Designing a Patient-Centric System for Secure Exchanges of Medical Data[⋆]

Thais Webber[0000-0002-8091-6021], Juan Mendoza Santana[0000-0003-1718-2480], Andreas Francois Vermeulen[0000-0003-2225-6851], and Juliana K.F. Bowles[0000-0002-5918-9114]

School of Computer Science, University of St Andrews
St Andrews KY16 9SX, UK
{tcwds,jjm20,afv,jkfb}@st-andrews.ac.uk

Abstract. Designing patient-centric healthcare systems which consider the smart integration of distributed medical data is challenging. This includes handling numerous architectural dependencies and requirements as a result of blending a variety of future generation technologies. Examples of recent approaches are proposals of a unified format for medical records to facilitate efficient healthcare provision, transparent data access control using blockchain technology, and emergent authentication mechanisms and privacy-preserving techniques for data analytics. The Serums project proposes an innovative design for a Smart Health Centre System in a distributed development effort. The goal is a comprehensive solution for integration and access of transnational medical records. This paper focuses on the architectural design workflow as a way of delivering artefacts for development iterations and contribute towards module integration planning in the software development process. Our experience shows that in data integration projects for healthcare provision, system architects and developers can profit from the designed viewpoints as artefacts to reveal integration challenges and highlight quality attributes.

Keywords: Healthcare information systems · Patient health records · Architectural design.

1 Introduction

The integration of Patient Health Records (PHRs) in a smart and secure environment for information retrieval can be an extensive challenge for researchers, system architects, healthcare providers and governments. Nonetheless, the vision of such an integration aims to enable, in the future, European-wide healthcare systems. In Europe, there is a clear trend towards patients becoming more empowered in healthcare processes and being increasingly aware of privacy and data ownership. For instance, patients may want to have more control over who has access to their data and to which parts of their data, whilst expecting the

⋆ This research is funded by the EU H2020 project Serums: Securing Medical Data in Smart Patient-Centric Healthcare Systems (grant code 826278).

system to comply with the EU General Data Protection Regulation (GDPR)[1] and other legal and ethical regulations.

The EU Horizon 2020 project Serums[2] aims to combine next-generation technologies, such as blockchain and data lake principles, in a patient-centric toolchain for accessing, storing, communicating and analysing distributed medical records in a secure and privacy-preserving way [8]. It specifies the transformation and aggregation of PHRs from distributed data sources (e.g., from hospitals, medical practices, healthcare databases, health monitoring devices) into a unified and universal format for metadata storage named *Smart Patient Health Record* (SPHR). Complex access rules to SPHR are expected to be written by patients and healthcare professionals on a user-friendly front-end as *data sharing agreements*, which are stored as smart contracts over a blockchain network.

The challenge of designing an architecture like Serums for projects that employ diverse and emerging technologies is even bigger when teams are not colocated. Successful integration projects with dispersed teams rely on a system-level modular design with well-defined requirements and documentation about each module and their inter-dependencies [14]. The degree to which it is possible to reduce ambiguity and guarantee cohesion of requirements in a project is also dependent on how the architectural design workflow is conducted and documented from early stages of development [14].

In this paper, we discuss the *Smart Health Centre System* (SHCS) architectural design workflow within Serums and the contribution of each designed artefact (e.g. requirements specification, viewpoint diagrams, interaction diagrams, and several others) towards identifying development challenges. The discussions about the workflow outcomes are directed towards software engineering researchers, project managers and architectural analysts. We also point out the role of viewpoints in identifying the expected quality attributes (i.e., compliance, traceability, auditability, testability and interoperability) for the effective integration of systems such as SHCS.

This paper is structured as follows. In Section 2, we present related work regarding integrated healthcare records, emerging technologies and the Serums perspective. In Section 3, we introduce the *architectural design workflow* defined for Serums, and present an overview of its *architectural design* in Section 4. Section 5 discusses the design workflow outcome towards the desired quality attributes for a seamless integration, and further summarises considerable observations from applying the first iteration of the design process. We conclude with suggestions for further work in Section 6.

2 Recent related work

In recent years, sharing PHRs across healthcare institutions and countries has become an essential requirement of future-generation effective healthcare provision. To successfully achieve this goal, different integration solutions have been

[1] Information on GDPR can be found at `https://gdpr-info.eu/`

[2] For more information see `http://www.serums-h2020.org`.

presented by researchers and software developers using different technologies such as Blockchain, Cloud Computing, IoT, Big Data, and so on [6], [9], [10], [17]. In addition, prototypes have been developed to evaluate the design of such personalised and decentralised medical records systems. MedRec [5] is an example of a recent system developed with blockchain technology to deal with sensitive medical information. It is mostly focused on empowering patients and providers in the choice to release metadata and in engaging medical stakeholders to participate in the network as miners. The mining reward is to access and aggregate anonymised data to perform advanced analytics. By contrast, the Serums project applies the concept of distributed ledgers mostly to store and retrieve patient records, focusing on the development of a robust and secure authorisation mechanism that allows patients and healthcare organisations to create personalised access to medical records. Moreover, the goal of Serums is to establish a tool-chain which is modular and integrates several next-generation technologies such as multi-factor user authentication schemes [2], and in the future, distributed privacy-preserving data analytics [11].

3 The Architectural Design Workflow

Serums follows an architectural design workflow in order to guarantee a clear integration process and adequate adherence to system requirements. An example of a basic design workflow for global software development (GSD) can be found in the work of Sangwan et al. [14].

The development of the Smart Health Centre System (SHCS) for Serums [8] can be classified as an example of a GSD project. The overall project assembles a total of 9 European partners within 7 countries including hospitals, academic and industrial partners. The coordinating team is responsible for the development and integration of the SHCS [8]. The *integration* task raises many challenges to the teams regarding sharing the understanding of system requirements, technological issues and various collaboration aspects such as exchanging concise technical reports to reduce design misalignment.

In addition, the project combines features of the *Rational Unified Process* (RUP) and incremental development models [15]. As described in [15], the development iterations are divided into four phases: *Inception*, *Elaboration*, *Construction* and *Transition*. The last is mainly concerned with the deployment of the system in the user community. In this paper, we focus on the *Elaboration* phase, aiming to include the integration task in the early stages of the design process. Fig. 1 illustrates the Serums architectural design workflow.

3.1 Elaboration Phase

The *Elaboration* phase focuses on *architectural design* activities that enable a common understanding of the requirements and the establishment of an overall system architecture. It enables software architects to identify and describe the principal modules (aka subsystems) and their relationships [15]. For instance, the

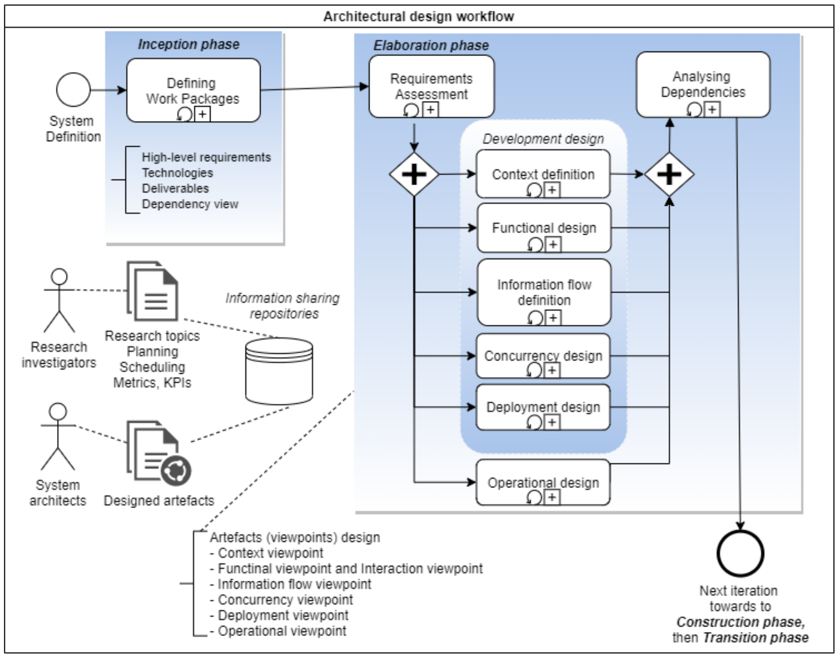

Fig. 1. Serums architectural design workflow

SHCS requires that only authorised users can access information from patient records which are usually distributed across multiple locations and may therefore present different privacy constraints that cannot be violated by the SHCS. Integrating both the requirements for the SHCS and the rules of each individual data source is an important integration challenge that must be addressed before reaching the *Construction* phase. This particular integration requirement can be refined on every development iteration assuming that for the first iteration only the rules of the SHCS are followed, whereas for the second iteration, the rules of each individual data source have to be integrated as well as part of the original requirement.

The SHCS architecture must be able to comply with this requirement that concerns not one but different scopes within the architecture such as the *security* modules and the *complex data storage and retrieval* modules. Thus, several artefacts need to be designed by each team to cope with this specification. Below, we describe each activity foreseen in the *Elaboration* phase of the architectural design workflow for Serums (see Fig. 1).

– *Requirements Assessment*: entails significant effort on the clear specification of system requirements and their coverage, generating a shared artefact called *Requirements Specification* (e.g., document). In practice, this document could also contain explicit remarks on how these requirements relate to the architecture towards supporting fundamental design decisions that impact non-functional aspects (i.e., quality attributes) [1].

– *Parallel design*: the listed activities in this workflow section could be partially performed in parallel, synchronously and/or asynchronously, even though they are depicted within a parallel gateway. In what follows, we give a brief description of each planned parallel activity and its suggested artefact.
  – **Context definition**: the design effort begins delivering the system *Context viewpoint*. It captures the interaction of the system with its environment, through relationships with other actors such as people and external systems. Sommerville [15] proposes context models as an artefact to identify the boundaries of a system. These models can be represented through use case diagrams, UML interaction diagrams (typically sequence diagrams), and in our particular case using a more general diagram [13]. Use case diagrams can only be created when a system requirements specification is (at least partially) available. Preliminary architecture descriptions can focus on views that show the system's modules with their major internal processes and explicit dependencies.
  – **Functional design**: aims to identify the functional blocks that will integrate the overall solution following the principles of high cohesion and low coupling [7]. In addition, this task includes the design of artefacts that describe the system's architectural elements that deliver the overall functionalities. Examples of artefacts generated from this activity are the functional and interaction viewpoints. These artefacts are needed to guide structural properties of the information, the concurrency, and the deployment whenever it is required. An incomplete system due to missing functional components can be a consequence of poor functional design. Additionally, modular development depends highly on this activity.
  – **Information flow definition**: allows us to specify in a high-level manner how information is stored, manipulated, managed and transferred within the system. At the same time, this flow enables us to expand on the nature of the connections between components and modules identified in the functional design, for we agree that components interact with each other by passing information. This viewpoint is a high-level view of data elements and information flow, it can include some elements of the structure of the data and observations about static information (which is not transferred across components). A diagram can capture the sequence in which the information is passed and transformed through the modules of the system. The information flow is particularly relevant because it also allows architects to visualise the responsibilities of each module and their expected functionality, and define an initial design of the interfaces that will be provided.
  – **Concurrency design**: aims to identify and describe the presence of concurrency within the system delivering a *Concurrency viewpoint* as artefact. A good concurrent design can help to identify bottlenecks and deadlocks generated by the interaction of different components. In addition, this design permits to identify convergence points in which the communication and the information flow can be synchronised.

  - **Deployment design**: captures the environment in which the system will be deployed through the different stages of the development process. It is directly related to the infrastructure needed by each of the functional modules capturing the hardware requirements and runtime dependencies. This activity delivers a *Deployment viewpoint* that can be summarised into high-level diagrams containing visualisations of the mappings between logical and physical elements, processing components, network interconnections, and reflections on the storage facilities required.
  - **Operational design**: captures how the system will be operated, administered, and supported when it is running in its production environment. It is a significant activity that must be considered and planned at design time to enable the definition of installation procedures, management, and normal operation of the system. Additionally, contingency plans can be introduced as means to mitigate problems resulting from unexpected functionality, which can be critical for healthcare systems.
- *Analysing dependencies*: marks the end of the *Elaboration* phase in terms of achieving a stable set of designed artefacts to follow up to the next phase of development cycle.

The parallel design activity above could also be named as *Development design.* The integration of these viewpoints entails an important step of architectural design, which will support the system development process. It aims to highlight the aspects of the architecture that are interesting to stakeholders involved in building, testing, maintaining, and enhancing the system. This perspective can be achieved by combining artefacts such as *Context viewpoint* to capture users and external agents, *Functional and Interaction viewpoints*, *Information flow viewpoint*, *Concurrency viewpoint* to define development units, dependencies and connections, and *Deployment design* to raise awareness about the system technical requirements. The *Operational viewpoint* will be evaluated towards the end of the *Construction phase* entering the *Transition phase.*

4 The Architectural Design for Serums

The architectural design plays a key role in system integration because its artefacts describe the responsibilities of each subsystem/module as well as the interfaces and steps required for their interaction [15]. The proposed SHCS architecture follows the workflow described in Section 3.

Context analysis: the *Context definition* within Serums' SHCS is shown in Fig. 2. There are three categories of users: hospital administrators, healthcare professionals, and patients. Each category can perform specific actions such as request a patient's SHPR and manage credentials or access rules. The diagram also shows that the system interacts with the data from three different hospitals (ZMC, FCRB and USTAN). Even though in a real scenario this interaction may differ from hospital to hospital, for the current prototype we assume that it happens through standardised APIs for each data source. In addition, an interaction

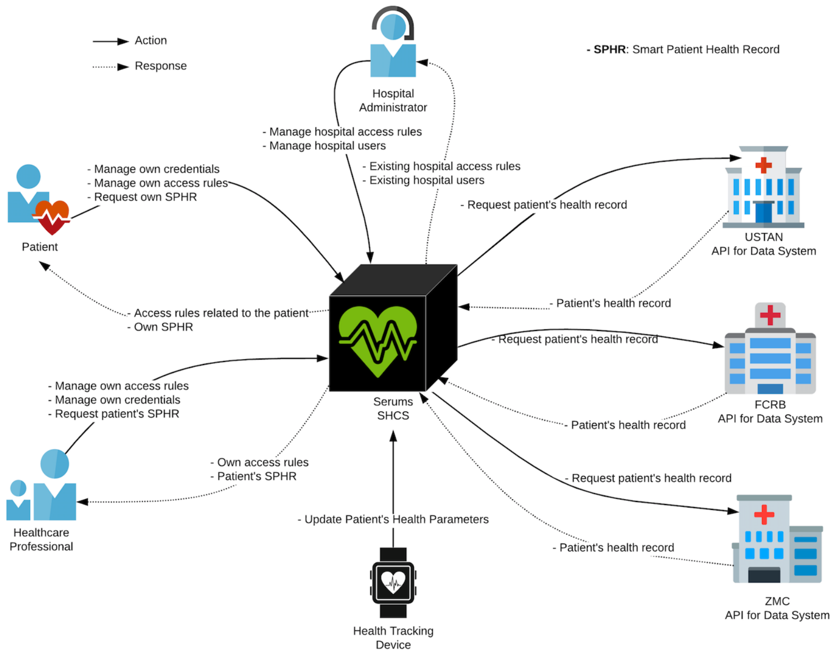
Action
Response
Hospital Administrator
- SPHR: Smart Patient Health Record
- Manage hospital access rules
- Manage hospital users
- Existing hospital access rules
- Existing hospital users
- Manage own credentials
- Manage own access rules
- Request own SPHR
Patient
- Request patient's health record
USTAN API for Data System
- Access rules related to the patient
- Own SPHR
- Patient's health record
- Request patient's health record
FCRB API for Data System
- Manage own access rules
- Manage own credentials
- Request patient's SPHR
Serums SHCS
- Patient's health record
- Own access rules
- Patient's SPHR
- Request patient's health record
Healthcare Professional
- Update Patient's Health Parameters
- Patient's health record
Health Tracking Device
ZMC API for Data System

Fig. 2. SHCS Context Viewpoint

with personal health tracking devices is depicted. The context viewpoint allows us to place actors, functionalities and external connections in a perspective in which details of internal system features (e.g. front-end receiving requests for the security layer and for data retrieval processes) are abstracted in a black-box.

Functional design: the *Functional viewpoint* of SHCS is shown in Fig. 3. In this diagram, modules are explicitly divided following their main goals and interfaces with the integration module. The integration module has two sub-modules: the front-end dealing with user interface and users requests; and the back-end for interfacing and coordinating other modules to get responses. The functional viewpoint translates the requirements into implementation modules that will inter-operate via APIs. This diagram is the foundation for the information flow as well as for the deployment and interaction viewpoints, mostly because they all depend on the definition of functional modules to capture their different perspectives.

Information flow definition: the *Information Flow* within the SHCS is shown in the diagram of Fig. 4. It formalises the challenge of gathering the distributed medical records into a flexible infrastructure of data processing (acquisition, transformation, storage and retrieval). The data flow viewpoint enables the mapping of critical design decisions related to these operations. For instance,

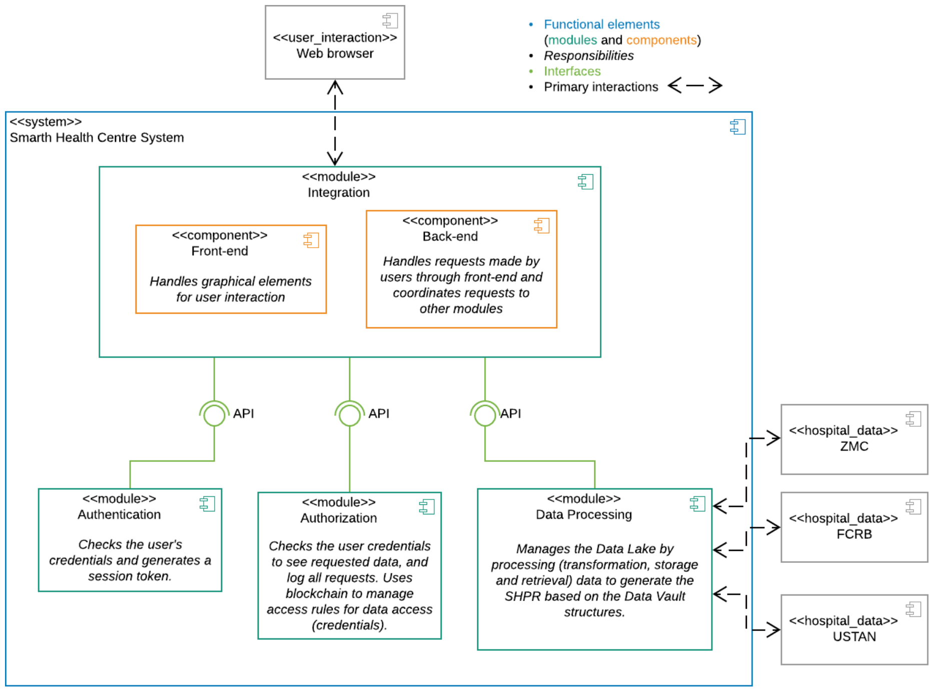
Functional elements
(modules and components)
Responsibilities
Interfaces
Primary interactions
<<user_interaction>>
Web browser
<<system>>
Smarth Health Centre System
<<module>>
Integration
<<component>>
Front-end
Handles graphical elements for user interaction
<<component>>
Back-end
Handles requests made by users through front-end and coordinates requests to other modules
API
API
API
<<module>>
Authentication
Checks the user's credentials and generates a session token.
<<module>>
Authorization
Checks the user credentials to see requested data, and log all requests. Uses blockchain to manage access rules for data access (credentials).
<<module>>
Data Processing
Manages the Data Lake by processing (transformation, storage and retrieval) data to generate the SHPR based on the Data Vault structures.
<<hospital_data>>
ZMC
<<hospital_data>>
FCRB
<<hospital_data>>
USTAN

Fig. 3. SHCS Functional Viewpoint

the responsibilities of the integration module are detailed here and it depicts all needed handlers (*back-end*) and response flows to users (*front-end*).

Development design: the *Interaction viewpoint* is shown in Fig. 5. It represents the setup that will allow all the different modules to interact with each other. This viewpoint together with the *Information flow*, the *Functional viewpoint*, and the *Deployment viewpoint* (shown in Fig. 6) shape the development design. All perspectives combined enable developers to make relevant decisions for the modules implementations.

The modular approach proposed in the *Functional viewpoint* enables the SHCS to deal with different technologies, such as blockchain, distributed databases, and so on. Each high-tech solution has specific requirements that must be satisfied; our *Deployment viewpoint* shows the solutions for this issue by means of *containerisation* [16]. Furthermore, an additional level of interoperability has to do with the communication between modular subsystems, which is directly addressed in our *Interaction viewpoint*. *Concurrency design* is shown in a simplified manner through the *Interaction viewpoint* and the *Information flow viewpoint*. The *Operational design* is not included here because this is still a very early stage of the project in which operation modes have yet to be explored before being documented.

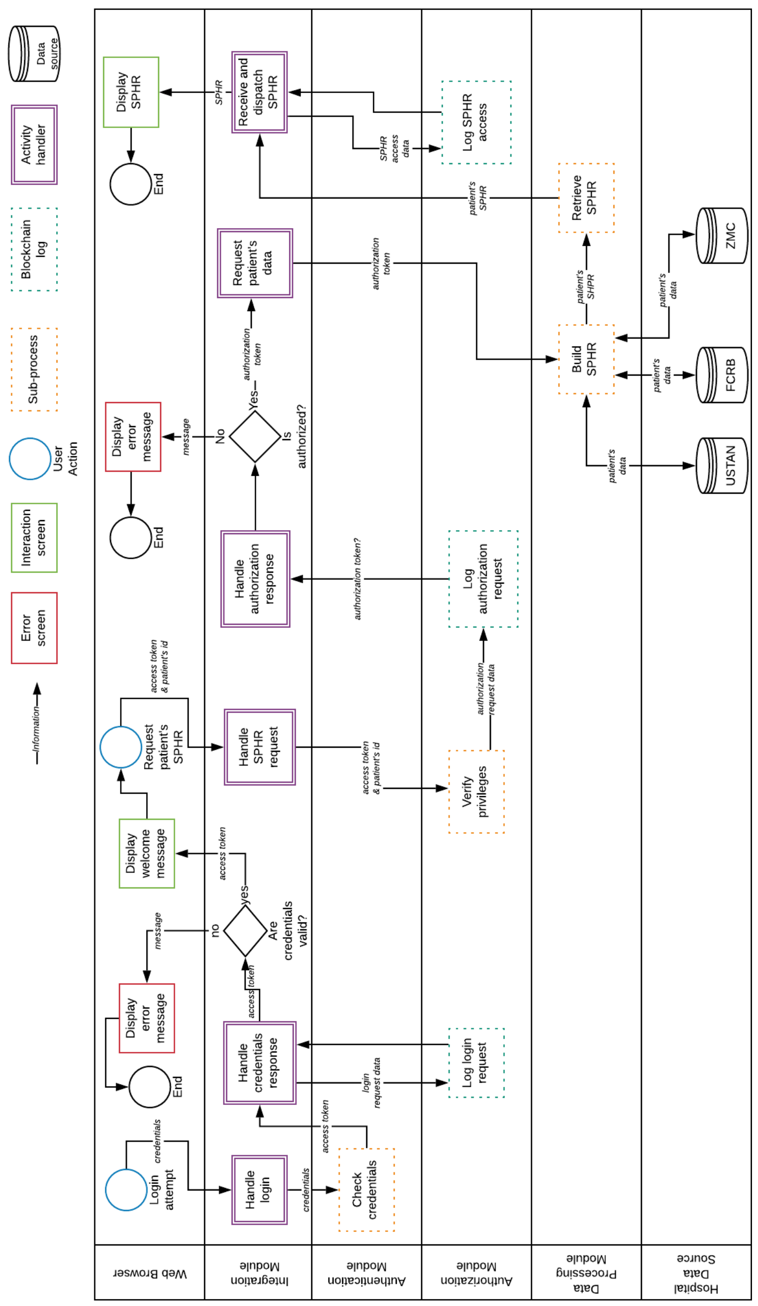

Fig. 4. SHCS Information Flow diagram

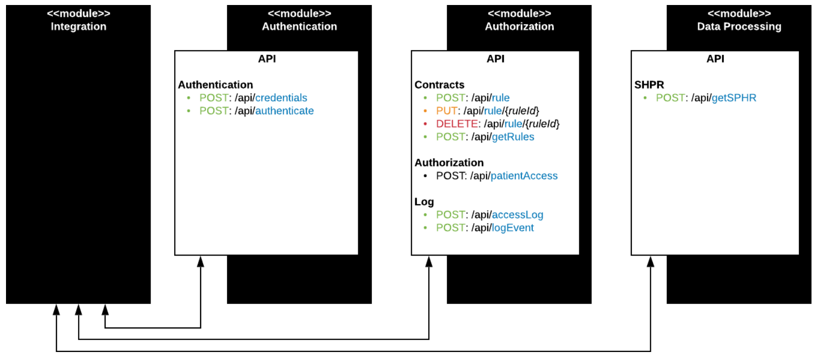

Fig. 5. SHCS Interaction Viewpoint

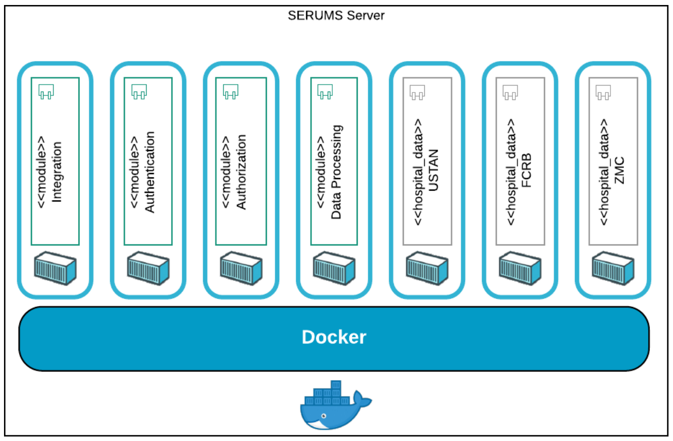

Fig. 6. SHCS Deployment Viewpoint

4.1 Architectural Design Overview

In the first software design iteration, we generated different viewpoints that can be aggregated in a high-level diagram showing the architectural overview, to gain an understanding of the required components of the SHCS. Fig. 7 presents the Serums architectural overview that assembles four different perspectives (or layers) of integration.

Security layer: is responsible for handling privacy issues to guarantee confidentiality, integrity and availability when processing medical data. It is responsible for the generation of secure access tokens to be associated with user sessions, which will enable information exchange within the system components. It is also responsible for the synchronisation of two modules: (i) *Authentication* to make sure only individuals with credentials can access and visualise (fully or partially) the requested patient's health record and (ii) *Authorisation* which is managed by the Blockchain network to control the credentials and access rules for infor-

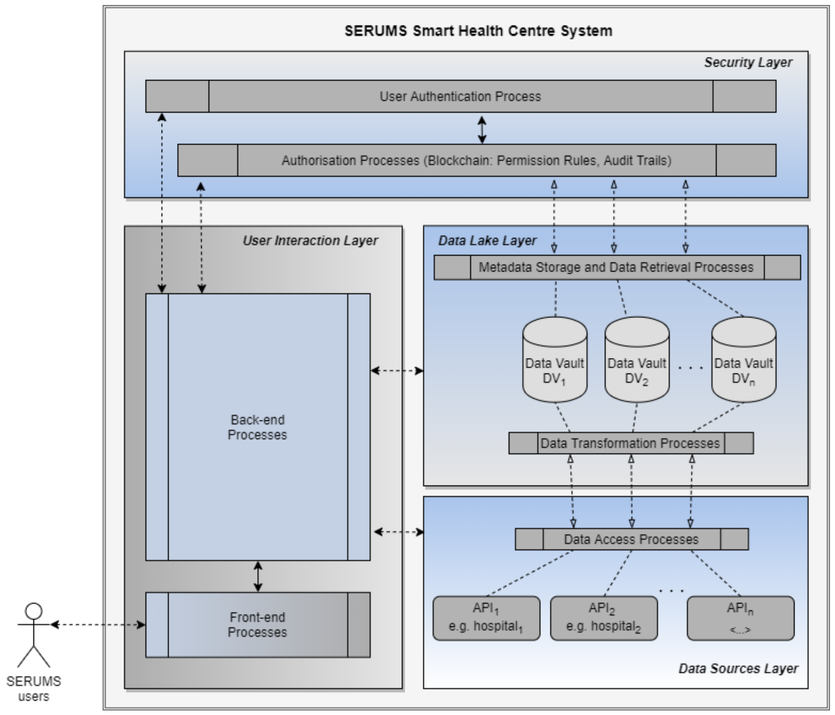
SERUMS Smart Health Centre System
Security Layer
User Authentication Process
Authorisation Processes (Blockchain: Permission Rules, Audit Trails)
User Interaction Layer
Back-end Processes
Front-end Processes
SERUMS users
Data Lake Layer
Metadata Storage and Data Retrieval Processes
Data Vault DV_1
Data Vault DV_2
Data Vault DV_n
Data Transformation Processes
Data Access Processes
API_1 e.g. $hospital_1$
API_2 e.g. $hospital_2$
API_n <...>
Data Sources Layer

Fig. 7. SHCS architecture overview

mation retrieval. Role-based access permissions will be programmed as well as processes for maintaining records of the performed transactions for audit trail.

User Interaction layer: is responsible for building the system's front-end and back-end modules, which comprise the central integration layer for demonstrating the usage of Serums. It implements processes such as the interface graphical elements (front-end connection to users and its connection to the back-end processes), user access rules and credentials management (back-end connections to Security layer and Data Lake layer). Serums authorised users (individual, group or organisation) connect and interact with the system via the front-end, in it the authentication process has to be included as the first step of the navigation flow, subsequently the back-end will communicate with the *Authorisation* module to resolve the *Role-based permission rules* that are created, stored and updated in the Security Layer.

Data Lake layer (intermediate layer between the Serums SHCS and the data sources of medical organisations responsible for data storage and retrieval): integrates the module responsible for distributed medical data processing. It must be able to dispatch segments of a patient's health record based on the user's permission. This layer is responsible for assembling the *Smart Patient Health Record* (SPHR) as a unified format to represent medical data from multiple sources. The data lake will not store explicit medical information, instead it will store pointers to the real data, which will be retrieved on demand. The SPHR will be constructed based on the metadata extracted from the real data, the extraction of the metadata is also responsibility of this layer. The Data Lake layer depends directly on the Data Sources Layer, and provides services to the User Interaction Layer.

Data Sources layer: this layer can integrate modules, systems, subsystems or APIs responsible for creating a safe environment for data acquisition from organisations to integrate SPHR. It consolidates distributed data servers (i.e., from organisations such as hospitals, national databases, the patient's home environment), implicating in several trusted and untrusted network connections for the acquisition of PHR data. This layer, though depicted in our architectural overview, is not under the control of the Serums team, and as seen in the Context Viewpoint (Fig. 2) is external to the system.

Fig. 7 is a valuable artefact delivered after a few iterations in the *Elaboration* phase of the design workflow. This output is the result of combining high-level requirements, preliminary design artefacts and several informal discussions between system architects of distributed teams.

5 Quality attributes for SHCS integration

The emphasis on non-functional requirements is usually orthogonal to functional design [4]. In this paper, we define quality attributes as non-functional requirements for SHCS integration. The design workflow should be able to produce artefacts that capture the desired quality attributes for the integrated SHCS: compliance, traceability, auditability, testability and interoperability. These properties are relevant for the project and they can, and should be, considered in the design of artefacts. The intersection between design viewpoints and quality attributes is summarised in Table 1.

Table 1. Desired quality attributes within integration perspective and major artefacts. Ticks indicate whether an artefact (viewpoint) is essential to inspect a quality attribute.

Quality attributes for SHCS integration

Artefact	Compliance	Traceability & Auditability	Testability V-model (testing approaches)				Interoperability
			Unit	Integration	System	Acceptance	
Requirements Specification	✓						
Context viewpoint	✓	✓				✓	✓
Functional viewpoint		✓	✓				✓
Information flow viewpoint	✓	✓		✓	✓	✓	✓
Interaction viewpoint				✓			✓
Concurrency viewpoint					✓		
Deployment viewpoint					✓		✓
Operational viewpoint							✓

Compliance determines that the integrated system is in accordance with established guidelines and specifications. From the collection of artefacts produced in the design workflow, compliance can be easily emphasised, for example, from the *Information flow viewpoint* refinement until it is thoroughly aligned with the requirements specification. In addition, compliance can be demonstrated with the help of the *Context viewpoint*, in which the actions and responses are described, thus enabling us to judge whether the system developed matches these actions. A more thorough analysis for compliance can be ultimately done with respect to the *Requirements Specification*.

Traceability in our context is defined as a quality attribute related to the ability to collect data regarding time records of transactions performed by users within the system. The *Information flow* and *Functional viewpoints* are valuable artefacts to ensure that this property will be fulfilled. In operation, the system should be able to produce transaction logs in the security layer that allow analysts to audit the system. Then, as a consequence of traceability, it is also possible to achieve Auditability, meaning that the integrated system can deliver information to verify if the requirements have been met and identify any non-conformance in the system operation. This can be achieved thanks to the security layer, in which a Blockchain implementation will automatically generate an entry into the ledger regarding information access. The *Information flow* diagram explicitly shows that the authorisation layer includes tasks related to the log entries in the Blockchain, just as it is mentioned in the *Functional viewpoint*. Complementary the *Context viewpoint* aids to capture what actions should be logged, hence strengthening traceability and auditability.

Testability is the degree to which a software artefact (module, requirements- or design document) supports testing in the test context. This quality attribute for Serums refers to the ability acquired through the viewpoints design to perform conclusive and reproducible testing. The *V-Model*[12] has been taken into consideration in order to align our architectural workflow and the generated artefacts to different stages of verification and validation according to that model. The context for *unit testing* can be related to the *Functional viewpoint*, in which individual components are responsible to perform specific tasks. *Integration testing* can be observed within the *Interaction viewpoint* in which all elements converge into the integration module. *System testing* can be placed within the *Deployment viewpoint* and the *Information flow*, that in conjunction provide a holistic perspective of the system. *Acceptance testing* can be done based on the actions and responses shown in the *Context viewpoint*. By enabling these testing stages related to the V-Model, our architectural workflow and artefacts contribute to develop testability as a feature of the integrated SHCS.

Interoperability is defined as the ability of systems to exchange and make use of information. SHCS has many challenges regarding module integration and how to define interfaces for secure data exchange. This quality attribute can be evaluated, for example, using the *Container viewpoint* to identify these interconnections. In our proposed architectural design, several other viewpoints (*Context*, *Functional*, *Information flow*) show the presence of external data sources related to hospital data. There is an evident demand for understanding interoperability, specially in the presence of dispersed data sources from partners and healthcare organisations. Irrespective of their specific implementations, the SHCS must be able to cope with each one of them seamlessly. It is at this point that standard protocols, such as *Health Level Seven* (HL7) [3] for delivering health information, can be considered.

Considering the five quality attributes highlighted as important in the context of the Serums project, we present a summary of observations gathered from the first iteration of the architectural design workflow.

1. Dispersed teams increase the challenges to specify and design modules because individual preconceptions have a greater weight in such project setup.
2. Sustaining informal discussions among dispersed teams often results in rework and misalignment of activities [14]. This is a challenge we have identified in our design workflow within the integration perspective.
3. Deriving formal specifications from high-level project requirements is not trivial because they focus on different aspects of the system.
4. Project requirements must be differentiated from system requirements and from individual requirements for each module.
5. *Requirements assessment* in the design workflow has the upmost relevance for acquiring all five quality attributes further in the remaining design activities.
6. There is a need for individual architectural design for each functional module besides the system's architectural design.
7. The relation between system architecture and module architecture must be clearly established to ease seamless integration.
8. *Functional viewpoint* granularity can vary depending on the perspective/responsibility of a development team.
9. The *Functional viewpoint* does not necessarily give a clear indication of all modules and operations involved in the overall architectural design. Several discussions in different communication channels were conducted during this modelling activity (not all captured in logs and documentation).
10. The *Functional viewpoint* and the *Information flow* are highly related to the SHCS desired quality attributes of traceability, auditability and testability.
11. *Information flow* is concerned with specific actions, thus there is no global description of the system's information flow.
12. The *Information flow viewpoint* is one of the richest artefacts to understand modules interfaces, data exchanges and security steps demanded by SHCS. It allows the module dependencies and internal processes to be further discussed and detailed amongst teams. Moreover, it could guide SHCS testing plan and expected system behaviour under user requests.
13. There are several approaches to interoperability, though we have proposed containerisation, we cannot guarantee at this stage that this will be sufficient for the entire system, additional development efforts are required to evaluate the effectiveness of containerisation.
14. Containerisation is being used instead of virtualisation, easing the deployment of modular components in a single shared unit of hardware.
15. Non-functional requirements and quality attributes have to be specified for the system as a whole and also for each individual module. The real con-

tribution of each component to the success of quality attributes has to be reflected individually.

The architectural design workflow has been illustrated with artefacts to help the relevant stakeholders make design decisions that impact SHCS. These impacts are first understood in the *Elaboration phase* when the system architecture can still be prone to change. If these potential issues are detected and dealt with in this early stage, the *Construction* phase will follow a smooth development. For example, if the interaction between components is opportunely defined, then development teams can focus on satisfying the expected APIs, while the team responsible for the integration can handle the other modules as black boxes, and thus have no concerns regarding their actual implementation.

6 Conclusion

This paper addresses how the architectural design workflow of a system can be used as the means to uncover integration challenges, in our case the SHCS for Serums. It is natural to assume that modular decomposition of a system results in more focused artefacts across distributed teams. However this decomposition should be done taking into consideration the need for future integration at the centre of the architectural design. Early identification of module dependencies from architectural design artefacts and their derived models is a rich contribution to understand the challenges of Serums integration. For instance, the design viewpoints accompanied with several support diagrams help to identify integration factors needed to achieve desired quality attributes. In addition, these artefacts could help dispersed teams to share a common understanding of the integrated system in more effective ways to lead next development iterations.

Some challenges are still open for discussion regarding the architectural design workflow and the architectural design itself. Access to patient health records is granted on the basis of interaction rules between SHCS users. These interactions are rather complex, because they depend on time, geographical and other regulatory restrictions that might result in conflicting rules. These conflicts have to be identified and resolved first from a specification perspective, and then from an implementation one, that is, decisions have to be made by the stakeholders to define how these conflicts should be resolved.

This paper highlights that designing integrated healthcare systems can be less complex when following a proper design workflow. The proposed workflow emerged from the architectural and technological dependencies and challenges inherent in our project. Defining unambiguous requirements that can be concisely understood and documented is the first step to have teams sharing the same architectural vision from early stages of the software lifecycle. Moreover, reviewing the architectural design against desired quality attributes and business goals could be a key point for a future successful system integration.

References

1. Balsamo, S., Di Marco, A., Inverardi, P., Simeoni, M.: Model-based performance prediction in software development: a survey. IEEE Transactions on Software Engineering **30**(5), 295–310 (May 2004)
2. Belk, M., Fidas, C., Pitsillides, A.: Flexpass: Symbiosis of seamless user authentication schemes in IOT. In: Extended Abstracts of the 2019 CHI Conf. on Human Factors in Computing Systems. ACM, New York, NY, USA (2019)
3. Benson, T., Grieve, G.: Principles of Health Interoperability SNOMED CT, HL7 and FHIR. Health Information Technology Standards (HITS), Springer (2016)
4. Blaine, J.D., Cleland-Huang, J.: Software quality requirements: How to balance competing priorities. IEEE Software **25**(2), 22–24 (March 2008)
5. Ekblaw, A., Azaria, A., Halamka, J.D., Lippman, A.: A case study for blockchain in healthcare : Medrec prototype for electronic health records and medical research data. In: Open & Big Data Conference, August 22-24, 2016. IEEE (2016)
6. Griggs, K., Ossipova, O., Kohlios, C., Baccarini, A., Howson, E., Hayajneh, T.: Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. Journal of Medical Systems **42**(130) (July 2018)
7. ISO/IEC/IEEE: ISO/IEC/IEEE International Standard - systems and software engineering–vocabulary. ISO/IEC/IEEE 24765:2017(E) pp. 1–541 (Aug 2017).
8. Janic, V., Bowles, J., Vermeulen, A., et al.: The Serums tool-chain: Ensuring security and privacy of medical data in smart patient-centric healthcare systems. In: 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019. pp. 2726–2735 (2019)
9. Karimi, L., Joshi, J.: Multi-owner multi-stakeholder access control model for a healthcare environment. In: 2017 IEEE 3rd Int. Conf. on Collaboration and Internet Computing (CIC). pp. 359–368. IEEE (Oct 2017)
10. Khan, S.I., Hoque, A.S.M.L.: Health data integration with secured record linkage: A practical solution for bangladesh and other developing countries. In: 2017 Int. Conf. on Networking, Systems and Security (NSysS). pp. 156–161 (Jan 2017)
11. Kumar, M., Rossbory, M., Moser, B., Freudenthaler, B.: Deriving an optimal noise adding mechanism for privacy-preserving machine learning. In: Anderst-Kotsis, G., et al. (eds.) Database and Expert Systems Applications, DEXA 2019. Communications in Computer and Information Science, vol. 1062. Springer (2019)
12. Mathur, S., Malik, S.: Advancements in the v-model. International Journal of Computer Applications **1**(12), 29–34 (2010)
13. Rumbaugh, J., Jacobson, I., Booch, G.: Unified Modeling Language Reference Manual, The (2nd Edition). Pearson Higher Education (2004)
14. Sangwan, R., Bass, M., Mullick, N., Paulish, D.J., Kazmeier, J.: Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series). Auerbach Publications (2006)
15. Sommerville, I.: Software Engineering. Pearson, 10th edn. (2015)
16. Syed, M., Fernandez, E.: A reference architecture for the container ecosystem. In: Proceedings of the 13th Int. Conf. on Availability, Reliability and Security. ARES 2018, ACM, New York, NY, USA (2018)
17. Zhang, P., Schmidt, D., White, J., Lenz, G.: Chapter one - blockchain technology use cases in healthcare. In: Raj, P., Deka, G. (eds.) Blockchain Technology: Platforms, Tools and Use Cases, Advances on Computers, vol. 111, pp. 1–41. Elsevier (2018)