International Journal of Computer Applications (0975 – 8887)
Volume 176 – No. 28, June 2020

A Decoupled Health Software Architecture using Microservices and OpenEHR Archetypes

Marcio Alexandre Pereira da Silva
Center for Informatics
Federal University of Pernambuco, Recife/PE, Brazil

Valéria Cesário Times
Center for Informatics
Federal University of Pernambuco, Recife/PE, Brazil

André Magno de Costa Araújo
Department of Information Systems
Federal University of Alagoas, Penedo/AL, Brazil

Paulo Caetano da Silva
Master Program in Computing
Salvador University, Salvador/BA, Brazi

ABSTRACT

Coupling is a challenge in software engineering, because errors or failures compromise the whole software execution. To address this issue in the healthcare domain, a Decoupled Health Software Architecture (DHSA) is proposed in this paper. This study presents the development of three components, a tool, and a formal metric. The Connector, Container and Archetype-based microservice (Archemicro) components make the DHSA, which is dynamically generated by the Microservice4EHR tool. For assessing, a legacy software used in Brazilian hospitals is migrated to the DHSA. A comparison is performed between three tools (MARCIA, Template4EHR, and EhrScape). The Archetype-based Software Architecture Coupling (ASAC) is a formal metric to measure the coupling level of Health software architectures. As a result, DHSA increases by 66,6% the decoupling index of the healthcare software. The healthcare domain therefore is benefited with a software architecture that maintains the software operation even if a component from the software architecture causes errors.

General Terms

Distributed Software Architecture.

Keywords

Health Information Systems (HIS); Distributed Software Architecture; Electronic Health Record (EHR); OpenEHR Archetypes; Microservices.

1. INTRODUCTION

Software engineering is the systematic application of engineering methodologies and approaches to the development of software from any domain. In turn, software development is a process by which software components are created using tools such as computer languages, libraries, and frameworks [1]. The union of these components are called software architecture. In literature, software architecture consists in defining the components of a software, their external properties and their relationships with other components [2]. There is a consensus that software architecture plays a central role in software development and an important role in the life cycle phases after software delivery [3].

For decades, this has been a challenge: How to specify the components of a software architecture in a way that software execution is not interrupted by an unexpected error or failure? One solution is to create software architectures based on "decoupled components", which means that all these components are dissociated, as in not interconnected or interdependent [4].

A decoupled software architecture allows (i) different parts of the software to perform its tasks independently, and (ii) its components remain completely autonomous and unaware of each other. Thus, unexpected behavior in one component should not impact any other components, because each component operates separately and its functionality is self-contained [5]. Historically, a way of building decoupled architectures has been the adoption of services [6]. Initially, organizations have used a finite set of services to execute their business, creating a new architecture style called SOA (Service-Oriented Architecture). SOA is the adoption of a set of software that performs as a service [7].

Due to the need for improvement and evolution in the way software architectures are designed, a new approach has emerged in software engineering. Known as microservices, this approach decomposes applications in basic functions, making them more decoupled. Each function operates as a service and can be implemented and deployed independently. This means that each individual service can run or experience unexpected behavior without compromising others. The microservice architecture proposes an application that can be decomposed in a set of microservices, giving the software a decoupled approach in relation to the software architecture [8].

In the healthcare software domain, decoupling is also important. The openEHR Foundation [9] has specified a standard called Archetype in order to provide low coupling at the data level in healthcare software. Archetype gives autonomy to healthcare data, because it represents it in an independent and standardized way [9]. However, most healthcare software applications are strongly coupled at the architecture level [10, 11]). Some tools reported in literature bring important contributions to state-of-art and state-of-practice in the healthcare domain [10, 11, 12]. However, these tools generate Health software architectures from a set of archetypes. There are some limitations about the decoupling of their generated software architectures. For example, if there is unexpected behavior (e.g., errors or failures) in any part of the software architecture, it impacts and interrupts the whole software system execution.

Thus, this paper proposes a Decoupled Health Software Architecture (DHSA) to support archetype-based Health software. This proposal specifies components that perform and interact through a web environment. These components are Connector, Container and Archetype-based Microservice

(Archemicro), which are detailed in Section 3. A tool called Microservice4EHR has also been developed to enable the redesign of Health legacy software into the DHSA. In this way, a legacy Health software can migrate its software architecture to a new decoupled one such as DHSA.

In order to validate the proposed solution in this paper, a real-world institution operating under the Brazilian Health Ministry [13] has been used to assess the decoupling of the software architecture built by cutting-edge tools such as: MARCIA [10], Template4EHR [11], EhrScape [12], and Microservice4EHR, the latter being proposed in this paper. To measure a coupling index of the software architectures, a metric called Archetype-based Software Architecture Coupling (ASAC) calculates the coupling index.

This paper is organized as follows: Section 2 describes the basic concepts used to develop this work, while Section 3 presents and describes the DHSA and the Microservice4EHR tool. Section 4 shows the assessment on a DHSA-based software architecture and three other software architectures generated by state-of-the-art tools. The main results are debated in Section 5. Finally, the conclusion and suggestions for future works are presented in Section 6.

2. BACKGROUND AND RELATED WORKS

This section contextualizes openEHR archetypes (Section 2.1), conceptualizes monolithic and microservice architectures (Section 2.2), and provides an analysis of related works identified both in Academia and Industry (Section 2.3).

2.1 OpenEHR Archetype

The openEHR Foundation – an international organization composed of several Health domain specialists - has specified archetype as a standard for the computational representation of the Electronic Health Record (EHR) [9].

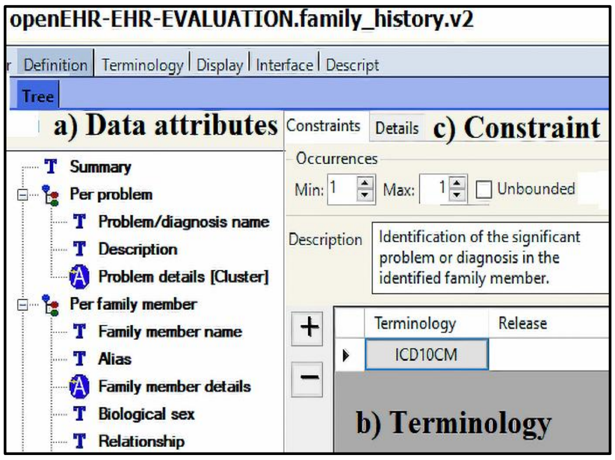
An archetype is based on a dual model to create a unique EHR, preserving the history and evolution of patient clinical data, whose exchange and reusability can be applied in other healthcare domains [11]. The dual model architecture allows the separation of (i) clinical and demographic properties, and (ii) standards and terminologies that give a semantic meaning to healthcare data. The first level of the dual model regards the components of programming language, the information exchange language, and all other components related to software development. The second level is represented by archetypes and templates [14].

In an archetype, attributes specification is performed through constructors whose data input are called generic data structures, which allow the representation of EHR heterogeneity through types such as: ITEM_SINGLE, ITEM_LIST, ITEM_TREE and ITEM_TABLE [11].

ITEM_SINGLE models a single data attribute (e.g., patient gender), while ITEM_LIST gathers a set of attributes into a list (e.g., patient address composed by street name, number and zip code). ITEM_TREE specifies a hierarchical data structure which is logically represented as a tree (e.g., physical and neurological evaluation of a patient). Finally, ITEM_TABLE models the data elements through lines for element definitions and columns for information values (e.g., a clinical report, whose clinical exams are shown in lines, and the respective values are presented in columns).

Each attribute from any data structure is characterized by a data type and may also have a set of domain constraints and associated terminologies. Terminologies give a semantic meaning to clinical data and can be represented by Health technical terms or textual information defined by a domain specialist.

Figure 1 depicts a sample of an archetype (i.e., family history.v2), which models the family history of a patient using the ITEM_TREE type. In this archetype, data attributes are organized in a multilevel hierarchical structure. The attribute Problem/diagnosis has a specific Health terminology (i.e., ICD10CM), which standardizes a given diagnosis. There is also a constraint (i.e., occurrences) which indicates that at least one diagnosis must be described.


openEHR-EHR-EVALUATION.family_history.v2

- Summary
- Per problem
  - Problem/diagnosis name
  - Description
  - Problem details [Cluster]
- Per family member
  - Family member name
  - Alias
  - Family member details
  - Biological sex
  - Relationship

Occurrences
Min: 1 Max: 1 Unbounded

Description: Identification of the significant problem or diagnosis in the identified family member.

Terminology	Release
ICD10CM	

Fig 1: Example of an Archetype structure.

Archetype can be described and specified by a formal language called Archetype Definition Language (ADL). ADL technology describe constraints based on domain content models and has three elements in its syntax: (1) dADL (data ADL) used for data definitions, (2) cADL (constraint ADL) can define domain constraints, and (3) FOPL (First-Order Predicate Logic), whose use is for building logic expressions based on first-order predicate. All archetypes are freely available on the Internet, on Clinical Knowledge Manager (CKM) and can also be used by systems in different formats (e.g., JSON, XML) [9].

Currently, the archetype standard has been adopted worldwide in both Academia and Industry [9]. Some state-of-art and state-of-practice tools propose the use of a set of archetypes in order to dynamically build software artefacts, e.g., Graphical User Interface (GUI), data schema, and Application Programming Interfaces (APIs) [10, 11, 12]. State-of-art studies suggest that building software from a set of archetypes is a good practice for maintaining healthcare software in accordance with international health standards.

Following this perspective, Microservice4EHR is specified to extract archetypes from an existing GUI, and build the DHSA from these archetypes, a different approach compared with software architectures generated by other cutting-edge tools.

2.2 Coupled and Decoupled Architecture

For decades, the use of software architectures whose components are interconnected and interdependent (i.e., coupled) is referred to as monolithic architecture. In the monolithic architecture shown in Figure 2, components in the business logic layer do not perform tasks independently. These components share the resources of the same machine (computer) and interoperate with a bounded set of other components, for example: other components that are contained in the respective software. For instance, in monolithic applications (i.e., applications built on a monolithic architecture), it is not possible to operate the business logic layer components of a software with the business logic layer components of another software [15]. To mitigate the problem of coupling and allow the decoupling of components from business logic layer (permitting the operation between them), studies on the use of a small service (or microservice) into business logic layers have been carried out. This means that in this approach, the business logic layer is made of microservices which perform tasks independently [8].

Figure 2 also shows an approach to develop a single software composed of a set of microservices. Each microservice represents either components from the business logic or data layers, and does not share the resources of the same machine (computer). Each microservice has its own database, and databases can be different from each other. The microservice is also able to interoperate with different sets of other microservices in order to reach the business targets. The access to each microservice is performed through Internet or local networking [16].

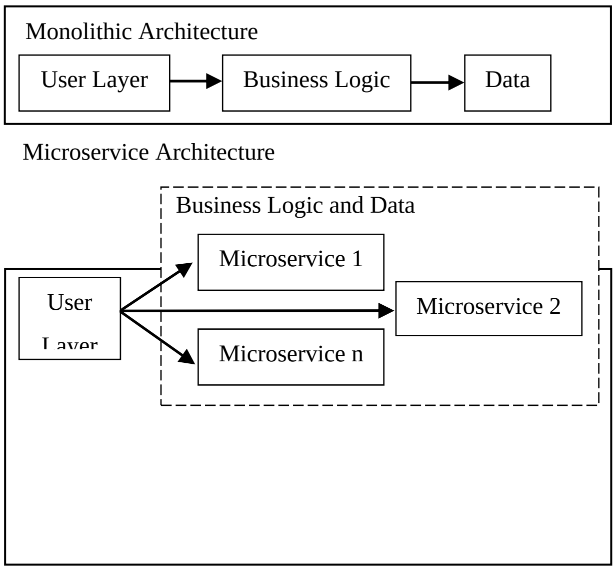


Fig 2: Example of Monolithic and Microservice architectures

The adoption of microservice architectures promotes software adaptation to new technological demands (e.g., cloud, big data) [8], as well as non-functional requirements (e.g., reusability, loosely coupled design, scalability, resilience) [15, 16]. Because of these features, in this study, a microservice architecture is used in the DHSA. Some works propose tools that build software architectures from a set of archetypes, which are shown below.

2.3 Related Works

A set of state-of-the-art studies that propose the dynamic generation of Health software artefacts from a set of archetypes are selected as related works.

The generation of software artefacts from archetypes has been performed for at least seven years and can be seen in the following studies: Duftschmid et al. [17] propose to map archetypes from a legacy database and then generate templates for a Health application called ArchiMed. Sundvall et al. [18] propose the use of REST architectures in Health applications that use archetypes for formatting their Healthcare data. Frade et al. [19] present a survey of the main tools that work with archetypes and Health Information Systems, namely: Liu EEE, OceanEHR Platform, OpenEHR Serv, OpenEyes, OpenHealth Multicare, and Think!EHR Platform.

In the last five years, a set of other tools have proposed the building of software artefact from a set of archetypes: Araújo et al. [20] built the PolyEHR tool, which promotes the storage of archetypes in two types of databases (relational and NoSQL), and does so through web-based services. Muslim et al. [21] propose a web-based service to transform health data from archetypes to the HL7 format. Araújo et al. [22] built a cloud service in order to create archetype-based graphical user interfaces, and store archetypes using database as a service (DBaaS). Gomes et al. [12] built an archetype to represent data from patients diagnosed with Chikungunya, and develop the MARCIA tool. MARCIA dynamically generates GUI from a set of archetypes. This architecture uses services to manage (store and query) Health data. The GUI is on the client-side, while API and Service are on the server-side.

Reis et al. [23] built a service-based framework to access health data from different sources (including legacy systems) while also allowing the integration with archetypes. Araujo et al. [10] developed the Template4EHR tool, which dynamically generates two software artefacts from a set of archetypes: a GUI and a Data Model. Template4EHR proposes the use of a software architecture based on Client-Server architecture and REST API. The client-side GUI sends a JSON document to the server-side REST API. This REST API processes all archetype-based data from the GUI, whose data is formatted by a Data Model also generated by the tool.

The following state-of-the-practice tools have been found: EhrScape [12], EtherCIS [24], and CloudEHRServer [25]. They enable the management of archetype-based Health data on a web environment. EtherCIS and CloudEHRServer propose resources for storing and querying data through a web environment (services and API). EhrScape provides resources to (i) dynamically build webforms and APIs from a set of archetypes, whose APIs run on a server that is provided by EhrScape and (ii) managing data through a web environment.

The above related works propose the execution of archetype-based Health software (or part of them) on a web environment. However, none of them use or enable the use of decoupled software architectures, where an error or failure in a part of the software architecture does not impact or create side effects for other parts of the architecture. In the following section, the proposed software architecture called DHSA is presented.

3. THE DECOUPLED HEALTH SOFTWARE ARCHITECTURE (DHSA) AND THE MICRO SERVICE 4EHR TOOL

This section presents the DHSA and the Microserivce4EHR tool. Section 3.1 shows an overview of the proposed architecture and Section 3.2 discusses Microservice4EHR functionalities.

3.1 The DHSA

Main objective of this study is to allow the migration of a legacy Health software built with coupled components into a new software version with decoupled components. Figure 3 illustrates Health software A and B. Health software A has its business layer composed of a set of coupled units (i.e., monolithic architecture). The business logic layer units are also based on a set of Health data standards (e.g., a set of archetypes). Each standard represents the healthcare data from each unit x, y and z. If one of these units presents unexpected behavior (an error or failure), it will affect the execution of other components, stopping the execution of the software.

Health software B business layer components are decoupled. Each decoupled component is based on a single healthcare data standard (e.g., a specific archetype). If there is an error or failure (an unexpected behavior) in one of them, this does not impact the execution of other components. This occurs because all of these components are decoupled and independent each other.

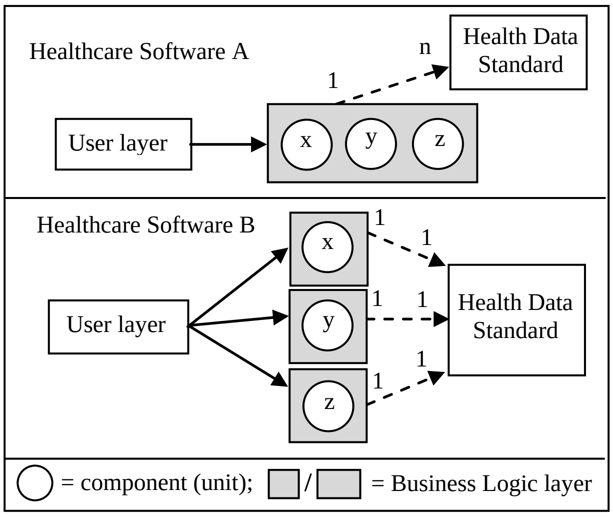


Fig 3: A coupled and a decoupled software architecture.

In this study, three components have been specified to compose the DHSA (Connector, Container and Archemicro), which are explained below.

3.1.1 *The Archetype-based Microservice (Archemicro)*

Archemicro is a server-side microservice whose scope of task is related to the data processing of a specific archetype. An archemicro task is autonomously performed on the Internet using CRUD (Create, Read, Update and Delete) operations. Microservice4EHR generates an archemicro for each archetype.

3.1.2 *Container*

Container is a server-side component which represents a standard unit of software that encapsulates the code and all its dependencies so that the archemicro runs reliably, with no interruption, in any computing environment. Container encapsulates an archemicro, ensuring that the environment needed for the archemicro is available.

3.1.3 *Connector*

Connector is a client-side component that directly intermediates the communication between the GUI and archemicro. Connector (i) receives data from GUI, (ii) identifies the set of archetypes used in the request, and (iii) sends the data to the respective set of archemicros. The role of the Connector is to read the set of archetypes used in the GUI and establish connections with the respective set of archemicros.

Figure 4 depicts healthcare software and its interaction with the openEHR archetypes on the Internet. In the user layer, the GUI is built based on archetypes. This GUI sends data to the Connector component, which processes data and sends it to an archemicro. In the business logic layer, the archemicro component (whose scope is CRUD operations of an archetype) receives data from Connector and processes them in accordance with their task. Finally, the result of this processing is returned to the GUI.

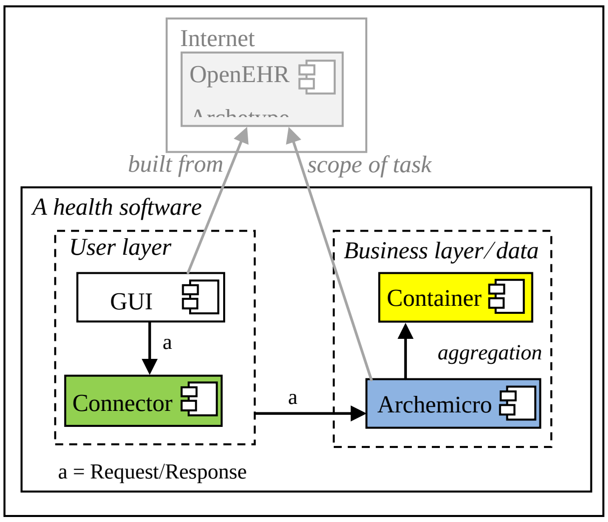


Fig 4: DHSA conceptual modelling.

3.2 The Microservice4EHR Tool

To build the DHSA, Microservice4EHR has as input data an existing GUI such as a legacy health software. Thus, there is an initial phase in order to build the GUI: (i) initially, the user (e.g., a software engineer) verifies the software requirements; (ii) after that, the user goes to a repository where archetypes are stored on the openEHR Foundation website - and chooses the set of archetypes that matches the software requirements; (iii) finally, the software engineer or developer writes the GUI based on these archetypes.

In the last phase, the Microservice4EHR tool is used to generate the health information system (HIS). The input is a legacy or new "Archetype-based GUI", which means a GUI that has been built from a set of archetypes. The archetype-based GUI must be included, as a data, into a JSON document.

Figure 5 depicts the input (JSON document) of the Microservice4EHR tool. After that, Microservice4EHR dynamically generates the Connector, Container and archemicro components.

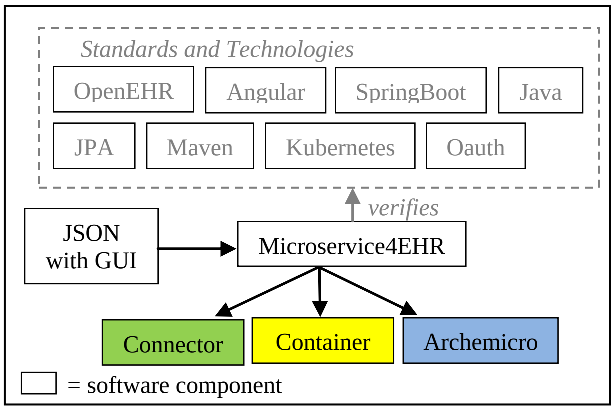


Fig 5: Input and Output of Microservice4EHR tool.

Also shown in Figure 5, all dynamic generation is based on industry standards widely adopted in software development worldwide, for example: openEHR for Health data; Angular for dynamic interaction between client-side and server-side; Spring Boot for building the Health software environment; Kubernetes for managing microservices on the Internet; JPA/Hibernate for Health data model mapping; Maven for dependencies in software application; Java as a programming language for writing the archemicro components, and OAuth2 for access security among components in the software application.

```
01 Algorithm Microservice4EHR_Tool
02 Input: a JSONDocument
03 Output: a JSONDocument
04 BEGIN
05   READ JSONDocument
06   SET host as URL
07   SET archetypeNames as Array to empty
08   SET htmlGUI as XMLDocument
09   FOR EACH jsonObject on the JSONDocument
10     SET host GET Host on the jsonObject
11     SET archetypeNames GET Archetypes
12                     on the jsonObject
13     SET htmlGUI GET Gui on the jsonObject
14   END FOR
15   SET datamodel as Object
16   FOR EACH node on the htmlGUI
17     SET datamodel APPEND Model
18   END FOR
19   SET jsonDoc as JSONDocument
20   SET jsonDoc APPEND buildConnector(host,
21                 datamodel, archetypeNames)
22   SET jsonDoc APPEND buildContainer (host)
23   SET jsonDoc APPEND buildMicroservice(host,
24                 datamodel, archetypeNames)
25   RETURN jsonDoc
26 END
```

Fig 6: Microservice4EHR tool algorithm.

The algorithm in Figure 6 shows the Microservice4EHR tool that generates the Connector, Container, and Archemicro components. A JSON document is the data input of the algorithm, which is instantiated in the memory to validate and extract its data (line 05). After that, a set of variables is instantiated in memory to manipulate data obtained from the input JSON document, such as host address, archetype names, and HTML-based GUI, as shown from line 06 to 14.

The data model and archetypes contained in the HTML-based GUI is retrieved and used in the respective functions (buildConnector, buildContainer, and buildMicroservice), which respectively generate the Connector, Container and Microservice components, as seen from line 15 to 24. Finally, the algorithm produces a JSON document with the programming language written components (line 25).

4. ASSESSING THE DHSA

This section describes the main results obtained in the assessment of the proposed solution. Section 4.1 shows an analysis of state-of-the-art and state-of-the-practice in relation to Microservice4EHR tool, while Section 4.2 discusses the metric used to calculate the coupling level in software architectures.

In the scope of this assessment, a real-word scenario of public hospitals in Northeastern Brazil are chosen. Their set of software requirements, which is ruled by the Brazilian Health Ministry, contains: registration of patient admission, blood pressure, and respiration.

Figure 7 illustrates a legacy desktop health software from a public hospital in Northeastern Brazil. Some observations can be obtained: (i) The software is not build based on openEHR archetypes; (ii) the GUI is developed through Java Swing component, which is meant for desktop software; (iii) the business logic layer is built with Java Bean components, which runs on a local machine; (iv) the database is MySQL Server; and, finally, (v) the software supports the following software requirements: registration of patient admission, blood pressure and respiration.

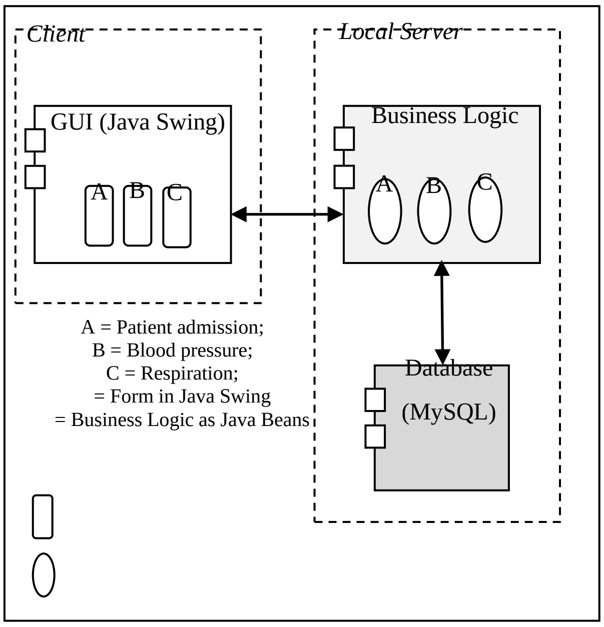


Fig 7: A public hospital legacy software.

To migrate this legacy software to a web environment, four tools that propose the dynamic generation of health software from a set of archetypes are considered: MARCIA, Template4EHR, EhrScape, and Microservice4EHR.

To make this assessment feasible, two phases are developed. In Phase 1, a set of openEHR archetypes is chosen to represent healthcare data. For the registration of patient admission, the archetype individual_personal.v0 was chosen, blood_pressure.v2 for the registration of blood pressure data, and respiration.v2 for the registration of respiration data. Still in this first phase, the same set of archetypes was used to dynamically build software generated by the four tools mentioned above.

In phase 2, to evaluate the decoupling aspect of the components from each software architecture, the output software of each tool was modelled, as described in Section 4.1.

4.1 Analyzing the Software Architectures Generated by Microservice4EHR and Related Tools

MARCIA tool generates a health software whose architecture is implemented according to a Client-Server architecture, a REST API and a service. To query and save data, this architecture uses services. As shown in Figure 8, there is a GUI on client-side, while the API and service are on server-side. MARCIA API scope of task is the data processing of all archetypes contained in the respective GUI. Even if there is more than one archetype in the GUI, all modules are performed inside the same API. These components are interconnected and interdependent, and cannot execute tasks independently. This means that if one module from one specific archetype changes, fails or generates an error, it will impact the other modules.

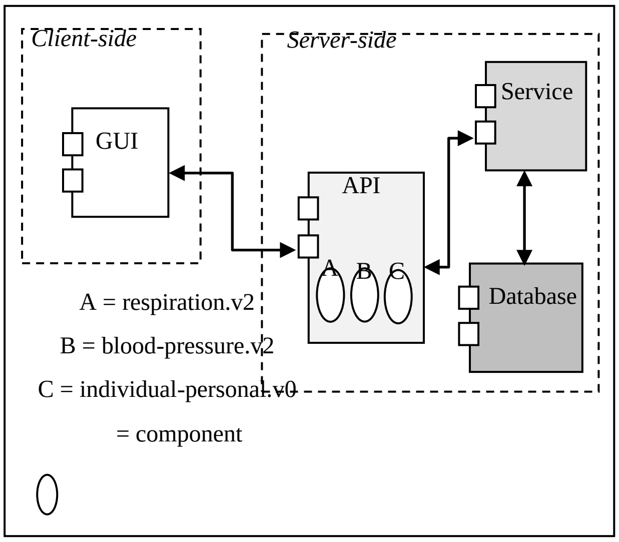


Fig 8: Software Architecture generated by MARCIA tool.

Template4EHR tool dynamically generates an archetype-based GUI and data Model. Its software architecture is implemented based on a client-Server architecture and a REST API. As illustrated in Figure 9, there is a GUI in the client layer, and a REST API in the server layer. The GUI interacts with the API, which interacts with a database. Similarly to the MARCIA tool, the task scope of Templates4EHR API is the data processing of all archetypes contained in a GUI. Even if there is more than one archetype in a GUI, all business logic layer components are performed inside the same API. These components are interconnected and interdependent, and cannot execute tasks independently. This means that if one module from one specific archetype has an unexpected behaviour (e.g., error or failure), it will impact the other modules.

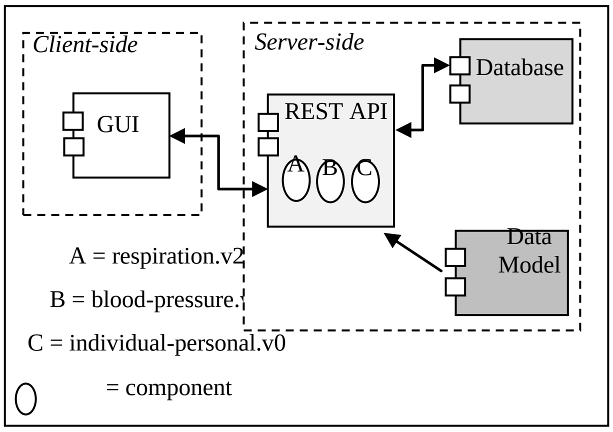


Fig 9: Software Architecture generated by Template4EHR tool

EHRScape tool generates health software whose architecture is based on a client-Server architecture and a REST API. As presented in Figure 10, there is a GUI in client layer, and an API and a database in the server layer. The GUI interacts with the API, which interacts with a database. EhrScape API processes data of all archetypes contained in the respective GUI. This processing is done in one module, even if there is more than one archetype in GUI. These components are interconnected and interdependent, and thus cannot execute tasks independently.

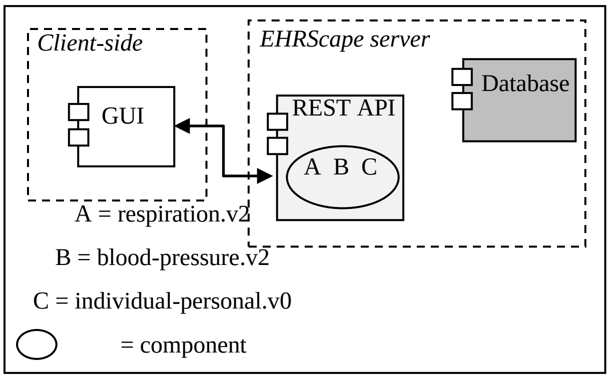


Fig 10: Software Architecture generated by EHRScape tool

The Microservice4EHR tool dynamically generates a DHSA made of three components: Connector, Container and Archemicro. As depicted in Figure 11, there is an archemicro for each archetype used in the GUI. The GUI (legacy or new) sends data to Connector, which intermediates the communication between the GUI and the set of containers and archemicros via a web environment. Each archemicro has its own database.

The decoupling aspect of the first three software architectures generated by MARCIA, Template4EHR, and EhrScape are similar. For instance, in the MARCIA software architecture, if there is an unexpected behaviour in component A (related to archetype respiration.v2), all other components (B and C) would be impacted by this behaviour (i.e., all execution of this API would be interrupted). This occurs because there is no separation or dissociation between them; they are interconnected and interdependent. In this way, the first three software architectures (MARCIA, Template4EHR and EhrScape) are described as monolithic architectures (as described in Section 2.2).

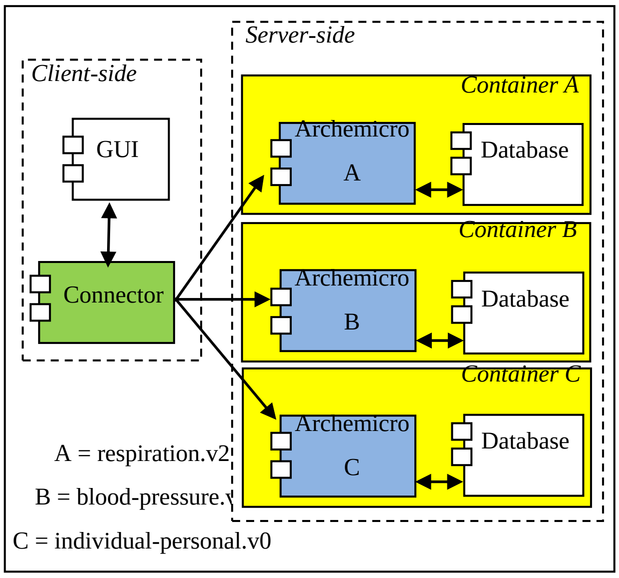


Fig 11: Software Architecture generated by Microservice4EHR tool.

On the other hand, in the DHSA (i.e., Microservice4EHR software architecture), all components from business logic are built as small, separate services (i.e., microservice). For example, if there is an unexpected behaviour (e.g., error or failure) in archemicro A, this does not impact any other components (B and C). This occurs because they are separated and execute tasks autonomously.

4.2 A Formal Method to Evaluate the Coupling level

A formal metric is specified, in this study, to evaluate the "coupling level" in healthcare software architectures. This metric is dubbed Archetype-based Software Architecture Coupling (ASAC).

This metric considers two quantities: (i) the number of archetypes used in a software, and (ii) the coupling coefficient of each component in a software architecture. In this study, "coupling coefficient" and "component" have particular meaning: "Coupling coefficient" means the impact that an unexpected behaviour (e.g., error or failure) generates from a component onto another one. Then, each impact represents the sum of 1 (one) point in the equation. If there is no impact, it is 0 (zero). "Component" means any processing unit from the business logic layer (e.g., a module, a method, a service, or a microservice).

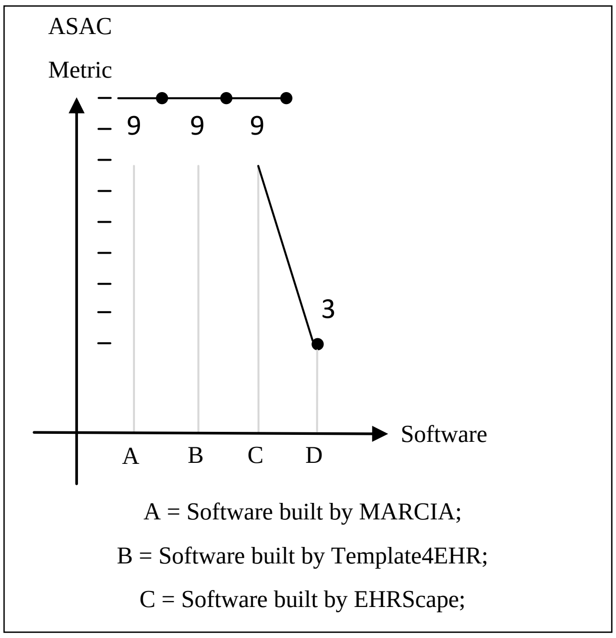
To exemplify this, let us assume that for each existing archetype in the client layer, there is a respective server-side component that processes it. Thus, {ASAC = (a1→i) + (c1→i)}, where a1→i is the number of archetypes used in each software, and c1→i is the sum of all coupling coefficients found in the server-side components.

The ASAC metric measures the coupling in software that uses a set of archetypes (i.e., there is a coupling between the software and archetypes). For this, the ASAC metric cannot be zero, there is an “initial coupling index” regarding the number of archetypes that compose the software. This occurs because if there is some change in archetypes, the software must be rebuilt to maintain the compliance between software and the health data standard (i.e., archetype). This compliance dependency, in this study, is seen as coupling. Finally, for the ASAC metric, a lower result means a less coupled (or more decoupled) software architecture.

MARCIA, Template4EHR and EHRScape software architectures have the same behaviour. As illustrated in Figure 8, 9 and 10, respectively to each software architecture, if there is an unexpected behaviour in component A, this will impact two other components (B and C). The same occurs with component B (impacts A and C) and component C (impacts A and B), which means each component (A, B and C) has two points of coupling coefficient. Thus, MARCIA, Template4EHR and EHRScape ASAC metrics are 3+(2+2+2). These health software are built from three archetypes, and the coupling coefficients of the components are 2. The results of all these three software architecture ASAC are 9 (nine).

Microservice4EHR software architecture: As presented in Figure 11, DHSA has a particular behaviour. Components A, B or C do not impact any other component, even if there is an error or failure in them. This means that each component has a coupling coefficient of zero. Thus, Microservice4EHR ASAC metric is 3+(0+0+0); there are 3 archetypes and the coupling coefficient of each component is 0 (no component impacts another). The DHSA ASAC result is 3 (three).

As illustrated in Figure 12, the first three software built by MARCIA, Template4EHR and EHRScape have equal ASAC metrics (9). The ASAC metric related to the DHSA-based software (built by Microservice4EHR) is 3.



D = DHSA-based Software built by Microservice4EHR.

Fig 12: ASAC metrics on generated software.

As a result, the DHSA is a software architecture 66,6% less coupled (or more decoupled) than other software architectures built by other state-of-the-art tools. This result is obtained by applying the rule of three on the ASAC metrics described above.

5. CONCLUSION

This paper introduces the Decoupled Health Software Architecture (DHSA), which is a software architecture generated from a set of archetypes (an international health data standard specified by the openEHR Foundation). DHSA is composed of autonomous and independent components, which perform tasks through a web environment. This study has also developed (i) the Microservice4EHR tool (to dynamically build the DHSA), and (ii) the Archetype-based Software Architecture Coupling (ASAC) metric. An assessment is performed in a real-world scenario in Brazilian public hospitals, whose result shows that DHSA is 66,6% more decoupled than software architectures generated by state-of-art tools like MARCIA, Template4EHR, and EHRScape. Thus, the healthcare sector is benefited with a solution that decreases the coupling of a health software architecture, and this means that unexpected behaviour (e.g., error or failure) in a part of the health software does not imply its interruption, improving its functionality for the Health community.

As future works, the behaviour of the DHSA components working with blockchain should be investigated.

6. ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) – Finance Code 171220.

7. REFERENCES

[1] Pressman, R., Maxim, B. (2019) Software Engineering: A Practitioner's Approach. McGraw-Hill Education, ISBN: 1259872971.

[2] Bass, L., Clements, P. and Kazman, R. (2012). Software Architecture in Practice (3rd ed.). Addison-Wesley Professional.

[3] Cha, J., Kim, J., and Jeong, Y. (2016). "Architecture Based Approaches Supporting Flexible Design of Self-Adaptive Software," 2016 Int. Conf. on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, pp. 1424-1425. doi: 10.1109/CSCI.2016.0280.

[4] Bjuhr, O., Segeljakt, K., Addibpour, M., Heiser, F. and Lagerström, R. (2017) "Software Architecture Decoupling at Ericsson," 2017 IEEE Int. Conference on Software Architecture Workshops (ICSAW), Gothenburg, 2017, pp. 259-262. Doi: 10.1109/ICSAW.2017.7.

[5] Knoche, H. and Hasselbring, W. (2018) "Using Microservices for Legacy Software Modernization," in IEEE Software, vol. 35, no. 3, pp. 44-49, May/June 2018. Doi: 10.1109/MS.2018.2141035.

[6] Qian, K., Liu, J., and Tsui, F. (2006) "Decoupling Metrics for Services Composition," 5th IEEE/ACIS Int. Conference on Computer and Information Science and 1st IEEE/ACIS Int. Workshop on Component-Based Software Engineering,Software Architecture and Reuse (ICIS-COMSAR'06), Honolulu, HI, 2006, pp. 44-47. Doi: 10.1109/ICIS-COMSAR.2006.30.

[7] Megha Mayreddy and B.tarakeswara Rao. (2015) Secure SOA Framework for Multi-Cloud Storage and Computing. International Journal of Computer Applications 114(8):10-16, March.

[8] Kakivaya, G., Xun, L., Hasha, R., et al., (2018) "Service fabric: a distributed platform for building microservices in the cloud," in Proceedings of the 13 EuroSys Conf. (EuroSys '18). ACM, New York, NY, USA, 2018, Article 33, 15 pages. DOI: 10.1145/3190508.3190546.

[9] OpenEHR Foundation (2020). OpenEHR. Accessed on: Apr.26, 2020. [Online]. Available: https://www.openehr.org.

[10] Think!EHR Platform (2017) EHR Scape. Accessed on: Apr. 26, 2020. [Online]. Available: https://www.ehrscape.com.

[11] Gomes, F., Paiva, J., Bezerra, A. et al. (2018) "MARCIA: Applied Clinical Record Management : Electronic Health Record Applied with EHRServer," 2018 IEEE 20th Int. Conf. on e-Health Networking, Applications and Services (Healthcom), Ostrava, 2018, pp. 1-6. doi: 10.1109/HealthCom.2018.8531096.

[12] Araujo, A., Times, V. and Silva, M. (2020) "A Tool for Generating Health Applications Using Archetypes," in IEEE Software, vol. 37, no. 1, pp. 60-67, Jan.-Feb. 2020. Doi: 10.1109/MS.2018.110162508.

[13] Brazil Ministry of Health (2002). Ordinance No. 2048. Accessed on: Apr. 26, 2020. [Online]. Available: http://bvsms.saude.gov.br/bvs/saudelegis/gm/2002/prt2048_05_11_2002.html (in Portuguese).

[14] Moner, D., Maldonado, J.A., Robles, M. (2018) "Archetype modeling methodology," J. of Biomedical Informatics, Volume 79, 2018, Pages 71-81, ISSN 1532-0464, https://doi.org/10.1016/j.jbi.2018.02.003.

[15] Chaitanya K Rudrabhatla. (2018) A Systematic Study of Micro Service Architecture Evolution and their Deployment Patterns. International Journal of Computer Applications 182(29):18-24, November.

[16] Larrucea, X., Santamaria, I., Colomo-Palacios, R., et al. (2018), "Microservices," in IEEE Software, vol. 35, no. 3, pp. 96-100, May/June 2018. Doi: 10.1109/MS.2018.2141030.

[17] Duftschmid, G., Chaloupka, J. and Rinner, C. (2013) "Towards plug-and-play integration of archetypes into legacy electronic health record systems: the ArchiMed experience, " BMC Med Info Decis Mak; 13: 11. doi: 10.1186/1472-6947-13-11.

[18] Sundvall, E., Nystrom, M., and Karlsson, D. (2013) "Applying representational state transfer (REST) Architecture to archetype-based electronic health record systems". BMC Medical Informatics and Decision Making. 13:57. DOI: 10.1186/1472-6947-13-57.

[19] Frade, S., Freire, S. M., Sundvall, E. et al., (2013) "Survey of openEHR storage implementations," Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems, Porto, 2013, pp. 303-307. Doi: 10.1109/CBMS.2013.6627806.

[20] Araújo, A., Times, V., Silva, M. (2016) "PolyEHR: A Framework for Polyglot Persistence of the Electronic Health Record, " 2016 Int. Conf. Internet Computing and Internet of Things. ISBN: 1-60132-439-1. CSREA Press.

[21] Muslim, A., Puspitodjati, S., Mutiara, A.B., et al. (2017) "Web services of transformation data based on OpenEHR into Health Level Seven (HL7) standards," 2017 Second International Conference on Informatics and Computing (ICIC), Jayapura, 2017, pp. 1-4. Doi: 10.1109/IAC.2017.8280571.

[22] Araújo, A., Times, V., Silva, M. (2018) "A Cloud Service for Graphical User Interfaces Generation and Electronic Health Record Storage". In: Latifi S. (eds) Information Technology - New Generations. Advances

in Intelligent Systems and Computing, vol 558. Springer. 2018. Print ISBN: 978-3-319-54977-4.

[23] Reis, L. F., Ferreira, D. G., Maranhao, P. A. et al. (2018) "Integration through mapping — An OpenEHR based approach for research oriented integration of health information systems," 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), Caceres, 2018, pp. 1-5. doi: 10.23919/CISTI.2018.8399258.

[24] Ripple Foundation (2019) Ether CIS. Accessed on: Feb. 02, 2020. [Online]. Available: http://ethercis.org.

[25] CaboLabs (2020) CloudEHRServer. Accessed on: Feb. 02, 2020. [Online]. Available: https://cloudehrserver.com.