Lappeenranta University of Technology
School of Engineering Science
Degree Program in Computer Science

FEDERAL STATE AUTONOMOUS EDUCATIONAL INSITUTION FOR HIGHER PROFESSIONAL EDUCATION NATIONAL RESEARCH UNIVERSITY
«HIGHER SCHOOL OF ECONOMICS»
Faculty of Computer Science
Master's Programme 'System and Software Engineering'

**Dmitry Tsyganov**

# FUNDAMENTAL PROPERTIES OF SAAS ARCHITECTURE: LITERATURE REVIEW AND ANALYSIS OF DEVELOPMENT PRACTICES

Examiners: Associate Professor Uolevi Nikula, LUT
Professor Dmitry Alexandrov, HSE

Supervisors: Professor Ahmed Seffah, LUT

Lappeenranta – Moscow
2018

# ABSTRACT

Lappeenranta University of Technology
School of Engineering Science
Degree Program in Computer Science

FEDERAL STATE AUTONOMOUS EDUCATIONAL INSITUTION FOR HIGHER PROFESSIONAL EDUCATION NATIONAL RESEARCH UNIVERSITY
«HIGHER SCHOOL OF ECONOMICS»
Faculty of Computer Science
Master's Programme 'System and Software Engineering'

Master's Program: Double Degree Program between LUT Computer Science and HSE University, Faculty of Computer Science

Dmitry Tsyganov

**Fundamental Properties of SaaS Architecture: Literature Review and Analysis of Development Practices**

Master's Thesis

55 pages, 4 figures, 6 tables

Examiners: Associate Professor Uolevi Nikula, LUT
Professor Dmitry Alexandrov, HSE

Supervisors: Professor Ahmed Seffah, LUT

Keywords: SaaS, Architectural properties, e-government, Estonia

Software as a Service is a software delivery and licensing model according to which the software is stored and maintained by a service provider in the cloud and accessed by service consumer through a web interface. SaaS has gained popularity among software providers in recent years, with old companies offering their products as a SaaS application and new companies specializing in SaaS from the beginning. SaaS model has been implemented in various businesses and state services. One of the examples of successful SaaS implementation are e-Government services in Estonia. However, the SaaS model introduces new challenges compared to a traditional on-premise model. There are various SaaS-related architectures, frameworks, and practices proposed in the literature. Such studies usually describe architectural properties which are considered to be related to SaaS. This study contains two parts: the first part is a literature review in which important SaaS architectural properties are identified based on a number of scientific papers in which those properties were mentioned. The second part is a case study in which Estonian e-Government services are analyzed against the properties which were identified in the literature review. The conclusions are made about the list of important properties and their significance.

АННОТАЦИЯ

Лаппеенрантский Технологический Университет
Факультет Инженерных Наук
Магистерская Программа по Компьютерным Наукам

Федеральное государственное автономное образовательное учреждение
высшего образования
"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ "ВЫСШАЯ ШКОЛА ЭКОНОМИКИ"
Факультет Компьютерных Наук
Магистерская программа «Системная и программная инженерия»

Master’s Program: Double Degree Program between LUT Computer Science and HSE University, Faculty of Computer Science

Дмитрий Цыганов

**Исследование архитектуры SaaS и анализ практик разработки приложений**

Выпускная Квалификационная Работа Магистра

55 страниц, 4 рисунка, 6 таблиц

Экзаменаторы: Уолеви Никула, доцент, ЛУТ
Дмитрий Александров, профессор, ВШЭ

Руководитель ВКР: Ахмед Сеффа, профессор, ЛУТ

Ключевые слова: SaaS, свойства архитектуры, электронное правительство, Эстония

Программное обеспечение как сервис (SaaS) – это модель доставки и лицензирования програмного обеспечения (ПО), согласно которой программное обеспечение хранится и обслуживается поставщиком ПО в облаке, а заказчик осуществляет доступ к ПО через веб интерфейс. В последнии годы модель SaaS стала популярной среди поставщиков ПО, существующие компании предаставляют свои продукты как SaaS решение, а также появляются новые компании, специализирующиеся на SaaS с момента основания. Модель SaaS была реализована для различных предприятий и государственных служб. Одним из примеров успешного внедрения SaaS являются сервисы электронного правительства в Эстонии. Однако модель SaaS содержит проблемы, которых нет в традиционной модели. В литературе описано множество различных архитктур, фреймворков и практик, связанных с SaaS. Подобные исследования, как правило, описывают свойства архитектуры, которые считаются типичными для SaaS. Работа состоит из двух частей: первая часть – это обзор литературы, в котором определяются существенные свойства SaaS архитектуры основываясь на количестве статей, в которых упомянуты данные свойства. Вторая часть – исследование сервисов электронного правительства в Эстонии на предмет наличия в них свойств архитектуры, которые были определены в первой части. Сделаны выводы о списке существенных свойств литературы и их роли для SaaS.

# Table of Contents

## LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| B2B | Business-to-business |
| BaaS | Backend as a Service |
| BaaS* | Blockchain as a Service |
| CA | Certification Agency |
| CRM | Customer Relations Management |
| DaaS | Database as a Service |
| DB | DataBase |
| DBaaS | Database Backend as a Service |
| DBMS | DataBase Management System |
| DDoS | Distributed Denial of Service |
| EaaS | Enterprise resource planning as a Service |
| EBaaS | Etherium Blockchain as a Service |
| EHIS | Estonian Education Information System |
| ERP | Enterprise Resource Planning |
| HaaS | Hardware as a Service |
| HTML | Hypertext Markup Language |
| IaaS | Infrastructure as a Service |
| IDE | Integrated Development Environment |
| KPI | Key Performance Indicators |
| MaaS | Malware as a Service |
| MAI | Multiple Application Instance |
| MBaaS | Mobile Backend as a Service |
| MSLD | Multi-Tenant Secure Load Disseminated SaaS Architecture |
| MTA | Multi-Tenant Architecture |
| NaaS | Network as a Service |
| OS | Operating System |
| PaaS | Platform as a Service |
| PIC | Personal Identification Code |
| QoS | Quality of Service |
| RaaS | Robot as a Service |
| RAM | Random Access Memory |

| | |
|---|---|
| REST | REpresentational State Transfer |
| SaaS | Software as a Service |
| SAI | Single Application Instance |
| SAIMSI | Single Application Instance and Multiple Service Instances |
| SLA | Service Level Agreement |
| SMS | Short Message Service |
| SOA | Service Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SOCCA | Service-Oriented Cloud Computing Architecture |
| VM | Virtual Machine |
| WS | Web Service |
| WSDL | Web Service Description Language |
| XML | Extensible Markup Language |

# 1 INTRODUCTION

## 1.1 Background

Software as a Service (SaaS) is a software deployment and licensing model in which software is running on the cloud and is usually accessible for users through a web interface. National Institute of Standards and Technology of USA defines SaaS as cloud service model in which service consumer uses providers applications that run on a cloud infrastructure [1]. It is one of the three main principles of cloud computing, alongside Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

One of the first and best-known companies that provide enterprise SaaS solutions is Salesforce [2]. The model of the company is Business-to-Business (B2B), it was founded in 1999 and specialized in Customer Relations Management (CRM) in the cloud. The solution was horizontal and targeted general business processes. Now Salesforce has a wide range of products for sales, marketing and other core business components. Its revenue in 2017 was 8.4 billion USD.

In 2016 cloud service market segment of SaaS, PaaS and IaaS combined was estimated by Gartner at approximately 71.0 billion USD (33.9% of total cloud service market) with SaaS having the largest share of about 38.6 billion USD [3]. In the forecast [3] SaaS market is expected to reach 75.7 billion USD by 2020. In a survey among companies [4] it was shown that SaaS adoption continues to grow with 73% of respondent organizations saying that 80% or more of their applications would be SaaS applications by 2020. Adoption among medium and enterprise sized businesses is lower than among smaller businesses, but, according to forecast, in future it would grow faster, which will allow medium and enterprise sized businesses to outrun smaller businesses in SaaS adoption by 2020. In 2016 SaaS Industry Market Report [5] it was revealed that SaaS is also developing vertically with industry-specific solutions, which sometimes are referred to as Industry Clouds. The industries which use these solutions are healthcare, energy, real estate, transportation and others. This trend indicates that there is a demand for SaaS outside of generic cases.

With the rise in demand for SaaS solutions the number of software companies which provide SaaS solutions increased. There are both new companies which specialize in SaaS from establishment and companies which had employed other models and added SaaS solutions to its product line. The examples of new companies are GitHub (founded

2008) and Slack (founded 2013). Github provides hosting service for version control management for software developers with additional tools for issue-tracking, forking, statistics, documentation. It has enterprise solution that can be stored on the private cloud. Slack provides cloud tools like a messenger with channels and storage, which are aimed for work-related communications among teams. [6] Among traditional companies moving to SaaS are Microsoft, Oracle, Cisco. Microsoft introduced Office Online – set of online versions of its office applications. Oracle moved its line of business applications, including ERP, CRM, SCM, HR, and payroll, to the cloud. Cisco offers video conferencing and collaboration services [6].

At the same time, there are challenges in moving to SaaS for service providers. According to a survey [4], the three main criteria for choosing SaaS application for customers are cost, security, and ease of use. In SaaS architecture there are security issues that are not present in other architectures. It is the service provider's responsibility to maintain security. Multi-tenancy architecture increases the possibility of security issues and Service Level Agreement (SLA) violations. Since SaaS is built on top of PaaS and IaaS, it is susceptible to security problems from all three levels. There are Virtual Machine (VM) security problems on IaaS level, Software Oriented Architecture and API security problems on PaaS level, web application vulnerabilities on SaaS level [7]. Other common security issues which affect providers include network security, data integrity, availability, authentication and authorization, data confidentiality, and data breaches [8]. Providers also have to host and maintain hardware, which leads to additional costs. Hardware level is usually implemented as a data center. Resource allocation is another task that needs attention, taking into consideration SLA, scalability and Quality of Service (QoS). As resources are shared among tenants, overall cost depends on the efficiency of allocation algorithm. Resource allocation in SLA is a complex task which can be performed with different strategies and various algorithms [9].

To fulfill customer needs, it is important for SaaS application to have an architecture that would address the issues and implement desirable properties listed above. Various computer architectures are proposed in the literature, which can have common properties. By composing a list of the most common properties in SaaS architectures it is possible to define key properties of SaaS architecture.

## 1.2 Goals and Delimitations

The goal of the thesis is to define a list of important SaaS architecture properties and conduct a case study to analyze an existing service for the presence of these properties. In future, the defined properties can be used in the forward engineering of a SaaS application or in studies that further analyze SaaS architecture properties. The case study is conducted to prove the practical importance of the properties through identifying them in a functioning service and to evaluate the service from an architectural perspective.

The tasks are:

1) Construct a list of articles about SaaS architecture and related topics
2) Analyze the articles and gather information based on extracted fields
3) List properties sorted based on the number of articles they were mentioned in
4) Select a service for a case study
5) Conduct a case study of the service

Based on the goal the main research question is: ”What are the important properties of SaaS architecture?”. The sub-questions are: ”What are the most mentioned properties in the academic papers?” and ”What architectural properties can be identified in the service chosen for the case study?”.

Architectural properties in this context are characteristics, patterns or paradigms which are common for SaaS applications based on the literature. In the thesis, the focus is more on SaaS and PaaS levels and less on IaaS level. The hardware level is not examined. The properties are chosen based on the number of papers from the preselected list in which those properties are mentioned. Some of the properties are extracted from case studies of industry applications, which proves their legitimacy from a practical standpoint.

The limitations of the study are the following. The nature of the correlation between SaaS value propositions and architectural properties was not studied. The properties listed have not been developed through requirement engineering, so they are not requirements. The thesis is not a systematic literature review, so the main focus is on retrieved data, not review methodology.

## 1.3 Thesis Structure

The thesis is structured as follows: in Chapter 2 state of the art is presented, including a more detailed description of cloud computing and SaaS in general, literature

review description and results, retrieved architectural properties and their descriptions. Chapter 3 contains the case study, Chapter 4 contains conclusions and future research proposals.

# 2 STATE OF THE ART

## 2.1 SaaS as a Software delivery model

SaaS is a software delivery and licensing model, according to which software provider stores and maintains software on the cloud, and software consumer (end user) accesses it through the web interface. Software delivery (or software deployment) is a process of transporting software to a user. The result of software deployment is ready-to-use software on the user side. Software deployment can also be defined as processes between the acquisition and the execution of software [10]. Even though SaaS is considered a delivery model, there is no fixed list of delivery models. The motivation for it in literature is that deployment process is a complex process that is unique for each product [10]. The following stages of deployment lifecycle are identified: Release, Installation, Activation, Updating, Undeployment (deinstallation) [10]. Table 1 shows how those stages are handled in SaaS and on-premise models.

**Table 1.** Deployment stages description in SaaS [10][11]

| Deployment Stage | On-Premise Description | SaaS Description |
|---|---|---|
| **Release** | The software is packaged with information and artifacts which are sufficient for it to be installed | The software is uploaded to the server and is accessible through a web interface |
| **Installation** | All dependencies are pre-installed if packaged with software, files are copied to user's machine, registry records are written, environment variables are set. | The software is accessed the same way as a regular website |
| **Activation** | The activation usually requires an internet connection and consists of either signing into your account on the platform the software is connected to or entering a serial key | Activation consists of choosing a subscription plan. Usually, there is a free plan with certain limitations enabled by default |
| **Updating** | Depending on software and | The software is updated on the |

| | | |
|---|---|---|
| | Operating System (OS), software is either automatically updated through the internet or reinstalled | server, the user always receives the latest version when he accesses the web client |
| **Undeployment** | Everything that was done during installation is being reverted | No action required except canceling the subscription |

## 2.2 Everything as a Service (XaaS)

Before analyzing the current state of SaaS it is important to define it in the context of cloud computing. The term cloud computing refers to the practice of using shared remote computational power. The term cloud refers to hardware and software that allows users to access and utilize that power. In some cases, the cloud can be defined as hardware and software of a data center [12]. A cloud can be built on an individual server or PC, but it's not common in the industry and not optimal from a practical standpoint. Clouds can be divided into two categories: public and private. Public clouds are accessible to the general public through the internet and private clouds are covered by a firewall and usually used by enterprises. In some articles, this categorization is divided further. Two more subcategories are proposed [12][13]: hybrid cloud and community cloud. Hybrid cloud is a combination of a public cloud and a private cloud. Such a combination can benefit from advantages of both public and private availability models. It can be used for services that require various accessibility levels implementation. For example, in [13] private cloud is used as a proxy and allows to perform duplication checks with respect to the level of privilege for each user. A community cloud is a combination of cloud and Grid Computing, Digital Ecosystems, Green Computing [14].

When describing cloud in the context of SaaS architecture, there are two concepts related to SaaS which have similar meanings and are used widely - SaaS application and Cloud Infrastructure. Both refer to all the software that supplies the service to the end users, but cloud infrastructure may also include hardware level [12].

In terms of cloud computing, SaaS can be defined as applications and services provided by cloud computing. This definition considers the whole software layer as SaaS application and excludes only hardware layer. Many articles divide SaaS application into three different levels - IaaS, PaaS, and SaaS. Such categorization is usually referred to as Everything as a Service or XaaS [12][15].

There are also alternatives, like the following hierarchy: cloud provider, SaaS provider/Cloud User, SaaS user [12]. SaaS user is an end user. Cloud provider administers hardware and software required for giving cloud users access to computational power. SaaS provider administers the application itself, which is built on top of the cloud. In the paper division into more narrow categories is acknowledged, but it is claimed that the boundaries between different levels are blurred, and, therefore, it is hard to construct a robust terminology. For example, in some studies, it is stated that PaaS is aimed at developers rather than end users [15]. At the same time, developers are end users of PaaS, so PaaS can be considered SaaS that is aimed at developers.

In [15] XaaS is employed as a view of cloud infrastructure layered architecture. Three levels are defined - infrastructure, platform, and application (called SaaS in XaaS notation). The lowest level - infrastructure - encapsulates various hardware and computing resources for cloud users to run their Operating Systems (OS) and low-level software on. It is targeted for administrators. The middle layer - platform - encapsulates various software tools to create a development environment. Developers can focus more on writing code since they use provider's utilities, middleware and other parts of environment they need to manage themselves otherwise. Finally, the last layer - application - is the application itself. It is built on top of the previous layer and is aimed at the end user. This layer contains all business logic of the application that is delivered as a service. This categorization is employed further in the thesis.

Apart from three main layers which make up a SaaS application, XaaS includes Hardware as a Service (HaaS), Backend as a Service (BaaS) (sometimes also Database Backend as a Service (DBaaS) or Mobile Backend as a Service (MBaaS)), Database as a Service (DaaS), Robot as a Service (RaaS), Malware as a Service (MaaS), Network as a Service (NaaS), Blockchain as a Service (BaaS, but will be abbreviated as BaaS* is this thesis to avoid confusion with Backend as a Service).

HaaS provides access to hardware components over the web. It allows interconnection of physical systems, virtualization and hardware emulation [16]. RaaS combines SaaS, Internet of Things (IoT) and robotics [17]. RaaS application is a SaaS application that provides access to robots. Services may include manual robot control, writing programs for robots. DaaS application provides users with the ability to create, access, manage data in the database administered by a service provider. DaaS is aimed primarily for companies, it allows them to reduce costs by eliminating the need to buy

hardware and to pay specialists to manage database [18]. BaaS is aimed for mobile and web developers. It provides simple configurable backend capable of common tasks (user management, database, etc) [19]. An example of using BaaS can be a mobile application that displays constantly changing data. This data can be stored in BaaS so that application retrieves data from the backend. The key difference between DaaS and BaaS is that DaaS provides full access to the database and in BaaS database is usually wrapped with API, client framework, GUI and is not accessed directly. The emphasis is on ease of using it as a backend and performing common simple tasks (e.g. add, delete, edit entities). MaaS is a way to use cloud computing for Distributed Denial of Service (DDoS) Attacks. DDoS attacks usually require a lot of resources, but with MaaS a hacker can simply choose a target and attack parameters (duration, power) and the attack would be conducted by service provider's hardware and software [20]. NaaS solves such problems as load balancing, it is aimed at companies which need network infrastructure. The value proposition is reducing costs by eliminating network maintenance [21]. BaaS* [22] is a relatively new addition to XaaS family. With rising popularity of blockchain technology small companies which want to utilize it started to look for a solution that would allow them to integrate blockchain into their software without having to implement it from scratch. Popular use cases for blockchain are smart contracts, distributed applications, and custom tokens. Microsoft Azure, one of the biggest service providers, partnered with Ethereum to create Ethereum Blockchain as a Service (EBaaS* or ETH BaaS*) on Azure [23].

An example of a SaaS application is Salesforce [2]. Salesforce is a complex CRM solution for businesses. The service Salesforce provides is a variety of different cloud based programs, which allow to manage customers, sales, products and perform other business-related tasks. Some companies that were offering only on-premise solutions before, have started developing SaaS alternatives. For example, office suites which include text processors, spreadsheets and presentation programs. Microsoft Office was originally delivered to clients through a traditional model, now it is also available as a SaaS product [24]. A similar office suite was released by Google, called Google Docs. It is delivered as SaaS from the establishment and there is no on-premise version [25]. An example of PaaS is Heroku [26]. Heroku is a platform that allows developers to store and deploy web applications. Heroku supports various programming languages, including JavaScript, Go, Ruby, Python, Java. Heroku provides solutions to day-to-day development activities, like

storing and accessing logs, keeping the application alive, uploading, building and running application automatically and some other. Heroku. also addresses fundamental issues - mainly scalability and accessibility. Heroku is categorized as PaaS as this service allows developers to build applications and services rather than giving them full access to the server, as in IaaS. An example of IaaS is DigitalOcean [27]. DigitalOcean provides developers with access to scalable virtual machines. It also solves the problems of accessibility and scalability. It is considered as IaaS as it grants users access to full infrastructure. Unlike in Heroku, there are no limitations on programming languages or types of applications. The downside is that deploying, keeping alive, storing logs and other similar tasks are the responsibility of the user, not the service provider. The example of Backend as a Service is Back4app [28]. Back4app is based on Parse Server, it provides the ability to store data, send push notifications to users, perform analytics, run cloud code and other. There are SDKs for both Android and IOS, which help to connect a mobile application to Parse Server backend.

As it can be observed from analyzing the examples, different XaaS models are distinguished by access level required by tasks this model allows to perform. The higher the level, the more functionalities are "wrapped" by the business logic. From a software, architecture standpoint reviewed XaaS models are extensions or variations of SaaS. Additional system components introduce additional challenges, but core architectural objectives and issues are shared among all XaaS models. This point is reinforced by alternative categorization, which includes only SaaS layer. Further in the thesis mainly the term SaaS will be used.

## 2.3 Key considerations in SaaS

### 2.3.1 SaaS value propositions and success factors

To understand SaaS popularity, it is important to determine what are the success factors and value propositions of SaaS to both service providers and service consumers. This section is mainly based on [29].

Success factors are areas which are critical to the successful adoption of SaaS among both providers and consumers. Value propositions are aspects of the service which lead customers to choose it over the competing one. In the context of SaaS, they are properties which create the additional value of SaaS over traditional on-premise solutions. While value propositions focus on positive aspects, success factors are areas which need

additional attention, including risky areas. Value propositions are unique to SaaS by definition. Success factors can be shared between SaaS model and traditional model.

The success factors which were identified in the reviewed paper [29] and number of articles in which they were mentioned are presented in Fig. 1.

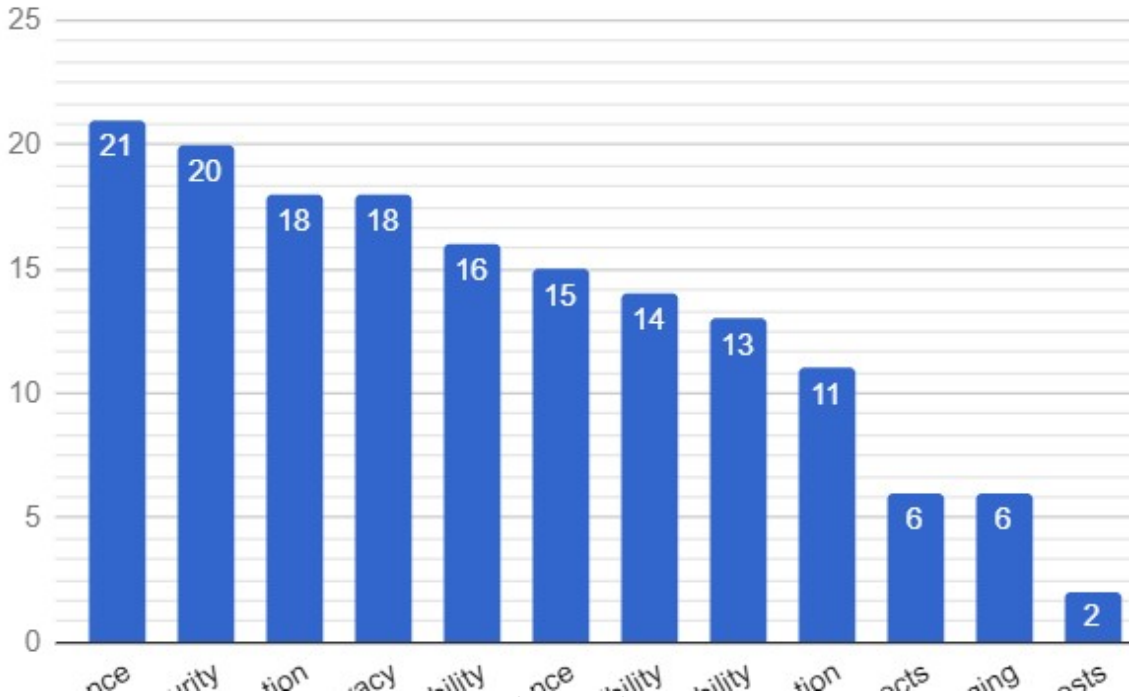


**Fig. 1.** Success factors of SaaS

*Performance* can be an issue since client-side code runs in the browser and, therefore, the application is essentially slower than the native application on the same platform. This factor is especially important on mobile platforms. Google Docs, for example, has a native application for mobile platforms, which means it's not pure SaaS on mobile devices. *Security* is a major concern in SaaS since this delivery model introduces more vulnerabilities. This includes DDoS attacks, data loss because of server failure, hackers gaining access to the servers and so on. *Individualization* (or *customization*) is a preprogrammed ability for the user to make changes to the application according to his/her individual needs. Since SaaS applications usually wrap some functionality and deliver it as a service, there is a risk to limit application customization options. *Privacy* is related to decentralization problem. Decentralization leads to privacy risks since the company has to trust the service provider with the data. Big companies prefer their data to be stored in an in-house database rather than anywhere else, as it prevents data decentralization.

*Availability* is critical since on-premise software is installed on customer's device, and, therefore, always available as long as the customer uses the device. SaaS is delivered through the web, so it has the same availability issues as traditional websites. Among less popular factors, the most relevant to this study is *flexibility*. Flexibility refers to flexibility in resources, meaning that each user of the service can obtain the optimal amount of storage and computational power. This is a unique SaaS property, as an amount of resources available to traditional on-premise software is determined by user's hardware it is installed on. In other studies, it is usually referred to as *Scalability*.

The value propositions which were identified in the reviewed [29] paper and the numbers of articles they were mentioned in are presented in Fig. 2.

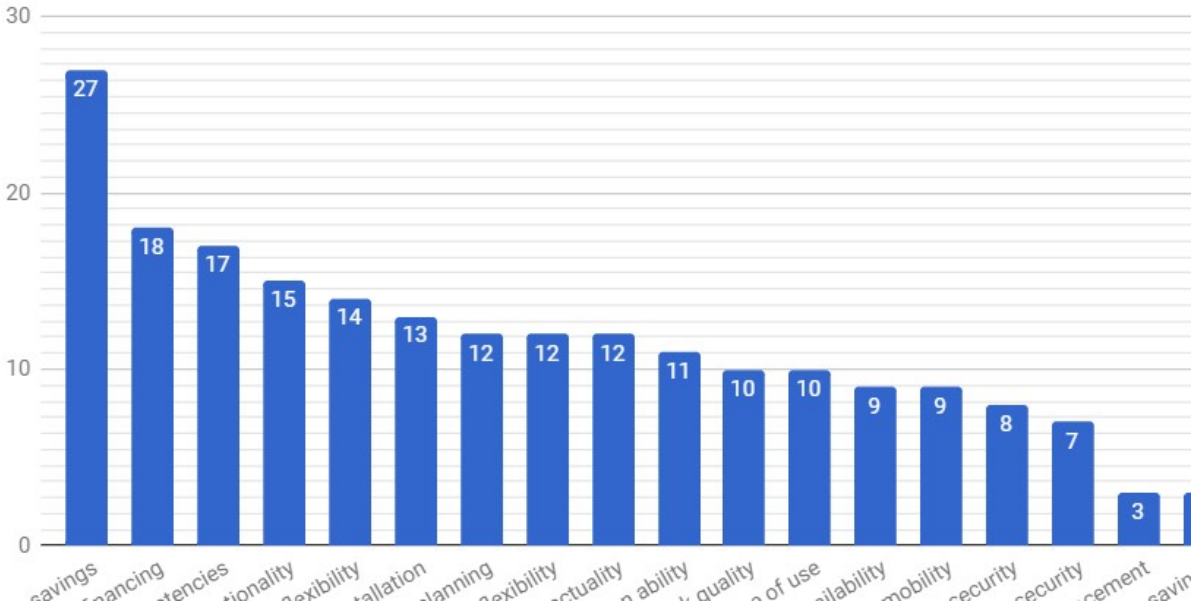


**Fig. 2**. Value propositions of SaaS

The first value proposition - C*ost savings* - was mentioned 1.5 times more than the second one. In an on-premise model, the user has to buy software licenses. If there is no subscription option, the license can be expensive. In SaaS model, there is a subscription or per-user payment options, and usually, a lot of different plans are offered, so that user can pick the most suitable one. It reduces software costs at early stages drastically. The user also does not need to buy hardware, which is especially important for small companies and startups. *Financing* is also related to SaaS payment model. It refers to initial financing of the project, which, as it was already described above, is much cheaper with SaaS model compared to on-premise. *Concentration on Core Competencies* means that a user does not

need to focus on IT. For example, a small company can save money on an IT department if a SaaS solution is employed. *Cost Flexibility* is another finance-related value proposition. It refers to user's ability to cancel the subscription and switch to another solution without paying for installation or uninstallation. *Installation* of a SaaS solution is always easier than an installation of an on-premise software, since SaaS is delivered through the web, and users can get instant access to it through browsers or mobile applications. In comparison, some on-premise solutions require specially trained IT personnel to install them. *Planning* is the fourth value proposition that is connected to financing. Users can plan their financial expenses better due to a subscription model. Strategic flexibility is achieved by the ability to cancel the subscription at any time. Canceling the subscription is usually instantaneous and does not introduce any additional costs. Actuality refers to the fact that users always have the latest version of the software since service providers manage updates of the software on servers.

Among less popular value propositions there is *Availability*, which is also a success factor. The advantage of SaaS is that a SaaS application is available on any device wherever there is an access to the internet. SaaS application is also estimated to have less downtime compared to on-premise software [11].

The main value proposition of SaaS appears to be lower costs compared to the traditional on-premise model. Many value propositions are possible because of subscription payment model.

### 2.3.2 SaaS model disadvantages

This section contains fundamenta internal disadvantages of SaaS model. Architectural issues are not considered, they would be analyzed in detail below.

- Decentralization. Many companies prefer to store data inside the company so that there is no trust factor involved in storing data.
- Security. SaaS architecture is more vulnerable to some security issues than traditional on-premise software. The fact that the data of several different tenants can be stored on one physical machine makes hardware vulnerabilities severe for cloud computing. For example, recently discovered meltdown attack [30] allows virtual machines to steal each other's data through a vulnerability in Intel processors. It means that an attacker running a malicious app on can steal all data stored in Random Access Memory (RAM) on the same hardware server regardless of virtualization delimitations. This might be a major concern for enterprise clients.

- Performance limitations. SaaS performance capabilities are affected by limitations of current web technologies, which means that SaaS model is not suitable for performance-sensitive applications, for example, computer games with advanced graphics.

### 2.3.3 Business Perspective

From a business perspective, SaaS is a representation of business agility trend. It can be viewed as a new wave of outsourcing [31] because software maintenance and updating are delegated to third parties. The key advantages for companies, which were identified in [32], were also mentioned in value propositions from 2.3.1: lower cost of entry, access to hardware resources without capital investments, scalability, new types of applications.

There are two main groups of stakeholders in traditional software model: software providers and software consumers. In cloud computing, four main groups are identified [32]: Consumers, Providers, Enablers, Regulators. Consumers (can be also referred to as subscribers or tenants) purchase the license to use the software. In the traditional model, they also own, maintain and update software. Providers own, maintain, update software, operate cloud computing systems, and set the price for cloud services. They have more responsibilities in SaaS model compared to the on-premise model. Enablers include parties which provide infrastructure, support SaaS or choose SaaS over traditional model. They are not providers or consumers, but they would benefit from SaaS success. Regulators are stakeholders which are responsible for regulating SaaS. One example is the government. SaaS model introduces new regulatory challenges, since the consumer and the provider can be in two different countries. In this case, it is not clear by which country's laws the whole process is regulated.

For utilizing the SaaS model successfully, providers must have an understanding of which applications are suitable for moving to SaaS and which are not. General-purpose applications like the ones that were already described above (Microsoft Office [24], Google Docs [25]) and cooperation applications (Google Docs [25]) proved to be suitable for SaaS model. More complex cases might require a case study. For corporate clients, core business process applications were the first to migrate to SaaS. Infrastructure management applications and development and integration tools also have the potential for SaaS migration [11].

In a B2B model, companies as software consumers have to create their business model taking into consideration particular qualities of SaaS model. Eight business model design choices can be defined in this case [31]:

1. Service characteristics
2. Value source
3. User target group
4. Data architecture configuration and tenancy model
5. Governance and demand/supply management core competencies
6. Cloud deployment model
7. Integration and provider strategy
8. Pricing structure

Another area that has SaaS adoption potential is enterprise solutions. Salesforce, which was already described above, specializes in enterprise solutions. One of the enterprise solutions types that were heavily affected by SaaS was Enterprise Resource Planning (ERP). ERP in IT industry is a software solution that manages core business processes. ERP as a Service's (EaaS) main value proposition is, again, lower costs. The main risks are security issues in management and data transfer [33].

### 2.3.4 Service Level Agreement

Service level agreement (SLA) is an agreement between service provider and service consumer on the terms of the service. SLA describes various aspects of a service such as quality of service, termination conditions, quality metrics, Key Performance Indicators (KPIs), Quality of Service (QoS). QoS describes measurable parameters which may serve as a representation of service's quality. There is a subtype of SLA for web services called Web Service (WS) Agreement. In [34] requirements for WS-Agreement are specified. According to them, WS Agreement must allow the use of any service term, allow the creation of agreements for existing and new services, allow the use of any condition specification language, provide symmetry of protocol, be composable with various negotiation models, be standalone, allow independent use of different parts of the specification. Such value propositions of SaaS as planning and strategic flexibility are possible mainly due to the versatile nature of SLA. The agreement can be implemented as a protocol or with smart contracts [35].

## 2.4 Key architectural properties

### 2.4.1 Review process

The goal of the literature review is to find important software architectural properties and problems of implementing them in different architectural styles. The importance is estimated based on the number of articles in which these properties are mentioned. Nine papers were chosen for the literature review. The papers were chosen based on relevance to the research question and initial contents analysis. Papers which describe SaaS architecture were chosen because architectural property implementation in a real architecture serves as a proof of the property's validity. Papers about SaaS maturity model were chosen because the maturity of the system is usually decided based on architectural properties. Papers which describe value propositions were chosen because if an architectural property has a correlated value proposition, that means the property has fundamental value and it directly influences SaaS success.

The extracted fields used for the review:

General extracted fields:

F1 - Title

F2 - Author(s)

F3 - Year

F4 - Abstract

F5 - Keywords

Extracted fields related to research questions:

F6 - Architectural properties

**Table 2.** Literature review extracted fields

| hor(s) | Year | Keywords | Architectural Properties | | | | |
|---|---|---|---|---|---|---|---|
| Tek TsaiEmail, XiaoYing Yu Huang | 2014 | software-as-a-service, SaaS architecture, customization, multi-tenancy architecture, redundancy and recovery, scalability | Customization | MTA (multi-tenancy architecture) | Scalability | Redundacy | |
| Joachim | 2011 | Service-oriented Architecture, SOA, literature review, literature search, adoption, governance, practices, impact, business value, research agenda, IS journals, IS conferences. | Customization | Modularity | Scalability | Interoperability | |
| astian Walther, Andreas nk, Torsten Eymann, Niraj gh, Gaurang Phadke | 2012 | Literature Review, Software as a Service, SaaS, Value Proposition, Success Factor, DeLone and McLean. | Customization | Virtualization | Scalability | | |
| shan Pervez, Sungyoung , Young-Koo Lee | 2010 | SaaS, Multi-Tenant, CloudComputing Architecture, ServiceOrientedArchitecture(SOA), ServiceLevelAgreement (SLA). | Customization | MTA (multi-tenancy architecture) | Scalability | | |
| tikainen, Gabriella, and Arto la. | 2014 | SaaS architecture, cloud, pricing, SaaS | Customization | MTA (multi-tenancy architecture) | Scalability | Virtualization | |
| -Tek Tsai, Xin Sun, Janaka asooriya | 2010 | Multi-tenacy, Service Oriented Computing, Cloud Computing | Customization | MTA (multi-tenancy architecture) | Scalability | Virtualization | |
| derick Chong and Gianpaolo raro | 2006 | Application Architecture, Software-as-a-Service (SaaS) | Customization | MTA (multi-tenancy architecture) | Scalability | | |
| ngseok Kang , Jaeseok | | | | | | | |

All extracted fields except F4 are represented in Table 2. Architectural properties that were mentioned more than once are color coded.

The short summaries of each reviewed article:

*Software-as-a-service (SaaS): perspectives and challenges* [*36*]

In the article three SaaS architecture properties which authors consider important (Multi-tenancy, Customization, Scalability) are defined and four SaaS architectures (Database-oriented, Middleware-oriented, PaaS-based, Service-Oriented) are described based on those properties. The three properties are further described. Customization is classified based on development levels, customization options (Fixed variation points and fixed options, Fixed variation points but allow tenant-supplied options, Allow tenants to create their own variation points and options, Intelligent customization, Customizable SaaS infrastructure, SaaS and PaaS configuration) are described. Tenant isolation design options for multi-tenancy database (DB) (each tenant has a DB with its own schema; Each tenant has his own DB, but all tenants use the same collection of schemas, possibly one schema for one application domain; each tenant has his own DB but all share the same schema; the DB is shared among tenants, but each tenant has his own schema; shared DB and schemas; each extension is a table; sparse columns for tenant information; hybrid solutions) issues are described, several design problems and solutions are stated, with the disclaimer that all conclusions are preliminary due to the lack of solutions tested in a real infrastructure. Two main scalability solutions are described, other common techniques are mentioned, scalability design principles are listed. Factors which affect scalability are described in detail. In addition to the main three properties, Recovery and Redundancy mechanisms are identified as a popular component of SaaS systems. Salesforce.com example and related technologies such as tenant awareness are described. The conclusion states that unique SaaS architecture requires special approaches for software engineering, databases and database management systems (DBMS's), requirement engineering and verification.

*A Literature Review of Research on Service-Oriented Architectures (SOA): Characteristics [37]*

In the article information from 40 articles about Service Orinted Architecture (SOA) is summarized and systematized. Among business related findings seven SOA architecture principles (Modularity/loose coupling, Implementation independence, (Open) standards, Service description, Interoperability, Platform independence, Service contract) and three IT benefits (Integration, Reuse, and Scalability) are identified. Scalability was

mentioned only in one article. Reuse and Integration were mentioned in several articles, they partially support Customization property, which was defined earlier. There is no architectural property that directly corresponds to the most popular benefit - Integration.

*Success Factors and Value Propositions of Software as a Service Providers [29]*

The article consists of theoretical background and definition of success factors and value propositions. The theoretical background contains definitions and brief descriptions of SaaS, a value proposition, and a success factor; the success model is used in the research. In the results section, 13 different success factors and 19 value propositions are covered. The most mentioned success factor and value proposition were, respectively, performance and cost saving. SaaS is categorized as a form of cloud computing. Individualisation is listed as a third most discussed critical success factor, flexibility is in seventh place. Individualisation corresponds to customization. The flexibility factor encapsulates scalability and virtualization. The two most popular factors - performance and security - are not specific to SaaS architecture, but require a special approach in a SaaS application. Section 2.3.1 of the thesis is based on this article.

*Multi-Tenant, Secure, Load Disseminated SaaS Architecture [38]*

The paper proposes a SaaS architecture called Multi-Tenant Secure Load Disseminated SaaS Architecture (MSLD). The five levels of SaaS maturity are described. MSLD is service-oriented, it consists of five services: Responder Service, Routing Service, Security Service, Logging Service, and Service Realization. The XML based protocol for communication between those services is proposed.

*SaaS architecture and pricing models [39]*

In the study dependency between SaaS architecture and pricing model is investigated. Case studies of five companies are conducted. A literature review is presented to give state of the art information on SaaS architectures and cloud maturity. Architectural properties of SaaS (scalability, customization, multi-tenant-efficiency) are listed. Advantages and disadvantages of multi-tenancy are described. Different types of SaaS maturity models are described. The conclusion on maturity is that universal maturity levels for SaaS application were not developed because each SaaS application can have different requirements which affect the final application architecture. Five respondent companies all have different levels of SaaS adoption, from no SaaS (on-premise applications) to SaaS with SOA. The conclusion is that pricing and SaaS architecture affect each other if the

company's value proposition is based on SaaS maturity, and, therefore, implemented architectural properties.

*Service-Oriented Cloud Computing Architecture [40]*

In the paper, a Service-Oriented Cloud Computing Architecture (SOCCA) is proposed. In the introduction, an overview of three main XaaS layers (IaaS, PaaS, SaaS) is given. The architecture is aimed to solve or mitigate the issues of current architectures listed in the paper. The issues are: users are often tied with one cloud provider, computing components are tightly coupled, lack of SLA support, lack of Multi-tenancy support. The architecture combines SOA and layered architectural styles. In the paper, multi-tenancy is described more than any other architectural property. SOCCA allows three different multi-tenancy patterns: Multiple Application Instance (MAI), Single Application Instance (SAI), and Single Application Instance and Multiple Service Instances (SAIMSI). SAIMSI is a new multi-tenant pattern presented in the paper.

*Architecture Strategies for Catching the Long Tail [41]*

The article provides definitions and description of SaaS. The three architectural properties of SaaS (customization, multi-tenancy architecture and scalability) and the SaaS maturity model are defined. The maturity model proposes that the most mature SaaS application has several instances and one load balancer, and, therefore, has multi-tenancy implemented. Four maturity levels have the following properties: 1 - none of the three properties; 2 - customization; 3 - customization, multi-tenancy; 4 - customization, multi-tenancy, scalability. The concept is proposed that the more mature the SaaS application, the less isolated tenant data is. A closer look at multi-tenant data model is given, architectural issues are listed.

*A General Maturity Model and Reference Architecture for SaaS Service [42]*

In the article key functions of successful SaaS are determined, SaaS maturity model and architecture are proposed. A case study of the main service vendors (Amazon, Salesforce, Microsoft, Google) is conducted. A table of common functions of SaaS is assembled, containing technical and business functions. The term technical functions, in this case, is the same as architectural properties. There are six technical functions in total, including three main properties from other papers (multi-tenancy, scalability, customization). A maturity model with four levels (ad hoc, standardization, integration, virtualization) is proposed. A scheme of the reference architecture is given.

*EasySaaS: A SaaS Development Framework [43]*

In the article, the framework for creating a SaaS application is proposed. Customization is addressed by adding a feature-request functionality to the application.

### 2.4.2 Extracted Architectural Properties

**Table 3.** Architectural properties and in which papers they were mentioned

| article \ property | Customization | MTA (multi-tenancy architecture) (efficiency) | Scalability | Redundancy | Virtualization |
|---|---|---|---|---|---|
| Software-as-a-service (SaaS): perspectives and challenges | ✓ | ✓ | ✓ | ✓ | ✗ |
| A Literature Review of Research on Service-Oriented Architectures (SOA): Characteristics | ✓ | ✗ | ✓ | ✗ | ✗ |
| Success Factors and Value Propositions of Software as a Service Providers | ✓ | ✗ | ✓ | ✗ | ✓ |
| Multi-tenant, secure, load disseminated SaaS architecture | ✓ | ✓ | ✓ | ✗ | ✓ |
| SaaS architecture and pricing models | ✓ | ✓ | ✓ | ✗ | ✗ |
| Service-Oriented Cloud Computing Architecture | ✓ | ✓ | ✓ | ✗ | ✓ |
| Architecture Strategies for Catching the Long Tail | ✓ | ✓ | ✓ | ✗ | ✗ |
| A General Maturity Model and Reference Architecture for SaaS Service | ✓ | ✓ | ✓ | ✓ | ✓ |
| EasySaaS: A SaaS Development Framework | ✓ | ✓ | ✓ | ✓ | ✗ |
| Total | 9 | 7 | 9 | 3 | 4 |

There are three main properties which were mentioned the most frequently: Customization - 9 articles, Scalability - 9 articles, MTA (multi-tenancy architecture) - 7 articles. The first paper, published in 2008, where the three properties were defined was

[41]. Many other reviewed studies reference that paper. Multi-tenancy has fewer mentions in reviewed articles, possibly because it has no tightly coupled value proposition nor a success factor. Scalability and Customization can be mentioned as success factors as well as architectural properties.

The properties are described in detail in the following sections (2.4.3 - 2.4.7).

### 2.4.3 Scalability

Scalability is an ability of a program to scale. A highly scalable program performs similarly on both small and big data sets. The term scalability is ambiguous in the context of SaaS and cloud computing. It may refer to scalability as a SaaS success factor [29], hardware scalability, data (database) scalability [33], architectural property (as it was in the literature review), and scalability in multi-tenancy [45].

A SaaS application is an alternative to multiple instances of on-premise applications. Multiple instances of one application are independent of each other, which means the number of users (number of instances) does not affect them in any way. To compete with on-premise applications on the market, SaaS application also has to perform the same regardless of the number of users, so it has to scale effectively, which means scalability is an essential property of a SaaS application.

Two main hardware scaling approaches are horizontal and vertical scaling, which can also be referred to as scaling out and scaling up respectively [44][41]. Vertical scaling, in this case, means moving an application to a server with more computational power, horizontal scaling means adding more servers. The vertical approach usually does not require any changes in the software. The horizontal approach may not be suitable for some architectures. For example, suppose we have an application which works with high read-write intensity data. If the data is stored in RAM and is dumped to the database on termination to achieve high performance, such application would not be horizontally scalable, because after user writes data, only the server that was serving that client would have it and other servers would not. At the same time, vertical scalability is applicable for that case.

Data or database scalability describes how database behavior changes with the increase of data amount. In SaaS application, one database can serve thousands of customers [41]. In that case query execution typically slows down. There are various methods to improve database performance, for example, indexing and partitioning [44].

There are several approaches to scaling out a database: master-slave replication, database clustering, and database sharding [44]. Outside of database in-memory storages and caching patterns can be used.

Scalability as an architectural property can be achieved by following these guidelines:

- Statelessness - any operation can be handled by any instance. The example of the stateful application was given in the previous paragraph. Statelessness is achieved by moving the state down the stack - usually to the database [44][41].
- Asynchronous I/O operations [41].
- Pools of resources. Resources can be threads, network or database connections. [41]
- Minimizing locking in database operations [41]

### 2.4.4 Multi-Tenancy

Multi-Tenancy is a software architectural property which indicates that some resources in the application are shared among multiple tenants [41][38]. Tenants, in that case, are users or groups of users who have control over the portion of resources provided by the application. Multi-tenancy can be achieved by either several instances or one instance of the application. Following these patterns, each tenant can have his own database or a share of a common database.

Multi-tenancy is tightly interconnected with the other two architectural properties - scalability and customization. The key challenges in multi-tenancy architecture are to provide data isolation, cost-efficiency, scalability and customization [41][45]. Isolation indicates that the data owned by one tenant cannot be accessed by another tenant and one tenant's customizations do not affect other tenants [42]. Customization in the context of multi-tenancy mostly refers to customization of the data schemas which can be stored for each user, more specifically database schemas [46]. Maturity models which were discussed earlier suggest that mature SaaS applications have some kind of a shared data schema, which leads to providing customization being a challenge. Scalability in multi-tenancy refers both to individual tenants being able to scale their applications if needed and to scale the whole system with the increasing number of tenants. For example, SaaS services usually have free and paid plans. Free plans offer fewer resources compared to paid plans. If a user moves to a paid plan from a free one, the application must grant him access to the higher amount of resources.

### 2.4.5 Customization

Customization as an architectural property in a SaaS application is a functionality that provides the user with an ability to make changes in certain parts of software through a specially designed interface [41]. Customization is aimed at addressing specific needs of each customer. The ad-hoc software can be created separately for each customer based on the customer's individual needs (this is relevant for corporate customers). To compete with this model SaaS applications also need a method to address those needs. Customization can be achieved in the following steps:

1. Identifying the application areas which may change according to customer's needs
2. Build an architecture that allows different options for these areas
3. Provide an interface for the user to manage the options for those areas

For example, PaaS Heroku has add-ons. This service allows users to choose an add-on that wraps some external service from a list of available add-ons and integrate it into the application. The service provider is responsible for integrating, managing add-ons, the user only chooses them through GUI. Add-ons include DaaS, monitoring and logging utilities, rendering utilities and more. An alternative to that model would be an on-premise server or IaaS, where the user can choose any external services, but he has to install, integrate, manage and update them manually (if they are not also provided via SaaS model). In [36] 6 possible customization levels are identified:

- Fixed variation points and fixed options - the example can be predefined layouts in online Integrated Development Environments (IDEs)
- Fixed variation points, which allow tenant-supplied options - a non-SaaS example is desktop wallpaper in an OS. A SaaS example is a themed mechanism in Telegram messenger [47]. Users in both cases can supply any item as long as it fits pre-defined requirements.
- Allow tenants to create their own variation points and options
- Intelligent customization - applications are created by a user from a list of modules
- Customizable SaaS infrastructure - in addition to the application, the user can customize SaaS
- SaaS and PaaS configuration - in addition to application and SaaS, the user can customize PaaS

### 2.4.6 Virtualization

A virtualization is an act and process of creating a virtual version of an object [42]. Virtualization allows achieving encapsulation and modulation in the system. In the context of SaaS, it is mainly used to divide resources between users (tenants). Also, the majority of IaaS providers, including DigitalOcean, employ virtual machines. In [42] virtualization is considered the highest level of SaaS maturity. In [40] virtualization is considered an approach to support multi-tenancy.

### 2.4.7 Redundancy

Redundancy is a replication of some elements of the system in order to increase its reliability [36]. Reliability is critical in SaaS. On SaaS level, redundancy can be achieved by having multiple load balancers and more servers than the minimum amount required to serve users. Databases can have replicas with duplicated data.

## 2.5 SaaS Maturity Models

SaaS maturity models define levels which describe SaaS application ability to provide SaaS value propositions and it's progress in success factors. In the reviewed papers [39][41][42] SaaS maturity models are based on the architectural properties. Each level of those maturity models describes an architecture that has zero or more SaaS architectural properties. The first level of a SaaS maturity model is usually Ad Hoc, which means software, that is made exclusively for each customer based on his requirements. On this level, Customization and Scalability are not considered as architectural properties but are achieved automatically due to the fact that each client receives his own application. In the articles in which SaaS maturity models are described, it is also proposed that SaaS architecture can be built incrementally based on those levels [41].

The maturity model proposed in [41] contains four levels with the following properties distribution: first - none of the three properties; second - customization; third - customization, multi-tenancy; fourth - customization, multi-tenancy, scalability. Forrester's model [39][48] contains six maturity levels, level one has none of the three properties, level two has multi-tenancy, levels three and four, in addition, have customization, levels five and six, in addition, have scalability. The maturity model proposed in [42] has four maturity levels like the one in [41]. The business aspect is considered as a service component. The first level is Ad Hoc. The second level is Standardization, on this level customization property appears. Level three, integration level, introduces multi-tenancy.

Finally, on the level four, virtualization level, virtualization property is added, customization is achieved through giving a set of functions to business process and scalability is achieved through a load balancer on top of a cloud architecture with several server instances. Full SOA is considered an architecture of the most mature SaaS.

## 2.6 Service Oriented Architecture

### 2.6.1 Overview

SOA is an architectural style in which an application consists of multiple services. This approach allows achieving loose coupling, implementation independence, decentralization and encapsulation [37]. SOA is easily scalable since components are loosely coupled and communicate through service protocol. The task of scaling an application is decomposed into scaling each of the services. A famous example of the service provider successfully employing SOA is Amazon. The applications created by different teams in Amazon communicate only through service interface [49]. Experts suggest that SOA improves flexibility for processes distributed over a large landscape of different systems and owners [50]. SOA is associated with Enterprise Service Bus (ESB) [37].

SOA has several disadvantages. The performance can be an issue, as the components interact with each other through the service protocol and all databases are wrapped with some kind of interface, and in some cases, the data has to go through the message bus [37]. In a smaller application, where performance is critical, it might be more suitable to have components in one application and querying a database directly. The development of SOA application is slower, especially on early stages, when the application is still small [50]. Error handling is more challenging because an error in one service has to be somehow handled in all the connected services [50]. Versioning is more complicated because each service has its own version [50].

### 2.6.2 Microservices

In the recent years, microservices and microservice architecture became popular amongst organizations [51]. The high-level definition of microservices is similar to the definition of SOA – an approach for developing an application as a set of small independently deployable modular services [50]. Microservices can be viewed as an approach for implementing SOA [50], which makes it one of the most popular SOA implementations approaches. However, microservices can also be defined as a separate

architectural style or ideal form of SOA or a refined subtype of SOA [50]. Advantages of microservices are scalability, availability, and isolation [51]. Microservices share the disadvantages which were listed above for SOA. In general, SOA is a broader term than microservices. One of the differences between microservices and SOA is that in microservices all the logic is at endpoints, while SOA is frequently considered to have a central part, for example, message bus [50][37].

There are two main approaches to implement SOA connecting different services - Simple Object Access Protocol (SOAP) and REpresentational State Transfer (REST) [52]. SOAP is an XML-based messaging protocol and REST is an architectural style of providing a web interface through a stateless application. Microservices are usually implemented with REST [50], while in practice SOA is associated more with SOAP [37].

## 2.7 Summary

The categorization that defines three levels of cloud infrastructure – IaaS, PaaS, and SaaS is employed in the thesis. The architectural properties which were found during the literature review and number of papers in which those properties were mentioned are presented in Table 4.

**Table 4.** Architectural properties and numbers of mentions

| Architectural Property | Number of papers |
|---|---|
| Customization | 9 |
| Scalability | 9 |
| MTA (multi-tenancy architecture) | 7 |
| Virtualization | 4 |
| Redundancy | 3 |

Customization, Scalability and Multi-tenancy were first defined as three main properties of SaaS architecture in [41] and then mentioned together in [36], [38], [39], [40], [42], [43]. In addition to being architectural properties, Customization and Scalability are considered success factors in SaaS [29]. Redundancy helps to achieve availability [36], which is also a value proposition and a success factor [29]. Virtualization supports MTA [38]. SaaS maturity models are tightly coupled with SaaS architectural properties. In the

maturity model defined in [41] and the Forrester's Maturity Model [39][48] maturity is determined based on which of the three main architectural properties (Customization, Scalability, and Multi-tenancy) the application has. In [42] the virtualization is considered an architectural property of the most mature SaaS. SOA is considered the most suitable architecture for a mature SaaS application [41]. The main architectural problem that was found during literature review is supporting customization and scalability in MTA [41][45].

# 3 Case Study

## 3.1 E-government

E-government aims to make citizens' interactions with government more efficient by providing online services which allow to solve the same problems and perform the same tasks which are carried out by traditional service centers now. In [*53*] it is proposed that four stages can be highlighted in e-government development: Cataloguing, Transaction, Vertical Integration, Horizontal Integration.

The first stage is called cataloging because the main activity of it is cataloging the government information. On this stage, the basic online presence of government is established. The resulting artifact is a read-only website (or multiple websites) that contains static cataloged government data. The examples might be either scans or text files of various form templates, executive orders, bills, legislation documents, laws, and other official papers.

At the second stage, an interactive website that serves as an interface to various government resources, such as databases is created. The example can be an online tax paying system. At this stage, citizens can make simple transactions with the database, that is why it can be called Transaction stage. The websites on this level have the core functionalities - retrieving and changing the data, but they cover only a small portion of government-related tasks.

The two remaining stages contain vertical and horizontal integration respectively. Vertical integration is an integration of different levels of government, such as district authorities, city authorities, country government. This implies that all levels interact and communicate with each other, have access to each other's databases or have the same

database. Horizontal integration is an integration of various government branches. The services which represent the branches are expected to communicate with each other if necessary. For example, if an application for a specific service, like gun licensing, requires no criminal record, the criminal record of a citizen can be checked automatically before providing him with an application form.

The most mature e-government consists of one service that can be used for any daily government-related task a country resident can have. Such a concept can also be viewed as the most mature SaaS according to [41] as it is a service with SOA architecture employed. Therefore, mature e-government can be used as an example of a complex SaaS application.

## 3.2 E-Estonia

### 3.2.1 Overview

E-Estonia is a movement to build a digital society in a European country Estonia. This section is based on E-Estonia website [54] and scientific articles which study the topic. Currently, Estonia is considered a world leader in e-government development and adoption. Currently 99% of state services are available online [54]. Estonia is a charter member of D5, the initiative to raise awareness of, develop and integrate electronic public services [55].

The electronic infrastructure of Estonia consists of multiple online services which represent different government structures and branches. The spheres represented by services are e-Identity, interoperability services, security and safety, healthcare, e-Governance, mobility services, business and finance, and education [54].

### 3.2.2 E-Identity

In Estonia legacy documents issued by the Soviet Union were first changed to new passports and then to ID cards with eID-functions. Every Estonian is assigned Personal Identification Code (PIC) [*56*].

PIC is used as a primary key in a citizens and residents table in the majority of government services databases. This allows easy communication between different services, which is an important factor of e-government success and an opportunity to establish services connection in SOA. The documents are issued by Certification Agency (CA).

The first installation of creating a digital id for online services in Estonia was Bank ID. Government e-services were accessible as external e-services from the internet bank. This method of authentication is still popular nowadays.

The next step was introducing new identity cards with authentication and electronic signature certificates. The card gives an access to various services and has additional applications, for example, it can be used for checking driver's license and as a public transportation ticket. The digital signature act allows users to sign various official documents online.

There is also a mobile identity service. Citizens need to install a specific SIM card provided by specific mobile operators into the phone.

The fact that the main means of identification is digital and most of the operations are done remotely without any personal contact allowed to create e-Residency [*54*]. E-Residency is a service that can grant anyone an electronic residency in Estonia. E-Residency does not give European Union residence permit nor does it grant an entry to the country, but it allows the holder to open a business in Estonia. There are several benefits of doing this, such as lower taxes, convenient online tax service, ability to move to another country without the need of reestablishing the company in a new country. This service brings a new source of income in Estonian budget from business owners which pay their taxes in Estonia.

In conclusion, e-Identity ties most of the other services together as it provides citizens and residents with a unique number and gives them the ability to sign documents with a digital signature, which can also be used in any service.

### 3.2.3 X-Road

According to the maturity model [53], in a mature e-government different services communicate with each other. Communicating in this context means an ability of a service to retrieve information from and write it to the databases of other government organizations. The three main tasks are to allow authorized services access the data, to maintain data consistency while transferring, and prevent third parties from getting the data being transferred.

X-Road is the solution to this problem that is used in e-Estonia services. X-Road is an exchange layer of the information system. It allows various services to interact with each other using SOAP protocol. X-Road is decentralized, which makes it highly scalable and more secure. Over 200 various services are connected to X-Road from over 900

different organizations, registers, and databases [57]. X-Road is also used in Finland since 2017.

The architecture of X-Road consists of servers which are both in the X-road center and in the users' organizations. Central server contains the register of all databases which use the system. There are two types of servers in client organizations: security servers and adapter servers. Adapter servers convert SOAP XML messages to database queries, mostly SQL, and back. Security servers are essentially firewalls. Services communicate with each other only through security servers. These servers use encrypted channels, which makes it impossible to steal the data while it is being transferred without having keys. Another function of security servers is logging all transactions, so the information which service requested, inserted or changed which data is available if needed [58].

The central server is situated in X-Road center. This center also contains the central servers of e-government and services description in Web Service Description Language (WSDL). WSDL is used for building automatic tools.

X-Road is a critical safety system since government services cannot function without it. There were incidents of cyber attacks already. In 2007 Estonia witnessed a series of DoS attacks [*55*]. Some of the state services were down as a result of the attack, but decentralized nature of X-Road allowed isolating the attackers.

SOAP is suitable for complex systems which require additional security layers [52], so it can be considered a suitable protocol for X-Road as opposed to REST.

The X-Road source code is available on GitHub [59]. The two main programming languages identified by automatic GitHub system in the repository are Java (67.1% of all lines) and Ruby (16.0%). Other languages have significantly smaller percentages (maximum 6.4%) and are used for frontend (Javascript and HTML) and supporting scripts (Shell, Python). A GitHub repository can have Issues. Issues are problems with the code or questions about the code posted by the repository owners or other users. Issues are disabled for X-Road repository.

### 3.2.4 E-Land Register and population registry

Population registry contains data about all Estonian citizens and residents. The data includes PID, name, date of birth, place of residence, nationality, native language, education, and profession. Residents can access the data and report mistakes if any.

E-land register contains data about immovable properties and related ownership. The data includes but is not limited to cadastral information, ownership relations, and mortgage information.

Both services are connected to X-Road. The information is always up-to-date. Examples of services which retrieve data from the population registry are transportation services, voting services, and school services. E-land register is used by other state services as well as private clients, who want to buy a new immovable property, and business clients, who want to confirm property ownership.

### 3.2.5 Sharemind

Sharemind is a data analytics platform for gathering and analyzing private data. Private data can be data that is classified by the government or is a trade secret of a company. Sharemind stores data so that it can analyze it, but third parties cannot read it, and, therefore, profit from it in any way. The areas of application are space, medicine, nuclear energy and more. Analyzing private data enables new possibilities in all of those fields.

Sharemind consists of the computational runtime environment and a library. It is an open source software. It allows building applications for processing private data from several parties using built-in protocols. New protocols can also be added as the source code is available. Encryption is implemented to ensure that hackers can not read the data during the protocol execution [*60*]. It is ensured that the parties involved can not read data of each other and Sharemind cannot read the data itself since the data remains encrypted while processing.

Sharemind is not a part of the e-government SOA architecture, but it is used for state purposes. It was used to collect tax and education records analytics data in 2015.

### 3.2.6 Security and safety

Security and safety branch of e-government includes e-law, e-court, and e-police. Apart from services it also contains KSI Blockchain - blockchain technology that was developed in Estonia and is used to ensure safety and integrity of state data. KSI Blockchain makes use of the fact that data stored in blockchain cannot be altered.

*E-law* is a service with a database that contains all Estonian laws. It has a web interface, so citizens can view the database information. The system employs blockchain technology to ensure that data is not altered. The service also allows users to follow council sessions online, which adds to the overall transparency of government operations.

*E-court* service is an attempt to automate judiciary. This is achieved with digital information system called Courtal developed by Estonian company Net Group. Through this service, users can post claims, submit evidence, answer questions, involve representatives, get verdict and more. It is accessible through the web after identity confirmation and also over a smartphone. Fines can also be paid automatically. Court supports several types of integration, including X-Road, SOAP, XML messages, web services. This allows it to retrieve information from both state services accessible only through X-Road (for example police databases) and external sources.

*E-Police* is a system which is aimed to provide high-level cooperation and communication tools to police officers. Police stations are receiving real-time positioning information about all police cars in the system. Cars are equipped with mobile workstations which allow accessing the information almost instantaneously. In 2006 in Louisiana, USA a convicted murderer Richard Lee McNair escaped from prison. He was stopped by a police officer near a prison, but officer let him go partially because of outdated, imprecise or wrong information about the escapee [*61*]. The quick access to all available information about suspects decreases the chance of making a mistake, which makes police more effective in general. The information was transferred by radio before the introduction of the service, so it required a dispatcher in the police office. Now dispatchers are not needed, therefore the service improves police efficiency as well.

### 3.2.7 Healthcare

Medical databases and ambulance management tools can be considered critical safety systems. If a doctor would not get important information about the patient in time or an ambulance would arrive late, it could lead to unfavorable consequences for the patient, including death. Therefore it is a matter of life and death to make those systems as effective and efficient as possible. Another medical-related area that can benefit from digitizing is prescriptions. It is common for drug dealers and drug addicts to use fake prescriptions in order to obtain controlled substances. For example, in Arizona USA officials are actively trying to fight fake prescriptions, but malefactors still find a way to fool the system. The main reason is that prescriptions which are issued on paper can be stolen or faked [62]. In e-Estonia, those issues are addressed with e-Health Records and e-Prescription services.

*E-health* records service contains a database with digitized health information. It employs KSI blockchain to ensure data integrity. This service allows doctors to access all

available necessary information about the patient. Patients can also access their own records as well as records of their children. Authentication is performed with e-Identity service. Statistics are gathered on service for spotting epidemics early on and tracking resources spending.

*E-prescriptions* is a centralized paperless digital system for issuing prescriptions. Medicine is issued to the resident after presenting an ID. Pharmacist access the prescriptions database to retrieve what medications were prescribed to the patient. Since the only way to get prescribed medications is by presenting the valid id, the system can be fooled only by hacking central database and change the data. It makes it almost impossible for malefactors to acquire the controlled substances.

### 3.2.8 E-governance

E-governance branch represents services directly related to governing the country and people who are responsible for it. It contains i-Voting, e-Cabinet, and State e-Services Portal.

Internet voting (I-Voting) is a process of voting through the Internet. Since the first German online election in 2001, there was an interest in the possibility for citizens to vote online without having to go to the polling stations. The demand for such systems grows as many countries experience a decreasing voter turnout, especially among younger voters. Internet voting is the most advanced type of voting in the topology that also includes Intranet poll site voting and kiosk voting. The concept has technical, political and legal issues, which include privacy, security, secrecy [63]. The voter needs to be identified. Since the process of voting is not supervised, it is harder to control forced voting. Estonian *i-Voting* service is a solution that addresses those challenges. It started in 1999 with an Internet- and web-based Election Information System. E-Identity and digital signatures are employed to identify voters. The forced voting issue is addressed by the fact that each voter can vote several times and each vote overwrites the previous one. This also mitigates the buying votes issue as it is very hard to control who the voter voted for since he can always vote again, which makes buying votes unreliable. A voter can later check with the mobile application if his vote reached the database and contains unmodified information. The secrecy addressed by the fact that when the votes are accessed by committee, all the voter identity information is erased from the votes. The votes are stored encrypted, and the decryption key is separated between the members of the voting committee so that they can access the votes only together. There is also an option to vote at a polling station, in that

case only paper vote will be counted. The company that developed I-voting, Cybernatica, also participated in the development of TIVI. TIVI is a solution that can be seamlessly integrated into an existing voting system. It features multiple authentication options and does not require e-Identity to function. TIVI can be deployed both on cloud and as on-premise [64].

State e-services portal *eesti.ee* [65] is an entry point to all other e-Services. After logging in to this portal the user stays logged in all the other e-Service available on the portal.

*E-cabinet* is an information system of government sessions which allow conducting government sessions in a paper-free manner. From a technical standpoint, e-cabinet consists of a decision-making tool, a scheduler, and a multi-user database. The system was introduced in 2000. At the time the hall where the sessions are held was equipped with stationary computers. As for now, ministers can use their mobile devices. Ministers receive all the relevant information and documents related to the topic that is currently being discussed in real time. Ministers vote using the same systems, if a decision has no objections, it is accepted automatically without debates, which saves time. According to e-Estonia website [54], e-cabinet allowed decreasing the average session time from four-five hours to 30-90 minutes.

### 3.2.9 Mobility services

*Mobility services* are a branch of e-government services that consists of public services which are aimed at increasing efficiency of transportation and transportation-related systems. They include intelligent transportation systems, mobile parking, and border queue management.

*ETC Estonia* is a network of intelligent transportation systems. It includes various transportation management systems. Harbour traffic-flow management system - Smart Port - provides various solutions to increase port capacity by reducing time spent on formal procedures. It contains on-line pre-registration and check-in, automated traffic management in the harbor and license plate recognition for passengers with vehicles. Incident handling platform allows emergency services to immediately access the data about vehicles which are involved in the incident. Ecofleet is a SaaS solution which helps organizations to manage vehicles and mobile employees. In future, the government plans to bring public transportation management to a new level by employing self-driven vehicles. It is legal to test self-driven vehicles in Estonia since March 2017.

*Border queue management* is a service that provides drivers with time slots for passing the border, which eliminates the need for physical queues. The service reduced waiting time on the border down to 30 minutes. The service was also implemented in Finland, Lithuania, and Russia after the initial success.

*Mobile parking* is a service that allows to pay for and manage parking with a mobile phone. The service provides several ways to pay for the parking. The first way is by sending SMS to a special number. SMS must contain the car number and the name of the parking area. To finish parking session driver has to call another phone number. The second way to pay is by using mobile application. The application has info about parking areas nearby, the parking session can be initiated and ended through the application interface. The application also allows the user to browse his parking sessions for the past six months. The third, most advanced way, is by connecting the car and the phone through Bluetooth. In that case, the session is initiated automatically when the car engine stops and it is stopped when the car starts moving again. Drivers also can reserve a parking spot within 20 to 30 minutes before parking. Parking zones are equipped with a complex of sensors which can check if there is a car on each spot. The system can notify officers about violations in case of a car being on the spot that was not paid for [66].

### 3.2.10 Business and finance

After separation from the Soviet Union, the government of Estonia wanted to create a favorable environment for entrepreneurship. One of the Estonian startups, a computer application for making calls on the internet, Skype, became extremely popular and was bought by Microsoft for 8.5 billion US dollars in 2011 [67]. The e-Estonia movement in the business sector is aimed to simplify and automate formal procedures in order to reduce bureaucracy to the bare minimum so that it would not get in the way of entrepreneurs who want to start or run new businesses. The simplified system can also attract businesses from other countries to pay taxes in Estonia through the e-Residence program. E-Estonia business and finance branch contains e-Tax, e-Banking, e-Business Register and Industry 4.0.

*E-Tax* is a service that allows paying taxes online. Paying taxes is a complicated process in some countries. For example, in USA tax paying is a high complexity process that requires hours of work; there are companies which specialize in tax preparation software and services [68]. With e-Tax, it takes approximately three minutes to file taxes online. All the information available from state databases is pre-filled in the form to save

user's time. It is made possible by e-Identity and X-Road technologies which were described above. With the help of this service, entrepreneurs can file their taxes from their country of residence. The website [69] has a separate sections for private and business clients, each tuned to serve individual needs of those two customer groups.

*E-banking* can be considered one of the frontiers of e-Estonia movement. One of the first ways to identify oneself online was through online bank account and it remains popular to this day. Banks are authentication service providers in X-Road, serving as an alternative to the ID card [*70*]. The banks started developing online banking services at the beginning of e-Estonia, which improved adoption of online services among the population in general. 99% of all transactions are made online, which saves a lot of time that would have been spent on visiting offices otherwise. The success of e-Banking in Estonia is driven by a combination of easy-to-use software, free-of-charge transactions, behavior changes, continuous and aggressive promotion in different media channels and in bank branches, using State-of-Art technologies, government efforts [71].

*E-business Register* is another service that makes it possible for foreigners who participate in e-Residency programme to manage a business in Estonia. It allows users to register a company, view and change data, file reports online, monitor processing data and record amendments, visualize relations between various companies and persons. It is not required for the owner by any procedure to be physically present in Estonia or to go to a specific department.

*Industry 4.0* is a concept of fully automated factories with integrated information systems which allow full control of the factory, data gathering, performance analytics and other useful functionalities. In the context of e-Estonia, industry 4.0 refers to the plans of adoption of this paradigm in future. Industry 4.0 is the last step in e-Estonia roadmap that is published on the website [54]. The key concept of Industry 4.0 is a Real-Time Factory, a factory that functions as one system and allows to measure Key Performance Indicators (KPIs) at real time. The Company SimFactory provides a complex solution that implements Real Time factory concept.

### 3.2.11 Education

Information technologies changed the education. New tools and practices were introduced and the goals of education have also changed. The information became more accessible, but it became harder to navigate through it. Modern children are familiar with gadgets from early years and educational system can view it as an opportunity to take

education practices to a new level. To achieve that goal it is important for the education system to keep up with the current technologies. E-Estonia educational solutions are e-School, DreamApply, Estonian Education Information System, and Eliis.

*E-School* is the service that digitizes all formal study-related processes. Teachers post homework on the portal, publish grades, evaluate behavior, and keep track of attendance. Parents can receive all this information directly from the portal, so they are always aware of the child's current situation at school. They also can write an explanation why their child was absent in school. Pupils can see homework, browse grades, and create an online portfolio. The portal can also be used for communication with school staff. As many other services of e-Estonia, e-School allows gathering statistics.

*Dreamapply* is a SaaS solution for admission and marketing management. It was launched in 2011 by the Estonian software company that has the same name. It is built mainly for universities which receive a lot of applications from foreign students. The platform is used in 20 countries.

*The Estonian Education Information System (EHIS)* is a service that is built around the main educational database. The database contains information about universities, students, lectures, teachers, graduation documents and other education-related information. The data gathering started in 2005. The service can be used by students, professors and other people who are involved in the education process for retrieving the data.

*Eliis* is an online service for organizing work of pre-schools and kindergartens. Government officials can retrieve statistics. Some of the functionalities of the platform: class journal, events calendar, surveys, statistical reports, document management, announcement board, communication platform. Similar to e-School, Eliis can be used by parents, teachers and government officials. Parents can contact teachers, browse pictures from events, read about children's activities. Teachers do journaling in the system.

## 3.3 Architecture analysis

### 3.3.1 General Overview

E-Estonia is a movement that involves many aspects of life. The solutions include both services, on-premise solutions, platforms, hardware solutions and more. Some of the solutions cannot be classified as e-Government (for example, Sharemind or DreamApply).

For this study, the e-Government part of e-Estonia is analyzed, excluding stand-alone services which are not connected to the main system. It would be referred to as e-

Estonia core further in the thesis. The core is considered to be services connected to X-Road. According to the information given above and the architectural schema, the core of e-Estonia combines SOA and Message Bus architectural styles [72]. In [37] Enterprise Service Bus (ESB) was mentioned as a popular technology that is often used to implement SOA. ESB is a variation of Message Bus style. According to the maturity model from [41], e-Estonia core can be considered the most mature SaaS.

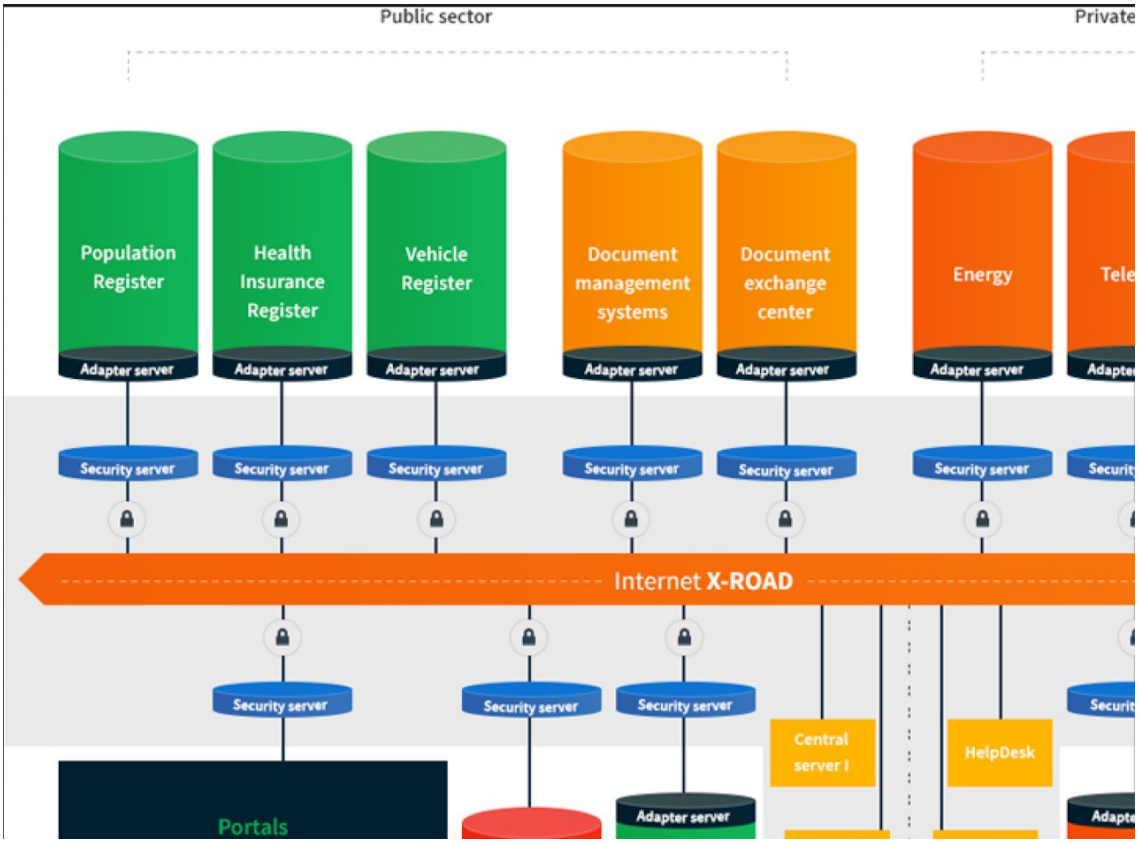


**Fig. 3.** The architecture of e-Estonia [73]

The following sections (3.3.2 - 3.3.6) contain E-Estonia analysis against architectural properties, which were retrieved from the literature review – Scalability, Customization, Multi-Tenancy (MTA), Virtualization, Redundancy. Two levels were taken into consideration in the analysis – e-Estonia as one service and individual services which make up e-Estonia core. Services which are not connected to X-Road, systems which include on-premise solutions, hardware and some other are not considered, they would require a separate case study.

### 3.3.2 Scalability

The population of Estonia is 1.3 million people and e-Estonia e-Residency is aimed to draw 10 million e-residents [54]. This raises the question of E-Estonia core scalability.

On the top level e-Estonia architecture – SOA – natively supports scalability [37]. Service bus X-Road allows connecting new services to the system easily. Decentralized nature of X-Road allows it to scale natively since the new services added to the system also bring their own security and adapter servers, therefore this load remains distributed despite the number of connected services [57]. The load on the centralized parts of X-Road (for example, X-Road certification service) still increases with the increase of the number of services.

On the individual service level, each service has to be analyzed separately. One possible bottleneck are e-Identity services since they are used by all the other services. The load on an individual service increases linearly depending on the number of users, the load on identity services increase can be roughly estimated as the number of users multiplied by the number of services, therefore e-Identity services experience the highest load in all the system. This problem is currently mitigated by the fact that there are different authentication options available. On the main portal of e-Estonia eesti.ee [65], there are options to authenticate with ID-card, mobile-ID or one of the six available banks (SEB, Swedbank, Danske pank, Nordea pank, Coop Pand, LHV pank). According to statistics from 2009 study, 80% of users were authenticating using bank services despite of more than one million issued identity cards [56].

The recent research proposes blockchain for e-Identity to solve potential legal issues [74]. As it was pointed out in the research itself, blockchain technology has known scalability problems. If blockchain technology would be implemented in e-Identity, scalability of this service would have to be reevaluated.

### 3.3.3 Customization

On the top level, the e-Estonia core is customizable because of SOA architecture and X-Road. SOA improves flexibility in complex systems [50], and new services can be added to X-Road if they match the requirements [70]. E-Estonia consists of services which were created by various different developers and providers. It is possible because of loose coupling in SOA architecture [37] – each service is encapsulated and can have any technologies as long as it is compatible with the general regulations.

On an individual service level, most of the e-Government services which are aimed for individual clients do not require customization higher than the lowest level from the classification that was described in [36] - fixed variation points and fixed options. There are several reasons for it. The first reason is that government procedures are usually highly standardized, and, therefore, do not allow variety. For example, paying taxes involves a standard form. The forms are different for businesses and individual taxpayers [75], but other than that each user has exactly the same form. In that case determining which type of taxpayer the user is, as shown in Fig. 2, and providing the form according to that information can be considered a fixed point of customization with two fixed options.

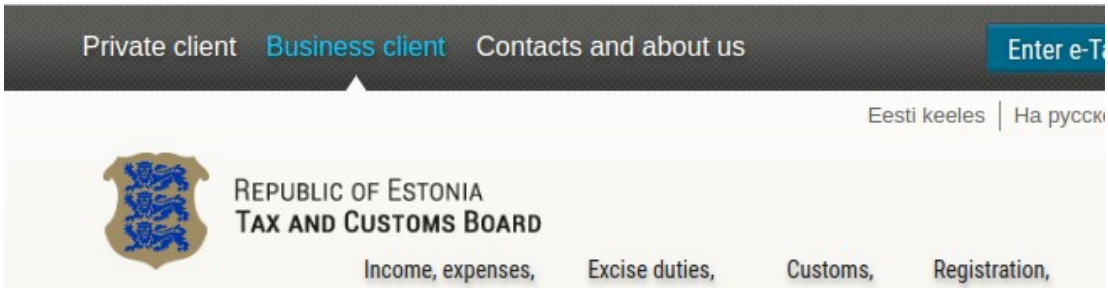


**Fig. 4.** The main menu of Tax and Customs Board of e-Estonia

The second reason is that in e-Government the tasks themselves are standardized. The need for customization comes from the individual needs of each client [29]. In e-government, the users need to perform standard procedures, which eliminates the need to customize the application for each user further than allowing each user to pick a task from a fixed list. The third reason is that e-Government does not include solutions with a level lower than SaaS (meaning PaaS or IaaS). According to classification [36], the lower the service level (SaaS, PaaS, IaaS), the higher the required level of customization. Having only SaaS application in system eliminates the need for a high level of customization.

### 3.3.4 Multi-Tenancy (MTA)

MTA can be considered only on a service level since this property describes resource sharing. In [56] it was mentioned that PIC is used as a primary key in several state databases, which means that user information is stored as database rows. It means that users share the schema and the database. This fact indicates that users do not share resources, they only access data, represented as rows in relational databases, which eliminates problems with customization and scalability in MTA.

Further analysis of this property would require access to the source code of the individual services.

### 3.3.5 Virtualization

Virtualization is a low-level property. It was not mentioned in reviewed sources.

### 3.3.6 Redundancy

It was established in the results of a literature review that redundancy supports availability. E-Government systems are critical safety systems, therefore availability is crucial. After the cyber attack in 2007 [76] the government of Estonia became especially concerned about cybersecurity. To ensure availability of e-services Estonia recently launched data embassies initiative [76]. Data embassies are data centers located in other countries which will duplicate data from e-Estonia databases and potentially application servers to ensure redundancy.

# 4 DISCUSSION AND CONCLUSIONS

## 4.1 Main findings

Table 5 contains the main information that was found about architectural properties both from the literature review and from the case study.

**Table 5.** The main findings

| Architectural property or architecture | Literature review findings | Case study findings |
|---|---|---|
| Scalability | • Value proposition [29]<br>• Indicates SaaS maturity [38][41]<br>• Has to be ensured on several levels [38][43] | • Is ensured by SOA architecture and X-Road on the top level |
| Customization | • Sucess factor [29]<br>• Indicates SaaS maturity [38][41]<br>• The lower the level of XaaS (SaaS, PaaS, IaaS), the higher level of customization required [36] | • Is ensured by SOA architecture and X-Road on the top level<br>• Only the lowest level of customization is required for the most services |
| MTA | • Indicates SaaS maturity [38][41]<br>• Supporting customization and scalability in MTA can be challenging [41][45] | • Users share the same schema and database [56] |
| Virtualization | • Can be considered a property on the most mature SaaS [42]<br>• An approach to support multi-tenancy [40] | • Was not found in the available information |
| Redundancy | • Ensures availability [36] | • There are plans to implement |

| | | |
|---|---|---|
| | | redundancy to ensure availability in case of cyber attacks and system failures [76] |
| SOA | • Supports customization and scalability [46]<br>• The architecture of the most mature SaaS [41] | • Is an architecture of e-Estonia core<br>• Supports customization and scalability |

SOA is shown to support scalability and customization. Redundancy is shown to ensure availability. According to the 4-stage model defined in [53], e-Estonia core is at the fourth stage, which means it is a mature e-Government service. It shows that SOA can be an architecture of a mature service. At the same time, customization and MTA, which were considered the properties which define SaaS maturity in the case study, are at the lowest levels possible according to classifications from [36]. Two possible conclusions can be drawn from that case:

1) The required level of SaaS application architectural maturity is determined by the requirements which depend on the specific purpose of each service. There is no strong correlation between service maturity and SaaS architecture maturity (the similar idea was proposed in [39])
2) The studied service is a degenerated SaaS, that fits the SaaS definition but lacks the main architectural properties

E-Estonia is a complex system that includes various solutions with various deployment models. The part that can be considered SaaS includes identity services, X-Road, and the state services which use identity services and are connected to X-Road. SOA contains service bus X-Road, which also ensures scalability and customization.

Table 6 contains the answers to the two sub-questions “What are the most mentioned properties in the academic papers?” and “What architectural properties can be identified in the service chosen for the case study?”.

**Table 6.** Architectural properties, numbers of mentions and results from a case study

| Architectural Property | Number of papers | Identified in the case study? |
|---|---|---|
| Customization | 9 | Yes |
| Scalability | 9 | Yes |
| MTA (multi-tenancy | 7 | Partially (see 3.3.4) |

| | | |
|---|---|---|
| architecture) | | |
| Virtualization | 4 | No |
| Redundancy | 3 | Yes |

It can be observed in Table 6 that the three main SaaS architecture properties which were first mentioned in [41] were shown to be important both by the literature review and by the case study. Visualization was mentioned in the papers but was not found in the case study, but there is a possibility that it might be identified if a source code analysis of individual services would be conducted. Redundancy was identified both in the literature review and in the case study. SOA is identified as an architecture of SaaS both in the literature review and in the case study.

The main research question is "*What are the important properties of SaaS architecture?*". Customization, Scalability and MTA were proven to be the three most important properties of a SaaS architecture. At the same time, case study results lead to the conclusion that SaaS maturity is not universal and it depends on requirements, the same concept was proposed in [39]. E-Estonia is a mature e-government according to the maturity model from [53], but according to SaaS maturity models [39][41][42] it is not mature, because the SaaS properties are degenerated. So, even though these properties are important, they do not necessarily need to be in every SaaS application for it to be considered mature. Redundancy was identified in both the literature review and the case study. Redundancy is an approach to achieve availability [36], which makes it more universal than the previous three properties, since availability is required for every service. At the same time redundancy is not required to achieve availability. Having availability in an architecture can be considered a good practice for SaaS application engineering. Virtualization was not identified in the case study since the information about architectures of services was insufficient, but the fact that Virtualization supports MTA reassures its importance.

## 4.2 Further Research

The important architectural properties were identified in the study. Architectural style SOA was proven to support two of the properties – customization and scalability, but other architectural styles were not analyzed. A study can be conducted to identify which

architectural styles are suitable for a SaaS application by analyzing which of the important architectural properties are compatible with which architectural styles and what are the possible problems of implementing architectural properties in different styles, like client-server [72]. It was identified that some of the architectural properties are related to value propositions and success factors, for example, scalability and customization are also success factors. Studying how each of the architectural properties is connected to success factors can help to evaluate fundamental value of the architectural properties. The problem of ensuring customization and scalability in MTA was identified. It can be researched what are the approaches of solving this problem. X-Road is available on GitHub. To check if X-Road has the desired properties, the code can be studied for vulnerabilities in the code and architectural properties and styles. The majority of e-Estonia services are not open-source. To identify architectural properties on lower level, a full open source system can be analyzed for those properties.

# REFERENCES


1. The NIST Definition Of Cloud Computing. 2011. Ebook. National Institute of Standards and Technology's (NIST) [Accessed 19 Feb. 2018]. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf.
2. "CRM Software & Cloud Computing Solutions - Salesforce UK". 2018. Salesforce.Com. https://www.salesforce.com/ [Accessed 19 Feb. 2018].
3. "Gartner Says Worldwide Public Cloud Services Market To Grow 18 Percent In 2017". 2017. Gartner.Com. https://www.gartner.com/newsroom/id/3616417 [Accessed 13 Feb. 2018].
4. Politis, David. 2017. "The 2017 State Of The Saas-Powered Workplace Report". Bettercloud Monitor. https://www.bettercloud.com/monitor/state-of-the-saas-powered-workplace-report/. [Accessed 13 Feb. 2018].
5. Lambert, Sebastian. 2016. "2016 SaaS Industry Market Report: Key Global Trends & Growth Forecasts – Financesonline.Com". Financesonline.Com. https://financesonline.com/2016-saas-industry-market-report-key-global-trends-growth-forecasts/ [Accessed 13 Feb. 2018].
6. "50 Leading Saas Companies - Datamation". 2018. Datamation.Com. https://www.datamation.com/cloud-computing/50-leading-saas-companies. [Accessed 27 Feb. 2018].
7. Almorsy, Mohamed, John Grundy, and Ingo Müller. 2016. An analysis of the cloud computing security problem. arXiv preprint arXiv:1609.01107.
8. Subashini, Subashini, and Veeraruna Kavitha. 2011. A survey on security issues in service delivery models of cloud computing. Journal of network and computer applications, 34(1), pp.1-11.
9. Wu, Linlin, Saurabh Kumar Garg, and Rajkumar Buyya. 2011. SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments. In Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (pp. 195-204). IEEE Computer Society.

10. Dearle, Alan. 2007. "Software Deployment, Past, Present And Future". Future Of Software Engineering (FOSE '07). doi:10.1109/fose.2007.20.
11. Dubey, Abhijit, and Dilip Wagle. 2007. Delivering software as a service. The McKinsey Quarterly, 6(2007), p.2007.
12. Armbrust, Michael, Ion Stoica, Matei Zaharia, Armando Fox, Rean Griffith, Anthony D. Joseph, and Randy Katz et al. 2010. "A View Of Cloud Computing". Communications Of The ACM 53 (4): 50. doi:10.1145/1721654.1721672.
13. Li, Jin, Yan Kit Li, Xiaofeng Chen, Patrick P.C. Lee, and Wenjing Lou. 2015. "A Hybrid Cloud Approach For Secure Authorized Deduplication". IEEE Transactions On Parallel And Distributed Systems 26 (5): 1206-1216. doi:10.1109/tpds.2014.2318320.
14. Briscoe, Gerard and Marinos, Alexandros 2009. Community cloud computing. In: First International Conference on Cloud Computing, 1-4 December 2009, Beijing, China.
15. Pallis, George. 2010. "Cloud Computing: The New Frontier Of Internet Computing". IEEE Internet Computing 14 (5): 70-73. doi:10.1109/mic.2010.113.
16. Stanik, Alexander, Matthias Hovestadt, and Odej Kao. 2012. "Hardware As A Service (Haas): Physical And Virtual Hardware On Demand". 4Th IEEE International Conference On Cloud Computing Technology And Science Proceedings. doi:10.1109/cloudcom.2012.6427579.
17. Merle, Philippe, Christophe Gourdin, and Nathalie Mitton. 2017. "Mobile Cloud Robotics As A Service With Occiware". 2017 IEEE International Congress On Internet Of Things (ICIOT). doi:10.1109/ieee.iciot.2017.15.
18. Hacigumus, H., B. Iyer, and S. Mehrotra. 2018. "Providing Database As A Service". Proceedings 18Th International Conference On Data Engineering. Accessed May 11. doi:10.1109/icde.2002.994695.
19. Lane, Kin. 2015. "Overview of the backend as a service (BaaS) space." API Evangelist.
20. Yan, Qiao, and F. Richard Yu. 2015. "Distributed Denial Of Service Attacks In Software-Defined Networking With Cloud Computing". IEEE Communications Magazine 53 (4): 52-59. doi:10.1109/mcom.2015.7081075.

21. Costa, Paolo, Matteo Migliavacca, Peter R. Pietzuch, and Alexander L. Wolf. 2012. "NaaS: Network-as-a-Service in the Cloud." In Hot-ICE.
22. Samaniego, Mayra, and Ralph Deters. 2016. "Blockchain As A Service For Iot". 2016 IEEE International Conference On Internet Of Things (Ithings) And IEEE Green Computing And Communications (Greencom) And IEEE Cyber, Physical And Social Computing (Cpscom) And IEEE Smart Data (Smartdata). doi:10.1109/ithings-greencom-cpscom-smartdata.2016.102.
23. Gray, Marley. 2015. "Ethereum Blockchain As A Service Now On Azure". Azure.Microsoft.Com. https://azure.microsoft.com/en-us/blog/ethereum-blockchain-as-a-service-now-on-azure/ [Accessed 22 Mar. 2018].
24. "Free Microsoft Office Online, Word, Excel, Powerpoint". 2018. Products.Office.Com. https://products.office.com/en/office-online/documents-spreadsheets-presentations-office-online [Accessed 23 Mar. 2018].
25. "Google Docs - Create And Edit Documents Online, For Free.". 2018. Google.Com. https://www.google.com/docs/about/ [Accessed 23 Mar. 2018].
26. "Cloud Application Platform | Heroku". 2018. Heroku.Com. https://www.heroku.com/ [Accessed 23 Mar. 2018].
27. "Digitalocean: Cloud Computing, Simplicity At Scale". 2018. Digitalocean. https://www.digitalocean.com/ [Accessed 23 Mar. 2018].
28. Back4App Inc. 2018. "Parse Server Platform - Backend Made Simple | Back4app". Back4app. https://www.back4app.com/.
29. Walther, Sebastian, Plank, Andreas, Eymann, Torsten, Singh, Niraj, and Phadke, Gaurang. 2012. "Success Factors and Value Propositions of Software as a Service Providers – A Literature Review and Classification" AMCIS 2012 Proceedings. 1. http://aisel.aisnet.org/amcis2012/proceedings/EnterpriseSystems/1 [Accessed 23 Mar. 2018].
30. Lipp, Moritz, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. 2018. "Meltdown." arXiv preprint arXiv:1801.01207.
31. Joha, Anton. 2012. Design Choices Underlying the Software as a Service (SaaS) Business Model from the User Perspective: Exploring the Fourth Wave of Outsourcing. Journal Of Universal Computer Science, 18(11), pp. 1501-1522.

32. Marston, Sean, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. 2011. "Cloud Computing — The Business Perspective". Decision Support Systems 51 (1): 176-189. doi:10.1016/j.dss.2010.12.006.
33. Lenart, Anna. 2011. "ERP In The Cloud – Benefits And Challenges". Research In Systems Analysis And Design: Models And Methods, 39-50. doi:10.1007/978-3-642-25676-9_4.
34. Andrieux, A & Czajkowski, K & Dan, Asit & Keahey, K & Ludwig, Hans & Pruyne, Jim & Rofrano, John & Tuecke, Steven & Xu, Mantao. 2004. Web services agreement specification (WS-Agreement). Global Grid Forum. 2.
35. Emanuele Di Pascale, Jasmina McMenamy, Irene Macaluso. 2017. "Smart Contract SLAs for Dense Small-Cell-as-a-Service". http://arxiv.org/abs/1703.04502 arXiv:1703.04502 [Accessed 28 Mar. 2018].
36. Tsai, WeiTek, XiaoYing Bai, and Yu Huang. 2014. "Software-As-A-Service (Saas): Perspectives And Challenges". *Science China Information Sciences* 57 (5): 1-15. doi:10.1007/s11432-013-5050-z.
37. Joachim, Nils. 2011. "A Literature Review of Research on Service-Oriented Architectures (SOA): Characteristics, Adoption Determinants, Governance Mechanisms, and Business Impact." In AMCIS.
38. Pervez, Zeeshan, Sungyoung Lee, and Young-Koo Lee. 2010. "Multi-tenant, secure, load disseminated SaaS architecture." In Advanced Communication Technology (ICACT), 2010 The 12th International Conference on, vol. 1, pp. 214-219. IEEE.
39. Laatikainen, Gabriella, and Arto Ojala. 2014. "Saas Architecture And Pricing Models". 2014 IEEE International Conference On Services Computing. doi:10.1109/scc.2014.84.
40. Tsai, Wei-Tek, Xin Sun, and Janaka Balasooriya. 2010. "Service-Oriented Cloud Computing Architecture". 2010 Seventh International Conference On Information Technology: New Generations. doi:10.1109/itng.2010.214.
41. Chong, Frederick, and Gianpaolo Carraro. 2006. "Architecture strategies for catching the long tail." MSDN Library, Microsoft Corporation, 9-10.
42. Kang, Seungseok, Jaeseok Myung, Jongheum Yeon, Seong-wook Ha, Taehyung Cho, Ji-man Chung, and Sang-goo Lee. 2010. "A General Maturity Model And

Reference Architecture For Saas Service". Database Systems For Advanced Applications, 337-346. doi:10.1007/978-3-642-12098-5_28.

43. Tsai, Wei-Tek, Yu Huang, and Qihong Shao. 2011. "Easysaas: A Saas Development Framework". 2011 IEEE International Conference On Service-Oriented Computing And Applications (SOCA). doi:10.1109/soca.2011.6166262.
44. Seovic, Aleksandar. 2010. Oracle Coherence 3.5. Birmingham: Packt Pub, 17-24.
45. Guo, Chang Jie, Wei Sun, Ying Huang, Zhi Hu Wang, and Bo Gao. 2007. "A Framework For Native Multi-Tenancy Application Development And Management". The 9Th IEEE International Conference On E-Commerce Technology And The 4Th IEEE International Conference On Enterprise Computing, E-Commerce And E-Services (CEC-EEE 2007). doi:10.1109/cec-eee.2007.4.
46. Bo Gao, Chang Jie, Guo Zhi, Hu Wang, Wen Hao An, Wei Sun. 2009. "Develop and Deploy Multi-Tenant Web-delivered Solutions using IBM middleware: Part 3: Resource sharing, isolation and customization in the single instance multi-tenant application". https://www.ibm.com/developerworks/library/ws-multitenant/ws-multitenant-pdf.pdf [Accessed 9 Apr. 2018].
47. "Telegram – A New Era Of Messaging". 2018. Telegram. https://telegram.org/ [Accessed 9 Apr. 2018].
48. "Saas Maturity Model According To Forrester". 2018. *Architects Rule!*. https://blogs.msdn.microsoft.com/architectsrule/2008/08/18/saas-maturity-model-according-to-forrester/ [Accessed 9 Apr. 2018].
49. "The Secret To Amazons Success Internal Apis". 2012. Apievangelist.Com. https://apievangelist.com/2012/01/12/the-secret-to-amazons-success-internal-apis/ [Accessed 12 Apr. 2018].
50. Pautasso, Cesare, Olaf Zimmermann, Mike Amundsen, James Lewis, and Nicolai Josuttis. 2017. "Microservices In Practice, Part 1: Reality Check And Service Design". IEEE Software 34 (1): 91-98. doi:10.1109/ms.2017.24.
51. Kwan, Anthony, Hans-Arno Jacobsen, Allen Chan, and Suzette Samoojh. 2018. "Microservices In The Modern Software World". Dl.Acm.Org. http://dl.acm.org/citation.cfm?id=3049915.

52. Tihomirovs, Juris, and Jānis Grabis. 2016. "Comparison Of SOAP And REST Based Web Services Using Software Evaluation Metrics". Information Technology And Management Science19 (1). doi:10.1515/itms-2016-0017.
53. Layne, Karen, and Jungwoo Lee. 2001. "Developing Fully Functional E-Government: A Four Stage Model". Government Information Quarterly 18 (2): 122-136. doi:10.1016/s0740-624x(01)00066-1.
54. "E-Estonia — We Have Built A Digital Society And So Can You". 2018. E-Estonia. https://e-estonia.com/ [Accessed 15 Apr. 2018].
55. Anthes, Gary. 2015. "Estonia: A model for e-government". Communications Of The ACM 58 (6): 18-20. doi:10.1145/2754951.
56. Martens, Tarvi. 2010. "Electronic Identity Management In Estonia Between Market And State Governance". Identity In The Information Society 3 (1): 213-233. doi:10.1007/s12394-010-0044-0.
57. "X-Road — Cybernetica AS". 2018. Cyber.Ee. https://cyber.ee/en/e-government/x-road. [Accessed 16 Apr. 2018]
58. Kalja, A., A. Reitsakas, and N. Saard. 2018. "Egovernment In Estonia: Best Practices". A Unifying Discipline For Melting The Boundaries Technology Management: doi:10.1109/picmet.2005.1509730. [Accessed 16 Apr. 2018]
59. "Ria-Ee/X-Road". 2018. Github. https://github.com/ria-ee/X-Road. [Accessed 16 Apr. 2018]
60. Bogdanov, Dan, Sven Laur, and Jan Willemson. 2008. "Sharemind: A Framework For Fast Privacy-Preserving Computations". Computer Security - ESORICS 2008, 192-206. doi:10.1007/978-3-540-88313-5_13.
61. Christopher, Byron. 2018. "Was Carl Bordelon Railroaded?". Byron Christopher. https://byronchristopher.org/2015/01/12/was-carl-bordelon-railroaded/. [Accessed 17 Apr. 2018]
62. GABRIEL News. 2018. "Officials Try To Stop Fake Prescriptions, But Addicts Remain Persistent". Pinalcentral.Com. https://www.pinalcentral.com/casa_grande_dispatch/area_news/officials-try-to-stop-fake-prescriptions-but-addicts-remain-persistent/article_5df7ebe0-4569-5fb7-bb86-38fb3a410030.html. [Accessed 18 Apr. 2018]

63. Kersting, Norber, and Harald Baldersheim. 2004. "Electronic Voting And Democracy", 97-109. doi:10.1057/9780230523531.
64. "TIVI Powered By Smartmatic And Cybernetica - Tivi.Io". 2018. Tivi.Io. https://tivi.io/ [Accessed 16 Apr. 2018].
65. "Eesti Riigi Infoportaal | Eesti.Ee". 2018. Eesti.Ee. https://www.eesti.ee/et/index.html. [Accessed 16 Apr. 2018].
66. Manni, U. 2010. "Smart Sensing And Time Of Arrival Based Location Detection In Parking Management Services". 2010 12Th Biennial Baltic Electronics Conference. doi:10.1109/bec.2010.5629891.
67. De Pommereau, Isabelle. 2018. "Skype's Journey From Tiny Estonian Start-Up To $8.5 Billion Microsoft Buy". The Christian Science Monitor. https://www.csmonitor.com/World/Europe/2011/0511/Skype-s-journey-from-tiny-Estonian-start-up-to-8.5-billion-Microsoft-buy https://tivi.io/ [Accessed 17 Apr. 2018].
68. "Why Doing Your Taxes Is Much Harder Than It Ought To Be". 2018. Washington Post. https://www.washingtonpost.com/news/wonk/wp/2013/04/15/why-doing-your-taxes-is-much-harder-than-it-ought-to-be/?noredirect=on&utm_term=.6e577354099b [Accessed 17 Apr. 2018].
69. "Äriklient | Maksu- Ja Tolliamet". 2018. Emta.Ee. https://www.emta.ee.
70. "X-Road Regulations". 2006. Confluence.Csc [Accessed 18 Apr. 2018]. https://confluence.csc.fi/download/attachments/37226092/X-Road_regulations.pdf?version=1&modificationDate=1392038668191&api=v2 [Accessed 18 Apr. 2018].
71. Mia, Mohammad, Mohammad Rahman, and Md Uddin. 2007. "E-Banking: Evolution, Status and Prospect.".
72. "Chapter 3: Architectural Patterns And Styles". 2018. Msdn.Microsoft.Com. https://msdn.microsoft.com/en-us/library/ee658117.aspx. [Accessed 19 Apr. 2018].
73. X-ROAD FACTSHEET. 2018. Ebook. https://www.ria.ee/public/x_tee/X-road-factsheet-2014.pdf [Accessed 19 Apr. 2018].
74. Sullivan, Clare, and Eric Burger. 2017. "E-Residency And Blockchain". Computer Law & Security Review 33 (4): 470-481. doi:10.1016/j.clsr.2017.03.016.

75. "Republic of Estonia Tax and Customs Board". 2018. emta.ee. https://www.emta.ee. [Accessed 12 May 2018].
76. Robinson, Nick, and Keith Martin. 2017. "Distributed Denial Of Government: The Estonian Data Embassy Initiative". Network Security 2017 (9): 13-16. doi:10.1016/s1353-4858(17)30114-9.