Caring is Sharing – Exploiting the Value in Data for Health and Innovation
M. Hägglund et al. (Eds.)

doi:10.3233/SHTI230286

Health Research Requires Efficient Platforms for Data Collection from Personal Devices

Erlend JOHANNESSEN[a,1], André HENRIKSEN[a], Eirik ÅRSAND[a], Alexander HORSCH[a], Jonas JOHANSSON[b] and Gunnar HARTVIGSEN[a]
[a] *Dept. of Computer Science, UiT The Arctic University of Norway, Tromsø, Norway*
[b] *Dept. of Community Medicine, UiT The Arctic University of Norway, Tromsø, Norway*
ORCiD ID: Johannessen 0000-0003-4860-9192, Henriksen 0000-0002-0918-7444, Årsand 0000-0002-9520-1408, Horsch 0000-0001-7745-0139, Johansson 0000-0001-7912-5786, Hartvigsen 0000-0001-8771-9867

Abstract. Data from consumer-based devices for collecting personal health-related data could be useful in diagnostics and treatment. This requires a flexible and scalable software and system architecture to handle the data. This study examines the existing mSpider platform, addresses shortcomings in security and development, and suggests a full risk analysis, a more loosely coupled component-based system for long term stability, better scalability, and maintainability. The goal is to create a human digital twin platform for an operational production environment.

Keywords. Infrastructure, Scalability, Human Digital Twin

1. Introduction

Physical activity (PA) trackers and smartwatches can be used for health data collection in research as an addition to existing methods [1], and the data collected could be used to support patient diagnostics and treatment [2], see overview by Henriksen et al. [3].

For collecting data from many different device suppliers, a flexible and robust solution is needed, that can also receive data from heterogenous sources. The mSpider (Motivating continuous Sharing of Physical activity using non-Intrusive Data Extraction methods Retro- and prospectively) system is an experimental tool designed for automatic and continuous collecting of health-related data recorded by consumer-based activity trackers [4]. It has been designed to collect data of various PA-variables from activity trackers from a range of different providers. Today's activity trackers are smart devices capable of collecting many PA-variable estimates and transferring them to a smartphone for persistent storage. In their study, Henriksen et al. [4] collected smartwatch data using the mSpider system.

The current mSpider architecture consists of two servers, an administrative user-facing system (front-end), and a back-end server for gathering data by using the

[1] Corresponding Author: Erlend Johannessen, Department of Computer Science, UiT The Arctic University of Norway, PO Box 6050 Langnes, N-9037 Tromsø, Norway. E-mail: erlend.johannessen@uit.no.

manufacturers' public APIs. In addition, a mobile application has been made for those manufacturers (notably Apple and Samsung) where data only are available through SDKs provided by the manufacturers. **Figure 1** shows the original mSpider architecture.

In mSpider, the *participants* are enrolled in a *study*, after which they only need to wear their smart watches to collect and share data. Their activity data are uploaded to the device manufacturers' respective clouds and then pulled to the mSpider back-end server through the manufacturers' APIs.

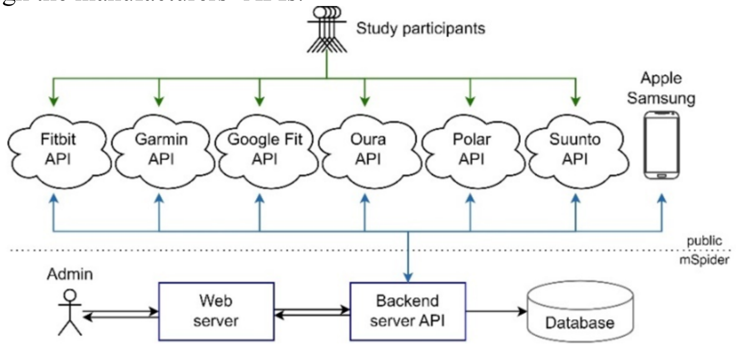

Figure 1. Original mSpider data collection architecture.

The programming language used for the mSpider back-end server was Go (go.dev, open source). The admin front-end used Node.js (nodejs.org, open source) for serving HTML, CSS, and JavaScript, and Angular (angular.io, open source) for creating the user interface. All data were stored in a MongoDB (mongodb.com, US) database. Both servers were run in a Docker (docker.com, US) environment using Docker-Compose on a single Ubuntu Linux (canonical.com, UK) server.

The goal of this study is to examine the mSpider system, identify problems and possible improvements, and to discuss an architecture capable of collecting data from a large population, over an extended period of time.

A digital twin is defined as a digital representation of physical entity. A *human* digital twin is maybe not fully realisable at the moment, but there is potential for creating small-scale human digital twins today, by joining different types of data from digital services and sources. The ambition aim is to add more data types and more data sources to the mSpider system, in order to approach a human digital twin system [5,6].

2. Methods

To analyse the state of the mSpider system and uncover problems, several experts were involved in interviews, and using the "think aloud" method, in a qualitative study approach [7], including the following steps:

1. Questioning the researchers using mSpider on how they experienced the system, and what they disliked or missed.
2. Discussing with the original developers and operational staff members behind mSpider, outlining the decisions governing the development of the system.
3. Reviewing the source code of the system, to understand the functionalities and state of the system.

3. Results

Observation done via researchers, developers (interview and code review), and operational staff are shown in **Table 1**. Priorities indicate the importance of the respective improvement. Issues are assigned to one type of actor, although some issues concerned or had consequences for several or all actors. Issues given low priority are not described in detail.

Table 1. Findings from researchers, developers, and operational staff, including code review. The rows are coloured differently to separate actors' observations from each other.

Actors	Observations	Priority
Operational staff	Security patches would be difficult to apply because of the way mSpider was developed and maintained	High
Operational staff	No assessment of threats or risks done on the system	High
Operational staff	Running system inside university network could theoretically put university network at risk	High
Operational staff	Every production update means deploying the full system. If something malfunctions the whole system may malfunction	Medium
Developers	Third-party components used got outdated and were no longer maintained and could not easily be replaced	High
Developers	Back-end server was a tightly coupled monolithic architecture	High
Developers	Back-end server was responsible for everything: participant consent dialogue, data collection, management, data extraction, batch runner for historical data	High
Developers	Device data were saved in the same storage, giving a format that did not suit all devices	Medium
Developers	Changing provider behaviour creates a new deployment of the full system	Low
Developers	Non-relational database may not be ideal for storage technology when there was need for combining several collections	Low
Developers	Different metadata were saved in the same storage collection	Low
Developers	Adding a new provider, initiated changes across the whole code base	Low
Researchers	A limited number of variables collected, e.g., daily step count, energy expenditure (kcal), and moderate or vigorous physical activity (PA)	Low / Medium
Researchers	Inefficient and cumbersome user interface	Low
Researchers	Only rudimentary data extraction from the system	Low
Researchers	Limited management functionality for study data	Low
Researchers	Limited management functionality for participant data	Low
Researchers	Data collection complexities with regards to when devices add their data to the manufacturer's cloud	Low

Several issues were uncovered from code reviews and from talking to developers. Device data from different manufacturers were saved into the same document storage, giving a general format that did not suit different data from different manufacturers. This required comprehensive mapping methods when reading from the different collections, since the various manufacturers have different data models. When used with a proof-of-concept system with limited data collection this worked but would be too complex when expanding on more data variables. Another issue was the use of third-party components in the source code. Using community code in your project is normally not an issue, but problems arise when packages get outdated and are no longer maintained. This could be due to lack of security fixes, but also because the package is outdated with regards to functionality as to what the package was meant to solve. As an example, the Golang package used for MongoDB database access did not work with newer (and more secure) versions of MongoDB. This package was deeply integrated with the code and could not be easily replaced.

The back-end server (see **Figure 1**) was implemented as a monolith, which normally is not a problem, but a tightly coupled monolithic system tends to end up as a "big ball of mud" [8]. The server was responsible for everything, including data collection from APIs, being a receiving API for the mobile mSpider clients (for Apple and Samsung), being a management and data extraction system for researchers and admin personnel, and a batch runner for gathering historical data from the device providers. This is a lot of responsibility for a system and is difficult to maintain.

Operational staff were mainly concerned with security and deployment of the mSpider system, and there were some problems with the solution running inside the university network. Every production update meant deploying the full system. Theoretically, if something malfunctioned the whole system could malfunction. Security patches would be difficult to apply because of the way mSpider was developed and maintained. From reviewing the mSpider project we found that several security measures were implemented in mSpider, but there was no assessment of threats or risks. Because of this, data was collected and stored anonymously. Running the current system on a single Linux server on the university premises was a problem with regards to security for mSpider but could also pose a problem for the university's security.

Based on the expert-interviews and the source code review, we identified several requirements for a new system version. The new system should:

1) Be ready for productive use in a professional health context.
2) Be scalable with regard to new devices.
3) Be scalable with regard to data volume.
4) Collect more diverse data or groups of data.
5) Make data interpretable and easily available to researchers from various disciplines.

4. Discussion

Risk assessments are used to expose undesirable incidents in systems and evaluate probability of occurrence. To be production-ready the new system needs a thorough risk assessment, and the security needs to improve for the system not to risk leaking collected data. The system also needs to be running continuously, so a stable set of services is necessary. This is dependent on how the software architecture is implemented, and some software principles are essential for this, among them separation of concerns and extensive use of interfaces when using the relevant programming languages.

Using a secure and capable runtime environment is key, and a suggestion for this is running the solution in Microsoft Azure or another cloud computing service.

An important feature in the new system would be scalability with regards to new devices. One way of solving this would be to create a service for each data provider so that change to one device provider's API, access method, or data collected only would affect a single service. This would also make it easy to add new providers to the system, in that the new provider could be developed and tested in isolation from the other services. Several services could also be added for each provider, so that the solution is scalable with regards to gathering increasing amounts of data from the population.

The storage system will be created such that data from each provider could be stored in their own databases. Creating the possibility for a separate database or localization of storage for each provider would increase scalability for storage, which again opens up for federated database technology [9].

Data collected are intended to be heterogeneous and the system should store as diverse and plentiful data as practically possible. The next version of mSpider should be expanded so that it covers more data types (e.g., pulse, sleep, temperature, body composition) and makes a low-resolution human digital twin [5] possible.

5. Conclusion

We have identified requirements for a production ready, scalable, and flexible data collection system for personal devices. One of the overarching goals for the new mSpider system is to enable it for population-based research investigating potential changes in a population's lifestyle and health. This would imply continuous data collection from the population to create data-driven analyses, working towards a human digital twin [6]. Researchers should be able to create a research project by initially setting some parameters for what data they want to be included in the study, such as steps, heart rate, weight, and sleep. These data could be extracted daily into a warehousing system [10]. This data collection system could give new opportunities for public health research.

References

[1] Brickwood K-J, Watson G, O'Brien J, Williams AD. Consumer-Based Wearable Activity Trackers Increase Physical Activity Participation: Systematic Review and Meta-Analysis. JMIR MHealth UHealth. 2019;7:e11819. doi: 10.2196/11819.

[2] Gwaltney C, Coons SJ, O'Donohoe P, O'Gorman H, Denomey M, Howry C, Ross J, ePRO Consortium. “Bring Your Own Device” (BYOD): The Future of Field-Based Patient-Reported Outcome Data Collection in Clinical Trials? Ther Innov Regul Sci. 2015;49:783–791. doi: 10.1177/2168479015609104.

[3] Henriksen A, Mikalsen M, Woldaregay A, Muzny M, Hartvigsen G, Hopstock L, Grimsgaard S. Using Fitness Trackers and Smartwatches to Measure Physical Activity in Research: Analysis of Consumer Wrist-Worn Wearables. J Med Internet Res. 2018;20:e110. doi: 10.2196/jmir.9157.

[4] Henriksen A, Johannessen E, Hartvigsen G, Grimsgaard S, Hopstock LA. Consumer-Based Activity Trackers as a Tool for Physical Activity Monitoring in Epidemiological Studies During the COVID-19 Pandemic: Development and Usability Study. JMIR Public Health Surveill. 2021;7:e23806. doi: 10.2196/23806.

[5] Johannessen E, Henriksen A, Hartvigsen G, Horsch A, Årsand E, Johansson J. Ubiquitous digital health-related data: clarification of concepts. Scand Conf Health Inform. 2022;52–58. doi: 10.3384/ecp187009.

[6] Miller ME, Spatz E. A unified view of a human digital twin. Hum-Intell Syst Integr. 2022;4:23–33. doi: 10.1007/s42454-022-00041-x.

[7] Eccles DW, Arsal G. The think aloud method: what is it and how do I use it? Qual Res Sport Exerc Health. 2017;9:514–531. doi: 10.1080/2159676X.2017.1331501.

[8] Big Ball of Mud [Internet]. [cited 2022 Nov 14]. Available from: http://www.laputan.org/mud/.

[9] Sheth AP, Larson JA. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput Surv. 1990;22:183–236. doi: 10.1145/96602.96604.

[10] Chaudhuri S, Dayal U. An overview of data warehousing and OLAP technology. ACM SIGMOD Rec. 1997;26:65–74. doi: 10.1145/248603.248616.