Axiomatic Design applied to CRUD matrix in Information Systems

Luís Cavique[1,2,5], Miguel Cavique[3, 4]

[1] Universidade Aberta, Rua da Escola Politécnica, 147, 1269-011 Lisboa, PORTUGAL
[2] LASIGE, Faculdade Ciências de Lisboa, Campo Grande, 1749-016 Lisboa, PORTUGAL
[3] R&D Unit for Mechanical and Industrial Engineering - UNIDEMI, NOVA School of Science and Technology, Campus de Caparica, 2829-516 Caparica, PORTUGAL
[4] Naval Academy, Base Naval de Lisboa, Alfeite, 2810-001 Almada, PORTUGAL
[5] Corresponding author, mailto: luis.cavique@uab.pt

Abstract. In Information Systems (IS), the search for a good design is still a relevant issue. In IS, two sub-systems coexist, applications and data. To articulate these sub-systems, some authors opt for the CRUD matrix. Similarly, Axiomatic Design (AD) theory studies how functional requirements are related to design parameters using a design matrix. In this paper, the application sub-system corresponds to the functional requirements, the data sub-system to the design parameters, and the CRUD matrix to the design matrix. The goal of the CRUD matrix is to maintain the independence of its items to minimize the information content of the design. Similarly, AD uses the design matrix to define a good design. This work aims to develop a theory to create object-oriented elements based on the CRUD matrix aligned with the business strategy.

1. Introduction

Axiomatic Design (AD) theory is a powerful systems design methodology based on matrix methods that analyze the pipeline from customer needs to process variables. For a thorough explanation see [11], [12], [13], [14].

Information Systems (IS) can be seen as a simple input/output mechanism that collects, stores, and distributes information. However, new legislation, new entities, and new interactions create new challenges in information management. A significant effort is given to align the business strategy with the technology, i.e., the process in which a business organization uses information technology to achieve business objectives [8].

Despite the high number of methodologies [16] in IS, the search for a good design is still a relevant issue, and the concept of a good design is still a subject of controversy. However, the essential SI elements remain constant. BSP (Business Systems Planning) [7] presented four essential elements for Information Systems Planning, coined as the Iron Cross: organization (people), applications (software), data, and technological systems (hardware).

To articulate the elements, applications, and data, we opt for the CRUD matrix. CRUD, achromic of <create, read, update, delete>, means the four basic operations possible on a data value. In SQL it corresponds to <insert, select, update, delete>. CRUD matrix was developed by IBM [7] and popularized by James Martin [9] in his book 'Managing the Database Environment'.

The CRUD matrix is an essential tool to perform diagnostics in software. Unfortunately, there are several practical examples with the CRUD matrix that corresponds to coupled designs. In [1] the authors use the CRUD matrix to know the current state (as-is) and to model the improved future state (to-be). In the current state, the CRUD matrix shows a poor design. After swapping rows and columns, cells are aggregated into macro-cells in a decoupled matrix in the improved state.

The CRUD matrix is still beneficial when integrated with more recent tools, like UML (Unified Modeling Language) [5]. UML in analysis and design allows the fulfillment of the system requirements with object-oriented design and relational databases models.

This work aims to add to CRUD the scientific support given by the Axiomatic Design theory. The design matrix brings value to the project implementation in IS. This work contributes to developing a theory to create object-oriented elements based on the CRUD matrix and aligned with the business strategy.

In a similar approach in enterprise architectures [2], the authors use data as FRs and applications as DPs. In this work, we follow the approach presented in chapter 5, Suh [13], where FRs correspond to applications and DPs to data.

The paper is organized into three additional sections. Section 2 introduces the CRUD matrix in information systems. Section 3 details the proposed CRUD design. Finally, in Section 4, conclusions are drawn.

2. CRUD Matrix

Enterprise Architecture is usually defined by the three-layers model [8]: business/process architecture, information system architecture, and technological architecture, as shown in Figure 1.

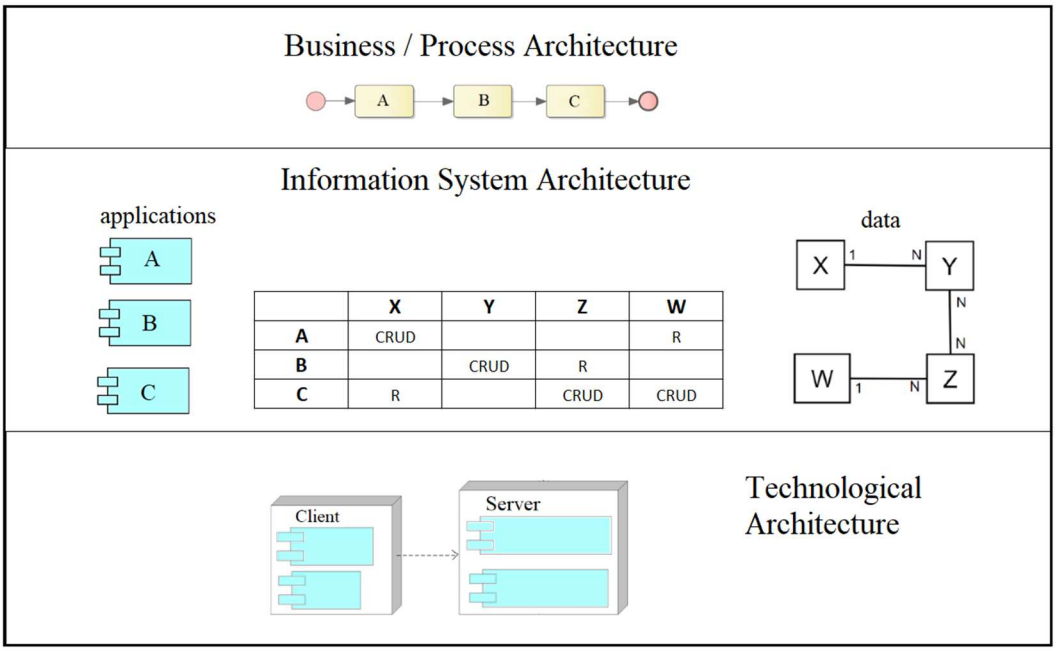

Figure 1. Three-layers model of Enterprise Architecture

In Enterprise Architecture, many synonyms depend on the layer, methodology, framework, or tool used, so it is crucial to find a synthesis with a reduced number of names.

In Cavique et al. [3], only three essential elements were chosen: the actors, the activities, and the data. Some of the synonyms are as follows: (i) actor is a synonym of lines-of-responsibility, (ii) activities is about applications, tasks, uses-cases, or operational-processes, (iii) data is a synonym of classes or informational entities.

In the following examples, we will use actors (α, β), activities (A, B, C), and data (X, Y, Z, W). This version of Enterprise Architecture with three layers includes:

- 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).

The CRUD matrix plays a vital role in the Information System Architecture by combining data between applications (A, B, C) and data (X, Y, Z, W). In a relational data model approach, data (X, Y, Z, W) correspond to tables of a database and the applications (A, B, C) to operations that maintain the data sets.

To develop straightforward projects, the authors Pereira and Sousa [10] and Vasconcelos [17] presented a set of three heuristic rules that guarantee the alignment of the information system. The heuristic rules can be summarized as follows:

#1 Each processing activity of the business process is automated by a single application;
#2 The data items must support the processes activities of the business processes;
#3 Each data item is managed (CUD) by a single application in the CRUD matrix.

In this work, rule #3 emphasizes developing a more managed CRUD matrix since data items updated by different applications are harder to manage.

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

Figure 2. CRUD matrix with counters

To ensure rule #3, Figure 2 shows the CRUD matrix with CRUD counters that validate possible inconsistencies [4]. Counter 1211 indicates that there are 1 Create, 2 Reads, 1 Update, and 1 Delete. There must be a single application that performs Create, Update and Delete, (CUD), and there may be multiple applications with the Read operator.

Preferably, the CRUD counters should be 1N11, i.e., a unique Create, Update and Delete and multiple Read operators. As previously stated, Read operations correspond to the SQL Select and usually involve many tables to obtain new information. The multiple Reads are allowed and do not produce concurrence; therefore, they will not be considered in the rest of our analysis.

The CRUD approach intends to obtain a close view of the system to avoid coupled sub-systems by finding a matrix with the main diagonal filled with the least number of elements.

With the three-layer model, Enterprise Architecture presents similarities with Axiomatic Design, which uses three mapping zones. Firstly, Axiomatic Design studies the transformation of the customer attributes into functional requirements, which corresponds to the transformation of activities into applications. Then, Axiomatic Design maps the functional requirements with a set of design parameters using the design matrix, corresponding to the CRUD matrix matching, between applications and data.

3. Proposed CRUD design

This section focuses on how Axiomatic Design can improve and better justify the CRUD structure in information systems projects. After clarifying the design matrix, the transformation to objected-oriented design is shown. Finally, we discuss the similarities between AD and IS rules.

3.1. Axiomatic Design approach

This section presents a practical example of the concepts presented in chapter 5, Axiomatic Design of Software [13].

In AD, Functional Domain and Physical Domain are associated as follows: functional requirements {FRs} and design parameters {DPs}. The most common designs maps between FRs and DPs, expressed by the design equation {FRs} =[A].{DPs}. Each DP must fulfill an FR, so the design matrix [A] 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 [15]. In the design matrix, the cells with operators CRUD are replaced by an X. Such as the axioms of AD, the goal of the CRUD matrix is to maintain the independence of the functional requirements, which usually correspond to minimize the information content of the design.

When there are more DPs than FRs, the design is either a redundant design or a coupled design, as stated by Theorem 3 [11]. Gonçalves-Coelho et al. [6] claim that redundancy designs belong to particular kinds of designs and propose seven theorems. Figure 2 shows a redundant design, as application C operates on tables Z and W. According to AD, the design equation is by Equation 1:

$$\begin{Bmatrix} FR\_A \\ FR\_B \\ FR\_C \end{Bmatrix} = \begin{bmatrix} X & & \\ & X & \\ & & X \quad X \end{bmatrix} . \begin{Bmatrix} DP\_X \\ DP\_Y \\ DP\_Z \\ DP\_W \end{Bmatrix} \tag{1}$$

However, an application makes a change on a table and then on another table. Therefore, 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*. Equation 2 shows the mentioned redundant design:

$$\{FR\_C\} = [X \quad X] . \begin{Bmatrix} DP\_Z \\ DP\_W \end{Bmatrix} \tag{2}$$

Equation 2 shows the first state of operation on Equation 3.

$$\{FR\_C1\} = [X \qquad] . \begin{Bmatrix} DP\_Z \\ DP\_W \end{Bmatrix} \tag{3}$$

And the other state by Equation 4.

$$\{FR\_C2\} = [\qquad X] . \begin{Bmatrix} DP\_Z \\ DP\_W \end{Bmatrix} \tag{4}$$

Application C can split in *C1* when addressing table *Z* and *C2* when addressing table *W*. It makes all applications belong to the same level of decomposing. Therefore, the design matrix can turn into a square matrix by the decomposition of FR-C as shown in Figure 3, using a square sub-matrix, where FR-C is decomposed into FR-C1 and FR-C2.

		DP-X	DP-Y	DP-Z	DP-W
FR-A		X			
FR-B			X		
FR-C	FR-C1			X	
	FR-C2				X

Figure 3. Decomposition of FR-C

3.2. Object-oriented design

Object-oriented (OO) techniques are some of the most potent paradigms widely used in computer science 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 crucial features: abstraction, encapsulation, inheritance, and modularity.

Object-oriented techniques encapsulate data and the operations that manipulate that data in a specific object. OO paradigm allows a higher level of abstraction, focusing only on those characteristics available by the object's interface and omitting the details of data and operations. The system's modularity is also achieved by decomposing each problem into loosely coupled intra-objects so that internal changes to one object do not affect another one.

In this work, the data items can be enriched with the OO approach. Four objects {A.X, B.Y, C1.Z, C2.W} can be found in the design matrix. Each cross ('X') in the design matrix [A], in Figure 3, corresponds to an object. Figure 4 shows the object {A.X} that associates the data DF-X with the operations/application FR-A.

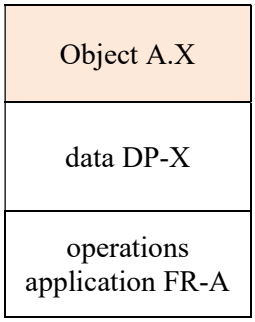

Figure 4. Object A.X

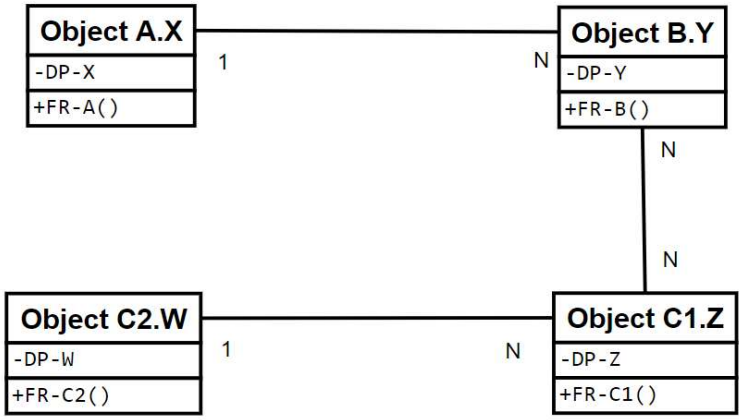

Figure 5. Object-oriented design supported by AD

The data schema presents in Figure 1 can be enriched with the functional requirements. The result is presented in Figure 5, where relations link the four objects. The object-oriented design supported by AD reinforces the OO paradigm to incorporate the advantages of modularity and reuse, providing savings in the software project development.

3.3. Discussion of the two approaches

Since there are similarities between IS and AD, some relationships between the axioms of Axiomatic Design and the heuristic rules in Information Systems are detailed in Table 1. Using AD theorems, a new rule, number 4, is proposed.

Table 1. Heuristic rules versus Axiomatic Design

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

Rule #1, a single application that automates each process activity, also corresponds to the mapping of Customer domain (process activities) and Functional domain (applications).

Rule #2, the data items must support the process activities of the business processes, or in order words, data items must support the applications.

Rule #3, each data item is managed by a single application in the CRUD matrix, as already seen, almost corresponds to the design matrix. In the transformation, two additional issues should be mentioned.

On the one hand, AD imposes a square design matrix, according to Theorem 4 [11]. Theorem 4 states that the number of DPs is equal to the number of FRs in an ideal design, so the CRUD should be square. To overcome this issue, we add rule #4, where the number of applications is equal to the number of data items.

On the other hand, using the CRUD counters, the CRUD matrix only allows uncoupled designs, showing decoupled matrices are not adequate for IS. This statement agrees with Theorem Soft 1 [13] that uncoupled software or hardware systems can be operated without precise knowledge of the design elements (i.e., modules) if the design is uncoupled.

Heuristic rules #1 and #2 ensure the vertical alignment with the business processes, and rule #3 ensure the horizontal alignment between applications and data.

4. Conclusions

Business strategy alignment with Information Systems and Information Technology is a longstanding problem because of the increasing complexity of the designs. In Information Systems (IS), the horizontal alignment of applications and data items is possible with the CRUD matrix.

However, regarding IS, the concept of a good design is still a subject of controversy. Adopting Axiomatic Design theory validates what a good design can be in information systems. AD helps to justify and simplify the design of the CRUD matrix of IS projects. We propose to adopt rule #4 that makes the number of applications equal to the number of data items, to create object-oriented modules.

By applying heuristic rule #3, the CRUD matrix can return a redundant matrix with more DPs than FRs. The re-design of the project using Axiomatic Design, by adding rule #4, allows creating object-oriented elements, improving the data design, and ensuring alignment with the business.

References

[1] Alves P, Vasconcelos A, Silva A 2011 Arquitecturas de sistemas de informação de referência para atendimento multicanal na administração pública [Reference information system architectures for multichannel service in public administration], *Proceedings of the Portuguese Association for Information Systems Conference*, Portugal

[2] Behrouza F, Fathollahb M 2016 A systematic approach to enterprise architecture using axiomatic design, *the 10th International Conference on Axiomatic Design, ICAD, Procedia*, Elsevier, **53** pp 158-165

[3] Cavique L, Cavique M, Mendes AB 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*, **1366**, Springer, Cham. https://doi.org/10.1007/978-3-030-72651-5_44

[4] 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

[5] 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

[6] Gonçalves-Coelho AM, Neştian G, Cavique M, Mourão A 2012 Tackling with redundant design solutions through axiomatic design, *International Journal of Precision Engineering and Manufacturing*, **13**, pp 1837–1843, https://doi.org/10.1007/s12541-012-0241-x

[7] IBM Corporation 1978 *Business System Planning Information - System Planning Guide*, International Business Machines Corporation, 2nd edition, New York

[8] Lankhorst, M 2013 *Enterprise Architecture at Work: Modelling, Communication and Analysis*, 3rd Editions, Springer, ISBN: 978-364-229-650-5

[9] Martin, J 1983 *Managing the Database Environment*, Prentice-Hall, Englewood Cliffs, New Jersey, ISBN: 013-550-582-8

[10] Pereira CM, Sousa P 2005 Enterprise architecture: business and IT alignment, In H. Haddad, L.M. Liebrock, A. Omicini, R.L. Wainwright, eds., SAC, pp 1344–1345, ACM

[11] Suh NP 1990 *The Principles of Design* (New York: Oxford University Press)

[12] Suh NP 1995 Design and operation of large systems, *Journal Manufacturing Systems*, 14 (3)

[13] Suh NP 2001 *Axiomatic Design: Advances and Applications* (New York: Oxford University Press)

[14] Suh NP 2005 *Complexity: Theory and Applications* (New York: Oxford University Press)

[15] Thomas J, Mantri P 2015 Axiomatic Design/Design Patterns Mashup: Part 1 (Theory), *the 9th International Conference on Axiomatic Design, ICAD, Procedia*, Elsevier, **34**, pp 269-275

[16] Van-Bon J, Verheijen T 2006 *Frameworks for IT Management: an Introduction*, ITSM Library, Publisher:Van Haren, ISBN-13 : 978-9077212905

[17] Vasconcelos, A 2017 Slides Aulas Arquiteturas Tecnológicas Empresariais [Slides of the Classes in Business Technological Architectures], DEI, Instituto Superior Técnico, Universidade Lisboa, Portugal