enterprise alignment.

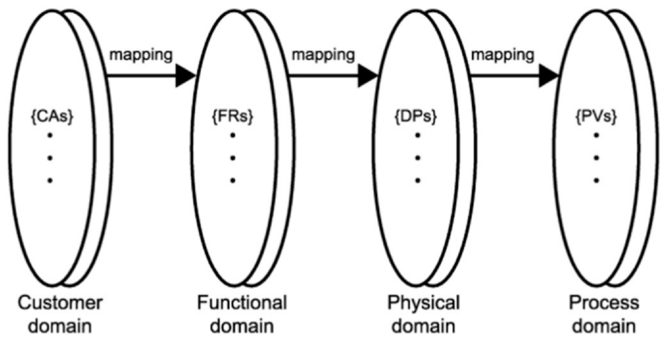


Fig. 4. The domains of axiomatic design by Suh (2001)).

shown in Fig. 4. Each domain has the corresponding vector of elements: customer attributes {CAs}, functional requirements {FRs}, design parameters {DPs}, and process variables {PVs}. AD applies to any new design, like engineering, industrial, organizational, or software.

According to AD, a good design needs the following two axioms:

- First axiom, the Independence Axiom: maintain the independence of functional requirements;
- Second axiom, the Information Axiom: minimize the information content of the design.

The mapping procedure expresses the relationships between every two sequential domains. The design is usually about the mapping from FRs to DPs. A good design needs to have the same number of FRs as the number of DPs. Therefore, the design matrix [A] is a square matrix, and the design equation is {FRs} = [A].{DPs}.

In the expanded form, the item A_{ij} relates FR_i with DP_j.

$$\begin{Bmatrix} FR1 \\ \cdots \\ FRn \end{Bmatrix} = \begin{bmatrix} A11 & \cdots & A1n \\ \vdots & \ddots & \vdots \\ An1 & \cdots & Ann \end{bmatrix} . \begin{Bmatrix} DP1 \\ \cdots \\ DPn \end{Bmatrix}$$

The design matrix can show three types of designs: uncoupled, decoupled, and coupled. The uncoupled design is the ideal design, and the decoupled is acceptable as a good design. A coupled design is a poor design. The corresponding design matrixes are respectively a diagonal matrix, an upper or lower triangular matrix, or a full matrix, as follows:

$$\text{Uncoupled}\begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} X & 0 \\ 0 & X \end{bmatrix} . \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix}$$

$$\text{Decoupled}\begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} X & 0 \\ X & X \end{bmatrix} . \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix}$$

$$\text{Coupled}\begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} X & X \\ X & X \end{bmatrix} . \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix}$$

If the number of FRs is greater than the number of DPs, length (FRs) > length(DPs), then the design is coupled, or some FRs cannot be fulfilled. When length(FRs) < length(DPs), the design is redundant, which can turn into a coupled, decoupled, or uncoupled design.

The design is a hierarchical process of decomposition from the top FR-DP to the leaf elements. The decomposition zigzag between two domains, from {FRs} to {DPs}, and then from DPs at a level to the FRs at the following lower level. Fig. 5 depicts the zigzag process.

The Information Axiom aims to minimize the information of the design. Given *p*, the probability of satisfying FR with DP, the information content is given by $I = log_2\left(\frac{1}{p}\right)$.

In an uncoupled design, the probability of satisfying FR1,., FRn with DP1,., DPn, is the product of the probability of independent

FR1		→	DP1	
	FR11	↙ →		DP11
	FR12			DP12
	FR13			DP13
FR2		→	DP2	
	FR21	↙ →		DP21
	FR22			DP22

Fig. 5. Zig-zagging.

events. Therefore, the information content is the sum of the information of accomplishing each FR:

$$I = \sum_{i=1}^{n} I_i = \sum_{i=1}^{n} log_2\left(\frac{1}{p_i}\right)$$

If the design is uncoupled or decoupled, the chosen design should have minor information content.

Therefore, it is possible to compare AD with the CRUD matrix. In AD, the CAs are the system requirements mapped into the applications. The applications correspond to the FRs of AD. The data architecture corresponds to the DPs and the CRUD matrix to the design matrix [A]. The goal of the CRUD matrix is to maintain the independence of the applications. Therefore, both methods try to find a design that corresponds to a diagonal matrix.

2.5. Re-design CRUD with axiomatic design

This sub-section focuses on how AD can improve the CRUD structure in IS designs. The similarities between AD and IS rules are discussed. This section ends with an object-oriented re-design by applying AD to Software [chapter 5, Suh (2001)].

As stated before, the design maps between FRs and DPs. In an ideal design, each DP must fulfill an FR so the design matrix [A] of the design equation {FRs} = [A]. {DPs} is square.

The {FRs} correspond to 'what' are the goals to achieve, and {DPs} match with 'how' to reach them. In this work:

- functional requirements {FRs} correspond to the applications {A, B, C}, that are aligned with the business architecture;
- the design parameters {DPs} match the data items {X, Y, Z, W}, that should support the business architecture;
- The design matrix [A], which maps FRs and DPs, coincides with the CRUD matrix.

In other words, {FRs} = [A]. {DPs} corresponds to {Apps}=[CRUD]. {Data}. DPs can also be seen as Design Patterns as referred in Thomas and Mantri (2015). In the design matrix the cells with operators CRUD are replaced by an X.

CRUD matrix might express a good design. The aim of CRUD and the design matrix is to maintain the independence of the functional requirements. Usually, an uncoupled design has less information content than a decoupled or coupled design. Therefore, an uncoupled design has a high probability of success than the other types of designs.

When the number of FRs is less than the number of DPs, the solution is redundant or coupled, as stated by Theorem 3 (Suh, 1990). Gonçalves-Coelho et al. (2012a, 2012b) claim that redundant designs follow seven theorems that allow classifying the design.

Fig. 1 shows a redundant design. Application C operates on data-items/tables Z and W. According to AD, the design equation is:

$$\begin{Bmatrix} FR\_A \\ FR\_B \\ FR\_C \end{Bmatrix} = \begin{bmatrix} X & & & \\ & X & & \\ & & X & X \end{bmatrix} . \begin{Bmatrix} DP\_X \\ DP\_Y \\ DP\_Z \\ DP\_W \end{Bmatrix}$$

In software engineering, an application can make a change on a table and then on another table. Therefore, it is possible to split an application in two. As a result, the concept of redundancy is more flexible in software than in many engineering fields. The redundant design of application C and data (Z, W) is equivalent to two different states of the same FR that can be fulfilled by *DP_Z* or *DP_W* as follows:

$$\{FR\_C\} = [X\ X].\left\{\begin{matrix} DP\_Z \\ DP\_W \end{matrix}\right\} \text{equals to}$$

$$\{FR\_C1\} = [X].\left\{\begin{matrix} DP\_Z \\ DP\_W \end{matrix}\right\} \text{and}$$

$$\{FR\_C2\} = [X].\left\{\begin{matrix} DP\_Z \\ DP\_W \end{matrix}\right\}$$

Application C can be divided into C1 when addressing table Z and C2 when addressing table W. The design matrix can turn into a square matrix by re-designing FR-C as shown in Fig. 6, using a square sub-matrix, where FR-C changes to FR-C1 and FR-C2.

2.6. Integration of the two approaches

Using the similarities between IS and AD, the axioms and theorems of AD can support new heuristic rules in IS. Table 1 shows the additional rule #4 based on AD and the similarities between heuristics rules and AD.

AD imposes a square design matrix in an ideal design (Theorem 4) (Suh, 1990). Therefore, the CRUD matrix should be square. As per rule #4, the number of applications is equal to the number of data items.

Rule #4, where the number of applications is equal to the number of data items, is a consequence of theorem 4 in AD. CRUD/AD matrix means a CRUD matrix improved with the first axioms of AD.

The CRUD matrix allows uncoupled designs. However, decoupled matrices seem not to be adequate for IS. This statement agrees with Theorem Soft 1 (Suh, 2001), which states that uncoupled software or hardware systems can operate without precise knowledge of the design elements (i.e., modules).

2.7. Object-oriented re-design

Object-oriented (OO) techniques are some of the most potent paradigms widely used in software design. An object is a bundle of data associated with a set of operations that act on that data. The object-oriented paradigm supports four critical features: abstraction, encapsulation, inheritance, and modularity.

OO techniques encapsulate data and the operations that manipulate data in a specific object. OO paradigm allows a higher level of abstraction, focusing on interface characteristics and omitting the details of data and operations. In addition, the system's modularity allows independence. By decomposing each problem into loosely coupled intra-objects, internal changes to one object do not affect any other.

In the afore presented Fig. 6, each cross ('X') in the design matrix [A], corresponds to an object. Each object *i*, corresponds to an {FR(i)}

	DP-X	DP-Y	DP-Z	DP-W
FR-A	X			
FR-B		X		
FR-C.1			X	
FR-C.2				X

Fig. 6. Re-design of FR-C.

= [A(i,i)]. {DP(i)} in the design matrix and consequently to the equivalent cell *i* in the CRUD matrix.

Therefore, four objects {A.X, B.Y, C1.Z, C2.W} exist in the uncoupled design matrix. The OO data items can be enriched because there is one relation between one FR and one DP. Fig. 7 illustrates the four objects.

The OO design supported by the AD concept of independence reinforces the OO paradigm. In addition, it can incorporate the advantages of modularity and reuse, providing savings in the software project development.

3. Proposed CRUD/AD model

In this section, the proposed model defines the diagrams for each layer and the vertical and horizontal alignment heuristic rules of the EA. A procedure to ensure alignment is detailed, and a run case-study is presented.

3.1. Heuristic rules to align enterprise architecture

Fig. 8 shows the three layers of EA associated with UML diagrams. The 1st layer (above) shows the sequence of activities in the Business/Process Architecture. The 2nd layer describes the list of applications (on the left) and the data diagram (on the right). The CRUD matrix merges applications and data items (in the middle) of the Information System Architecture. Finally, the 3rd layer illustrates the Technological Architecture of the infrastructure.

A UML activity diagram with partitions (swim-lanes) or a simplified version using a UML use-case diagram can represent the process diagram.

Entity-Relationship diagram or a class diagram in UML can represent the data. Also, in the 2nd layer, there is a list of applications and a CRUD matrix that unify applications and data.

Finally, an architecture diagram or UML implementation represents the infrastructure diagram using components and nodes.

The set of diagrams must be aligned to assure consistency in their articulation. Two alignments are generally considered vertical and horizontal alignment.

In the business area, the description of the architecture starts with the "Business Area Narrative". The narrative can be obtained through meetings, focus groups, interviews (structured or unstructured), or case studies. The narrative generally includes a survey of the 'as-is' system (past and present) and the intended or 'to-be' (future) design associated with a set of functional requirements.

The system architect needs to address the design of two complementary views: the view of the functional analyst about process architecture who deals with the end-users (1st layer), and the system analyst approach focused on the IS/IT architecture (2nd and 3rd layer).

Fig. 9 illustrates a set of rules to achieve the Enterprise Architecture alignment. Pereira and Sousa (2005) proposed three heuristic rules that guarantee the alignment of the information system. In addition, rule #4 was introduced with AD, and rules #0 and #5 connect the additional UML diagrams. The revised heuristic rules are as follows:

#0 The process architecture must support the narrative of the business area;.

#1 The data architecture must support the narrative of the business area;.

#2 Each processing activity is automated by a single application;.

#3 Each data set is managed (CUD) by a single application;.

#4 Re-design of CRUD matrix according to the concept of independence of AD;.

#5 Each infrastructure is associated with one or more applications.

Table 1
Heuristic rules versus Axiomatic Design.

Heuristic rules in IS	Axiomatic Design
#1 the data items must support the processes activities of the business narrative	data items correspond to {DPs}
#2 a single application automates each process activity of the business processes	applications correspond to {FRs}
#3 each data item is managed by a single application in the CRUD matrix	design matrix [A] ensuring the independence of FRs
#4 the number of applications is equal to the number of data-items	Theorem 4 in Axiomatic Design:length(FRs) = length(DPs)

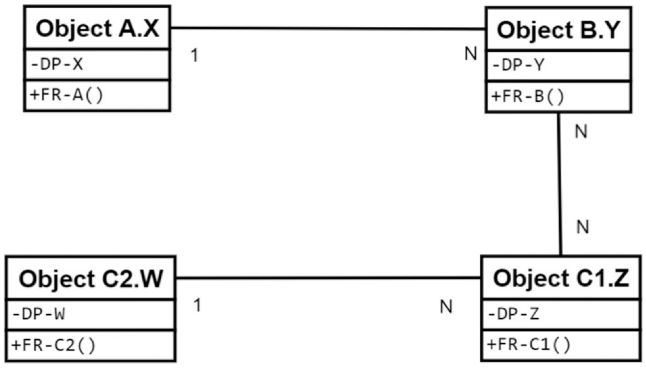


Fig. 7. Object-oriented re-design supported by AD.

Two heuristic rules apply to the business narrative. Rule #0 regards process architecture, and rule #1 to data architecture.

The business activities that can be automated are called applications and must follow rule #2. Moreover, rule #3 applies to data items and applications by the CRUD matrix.

After the survey of the Business Narrative, rules #0 and #1 initiates the two complementary views of the system. Then rule #2 allows the digital transformation of activities into applications. Rule #3 uses data and applications to create the CRUD matrix. Rule #4 redesigns the CRUD matrix using AD. Finally, rule #5 associates a technological infrastructure with one or more applications.

Rules #0, #1, #2, and #5 guarantee the vertical alignment between layers, rule #3 ensures the horizontal alignment within the layer, and rule #4 improves the core of the IS.

3.2. Procedure to ensure alignment

Procedure 1 ensures the alignment in the Enterprise Architecture numbered by the layer number: (1) for the 1st layer of Business processes, (2) for the 2nd layer of information system, IS, and (3) for the 3rd layer, technological architecture.

Procedure CRUD/AD:

(P0) define the business narrative;.

(P1) define the business architecture: UML activity diagram;.

(P2) define the information system architecture:

(P2.a) define the data architecture: class diagram;.

(P2.b) define the application's architecture: applications;.

(P2.c) define the CRUD matrix and re-design it allowing AD independence;.

(P2.d) detail CRUD cells with sequence diagrams;.

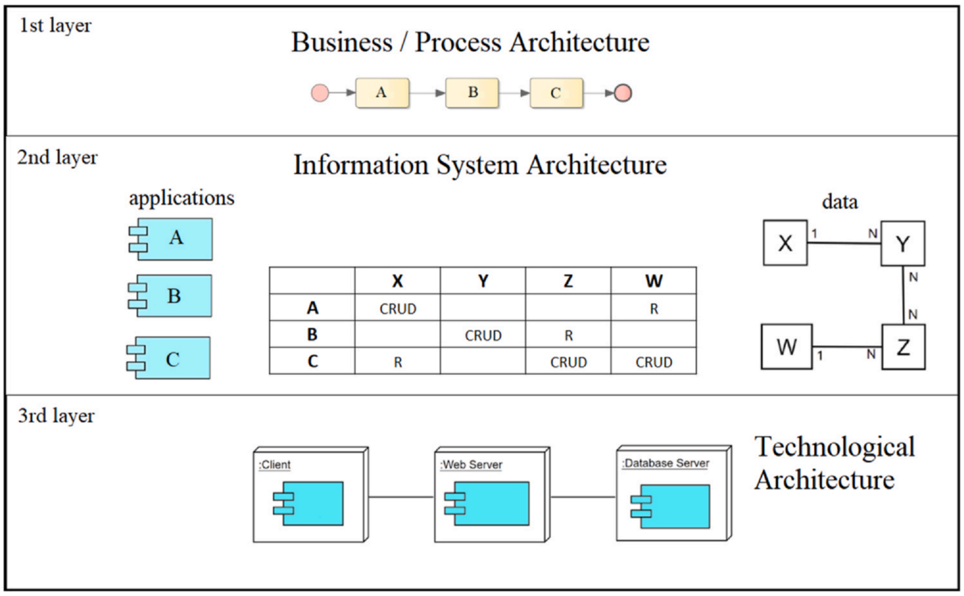
(P3) define the technological architecture: deployment diagram.

This procedure is further detailed in the following paragraphs:

(P0) Define the business narrative. The narrative should refer to the actors, activities, and data and how they are articulated.

(P1) Define the process architecture (or sequence of activities). List the activities, list the actors and fill in the matrix activities versus actors. Develop a UML use-case diagram or a UML activity diagram with lines of responsibility with the system actors. Apply rule #0, where the process architecture must support the narrative of the business area.

(P2.a) Define Data architecture. Create an Entity-Relationship diagram or a UML class diagram for the data architecture representing the necessary data that correspond to the requirements referred to in the narrative. Apply rule #1, where the data architecture must support the narrative of the business area.



	X	Y	Z	W
A	CRUD			R
B		CRUD	R	
C	R		CRUD	CRUD

Fig. 8. Three layers of the EA.

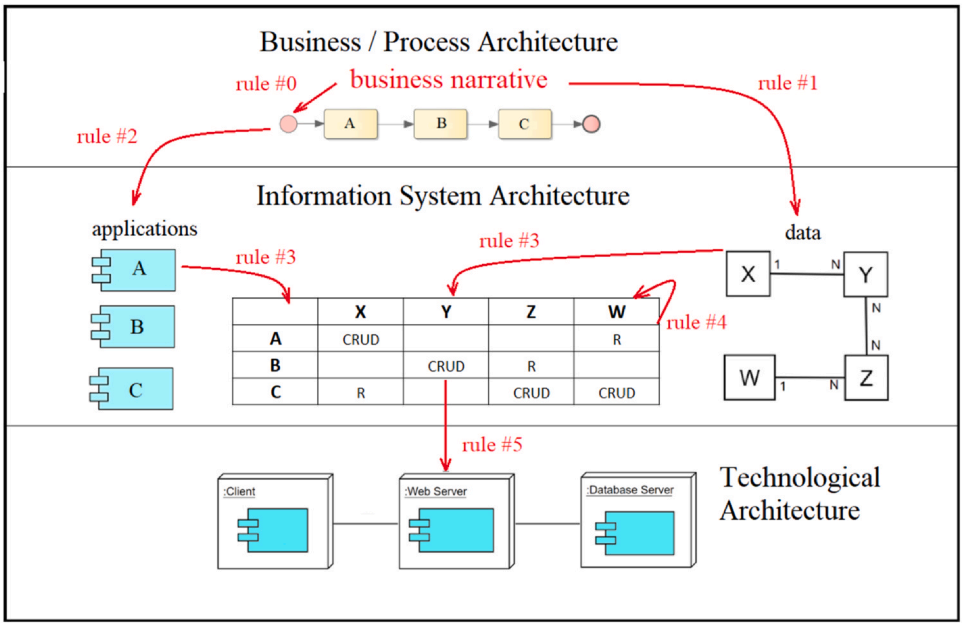


Fig. 9. Heuristic rules to align the Enterprise Architecture.

(P2.b) Apply rule #2, so each automated activity must correspond to a single application. Produce the CRUD matrix representative of the relationship between applications and classes (data).

(P2.c) Apply rule #3, making each data set managed (CUD) by a single application. By applying rule #4, re-design CRUD and the class diagram with Axiomatic Design.

(P2.d) Detail the CRUD matrix for each application, associating a sequence diagram, which details the message sequences between the classes.

(P3) Define technological architecture. The UML deployment diagram includes technological platforms, servers, client computers, databases, and operating systems. Next, apply rule # 5, associating one or more applications to each infrastructure.

Finally, summarize the three lists of the essential system elements (actors, activities, and data classes), and check the vertical and horizontal alignment of heuristic rules.

3.3. Running Case-Study

The importance of fighting COVID-19 motivates this student-based case-study. The study centers on the information model for vaccine distribution logistics and the national vaccination program user vaccination. The study starts with Information System Narrative based on a requirement report, meeting, or interview with the decision-makers. The business narrative is as follows.

3.3.1. (P0) The business narrative

The system to fight against Covid-19 has the following needs.

- To vaccinate about 10 million healthcare users;
- There are more than a hundred vaccination centers around the country and a logistic hub that coordinates all the operations;
- The case-study focuses on a vaccination center;
- The users must have an identifier, name, contacts, age, and other medical information;
- The data regarding vaccination centers should include the address, vaccine capacity, and number of users;
- There are several types of vaccines; some need two doses and others in a single one;
- User records in the vaccination center include of vaccine and the shot dates;
- Number and type of vaccines available at each vaccination center;
- Every time a user is vaccinated, the information is recorded in the system;
- In each vaccination center, there are daily entries and exits of vaccines;
- Entries correspond to new boxes of vaccines carried by the police;
- Each box has hundreds of vials; at each vial corresponds to 5 or 6 vaccine doses;
- Each box belongs to a vaccine lot number;
- The vaccine exits should correspond to the vaccines taken by the users.

The logistic hub provided the maintenance of the vaccination center, the different types of vaccines, and the users' identification.

There are different groups of employees that manage the IS in the vaccine center: employees who maintain information about users and vaccination centers; employees who update vaccine entries at the centers; employees who update vaccine exits at the centers; and employees who update the data of vaccinated users.

The vaccine center manager wants a daily report about the users vaccinated and the number of vaccines available: number of vials of the vaccines' entries, number of vials of the vaccines' exits, and vaccinated users (divided by a factor of 5).

3.3.2. (P1) Business architecture

First, it is necessary to create is a glossary of specific terms for the IS. Then, the system needs a list of activities and actors. Thus, the matrix of Use-cases (using verbs) versus Actors (using nouns) is built as per Fig. 10.

The matrix of use-cases versus actors is the input to build the UML activity diagram. The UML activity diagrams with swim lanes are shown in Fig. 11. The manager of each center has information to control the number of vaccines available, used, and eventually wasted.

use-cases \ actors	users' employee	vaccine entries employee	vaccine exits employee	vaccine center manager	logistic hub
maintain vaccination centers					X
maintain types of vaccines					X
maintain users' information					X
maintain vaccinated users	X				
maintain vaccine entries		X			
maintain vaccine exits			X		
generate daily report				X	

Fig. 10. Matrix of Use-cases versus Actors.

3.3.3. (P2) Information architecture

Rule #0 uses the SI narrative to create the matrix use-cases versus Actors, accomplished by the functional analyst. On the other hand, rule #1 is performed by the systems analyst, who transforms the narrative into a UML class diagram (P2.a), shown in Fig. 12.

The look-up classes are Users, Vaccination Center and Vaccines. The intermediate class Vaccine-Entries links the classes Vaccines and Vaccine-Exists. Finally, the fact classes are the ternary class Vaccinated-users and the class Vaccine-Exists.

In the UML class diagram, the number of symbols should be as small as possible, where the Inheritance, Association, and Aggregation are presented. In the example, only associations link the classes.

By applying rule #2, the list of applications/use-cases can be found, corresponding to the P2.b step. In the following step, P2.c, the CRUD matrix is defined. Fig. 13 shows the CRUD matrix representing the relationship between applications and classes/data.

Rule #3 says a single application manages each data item, which the CRUD counters show as 1N11 type.

In the procedure to ensure alignment, multiple iterations can occur. If in a previous iteration the business narrative referred to the maintenance of the vaccinated users and vaccines exits, according to AD, the FR and DP would be as follows:

FR1: maintain vaccination centers.
FR2: maintain types of vaccines.
FR3: maintain users' information.
FR4: maintain vaccine entries.
FR5: maintain vaccinated users and vaccine exists.
DP1: vaccination centers.
DP2: vaccines.
DP3: users.
DP4: vaccinated users.
DP5: vaccine entries.
DP6: vaccine exits.

With five FRs and six DPs, the design matrix would be redundant, where FR5 maintains two design parameters, DP4 and DP6, as follows:

$$\begin{Bmatrix} FR1 \\ FR2 \\ FR3 \\ FR4 \\ FR5 \end{Bmatrix} = \begin{bmatrix} X & & & & & \\ & X & & & & \\ & & X & & & \\ & & & & X & \\ & & & X & & X \end{bmatrix} \cdot \begin{Bmatrix} DP1 \\ DP2 \\ DP3 \\ DP4 \\ DP5 \\ DP6 \end{Bmatrix}$$

According to AD, rule #4, an ideal design follows Independence Axiom. The design matrix should be re-designed into a square

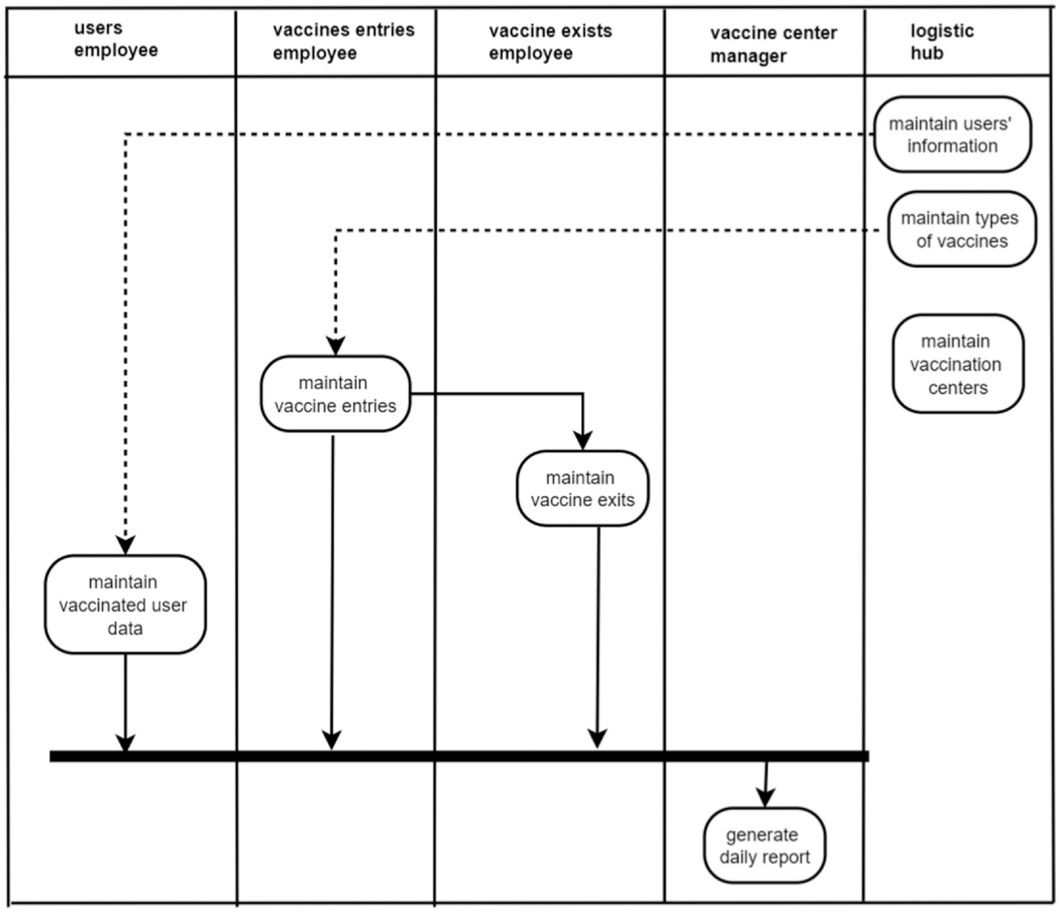


Fig. 11. UML activity diagram with swim-lanes.

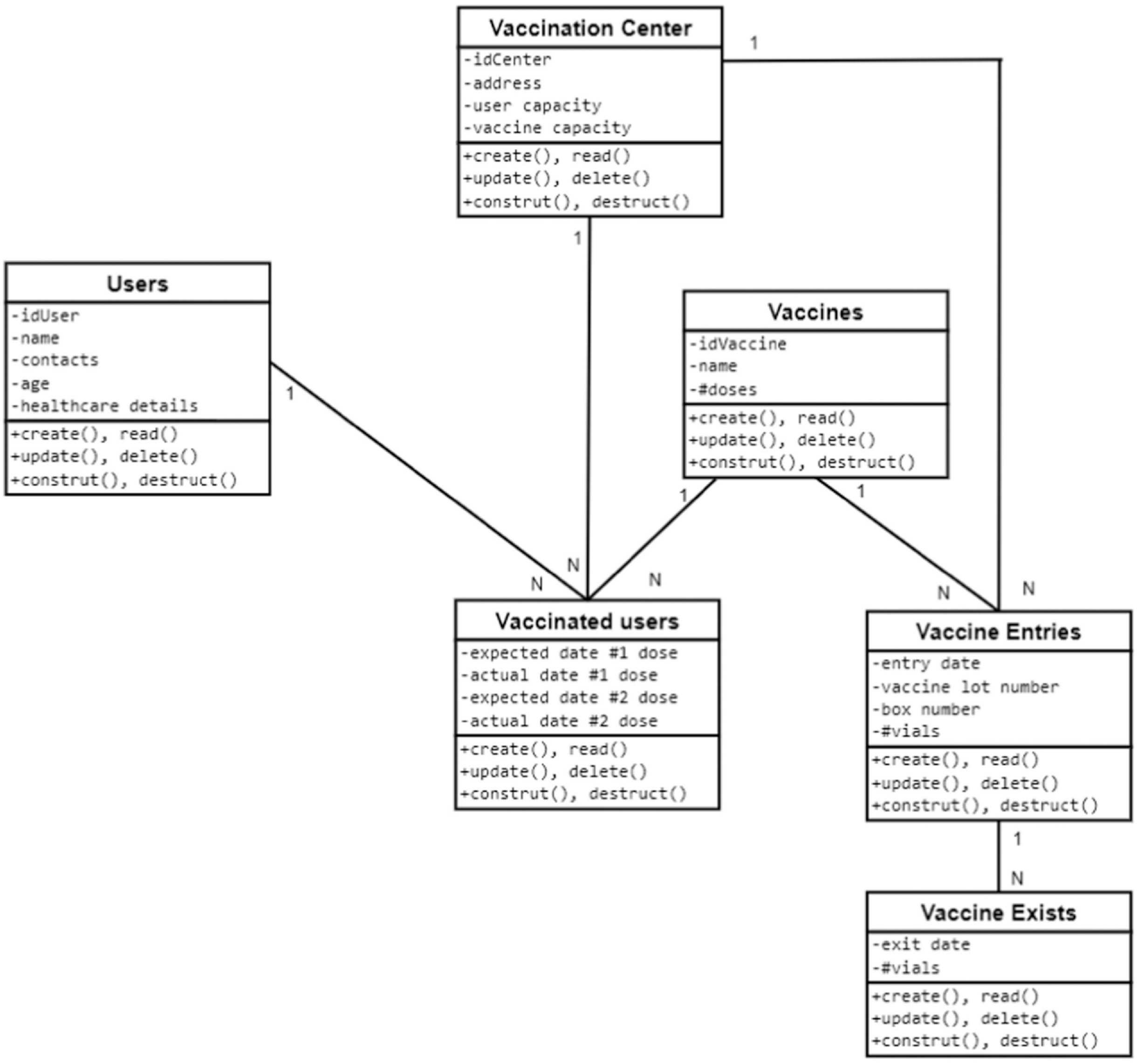


Fig. 12. UML class diagram.

use-cases \ classes	vaccination centers	vaccines	users	vaccinated users	vaccine entries	vaccine exits
maintain vaccination centers	CRUD					
maintain types of vaccines		CRUD				
maintain users' information			CRUD			
maintain vaccinated users		R	R	CRUD		
maintain vaccine entries		R			CRUD	
maintain vaccine exits		R			R	CRUD
generate daily report				R	R	R
CRUD counters	1111	1411	1211	1211	1311	1211

Fig. 13. CRUD matrix.

matrix by slitting FR5 into FR5a to maintain vaccinated users, and FR5b maintain vaccines exists, as shown in Fig. 12. The OO re-design can the performed over the class diagram.

In the example, a diagonal matrix was found. Each element of the diagonal is an object-oriented class re-designed that corresponds to the CRUD cells. Each CRUD cell can be presented in a Sequence Diagram at the intersection of a use-case and a data item, as defined in step P2.d.

As a CRUD cell example, Fig. 14 shows the CRUD cell of the Use-case maintaining vaccine that exists using three classes. The classes Vaccines and Vaccine-entries provide information, using the Read operator, and class Vaccine-exits is the target to perform the CRUD operators.

There is consistency with the CRUD matrix and the use-case diagram/ matrix (use-cases, actors), ensuring vertical alignment with two levels for each sequence diagram.

3.3.4. (P3) Technological architecture

This work proposes rule #4 to force the number of applications equal to the number of data items. Consequently, rule #4 re-designs the OO elements, ensures a good design, and establishes horizontal alignment with the business.

Rule #5 makes the concept of CRUD cells at the technologic level, or re-designed classes are called in this subsection 'service' using the SOA (service-oriented architecture) terminology. SOA consists of many modules that encapsulate well-delimited units of functionality, enabling data exchange with Web services (Romero and Vernadat, 2016).

There are advantages if services are loosely coupled to the system associated with the computing resource of virtual machines. The benefits of physical independence include performance isolation, easy resource management, and the on-demand deployment of

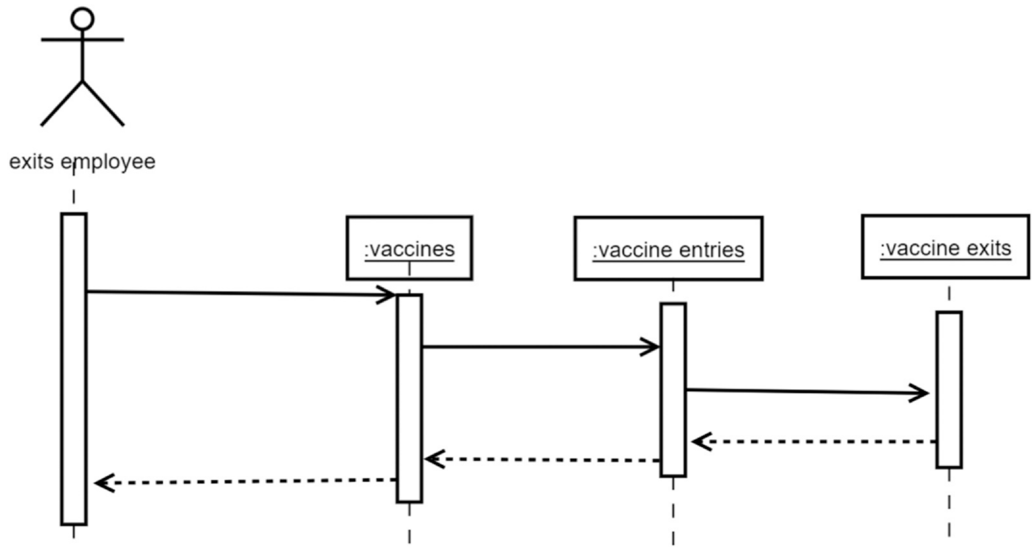


Fig. 14. UML sequence diagrams of the use-case maintain vaccine exists.

computing environments. Depending on the needs, different modules can be joined or withdrawn in this environment, plugged or unplugged from the system. It guarantees flexibility that allows the dynamic allocation of resources to balance the system's effective workload.

The service concept can be seen as an architecture paradigm that integrates complex engineering systems. Autonomy and loose coupling allow a system to adapt to the dynamic nature with permanent re-design needs.

In the technological architecture, the services can be specified using the UML deployment diagrams.

3.4. Summarizing the model

Business strategy alignment with IS and IT is a longstanding problem because of the increasing complexity of the designs. Moreover, the IS concept of a good design is still a subject of controversy. Adopting AD theory allows defining what a good design is in IS. AD helps to design the CRUD matrix of IS projects. As stated before, CRUD/AD means the CRUD matrix ensured by the Independence and Information Axioms.

The problem of many synonyms for the same element, depending on the viewpoint, is an IS issue. In the example, the actor has no synonym; the synonyms of activity are use-case and application; the synonym of the class is data-item; the synonyms of the object are CRUD cell and service.

In this model, the CRUD/AD matrix is the core of the Enterprise Architecture. The main inputs are the UML activity diagram and the UML class diagram. Its direct output is the UML sequence diagrams close to the applications that would run in an environment defined by the UML deployment diagrams.

To summarize the metric of the proposed Enterprise Architecture model, the elements belong to the three layers: Model (1st layer, 2nd layer, 3rd layer).

Then, the following elements are presented in the layer, where some of them are synonyms: Model ((actors, activities); (applications, classes); services).

In a balanced design, the number of activities should be equal to the number of applications (rule #1), the number of applications should be equal to the number of classes (rule #4), and finally, the number of services should be equal to the number of applications (rule #5). Thus, the metric of good design follows the expression Model ((M, N); (N, N); N).

The number of elements that express the model for the Vaccine case is ((5,6); (6,6); 6), with M = 5 and N = 6. Therefore, the repetition of N = 6 is according to the definition of a good design.

To obtain a unified view with the minimum jargon of the Enterprise Architecture, the model uses a minimal subset of the UML diagrams, shown in Table 2. Minimal jargon is a way to avoid errors in the use of the model.

4. Discussion

The CRUD/AD model is proposed in the previous section. The model suggests creating the CRUD/AD architect that maintains a unified view of the EA with the minimum jargon. The design software processes claim to use Design theories and Project Management. It is necessary to clarify the ontological differences between both.

This section discusses the following topics. First, the concepts of Design Theory and Project Management are distinguished. Then CRUD/AD architect is emphasized. Finally, Data Dictionary urges attention to increase the holistic view of the IS.

4.1. Design theory and project management

Hubka and Eder (1996) contribute to defining a taxonomy for Design Science. They proposed a two-by-two crossover framework. Technical systems and design processes cross with methodological declarations, descriptive and prescriptive. The descriptive description of the design allows defining the Design Process (or Project Management) in a socio-technical context. The prescriptive description includes methodologies and Theories of Design.

Agile methods are recent methodologies to foster a design. They help the design management, mainly by defining how the design team must work. However, they do not give a theoretical foundation

Table 2
A subset of UML diagrams.

Layer	UML diagrams and CRUD/AD
1) Business / Process Architecture	- UML use-case / activity diagram
2) Information System Architecture	- UML class diagram - CRUD/AD matrix - UML sequence diagram
3) Technological Architecture	- UML deployment diagram

to develop the product. Scrum is one of the most well-known Agile methods in the software industry. It promotes team creativity as well as offers rules for action (Weber et al., 2017). Scrum encourages iterative and incremental design development allowing changing the customer needs during the design process.

V-Modell (2021) has management mechanisms and tools for project execution. The management mechanisms include project organization and project planning. The project execution addresses the specification and subdivision at the V's downward branch, followed by realization and integration on the upward component.

Regarding the taxonomy mentioned above, Agile and V methods are mostly methodologies for design management. However, both methods are in the middle of the prescriptive and descriptive definition of the design process. On the contrary, AD is a pure design theory. The need for integration between Agile and V methods with AD comes from an industrial need. However, integration is not an ontological need and may cause interference in theory application. Using AD with Agile methods may create coupling and controversy. Thus, design can start using AD rules and end in the latter stages using Agile methods (Puik and Ceglarek, 2018). It is a way to decouple methodological declarations from prescriptive ones.

In any case, AD can benefit from using Agile and V-Modell methods to succeed in enterprise applications. Furthermore, AD can benefit from the integration fundamentals of V-Modell. The advantage of Agile and V-Modell methods is providing a methodology for creating a design team and guiding it to make it work. It is essential as an innovation method, but it does not provide a way to evaluate a solution. AD provides a way to classify a solution.

AD can help to create methodologies as it is a Design theory. Therefore, AD can help create design guidelines for production methods (Rauch et al., 2015). Project management might be independent of the Design theory.

In this work, the CRUD/AD method, with the six rules (#0.#5), is an example of Design theory. On the other hand, the CRUD/AD procedure, with four basic steps (P0, P1, P2, P3), gets closer to Project management.

4.2. CRUD/AD architect

This paper aims to define an Enterprise Architecture with vertical and horizontal alignments to ensure consistency. It uses UML, the CRUD matrix, and Axiomatic Design as a theoretical framework. The CRUD/AD matrix is the core of the IS design. Regarding subjects closer to Project management in the CRUD/AD model, two appointments should be added about the role of the CRUD/AD architect and the relevance of the Data Dictionary.

Some methodologies use the intricate concepts of viewpoint and view, integrating diverse architecture descriptions. A viewpoint essentially corresponds to the concepts, models, and techniques. Moreover, a view usually focuses on the concerns of the stakeholders. As an illustration, the methodologies of TOGAF and ArchiMate include 30 views and 20 viewpoints, respectively (Lankhorst, 2013). Thus, it generates a long and multifaceted dialect that is hard to apply.

IS has two classic views of the enterprise, the functional analyst view and the system analyst view. The functional analyst creates a bridge between the final users and the IS, specifying requirements, defining business processes, and realigning roles. UML activity diagrams are a standard tool for functional analysts. On the other hand, the system analyst is responsible for specifying the transformation of the business narrative, based on documents and requirements, into a software model. An example of a system analyst tool is a UML class diagram. Thus, the understating of the business generates two diverse views of IS, the functional and system analysts. The CRUD matrix can merge both views.

The proposed model asks for a CRUD/AD architect. The CRUD/AD architect summarizes the applications and data views and enhances the system using Axiomatic Design. In other words, the CRUD/AD architect generates the EA integration, creating a core between the three enterprise layers. The business functions and data items are aggregated in the CRUD/AD cells. The AD concept of ideal design is essential to make it possible. Each CRUD/AD cell is then deployed in the technological layer, ensuring a loosely coupling.

4.3. Data dictionary

In the early database systems, the normalization of names in the IS was encouraged to define all the enterprise names in the Data Dictionary. A data dictionary collects the names, definitions, and attributes of the data items, the application, the actors, and the physical elements. The data dictionary is more recently known as metadata, i.e., the data about the data. Given the multiple viewpoints for the same subject, this work suggests a glossary of terms of the enterprise to manage the synonymous.

With the evolution of search capacities in the IS, the data dictionary has lost relevance. Nowadays, the emphasis is given to iterative and interactive perspectives, including the version control automated, quality assurance, and agile tools, disregarding the data dictionary. As a result, the IS has unavoidable consequences of the duplication and redundancy of the central elements.

To summarize this section, we consider that the CRUD/AD architect assures the system's stability against the hustle and bustle of daily life, maintaining EA design with Data Dictionary accountability. The new CRUD/AD role supported by the business narrative and mission should integrate:

- the Data Dictionary tool: data, applications, actors and technological support;
- the consistency of the system using the CRUD/AD method.

This work proposes a CRUD/AD architect assisted with the Data Dictionary. The Project Management phase benefit from it. CRUD/AD can coexist with complementary methodologies such as Agile and V-Modell.

5. Conclusions

Organizations are getting more global, more extensive, with more relationships, and consequently more complex. Therefore, a unique and holistic view of Enterprise Architecture (EA) is increasingly necessary.

However, scientific literature proposes solutions with many synonyms depending on the layer, methodology, framework, or tool to answer these challenges. There is not a broad view of the system. Additionally, new languages evolve into more specific fields, increasing the lexical complexity and losing the necessary simplicity of the architecture. Instead of creating an overview of the organization, most EA methodologies create new diagrams with new symbols that require many hours of learning. Recent studies conclude that the EA process is costly and subject to errors, causing enterprises not to adopt EA.

The proposed procedure that checks the alignment in EA has three basic heuristic rules that guarantee the alignment of the Information System (IS). The proposed model extended the rules by connecting the UML diagram, the CRUD matrix, and the Axiomatic Design (AD) theory.

EA, founded on the three layers, Business/ Process Architecture, Information System Architecture, and Technological Architecture, promises companies to deal with digital transformation. The three layers approach allows a clear presentation of EA. Moreover, it permits horizontal and vertical alignment of business and information technology (IT) holistically.

This work proposes the CRUD/AD method. The approach deals with the three layers of EA, the four elements of the Iron Cross, and the six rules for organizational alignment. The model uses UML diagrams since a vast community of professionals commonly uses them. Although the CRUD matrix is not a UML tool, it can complete a system view. The proposed model firstly defines a set of UML diagrams for each layer of the EA. Then heuristic six rules are detailed to ensure vertical and horizontal alignment.

This work attempts to reduce and standardize the multiple synonyms in IS and establish rules to find good practices in design. We integrate UML diagrams with CRUD from an EA perspective, allowing a holistic view of the organization. The addition of AD ensures a comprehensive, concise, and consistent design.

The proposed procedure has three steps to check the alignment in the EA, each procedure for each organizational layer. Firstly, the business narrative was defined. Then, business architecture, information architecture, and technological architecture apply the (1) use-case/activity diagrams, (2) class diagrams, CRUD matrix with sequence diagrams to detail the applications, and (3) infrastructure diagram, respectively. The steps between diagrams are supported by heuristic rules, allowing the alignment of the system.

The number of elements in each layer can summarize the CRUD/AD model: Model ((actors, activities); (applications, classes); services). A good design has the same number (N) of activities, applications, classes, and services. A metric of good design is synthesized through the expression Model ((M, N); (N, N); N).

We believe the model gives a holistic view of the EA through the three layers. Moreover, the proposal of the CRUD/AD architect provides the needed integration of the IS by creating a core between the layers.

CRediT authorship contribution statement

Luís Cavique: Conceptualization, IS Methodology, Supervision. **Mariana Cavique:** Writing, Writing – review & editing. **Armando B. Mendes:** Writing – review & editing. **Miguel Cavique:** AD Methodology.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Barros, A., Duddy, K., Lawley, M., Milosevic, Z., Raymond, K., Wood, A., 2000, Processes, Roles, and Events: UML Concepts for Enterprise Architecture, In: Evans A. , Kent S., Selic B. (eds), UML 2000: The Unified Modeling Language, Lecture Notes in Computer Science, Springer, vol. 1939, pp. 62–77.
Cavique, L., Cavique, M., 2021, Axiomatic Design applied to CRUD matrix in Information Systems, the 14th International Conference on Axiomatic Design, ICAD, Lisboa, Portugal.
Cavique, L., Cavique, M., Mendes, A.B., 2021. Integration of UML diagrams from the perspective of enterprise architecture. In: Rocha, Á., Adeli, H., Dzemyda, G., Moreira, F., Ramalho Correia, A.M. (Eds.), Trends and Applications in Information Systems and Technologies. WorldCIST 2021. Advances in Intelligent Systems and Computing, vol. 1366 Springer, Cham. https://doi.org/10.1007/978-3-030-72651-5_44
Cavique, L., 2020, Modelação de Sistemas de Informação: Elementos essenciais na visão integrada das ferramentas do UML com a matriz CRUD [Information Systems Modeling: Essential elements in the integrated view of UML tools with the CRUD matrix], Recursos Educativos, Universidade Aberta, Portugal.
Desfray, P., Raymond, G., 2014, Modeling Enterprise Architecture with TOGAF: A Practical Guide Using UML and BPMN, Elsevier Inc., ISBN 978–0-12–419984-.
Fowler, M., 2003, UML Distilled: A Brief Guide to the Standard Object Modeling Language, Addison-Wesley Professional, 3rd Edition, ISBN: 978–032-119–368-1.
Gonçalves-Coelho, A.M., Neştian, G., Cavique, M., Mourão, A., 2012a. Tackling with redundant design solutions through axiomatic design. Int. J. Precis. Eng. Manuf. 13, 1837–1843. https://doi.org/10.1007/s12541-012-0241-x
Gonçalves-Coelho, A.M., Neştian, G., Cavique, M., Mourão, A., 2012b. Tackling with redundant design solutions through axiomatic design. Int. J. Precis. Eng. Manuf., vol. 13, 1837–1843. https://doi.org/10.1007/s12541-012-0241-x
Hinkelmann, K., Gerber, A., Karagiannis, D., Thoenssen, B., Merwe, A., Woitsch, R., 2016. A new paradigm for the continuous alignment of business and IT: Combining enterprise architecture modeling and enterprise ontology. Comput. Ind., vol.79, 77–86. https://doi.org/10.1016/j.compind.2015.07.009
Hubka, V., Eder, W., 1996. Design Science. Springer-Verlag, London Limited.
IBM Corporation, 1978, Business System Planning Information - System Planning Guide, International Business Machines Corporation, 2nd edition, New York.
Krasner, H., 2021, The cost of poor software quality in the U.S.: a 2020 report, Consortium for Information & Software Quality.
Lankhorst, M., 2013. Enterprise Architecture at Work: Modelling, Communication and Analysis, 3rd ed. Springer ISBN: 978364-229-650-5.
Martin, J., 1983. Managing the Database Environment. Prentice-Hall, Englewood Cliffs, New Jersey ISBN: 013-550-582-8.
Pereira, C.M., Sousa, P., 2005. Enterprise architecture: business and IT alignment. In: Haddad, H., Liebrock, L.M., Omicini, A., Wainwright, R.L. (Eds.), SAC. ACM, pp. 1344–1345.
Perez-Castillo, R., Ruiz-Gonzalez, F., Genero, M., Piattini, M., 2019. A systematic mapping study on enterprise architecture mining. Enterp. Inf. Syst., vol. 13 (5), 675–718.
Puik, E., Ceglarek, D., 2018, Application of Axiomatic Design for Agile Product Development, MATEC Web of Conferences 223, 01004, 12th International Conference on Axiomatic Design, ICAD, https://doi.org/10.1051/matecconf/201822301004.
Rauch, E., Dallasega, P. and Matt, D., 2015, Axiomatic Design based Guidelines for the Design of a Lean Product Development Process, 9th International Conference on Axiomatic Design, Procedia CIRP 34, p. 112 – 118.
Rocha, Á., Freixo, J., 2015. Information architecture for quality management support in hospitals. J. Med. Syst. 39, 125.
Romero, D., Vernadat, F., 2016. Enterprise information systems state of the art: past, present, and future trends. Comput. Ind., vol. 79, 3–13.
Silingas, D., Butleris, R., 2009, Towards customizing UML tools for enterprise architecture modeling, In: Nunes M.B., P. Isaías, P. Powell (eds), IADIS International Conference Information Systems, ISBN: 978–972-8924–79-9.
Spewak, S., Hill, S.C., 1995. Enterprise Architecture Planning: Developing a Blueprint for Data, Applications, and Technology. John Wiley & Sons, New York City.
Suh, N.P., 1990. The Principles of Design. Oxford University Press.
Suh, N.P., 1995. Design and operation of large systems. J. Manuf. Syst., vol. 14 (3)), 203–213.
Suh, N.P., 2005. Complexity: Theory and Applications. Oxford University Press, New York.
Suh, N.P., 2001. Axiomatic Design: Advances and Applications. Oxford University Press ISBN 019-513466-4.
The Standish Group International, 2010, Modernization, clearing a pathway to success.
Thomas, J., Mantri, P., 2015, Axiomatic Design/Design Patterns Mashup: Part 1 (Theory), the 9th International Conference on Axiomatic Design, ICAD, Procedia, Elsevier, vol. 34, pp. 269–275.
TOGAF, 2011, The open group architecture framework (Version 9.1), The Open Group.
V-Modell, X.T., 2021, Part 1: Fundamentals of the V-Modell, accessed September 2021, ⟨ftp://ftp.heise.de/pub/ix/projektmanagement/vmodell/V-Modell-XT-Gesamt-Englisch-V1.3.pdf⟩.
Valorinta, M., 2011. IT alignment and the boundaries of the IT function. J. Inf. Technol., vol. 26, 46–59.
Van-Bon, J., Verheijen, T., 2006, Frameworks for IT Management: an Introduction, ITSM Library, Publisher: Van Haren, ISBN-13: 978–9077212905.
Ward, J., Peppard, J., 2002. Strategic Planning for Information Systems, 3rd edition.,. John Wiley and Sons Ltd. ISBN: 0470841478.
Weber, J., Förster, D., Stäbler, M. and Paetzold, K., 2017, Adapt! – Agile Project Management Supported by Axiomatic Design, MATEC Web of Conferences 127, 01018, 12th International Conference on Axiomatic Design, ICAD 2017, https://doi.org/10.1051/matecconf/201712701018.
Zachman, J.A., 1987. A framework for information systems architecture. IBM Syst. J. 26 (3), 276–292.