Elements of Sustainability for Public Sector Software – Mosaic Enterprise Architecture, Macroservices, and Low-Code

Manu Setälä[1], Pekka Abrahamsson[2], and Tommi Mikkonen[2,3]

[1] Solita, Tampere, Finland
manu.setala@solita.fi
[2] University of Jyväskylä, Jyväskylä, Finland
pekka.abrahamsson@jyu.fi, tommi.j.mikkonen@jyu.fi
[3] University of Helsinki, Helsinki, Finland
tommi.mikkonen@helsinki.fi

Abstract. Public sector is a large consumer for software. In countries such as Finland, many of the systems are made to order by consultancy companies that participate in public tenders. These tenders initiated by the state, cities, and other public sector organizations. Furthermore, as public sector tasks are often decomposed to various actors, each and every one of them makes their purchase based on their own needs. In this paper, we argue that to maintain software sustainability in this context, there is a need for three key elements. Firstly, there is a need for an enterprise architecture where independent services from various vendors are can be easily deployed and integrated. Secondly, these services are build in a such manner that they can interact via well-defined APIs, but need no direct access to other services. Finally, techniques that support systematic, rapid development and deployment are needed.

Keywords: Public sector software · mosaic architecture · macroservices · software sustainability.

1 Introduction

Public sector is a large consumer for software that public sector has multiple, conflicting, and often intangible goals [2]. In countries such as Finland, many of these systems are made to order by consultancy companies that participate in public tenders, initiated by the state, cities, and other public sector organizations. Furthermore, as public sector tasks are often decomposed to various actors, each and every one makes their purchase based on their own needs.

This approach has led to surplus of information systems. For instance, a recent study on information systems at a city of Kerava, Finland, with 35.000 inhabitants found out that here were 93 information systems that interacted in one way or another [13]. Extrapolating from this, as there are over one hundred cities in Finland, one can conclude that there are thousands of information systems, many of which made to order. Furthermore, while designing systems to

order based on a public tender arguably fosters competition, the model has also been considered time consuming and error-prone [7].

While software is (almost) free to copy, there is no point in copying a piece of software that is specifically crafted for single client, solving a single problem that no-one else has. Since all software needs maintenance, this model leads to overly expensive use of software, and replacing it is next to impossible, because the replacement should be done following the same model – public tender and bidding for contracts. In other words, vendor lock-in forces municipalities to use a certain product or service, regardless of its quality, because switching away from it can be challenging, and the switching costs may be substantial [14].

To understand the scale of the public sector software we are talking about, let us consider software company Gofore[4] as a representative consultancy company operating with public sector based on public information[5]. It is pointed out that year 2019, 70% of the revenues of the company, totalling 64Me, resulted from public sector. In addition, it is pointed out that during years 2017-2019, public sector has increased on average 45%, whereas private sector only reached 24% at the same time. With several other companies similar to Gofore in the Finnish ICT ecosystem, public sector software has huge potential for exports as well. In total, the state of Finland only made a procurement worth over 1000Me [8] in the field of ICT. Much of the associated software is made to order, and focuses on problems of a single organization only.

In this paper, we argue that to maintain software sustainability in this context, three core elements must be considered:

- mosaic-like enterprise architecture – Mosaic EA for short – for public sector that allows integrating services from various vendors;
- ability to deploy subsystems in a fashion where they can liberally interact, but need no direct access; and
- systematic, affordable way to build subsystems that meet stakeholders' requirements.

The paper is based on observing public sector projects and related tenders as well as on bidding for such projects. The three authors have jointly more than 75 years of experience in practice and in academia. The experiences are drawn mostly from Finland. However, Finland is a good representative of a Northern European highly digitalized welfare state inside the European Union. Importantly, we consider the export possibility as a mechanism to foster sustainability.

2 Background and Motivation

While in many businesses, IT forms the core of all operations, the public sector often needs to outsource the whole solution. For instance, resources are often allocated such that the organization can run its operations but not design and

[4] https://gofore.com/en/
[5] https://gofore.com/vuosi-2020-julkisen-sektorin-kumppanina/

implement new solutions. To foster competition, public sector information system projects are often based on public tenders. Typically, these are formal, structured procedures for generating competing offers from different potential suppliers or contractors, who seek to win a service contracts. The contract can involve several phases of software development and operations. For instance, specification phase can be a separate contract, separated from implementation. Similarly, the contract can also include running daily operations [6].

Upon placing an open tender for the development, organisations often seek to avoid vendor lock-in. However, this is not always successful. Since tendering is only the beginning of the implementation process, chances are that it takes a considerable amount of time before the new system is operational, and during this time, the situation might change. Furthermore, those that loose in the tendering process can slow this even more with claims of unfair tendering. Hence, public sector actors by necessity must provide their services with various systems, some of which are new and some of which may have a long history. This, with the growing tendency to build systems that rely on other systems – for instance using AWS for certain routines via a well-defined API – adds complexity to managing public sector software.

Oftentimes, these tenders are based on a organization-centric view, simply because the acquiring organization focuses the tendering process to its own needs. For an end user citizen that needs such service, the result is that she needs to access the particular information system, and the service cannot be easily linked to other, related services. Hence, the end user needs to access the different systems independently, and ensure that her data in the different systems are in sync.

Based on the above, we argue that to manage its information systems a transition is needed from organization-centric to a citizen-centric model. To implement such, public sector organizations need an enterprise architecture (EA) that fosters open competition for tenders as well as supports multi-vendor operations. Furthermore, this EA must be defined and under the control of the organization in question, as otherwise it no longer can operate independently. We call such model Mosaic EA, since each party can provide individual pieces to the mosaic, whereas the public sector actor(s) fundamentally control and maintain the system as a whole.

3 Proposed Architecture and Approach

Today, there is a common tendency to acquire information systems such that one vendor delivers the whole solution. That way, the development is under the control of the single vendor, who can decide on many of the technical details. However, these systems do not constitute a coherent enterprise for public sector, but each one of them is a separate entity, with minimal interaction with other systems. To improve, we propose that systems are acquired in a fashion where their interfaces enables interaction across the different systems. Based on our view, this calls for reconsidering the role of each vendor and the public stake-

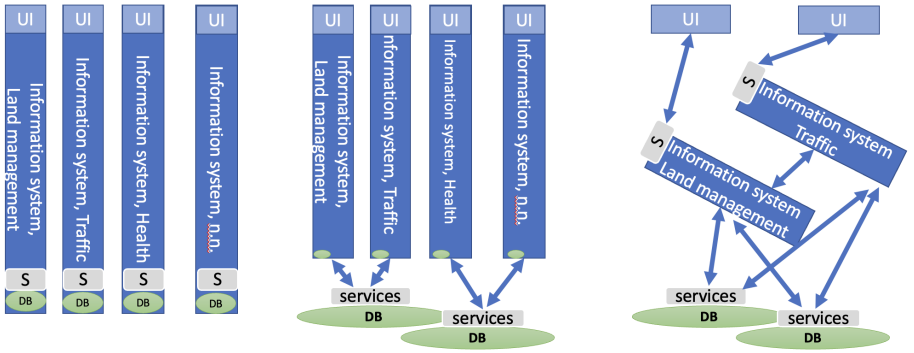

Fig. 1. Mosaic EA illustrated. Left: traditional software subcontracting model: each domain has own solutions, and nothing is shared. Center: Situation in Finland today: mix of shared and private databases, applications made to order. Right: true Mosaic EA, where databases and services are shared, based on open APIs.

holders, so that data and services can be shared across different applications. This is illustrated in Figure 1.

Mosaic Enterprise Architecture. Echoing the findings of [14], the development of a unified enterprise architecture is critical to the success of public sector software systems. The fashion we propose implementing this uses a mosaic as a metaphor. In a true Mosaic EA, the system consists of independent services that can originate from various vendors. A Mosaic EA defines common guidelines that are needed for monitoring and updating the services. Furthermore, openness, transparency and the ability to connect to other systems are at the core of Mosaic EA. In addition, Mosaic EA is used to define APIs for different services. These APIs are the only way to interact with other services, and each API can be implemented by several vendors. Therefore, services can be replaced with new, updated ones, given that the APIs remain unchanged.

Macro-Service Decomposition. The services defined by Mosaic EA have been inspired by micro-services and micro-service architecture [10], but they are of coarser granularity. Hence we call them macro-services. While a micro-service is something that a team of developers can single-handedly design and deploy, macro-services are of size and complexity that can be easily covered by a single public tender. Furthermore, a common requirement is that the macro-service API remains well-defined and maintained, so that it can evolve over time, and they can be replaced without major complications. Furthermore, with well-defined and maintained API, also reusing testware is possible. Hence, macro-services are a tool for the public sector when considering acquiring new information systems. While not an advocated practice in general, real-life examples of such systems are numerous, already when considering only Finland. These include regulatory and legal principles – such as registry legislation or taxation and anti-corruption regulation, as well as national registries, digital health care services, and digital authorization and mandates. In more detail, the Digital Fact Sheet

2019 Finland [3] lists the following services we characterise as macro-services: (i) the eAuthorisations service verifies a person's or organisation's authorisation to use digital services; (ii) a semantic interoperability workbench was implemented; (iii) Two new important registers have been created, incomes register and register of housing company shares. In general, the report presents a number of potential macro-services that seem like ready-made components for Mosaic EA. Obviously, these are just the tip of the iceberg, and there are several candidates when digging deeper in the existing services.

Low-Code Development. The final piece in the proposal is relying on low-code development model, where tools enable rapid development of new systems. The model has been found helpful in automating processes in e.g. manufacturing [16, 12]. A recent Gartner report [15] proposes certain essential use cases for the technology, where productivity gains are identified for professional and citizen development. In addition, considerable benefits are also seen in as speed of delivery. At present, tools for low-code development are plentiful. Often, they support rapid application development, one-step deployment and execution, and management using declarative, high-level programming abstractions. Model-driven development and metadata often play an important role. In parallel, also new development practices, such as opportunistic reuse [4], is opening doors to faster development and deployment, although arguably with a higher risk rate [9]. Both approaches offer faster development and deployment as ever before, to the extent that the whole public sector is at a brink of a potential disruption. At the core of this disruption is true openness at interface level, which combines macro-services and low-code development in the proposed model.

4 Discussion

Our key assumption is that no organization is interested in purchasing and managing small subsystems. Hence, to avoid large, vendor specific subsystems, a supporting enterprise architecture is needed, which we call Mosaic EA. This EA enables managing and operating the system, so that several vendors can participate in the development. In addition, since the organization operating the system needs assume its control, this liberates public organizations from vendors that aim at taking control over the systems. However, the public sector players have to have their own technical expertise to define and maintain the Mosaic EA. Currently, they continue relying on vendor's technical expertise [14], which needs to be changed for the new approach to work.

With this approach, if and when the public sector wants to buy complete systems, any vendor can be the lead in the process, as long as there is a healthy ecosystem producing and operating subsystems for the MEA. This consists of API economy based on clouds [11], reliance on open source [5], and strategic decision for a public actor to avoid vendor lock-in. The main foreseen negative consequence concerns large companies as this lowers the barrier of entry to the field. Indeed, one does not need to be a Global giant to serve governments.

The new model can also catalyst new income in terms of exports, which can be based on the same model. So far, the most focused approach is EU's Gaia-X initiative [1]. At present, the initiative is not yet at the level of Mosaic EA, but it is under active development, aiming to shape the whole EU software industry. We expect that this will inspire new types of software ecosystems, where smaller actors can participate in projects in a key role.

Supporting the proposed more generic approach requires a change in business processes as well. Current business models favor the large vendors and the large subsystems. Moreover, interoperability issues with completely closed, single-vendor systems traditionally introduce problems related to reuse or replacing the system, which the proposed model is trying to eliminate. Similarly, parallel use of several systems, which address the same issue but are designed by different vendors, is often complicated.

Finally, from the technical perspective, it is important to notice that macro-services represent a level of abstraction that can be conveniently specified at a contractual level, not necessarily something that is convenient to implement as such. Instead, macro-services' practical implementation requires their decomposition into smaller entities. Some of these can be new, but interfacing with existing macro-services is also possible. Furthermore, many of the existing implementations rely on open source components, which can be implemented by some other actors than the company that delivers the service to the public sector. In addition, in certain cases relying on AI requires a related approach, where AI related features are separated from the rest of the system, to avoid potential problems related to confidentiality, for instance.

5 Conclusions

In conclusion, when public sector acquires software systems, the requirements are often based on the intermediate needs at hand. This leads to an architecture model, where each system is tailored for a particular actor, and offers little opportunities for reuse. In this paper, we clam that this model is not sustainable for public sector in the long run, but a more ecosystem centric view is needed. As the technical solution that is inline with such ecosystems, we propose following a mosaic-like approach called Mosaic EA as the starting point of both the development and tendering process to initiate it.

With this framework in place, the development of functions can advance from one tendering process at a time. However, tendering and development needs to follow the ideology defined by the Mosaic EA, with guidelines how to deal with open APIs in the technical sense and ecosystem formation guidelines and operations in the process and organizational view. Moreover, since there are several open source projects as well as new methodologies such as low-code development, we expect that the new model will decrease IT costs, due to the increased competition and the ability to select the services in smaller, more understandable, upgradeable pieces, instead of acquiring a single system that must also be upgraded as such.

Achieving this, however, requires a new mindset, where public sector organizations assume responsibility over their information systems. This calls for increased IT and EA competence in the public organizations.

References

1. Braud, A., Fromentoux, G., Radier, B., Le Grand, O.: The road to european digital sovereignty with gaia-x and idsa. IEEE Network **35**(2), 4–5 (2021)
2. Caudle, S.L., Gorr, W.L., Newcomer, K.E.: Key information systems management issues for the public sector. MIS quarterly pp. 171–188 (1991)
3. European Commission: Digital Government Factsheet 2019: Finland (2019)
4. Hartmann, B., Doorley, S., Klemmer, S.R.: Hacking, mashing, gluing: Understanding opportunistic design. IEEE Pervasive Computing **7**(3), 46–54 (2008)
5. Jokonya, O.: Investigating open source software benefits in public sector. In: 2015 48th Hawaii International Conference on System Sciences. pp. 2242–2251. IEEE (2015)
6. Koski, A., Mikkonen, T.: On the windy road to become a service provider: Reflections from designing a mission critical information system provided as a service. In: 2016 International Conference on Information Systems Engineering (ICISE). pp. 51–56. IEEE (2016)
7. Koski, A., Mikkonen, T.: What we say we want and what we really need: Experiences on the barriers to communicate information system needs. In: Requirements Engineering for Service and Cloud Computing, pp. 3–21. Springer (2017)
8. Kähkönen, H.: Growth in government ICT procurement slowed – We list the top 20 suppliers. TIVI, March 12, 2021 (in Finnish). (2021), https://www.tivi.fi/uutiset/valtion-ict-hankintojen-kasvu-hidastui-listasimme-top-20-toimittajat/0b7d7b29-6250-4c8d-aaa3-b21da75ff785
9. Mäkitalo, N., Taivalsaari, A., Kiviluoto, A., Mikkonen, T., Capilla, R.: On opportunistic software reuse. Computing **102**(11), 2385–2408 (2020)
10. Nadareishvili, I., Mitra, R., McLarty, M., Amundsen, M.: Microservice architecture: aligning principles, practices, and culture. " O'Reilly Media, Inc." (2016)
11. Sallehudin, H., Razak, R.C., Ismail, M.: Factors influencing cloud computing adoption in the public sector: an empirical analysis. Journal of Entrepreneurship and Business (JEB) **3**(2), 30–45 (2015)
12. Sanchis, R., García-Perales, Ó., Fraile, F., Poler, R.: Low-code as enabler of digital transformation in manufacturing industry. Applied Sciences **10**(1), 12 (2020)
13. Vilpponen, H., Grundström, M., Abrahamsson, P.: Combining social service and healthcare as the first country in the world : Exploring the impacts on information systems. Journal of Advances in Information Technology **9**(4), 84–88 (2018)
14. Vilpponen, H., Grundström, M., Abrahamsson, P.: Exploring the critical success factors in social and health care information systems project procurement. In: Goel, A.K. (ed.) Recent Developments in Engineering Research Vol. 8. Book Publisher International (2020)
15. Vincent, P., Iijima, K., Driver, M., Wong, J., Natis, Y.: Magic quadrant for enterprise low-code application platforms. Gartner report (2019)
16. Waszkowski, R.: Low-code platform for automating business processes in manufacturing. IFAC-PapersOnLine **52**(10), 376–381 (2019)