AKADÉMIAI KIADÓ

Pollack Periodica • An International Journal for Engineering and Information Sciences

17 (2022) 1, 7–11

DOI: 10.1556/606.2021.00372

ORIGINAL RESEARCH PAPER

The impact of the software architecture on the developer productivity

Grácián Kokrehel* and Vilmos Bilicki

Department of Software Engineering, Faculty of the Science and Informatics, University of Szeged, Dugonics tér 13, H-6726, Szeged, Hungary

Received: December 31, 2020 • Revised manuscript received: June 24, 2021 • Accepted: June 29, 2021
Published online: November 1, 2021

ABSTRACT

Distinct technological trends seriously influence the modern software architectures. In this paper, four different software architectures and framework combinations were generally compared. The basis for the analysis is the developer's productivity.

In a 3 year-long research and development project, a real-world telemedicine application was efficiently implemented four times with various software architectures and architectural patterns. More than 5,000 person-hours were spent on carrying out them.

At present, a unique dataset is available, which provides the opportunity to compare the cost of spent person-hours in different approaches.

The goal of this research is to describe the measurement approach, the dataset and the applied architectures considering the software developer's productivity.

KEYWORDS

software architectures, developer productivity, full stack, framework, telemedicine

1. INTRODUCTION

Modern technological trends seriously influence software architectures. Typically, software developers have to select the best from an enormous variety of possible approaches and extract information from blogs and Git repositories. Unfortunately, these sources are technology-oriented since they focus on the How, not on the Why questions. There are few phases of software development: requirement, design, coding/implementation, testing, deployment, maintenance. This paper focuses on a software design and within that the architectural design. In a 3 years long Research and Development (R&D) project, a real-world telemedicine application was efficiently implemented four times with various software architectures and architectural patterns. More than 5,000 person-hours were spent on carrying out them. At present, a unique dataset is available, which provides the opportunity to compare the cost of spent person-hours in different approaches. The basis for the analysis is the developer's productivity.

Is a server-less architecture more efficient than classical server-based software architecture? Questions like this can be answered under these circumstances:

- Dataset with Ticket system;
- Same team through the development process;
- Same or similar architectures;
- Similarly experienced team.

*Corresponding author.
E-mail: kokrehel@inf.u-szeged.hu

2. STATE OF THE ART IN SOFTWARE ARCHITECTURE

It is all too common for developers to start coding an application without a formal architecture in place. Without a clear and well-defined architecture, most developers and

architects will resort to the de facto standard traditional layered architecture pattern, Mark Richards [1], focuses on five architectures that are commonly used to organize software systems: Layered (n-tier) architecture, Event-driven architecture, Microkernel architecture, Microservices architecture and Space-based architecture.

The Layered Architecture style is focused around dividing software functionality into distinct layers that are interacted and stacked vertically on top of each other. Functionality within each layer is related by a common role or responsibility. Communication between layers is explicit and loosely coupled. The main application of layering helps to support great separation of concerns that in turn, support maintainability and flexibility [2].

Microservices are an architectural style in which a single application is built by using multiple small services. Each service runs in its own process and communicates with other services by using lightweight mechanisms, often REST web services. Each service is a small component, which performs a single business purpose - authentication, notification, payment processing, etc. The different services may use different technology stacks - databases, frameworks or programming languages. They are also independently deployable and scalable [3].

Event-Driven Architecture (EDA) is a popular distributed asynchronous architecture pattern, which can be used to overcome the distributed data challenges [3].

Server-less computing is any computing platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running [4]. Based on Eismann, et al. [5] analysis of 89 server-less applications, they find that the most commonly reported reasons for the adoption of server-less are to save costs for irregular or bursty workloads, to avoid operational concerns, and for the built-in scalability. Server-less applications are most commonly used for short-running tasks with low data volume and bursty workloads but are also frequently used for latency-critical, high-volume core functionality. The Berkeley team [6] from the University of California predicts that server-less use will skyrocket. They also project that hybrid cloud on-premises applications will dwindle over time, though some deployments might persist due to regulatory constraints and data governance rules. So far, very few sources were found by comparing server-based efficiency with serverless.

3. STATE OF THE ART IN THE FIELD OF PRODUCTIVITY MEASUREMENT

In software engineering, productivity is frequently defined, from an economic viewpoint, as the effectiveness of productive effort, measured in terms of the rate of output per unit of input. Consequently, direct measures do not characterize the construct of productivity in an economic sense of the term, i.e., by means of an association between some input effort and the quantity of output obtained as a result [7, pp. 77]. Enterprise resource planning-system is one of the instruments to implement disruptive ideas in business and create long-time competitive advantages [8]. Fuzzy based production planning became more accurate and more realistic than the result of a traditional network model due to the factors of unpredictability of the human workforce and those of unforeseeable interruptions in the production [9].

Measurement techniques used worldwide to measure developer's productivity [10]: Sprint Burndown, Team Velocity metric, Throughput, Cycle Time, etc.

The throughput method indicates the total value-added work output by the team. They are represented by tickets completed by the team within a specified time. The cycle time is the total time that has elapsed from the moment a ticket started until the task is completed. A mixture of these two methods was used in the measurements. All of the methods mentioned have been tried, but the data set that stores the data in the form of Tickets has limited and made it impossible to make measurements in some cases. There are many variable names for measuring Time, Effort, Task, Function Points and Lines of Code. The diversity between Time and Effort occurred because they were measured with different time units. Function Points and Lines of Code had different names because of the different methods or strategies used by researchers [7, pp. 83]. A version of the less popular Task/Time method was used.

4. MEASUREMENT APPROACH

Over the 3 years, 4 projects ran in parallel and 5,000+ man-hours of logs were collected. Many beginners with no experience have been involved in the development process. Developers have created custom guides over the years: clean code, objective oriented programming, programming paradigms, design patterns and architectural designs. This also allowed new trainees and students to integrate faster. In the research project all the architectures mentioned later have been tested over the years. The other three projects were evaluated with just one of the architecture. The research project is a real world Telemedicine application. Throughout this project, the managers were the same and the Kanban [11] development process was used. Everyone receives tasks in the form of tickets and has been rated based on work-logs/completed tickets. There are more than 238 tickets in this project alone.

The FHIR [12] standard for medical purposes is used for data modeling. Table 1 shows the developer team experience. Software Architecture version number is SAvn.

Considering that this is an R&D project, so the individuals involved in the process are still students or trainees with little or no experience. The measurements were not biased by this fact.

5. APPLIED SOFTWARE ARCHITECTURES

The main task of the Custom API(SDK) is to connect Firebase and frontend. All queries, view tables, cloud

Table 1. The developer team experience

SAvn/ number of the developers	SAv1 (May 2018 - Jan 2019)	SAv2 (Feb 2019 - Sep 2019)	SAv3 (Nov 2019 - Feb 2020)	SAv4 (Mar 2020 - Sep 2020)
Beginner	16	8	3	0
1YearOld	0	0	5	2
2YearOld	0	0	0	6

functions, triggers, email functions, etc. are stored in one npm package (Fig. 1).

The so-called "Custom LIB" is a set of similar but parameterizable GUI elements that can then be used in other projects. It consists mainly of (simple) components suitable for visualization purposes. It also contains (powerful) widgets and model-specific elements with complex functions (Fig. 2). Table 2 shows the used technologies per version.

6. RESULTS AND DISCUSSION

Figure 3 shows the total time spent by developers on different architecture versions:

$$\text{Time spent}(H) = \sum\left(\frac{\text{Ticket}(H)}{\text{Developer}}\right), \tag{1}$$

where Ticket (H) is the time spent on ticket/task.

The difference is huge but other factors need to be considered in SAv1:

– testing and implementation of agile and kanban methods;
– inexperienced team;

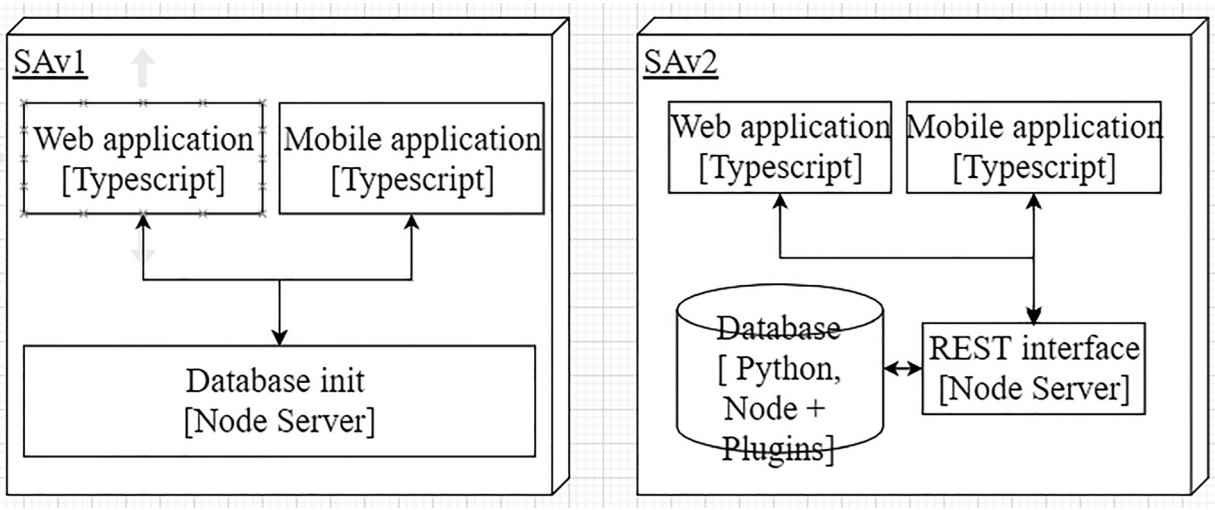

Fig. 1. SAv1 and SAv2

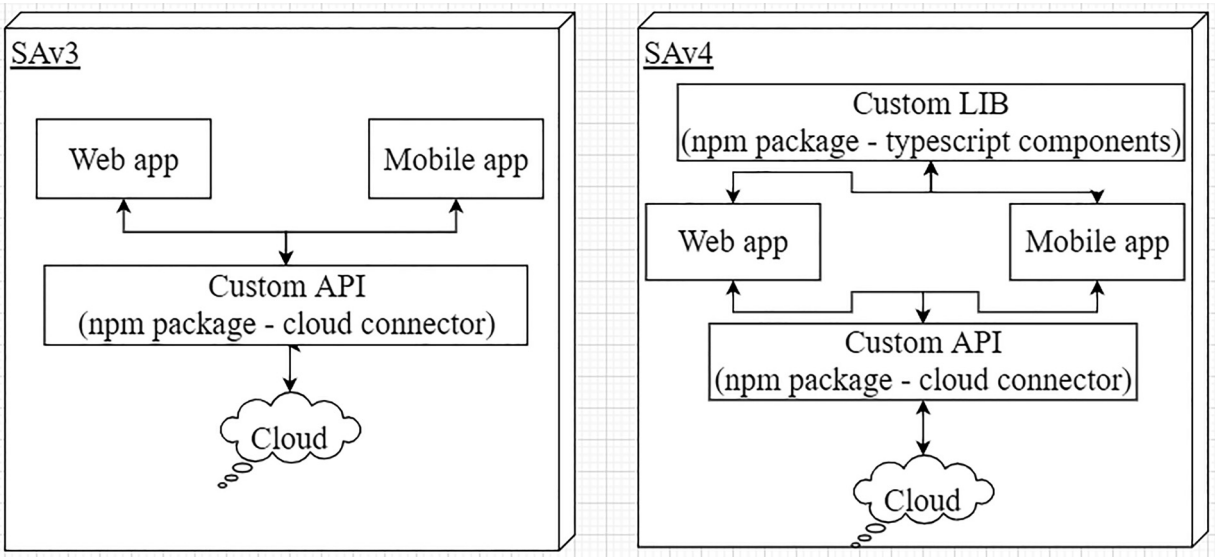

Fig. 2. SAv3 and SAv4

Table 2. Technologies per version

	SAv1	SAv2	SAv3	SAv4
Web app.	Angular Material	Angular Material	Angular Material	Ionic 4
Mobile app.	Ionic 3	Ionic 3	Ionic 3	Ionic 4
Database	Apache Cassandra	Apache Cassandra	Firebase	Firebase
Plugin	Lucene index	Lucene index	-	-
Server/Cloud	Loopback	Loopback	Firebase	Firebase
Connector	-	-	Angular Firestore	Angular Firestore

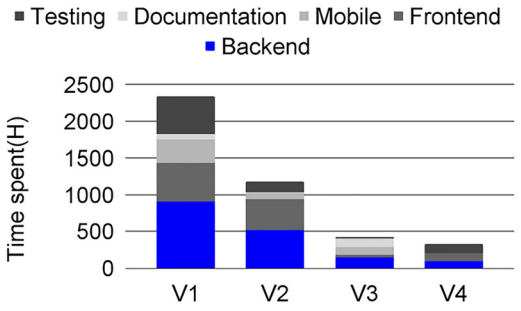

Fig. 3. Total time spent by developers on different architecture versions

– incomplete functional;
– requirement and design plans.

It can be clearly seen that the time between SAv1 (2341 H) and SAv4 (191 H) differs greatly, the latter being completed 8 times faster.

The difference between server-based and non-server-based systems is clearly indicated by the large change between SAv2 and SAv3. Developers spent a lot less time learning and setting up the day-to-day system, so a lot more tickets were completed.

Figure 4 shows the time spent by developers on different categories. It is clear that Backend-related tasks have taken up the most time. The frontend was refactored only during SAv2, later there were only bug fixes. The mobile application has been completed in SAv3.

Figure 5 shows the subtasks that consumed the most time. SAv4 tasks: 55.5 h to implement own server to store patient data, 42 h for Firebase analytics.

Figure 6 shows the completed functions on different architecture versions. For SAv4, all functions are 100%.

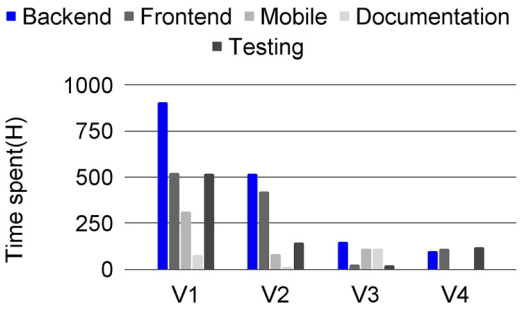

Fig. 4. Time spent by developers on different categories

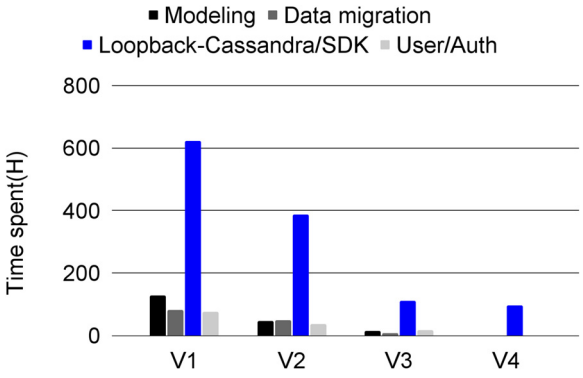

Fig. 5. Subtasks that consumed the most time

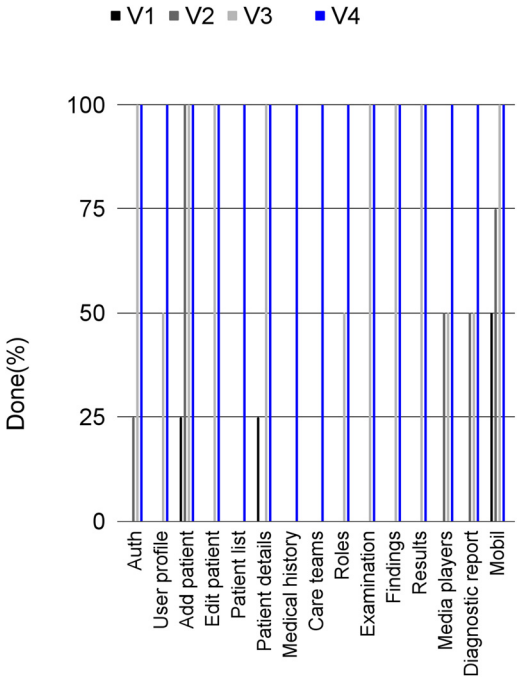

Fig. 6. Completed functions on different architecture versions

These requirements have changed and expanded over time. Yet in the post-SAv3 period, it was much easier to implement and demonstrate the new features described by customers and managers. It is important to mention that the maintenance of existing functions has also been greatly simplified.

7. CONCLUSION

The results show that in our case, Server-less Firebase-based architectures can be implemented in up to 8x less time than private cloud IaaS level architectures.

Based on the measurements our findings are the following: Server-less architectures are much more efficient than IaaS level architectures. Based on the results, it is easy to state that experience and skilled management can give completely different results for some architectures. If possible, it would be worth running the experiment again based on data from larger companies.

Future plans are:

- Command line interface for faster application initialization;
- Measure how fast the custom library can generate projects;
- How much code review affects productivity;
- Visualize metrics that motivate developers.

ACKNOWLEDGMENTS

The research project took place at the University of Szeged, Hungary. This research was supported by the EU-funded Hungarian grants EFOP-3.6.1-16-2016-00008, EFOP-3.6.3-VEKOP-16-2017-00002 and GINOP-2.2.1-15-2017-00073.

REFERENCES

[1] M. Richards, *Software Architecture Patterns*. O'Reilly Media, Inc., 2015.

[2] F. Akmel, E. Birhanu, B. Siraj, and S. Shifaw, "A comparative analysis on software architecture styles," *Int. J. Foundations Comput. Sci. Technol.*, vol. 7, no. 5/6, pp. 11–22, 2017.

[3] S. Zelev and A. Rozeva, "Using microservices and event driven architecture for big data stream processing," *AIP Conf. Proc.*, vol. 2172, no. 1, 2019, Paper no. 090010.

[4] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of server-less computing," *Commun. ACM*, vol. 62, no. 12, pp. 44–54, 2019.

[5] S. Eismann, J. Scheuner, E. van Eyk, M. n Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, "Server-less applications: Why, When, and How?," *IEEE Softw.*, vol. 38, no. 1, pp. 32–39, 2020.

[6] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, J. E. Gonzalez, R. A. Popa, I. Stoica, and D. A. Patterson, *Cloud Programming Simplified: A Berkeley View on Server-Less Computing*, UC Berkeley, 2019.

[7] E. Oliveira, D. Viana, M. Cristo, and T. Conte, "How have software engineering researchers been measuring software productivity?" in *Proceedings of the 19th International Conference on Enterprise Information Systems*, vol. 2, Porto, Portugal, April 26-29, 2017, pp. 76–87.

[8] M. A. Razzhivina, B. A. Yakimovich, and A. I. Korshunov, "Application of information technologies and principles of lean production for efficiency improvement of machine building enterprises," *Pollack Period.*, vol. 10, no. 2, pp. 17–23, 2015.

[9] L. Pusztai, B. Kocsi, and I. Budai, "Making engineering projects more thoughtful with the use of fuzzy value-based project planning," *Pollack Period.*, vol. 14, no. 1, pp. 25–34, 2019.

[10] Top 10 software development metrics to measure productivity, [Online]. Available: https://www.infopulse.com/blog/top-10-software-development-metrics-to-measure-productivity/. Accessed: Dec. 22, 2020.

[11] D. J. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*, Blue Hole Press, 2010.

[12] Index - FHIR v4.0.1. [Online]. Available: http://www.hl7.org/fhir/. Accessed: Dec. 22, 2020.