Defining an Architecture for Evolving Environments

Jarkko Hyysalo, Gavin Harper, Jaakko Sauvola, Anja Keskinarkaus, Ilkka Juuso, Miikka Salminen and Juha Partala
Faculty of Information Technology and Electrical Engineering, University of Oulu
P.O. BOX 8000, FI-90014 University of Oulu, Finland
jarkko.hyysalo@oulu.fi, gavin.harper@student.oulu.fi, jaakko.sauvola@oulu.fi, anjakes@ee.oulu.fi, ilkka.juuso@ee.oulu.fi, miikka.salminen@ee.oulu.fi, juha.partala@ee.oulu.fi

ABSTRACT

The architecture of a system specifies how the system should be designed and built. However, shortcomings are identified in current architecture process frameworks concerning evolving domains like healthcare. We claim that an iterative architecture process is required, where the technical concerns are separated from the non-technical ones. Furthermore, a strong guiding vision is required. Based on our experiences from a biobank IT infrastructure process, we present an architecture process that is modular, interoperable, controlled and abstracted, thus being capable of handling complex systems with large uncertainties.

CCS Concepts

• *Software and its engineering~Software architectures*

Keywords

Architecture; design; lessons learned; post-mortem; process.

1. INTRODUCTION

There exist several architectural frameworks that provide guidance on how a system is to be designed and implemented that are available. However, in domains that are not established or stable, there exist variables that may cause changes and unexpected events that require non-routine solutions. The wider the scope of the project and the more stakeholders that are involved, the more difficult the architecture definition is [1]. If the development problem is not well-structured, it becomes increasingly more challenging to address and communicate [2]. Healthcare is one such domain that is constantly evolving. There exist several stakeholders from different domains, various laws and regulations with many still emerging and service models still being refined. These factors result in an inherent volatility within this particular domain.

We claim that an incremental and iterative process is necessary, where the outcome is built gradually. This facilitates observation of the evolution of the design and implementation allowing a better understanding of its requirements and potential. Feedback can then gradually be gathered and incorporated into the development process.

SAC 2017, April 03-07, 2017, Marrakech, Morocco © 2017 ACM. ISBN 978-1-4503-4486-9/17/04…$15.00 DOI: http://dx.doi.org/10.1145/3019612.3019902

In order to define such a process, we propose the following research question: *What form of architecture process is suitable for evolving environments?* To address our research question, we used a post-mortem analysis to study the process of building a biobank IT infrastructure. The remainder of the paper is organised as follows. Section 2 studies the background; Section 3 presents the research approach; Section 4 discusses the new process model and the empirical study; Section 5 summarises the study.

2. BACKGROUND AND RELATED WORK

The healthcare domain is constantly evolving by means of new technological innovations, new requirements for efficiency and cost and new regulations being introduced. There also exists the continued interaction and dependency on legacy systems and data formats. Legacy systems and data formats that are widely utilised in the medical domain create challenges by means of potentially isolated and non-interoperable systems. A technical burden through legacy may arise from e.g. used data formats, processes, tools, applications or service-level agreements, which may each affect interoperability. Thus, there exists a need for migration and renewal strategies in addition to strategies that enable complying with the legacy systems.

Architecture design is heavily guided by requirements regarding efficiency and cost. Emerging technologies may provide better and more efficient solutions to current challenges, public-private partnerships (PPP) funded by a partnership of government and a number of private sector companies and new principles like 4P medicine, referring to preventive, predictive, personalised, and participatory medicine [3]. 4P medicine is also sometimes referred to as personalised or precision medicine with the goal to enhance the health outcomes with integration of evidence-based medicine and precision diagnostics into clinical practice.

Through this holistic approach, in combination with several divergent stakeholders and new technologies, it is easy to see that the design environment may become fragmented and volatile. Furthermore, the healthcare sector is evolving with new strategies and business models such as PPP, where the strategic and business drivers are diverse. The gradual evolution of legacy systems towards new solutions must facilitate the continued use of existing systems that are currently integrated into the current environment. This may result in a complex environment with combination of both legacy systems and modern solutions [4]. The ongoing evolution through changing legislation, regulations and improvements in medical practices creates an environment that is constantly changing.

Various frameworks have been proposed to address different design realities, including standards such as ISO/IEC/IEEE 42010. In addition a general model for architecture design is presented in [5]. Architectural frameworks have been analysed extensively [cf. 1, 6, 7], and a recurring theme across the frameworks is that each describes the role of the architecture in the product development process as a "*systematic analysis and design of related information to provide a model for guiding the actual development of information systems*" [6]. Many architecture frameworks discuss, but few focus primarily on, the architecture creation process [7]. The value of the processes is shown in the literature and it has been suggested that processes ensure that activities in an organisation are performed consistently and reliably [8]. Evaluations of architecture frameworks are also provided in [1, 7] that can be used to select an approach for different sets of requirements. While there are pros and cons for each method, common deficiencies in the architecture frameworks can be identified [6]: 1) The level of details required in models is not specified enough. 2) Rationales are not considered in models, thus no verification is possible. 3) Non-functional requirements are not considered in all frameworks. 4) Software configuration is not considered in all frameworks.

Moreover, architectural design involves often complex trade-off analyses that may require expertise in several domains. Further, the environment may be more variable and dynamic than current processes can support. It is even possible that not all stakeholders are known or they may not be fully aware of what the product is intended to accomplish. Thus, the guiding vision for the product is impossible to be fully defined during the early stages of development. Instead, it is suggested that it is built incrementally. In conclusion, there appears to be a need for an architecture process that addresses the identified shortcomings including but not limited to evolving environments, the availability of specific details, design rationales, non-functional requirements and software configurations.

3. RESEARCH APPROACH

The results are based on experiences gathered during a biobank IT infrastructure development project. The research consisted of studying several organisations related to biobank activities with the aim of defining architecture for a biobank and finally implementing a functional infrastructure. Managing the large number of stakeholders and constantly changing environment required a new approach for architecture development thus creating the need and basis for this work.

During the project several challenges were identified. A post-mortem analysis was conducted to analyse these findings and to identify ways of improving the architecture process. A post-mortem analysis is a study method that may be used to gather empirical knowledge. The benefits of a post-mortem analysis include revealing findings more frequently than other methods such as project completion reports. Post-mortem analysis may be used as a project-based learning technique [9, 10]. In addition to finding impediments of the process, a post-mortem analyses may be used to improve methods and practises [11]. During this research project, a post-mortem analysis was used to study our development process to facilitate identifying potential sources for improvement or optimisation. Our post-mortem analysis follows the general iterative post-mortem analysis, as shown Fig. 1.

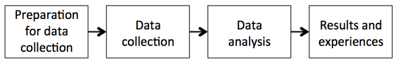

Figure 1. General post-mortem analysis process [9].

Our research started with an initial preparation stage where we identified key participants involved with our effort and considered which methods and procedures would be applicable. Project history was examined with the participants involved in the project (primarily project managers) and project documents were studied. Then our goal for the post-mortem analysis was determined–to identify potential sources for improvement and optimisation. Data collection involved gathering relevant project experiences from team members and key stakeholders. Participants of our data collection and analysis sessions were project managers (1), researchers (2) and system architects (2). A lightweight post-mortem analysis was used, as it fits the project size best [cf. 12]. KJ sessions [13] with thematic analysis [14] were utilised to gather and organise ideas and data. In the analysis phase, findings and ideas were organised into groups based on their relationships. Post-It notes were used to record the ideas and findings and related notes were then grouped together. Based on our results, we modelled an architecture process for evolving environments, which was finally reported in the results and experiences phase.

4. OUR ARCHITECTURE PROCESS

Our approach to architecture is a) modular, b) interoperable, c) controlled and d) abstracted. This way it can handle complex systems with inherent uncertainties. The design philosophy is that when designing the architecture, the requirements are separated into technical and non-technical requirements. Incremental and iterative development is suggested as it allows observing the outcome and improving it as new information becomes available. Our proposal is a Continuous Renewal architecture model, which is intended to be general such that it does not mandate how each abstraction level should be modelled. This allows several architectural styles and notations to be utilised. More important is that all the necessary views to a development are addressed. The architecture is modular to allow flexibility and extendibility. Modularity allows the reconstruction of any part of the system such that an area-of-effect can potentially be localised to just those components directly connected to the modified region. In the case of the biobank, the system was designed such that successive system component regions typically form a directional data flow through standardised interfaces. Interoperability is achieved through the specification of interfaces defining various domains with utilisation of open-standard communication protocols. Controllability comes from the rigorous process and from the Master Architecture that guides the development and verifies the outputs against the set targets. The architecture should initially be defined at a high level of abstraction. It is not necessary to define rigorously the exact transformations that shall occur on the data passing through the system. It suffices to consider only its input type and output type as it traverses a domain. The content of the data is largely irrelevant in most cases. This is highly beneficial in an evolving healthcare environment whereby the content of a data set may frequently be in a state of flux while the laws and regulations surrounding the data set are interpreted. Increasing or decreasing the level of abstraction as required allows the examination of the system from different perspectives. The separation of technical concerns from non-technical concerns allows adapting to future needs, as the design is not relying on specific technologies or solutions.

In evolving environments, constant comparison to the Master Architecture is required. It allows for the verification of compliance for all the relevant inputs and design choices, even if those vary during development. It also provides the goals towards which the effort is pushed as well as the guidelines that determine how those goals should be reached. Design rationales guide the overall work and are kept up-to-date by continuous communication with the stakeholders who see the system being defined incrementally. Continuous communication aids in building trust between stakeholders. This allows insight into the rationale for the design and implementation decisions and consequently allows context to more localised. Additionally, the Master architecture is updated accordingly. Comparing the results to the Master Architecture enables a constant feasibility analysis, and corrective actions if necessary.

In our architecture, domains are fundamental units and the communication pathways between the domains indicate connectivity between domains. The internal structure of a domain remains unspecified in the highest level of abstraction. It is specialised once the requirements for that domain are defined. For example, we can consider the anonymising encoding service of the biobank not as a part of the architecture, but as a specialisation of a domain for a specific task. Thus, if for example the law changes, the specialised components of the domain may be updated or replaced with minimal impact to the architecture assuming the new specialisation utilises the existing connection path and communicates using compatible data storage and communication formats. It then follows that any connected domains from which it receives from or transmits to must be able to accept that communication readily.

This is accomplished by initially designing the system at a high of abstraction, modelling the transformations that occur in a domain as a function with an argument type $\boldsymbol{T}$ that maps to some other type $\boldsymbol{U}$ where $\boldsymbol{T}$, $\boldsymbol{U}$ may have some structure or may represent a collection of different data types. It is also important to note that type identifiers such as $\boldsymbol{T}$ or $\boldsymbol{U}$ are arbitrarily chosen and the label communicates only the preservation, or lack thereof, of the structure of the input data. The labelling of an input and output type is defined such that if the input and output types of a domain are identical as is the case in a mapping from $\boldsymbol{T}$ to $\boldsymbol{T}$ then the transformation that occurs is said to be structure preserving such that the output contains an identical structure to the input type. An example of this could be structured tabular data with given column headings. If the transformation does not modify this table structure, instead only reading the contents or modifying the table contents then it is said to be structure preserving. It is then possible once a directed graph of each transformation is obtained to perform algebra upon this graph. Such operations may include the simplification of the structure through composing transformations or identifying potential incompatibilities between domains through type mismatches. Each domain in the architecture is constructed from one or many transformations such that the functionality of a domain is defined by the composition of these transformations.

In practise, many concepts may not map naturally to this model. Examples include data storage on disc and databases. In such cases, it is possible to map these as either state machines or simply as entities in the data flow that label a particular complex process. As requirements stabilise and become readily available, the intent is for an architecture defined in this abstract manner to reduce down to a traditional architecture specification.

Designing the system this way allows us to largely disregard the shifting external environment and design a system around the modelled data flow rather than the specific form of the transformations until such time that information exhibits stability. It is only required that information regarding what transformations are required exists. This way, the architecture is largely resilient against variation in both non-functional and many non-technical requirements as each domain is intended to be entirely self-contained with all state being local to that domain and any information that enters the domain is passed directly to it and the given output from a domain depends only upon information contained within that domain.

We propose that the architecture specifies open standard protocols for communication between interfaces of each domain. It is specified that all data be retained so any variation in requirements downstream can be trivially propagated through the signal chain or the entire data set can be rebuilt at any time if a failure occurs somewhere. Similarly, by defining the interfaces between domains, it is possible to enforce properties such as strong and guaranteed cryptography on communications and storage in addition to simple topology modifications due to a standardised interface between domains. While this requires additional work in the implementation stage, by communicating through a unified routing system, it ensures that future software replacing legacy or unsuitable components may develop against a known, open communication protocol removing the possibility that proprietary vendor communication methods hamper third-party inclusion into the architecture. There are many benefits of this approach, as shown in Table 1.

Table 1. Benefits.

- Since the creation of abstract domains is largely trivial and the communication between those domains follows open standards, each domain is fully knowable and may be audited. The system may then easily adapt by localising changes to only the affected domains.
- Adapting to future needs is made viable using this architecture, as it doesn't matter whether the software used to power a particular domain is open source or proprietary as long as it conforms to the open standard data storage formats and communication protocols, it can be replaced or upgraded.
- There is much less chance of a given software company creating a monopoly in the business domain by providing a large monolithic system that is proprietary and does not allow (or limits) the ability for third-parties to build upon or interface with it.
- There is opportunity for innovation because anyone can develop candidate solutions for domain specialisation without needing to invest effort in satisfying criteria regarding licensing other vendor APIs. It also allows for larger scale international collaboration.
- The organisation is free to choose any software, open source or proprietary to specialise each domain. We specify in our prototype biobank implementation architecture open source software because for our purposes existing solutions exist for many of the domain specialisations and it is possible to implement new functionality upon the existing code bases with relative ease. However, the client remains free to choose the software solutions they deem adequate. The only requirement is that the communication between domains follows open protocols with implementations provided either by an existing library or directly as part of the core infrastructure.
- There is a potential for reduced maintenance costs. If a decision is made to deploy an entirely open source system, not only does there not typically exist a license cost, there may exist multiple options regarding organisations able to support and maintain the system. That way they can receive quotes and optimise expenditure based on the value each quote offers.
- Since rigid software design processes may stagnate and impede innovation. By having a modular system, any organisation may be required to innovate whether it is by feature set or cost as there may not exist a possibility to implant a system at the project's inception and rely on the difficulty of switching to a competing product as a source of longevity in the deployed infrastructure.

With this in mind, we believe that this approach is not limited to a single field (pathology, genetics or similar) and does not depend on a single company. This is a general model that can apply to any domain of any size. This is where the novelty and innovation of this approach lies. We suggest a system design method that is resilient to changing requirements and constraints while being dependent only upon technological requirements. This method can potentially adapt and grow to any scale and is both modular and knowable.

There do, however exist limitations to our study. We had limited access to end-users as only a limited number of healthcare professionals were directly participating in the process. We thus had to rely on application and service providers, who served as an intermediary between the researchers and the end-users. However, the application and service providers are established and well known in their domain and have a strong knowledge of the end-users' needs and requirements. We can thus rely on their experience for making informed decisions. Furthermore, the practical generalisability of our results is limited until the process is used in other domains. We believe that generalisation issues are likely to be negligible, as this property constitutes a fundamental design philosophy in our approach.

5. SUMMARY

A post-mortem analysis was conducted for the biobank infrastructure process, where several challenges are encountered. Challenges were presented e.g. by the strict requirements for privacy and anonymity as well as rigid processes involved with the patient data. We have identified several challenges and solution proposals. Table 2 shows the overview of proposed solutions and summarises how the challenges may be addressed.

Table 2. Overview of proposed solutions.

- Changing requirements, components and environments are tackled with an iterative process that builds a shared understanding, shaping of goals and permits reaction to changes. Furthermore, a Continuous Renewability approach enables constant feedback and mitigates the effects of changes as the process progresses.
- Several communications related issues are tackled with an iterative process, as it allow stakeholders to see the system evolve. Improved communication practices also are necessary from the initial stages of the project to create a common vision that continues through the entire development cycle. Constant communication also builds trust between stakeholders.
- The Master Architecture provides the scope and guidance for the development work. The first draft of the architecture serves as a guide for development. The basic data flows are then defined. The Master Architecture provides a checkpoint from which the design may be verified. Designing the system around data flows limits complexity as well as the effects of a changing external environment.
- An iterative approach builds gradually towards the final result. The modular and abstract system architecture allows modifying the design with minimal effort and impact to other parts of the system. Abstraction and separation of concerns allows for an adaptable design that scales to accommodate future needs.
- Separation of non-technological issues from technological issues simplifies the design as it limits the effects of internal politics and rigid processes from the technological concerns.
- The separation of the architecture into isolated domains connected through a common interface can serve to restrict the propagation of errors through the system in the event of component failure or modification. This in turn has the potential to offer greater flexibility and expansion of the system to meet future needs.
- Interfaces between the domains utilise open-standard communication protocols. This ensures that the components can be changed according to future needs.

By following these proposals the resulting architecture will be a) modular, b) interoperable, c) controlled and d) abstracted. This approach is general by definition and should be easily adapted to other domains. It is suitable for evolving environments, thus addressing our research question. The resulting architecture from this process is discussed at [15].

6. REFERENCES

[1] Leist, S. and Zellner, G. 2006. Evaluation of current architecture frameworks. In *Proceedings of the 2006 ACM symposium on applied computing*. 1546-1553.

[2] Brancheau, J.C., Schuster, L. and March, S.T. 1989. Building and implementing an information architecture. *ACM SIGMIS Database 20(2)*, 9-17.

[3] Auffray, C., Charron, D. and Hood, L. 2010. Predictive, preventive, personalized and participatory medicine: back to the future. *Genome Med 2(8)*, 57.

[4] Ferrara, F.M. 1998. The standard 'healthcare information systems architecture and the DHE middleware. *International Journal of Medical Informatics 52(1)*, 39-51.

[5] Hofmeister, C., Kruchten, P., Nord, R. L., Obbink, H., Ran, A., and America, P. 2007. A general model of software architecture design derived from five industrial approaches. *Journal of Systems and Software*, *80*(1), 106-126.

[6] Tang, A., Han, J. and Chen, P. 2004. A comparative analysis of architecture frameworks. In *11th Asia-Pacific Software Engineering Conference, 2004*. 640-647.

[7] Franke, U. Höök, D., König, J., Lagerström, R., Närman, P., Ullberg, J. ... and Ekstedt, M. 2009, EAF2-a framework for categorizing enterprise architecture frameworks. In *10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*, 2009. 327-332.

[8] Mangan, P. and Sadiq, S. 2002. On Building Workflow Models for Flexible Processes. In *Proceedings of the 13th Australasian Database Conference*, 103-109.

[9] Birk, A., Dingsoyr, T. and Stalhane, T. 2002. Postmortem: Never leave a project without it. *IEEE Software 19(3)*, 43-45.

[10] Myllyaho, M., Salo, O., Kääriäinen, J., Hyysalo, J. and Koskela, J. 2004. A review of small and large post-mortem analysis methods. In *Proceedings of the ICSSEA*, 1-8.

[11] Collier, B., DeMarco, T. and Fearey, P. 1996. A defined process for project postmortem review. *IEEE Software 13(4)*, 65-72.

[12] Dingsøyr, T. and Moe, N.B. 2001. Augmenting experience reports with lightweight postmortem reviews. In *Product Focused Software Process Improvement*, 167-181.

[13] Scupin, R. 1997. The KJ Method: a technique for analyzing data derived from Japanese ethnology. *Human Organization, vol. 56*, 233-237.

[14] Cruzes, D.S. and Dyba, T. 2011. Recommended steps for thematic synthesis in software engineering. In *International Symposium on Empirical Software Engineering and Measurement*, 2011, 275-284.

[15] Hyysalo, J., Keskinarkaus, A., Harper, G. and Sauvola, J. 2017. Architecture Enabling Service-oriented Digital Biobanks. In *Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS-50)*, January 4-7, 2017, Hawaii.