Tampereen yliopisto

Turo Hyppönen

HOW TO BUILD AN EVENT-DRIVEN CAPABILITY

Master of Science Thesis
Faculty of Engineering and Natural Sciences
May 2020

TIIVISTELMÄ

Turo Hyppönen: How to build an event-driven capability
Diplomityö
Tampereen yliopisto
Johtamisen ja tietotekniikan DI-tutkinto-ohjelma
Tarkastajat: Prof. Marko Seppänen ja apul.prof. Kari Systä
Toukokuu 2020

Suomalaisen teleoperaattorin IT-yksikön kokonaisarkkitehtuuriosasto on keskusteluissa eri osapuolten sekä markkinaseurannan myötä havainnut vanhan paradigman uudelleennousun. Koetaan, että yrityksen järjestelmät ovat riippuvaisia toisistaan mikä aiheuttaa haasteita muun muassa järjestelmien kehityksessä sekä käytönaikana. Kun jotakin järjestelmää huolletaan tai järjestelmässä on vika, heijastuvat nämä vaikutukset myös muihin järjestelmiin. Tämä johtaa esimerkiksi menetettyyn myyntiin, koska asiakkaita ei pystytä palvelemaan. Ratkaisua haluttiin lähteä tarkastelemaan tapahtumapohjaisen arkkitehtuurin kautta.

Tapahtuma on merkittävä tilanmuutos ja tapahtumaa voidaan tarkastella useassa eri kontekstissa, kuten esimerkiksi systeemisuunnittelun tai ohjelmistokehityksen. Tapahtumat voidaan luokitella useampaan eri kategoriaan. Tässä työssä luokitellaan tapahtumat järjestelmien sisäisiin järjestelmätapahtumiin sekä liiketoimintatapahtumiin. Järjestelmä-tapahtumilla käsitetään tapahtumia, jotka ovat sidoksissa ohjelmiston sisäiseen logiikkaan kuten esimerkiksi sisäinen loki-tapahtuma, joka kertoo mitä järjestelmässä on tapahtunut. Liiketoiminta-tapahtumalla käsitetään esimerkiksi tapahtumaa, joka syntyy käyttäjän tehdessä tilauksen verkkokauppaan. Tilaus on tässä tapauksessa tarkasteltava tapahtuma.

Tapahtumapohjainen arkkitehtuuri on tapa tuottaa ohjelmistoja sekä systeemejä, jotka ovat löyhästi kytköksissä toisiinsa täten mahdollistaen toisistaan riippumattoman kehittämisen. Lisäksi mahdollistuu systeemi missä järjestelmien viat eivät heijastu toisiin järjestelmiin. Tapahtumapohjaisuus mahdollistaa lisäksi tietojen välittämisen useampaan paikkaan samanaikaisesti sekä luo edellytyksiä tietojen nopeammalle tiedon prosessoinnille.

Työssä selvitetään, mitä on tapahtumapohjainen arkkitehtuuri ja miten suunnitella tapahtumapohjainen arkkitehtuuri. Tämän lisäksi selvitetään, minkälaisia haasteita ja mahdollisuuksia yrityksellä on tapahtumapohjaisen arkkitehtuurin käyttöönottoon.

Tapahtumapohjaisen arkkitehtuurin määritelmään hyödynnetään kirjallisuustutkielmaa. Tapahtumapohjaisen arkkitehtuurin suunnitteluun hyödynnetään sekä kirjallisuustutkielmaa sekä luodaan kirjallisuuden pohjalta kyvykkyysviitekehys, jonka avulla luodaan tiekartta tapahtumapohjaiseen arkkitehtuuriin. Yrityksen haasteita ja mahdollisuuksia kartoitetaan haastatteluiden avulla. Haastattelun tuloksia sovelletaan kyvykkyysviitekehykseen, jolloin saadaan yritykselle räätälöity tiekartta tapahtumapohjaisuuteen.

Diplomityön tuloksena luotiin kyvykkyyspohjainen strateginen tiekartta tapahtumapohjaiseen arkkitehtuuriin. Tiekartta toimii ohjenuorena sekä viitekehyksenä IT organisaatiolle kehityksen yhteydessä. Tiekarttaa varten tehtiin kirjallisuustutkielma tapahtumapohjaiseen arkkitehtuuriin sekä haastateltiin yrityksen asiantuntijoita.

Avainsanat: tapahtumapohjainen arkkitehtuuri, kyvykkyys

ABSTRACT

Turo Hyppönen: How to build an event-driven architecture
Master of Science Thesis
Tampere University
Faculty of Engineering and Natural Sciences
Examiners: Prof. Marko Seppänen and Assoc. Prof. Kari Systä
May 2020

During the discussions with different internal stakeholders and from market review, Telecommunications operator's architecture team of IT department has concluded that there is a need to focus on old paradigm. Some of the operator's systems are coupled with each other that causes issues during system development and during normal operations. If a system is malfunctioning or has a planned service break, it might have impacts to other systems and cause loss of sales since customers cannot be served. There was a wish to view the issue through event-driven architecture.

An event is a significant change of state and the event can be viewed from system design point of view or computer science point of view among others. In this thesis, the events are limited to system and business events. A system event occurs inside of the software component, for instance, an internal log event. A business event reflects to real life events such as customer's order to eCommerce platform where the order is the event of the interest.

Event-driven architecture is a way to create systems or software that are highly decoupled thus enabling parallel development cycles. In addition, it enables a system where malfunctioning piece of system does not affect to other systems. Event-driven architecture enables also faster data transfer and a way to create new information by combining events.

The goals of this thesis were to find out what is event-driven architecture, how to design and implement event-driven architecture through business cases and what kind of obstacles and possibilities a company may have. Literature review was used to define the meaning of the event-driven architecture. For design and implementing the event-driven architecture, both literature review and capability-based framework were used. The framework helps to create a roadmap for event-driven architecture for the company. The obstacles and possibilities are charted via expert interviews on the company's personnel. The results of the interviews are applied to the capability-based and a tailored roadmap for the company was created.

As summary, the result of the thesis is a roadmap for event-driven architecture. The roadmap was achieved by using expert interviews and capability-based framework that IT organization can use while engaging in development activities.

Keywords: event-driven architecture, capability

ALKUSANAT

The thesis ideation began in autumn of 2018. The actual implementation of the thesis took approximately a year and half. I would like to thank my colleagues, my superior, DNA and my supervisors.

I also would like to thank my family for understanding and making this possible.

Helsinki, 5.5.2020

Turo Hyppönen

TABLE OF CONTENTS

1. INTRODUCTION

The thesis was done for Finnish telecommunications operator DNA and for its enterprise architecture department. During discussions with different stakeholders, the enterprise architecture team has come to conclusion that one of the critical needs in the future is enhancing data transfer speed to enable faster reaction for different business events. Business should be able to react faster to changes on competition at operational level such as price changes and different campaigns. Currently most of the data transfer happens batches in predetermined intervals which means that data is not up to date on systems that consumes the data. This leads to bad user experience since user cannot rely on timeliness of the information.

In addition, some parts of the architecture are coupled so there is a need for patterns for decoupling in order to prevent droppage of service level due breakage of some service. This has led to loss of sales in some situations.

1.1 Goals of thesis

Supply chain management is essentially managing processes which have inputs and outputs. Basically, input and outputs are events such as order, stock addition, stock reducing and so on. Streaming process and event-driven architecture are tools that enable to process a vast number of events simultaneously. Event-driven architecture enables efficient data transfer between services and systems. It enables highly decoupled architecture where the components are not depended on each other and parallel data processing is streamlined.

This thesis tries to understand how to implement such type of architecture through business cases and how to transform part of organization to utilize event-driven architecture. Main questions of the thesis are

1) what event-driven architecture is?
2) how to design and implement event-driven architecture through business cases
3) what kind of obstacles and possibilities, both technological and organizational, a company may have while implementing event-driven architecture?

1.2 Schedule and implementation

The thesis consists a literature review, interview and roadmap how to build event driven capabilities. The literature review gathers information how to design event-driven architecture and what kind of enablers others have found. The review emphasises case studies on event driven, big data, complex event processing and other related architectures where the amount of the data is vast and the requirement for decision making is fast.

The interviews consist of two business cases and analysis of the interviews. For the interviewees are given some information about event-driven architecture and the business cases. The questions of the interviews reflect the characteristics of event-driven architecture and with the questions is tried to understand what kind of enablers event-driven architecture requires in organization, both technical and organizational.

The theoretical background information and the business cases are prepared in spring of the year 2019. The interviews are held in the winter. the analysis of the interviews and literature review is done on autumn and the finalisation on spring of 2020.

1.3 Structure of thesis

The thesis explorers first the theory behind IT architecture and event-driven architecture to give the foundation for the actual business cases. Before the actual business cases are introduced, is given background information for the business drivers what event-driven architecture tries to solve. The business cases describe the issues to be solved and proposes solutions with event driven context. After introduction of the business cases interviews captures the opinions and views on the solution.

2. INTRODUCTION TO EVENT-DRIVEN ARCHITECTURE

2.1 Introduction

The events happen everywhere, and they can be categorized as events in business environment, operational, innovation and enterprise change (Cummins, 2009). Business environment events originate from customer actions such as change in credit score might affect customer ability to buy. This affects to companies will to sell. Other events in business environment relate to supply chain, economy, competitors, political, social, nature and technical. Events in supply chain such as changes in stock or availability might affect to resell prices.

Events can also be defined as an object or records in database. The records in database represent an activity that has happened or is thought to be happen. In this context events are tied to the IT systems and what happens inside the systems. These definitions correlate with each other and events in IT systems are usually representation of real-world events (Luckham, 2011).

In the middle of the decade of 2010 businesses have made attempts to move faster. In Nordic countries this phenomenon has been called as digitalization. This leads that the organizations have been seeking for holistic transformation. This transformation concerns also data transition: it must evolve from patch driven mode to more real time mode (Nandico, 2016).

The business environment is full of events and the business must be able to recognize and capture the meaningful events. After capturing the business must analyse the information and turn the information to knowledge and to wisdom. Event capturing follows data pyramid which describes the relationships between data, information, knowledge and wisdom (Figure 1).

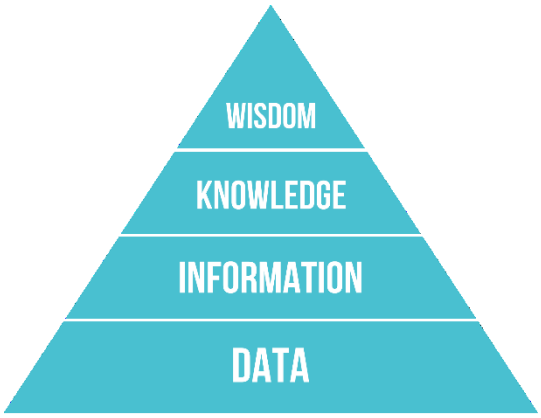


Figure 1 Data pyramid

Telecommunications industries have had long time access to vast amounts of user data. The big data initiatives don't bring profits if the data mastering tools are not in place and they don't support the scale of big data. The initiatives require also capable personnel and right tools for analysts (Bughin, 2016).

Events can be divided to at least two categories depending on the source. One division in the IT architecture context is to divide them to events that happen between systems and in business events such as order of new phone from the ecommerce site (Michelson, 2011).

Other sources divides events to types such as business, application, infrastructure, internal or exchange type. Business layer represents events that originates from processes such as orders. Application layer events are generalisations from business events and infrastructure events are low-level events such as information from network and its devices. Internal events are tied to certain product and its internal proprietary event type and formats while exchange event types focus how to transfer data between systems (Becker, Matzner, Müller, & Walter, 2011).

Event-driven architecture can be viewed via other theories as well such as computer science, information systems science or designs science. Computer science emphasizes how to the software is built, in system design point of view how information systems are built. Event-driven architecture enables simultaneity in supply chain management theory by providing same information same time to different processes. Design science and management science are also interlinked in sense that how organization can exploit event-driven architecture to achieve its business goals. (Overbeek, Janssen, & van Bommel, 2012)

Since events can be used in different context in this thesis is used term business events when referring to events that are in context of management or design science. Term

system event is used when referring to events that happen between systems, are in context of computer, information systems science, are application or infrastructure layer.

2.2 Enterprise architecture

IT architecture is typically described as models that helps to manage the change and understand the complexity of enterprise (Ross, Weill, & Robertson, 2006). IT architecture is the foundation for the execution of enterprises. Architecture is about communicating the vision of a complex system to be to systems stakeholders. While creating a system there is usually multiple stakeholders with various backgrounds. For instance, there is system owner, subcontractors, planners, designers and implementers who everybody has different kind of view for the system and they might have their own lingo. Architecture deals with both technical issues and social issues such as which kind of roles or personas should use a system. (Bente, Bombosch, & Langade, Collaborative Enterprise Architecture, 2012)

Many industries and companies have described their own framework for managing the architecture. NATO has NATO architecture framework (NAF), British Ministry of Defence has their own Architecture Framework (MODAF) and there is more commercial oriented framework such The Open Group Architecture Framework (TOGAF). Some enterprise architecture frameworks are criticized of being too vague and not providing any measurable benefit. Frameworks doesn't necessary tell how the model should be implemented thus leading various results (Kotusev, 2018).

Enterprise architecture frameworks however have drawn criticism for their complexity and deep roots in IT. Frameworks offer comprehensives guides for transformation but since the software development cycle has the role of the frameworks is different ((Goerziga & Bauernhansla, 2017)

IT Architecture in enterprise context is strategic tool for leading, improving and developing processes and systems. Thus, architecture is not entirely software development but rather taking a holistic and systemic view on organization and its parts.

Handling architecture in big corporates has coined a term Enterprise architecture which is sometimes perceived negatively due to approach that is perceived too academic and brings only disappointments (Bente, Bombosch, & Langade, Collaborative Enterprise architecture, 2012).

Agile software development process Agile software development emphasizes independent teams that are capable to do their own decisions and autonomously. One of the core value of agile software development is capability to react changes. Agile software

development focuses more on continuous flow of work instead of doing projects. However, this doesn't exclude the projects tool to manage the work, but mindset is shifted. In this context enterprise architecture can create conflicts if the implementation of the architecture is too strict and focuses how to get most of the existing tools and processes. On the other hand, too autonomous teams can lead to sub-optimal results in enterprise level such as duplication of effort, lack of governance and lack of innovation. From the enterprise architecture point of view, agile software development can seem as emergent architecture that is out of control. Enterprise architecture is associated with waterfall type of projects (Figure 2) while agile software development emphasized iterative development (Figure 3).

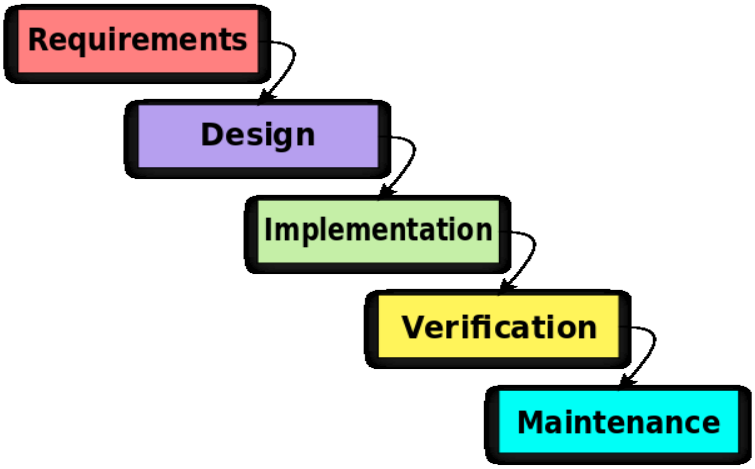


Figure 2 Waterfall project (Smith P. , 2009)

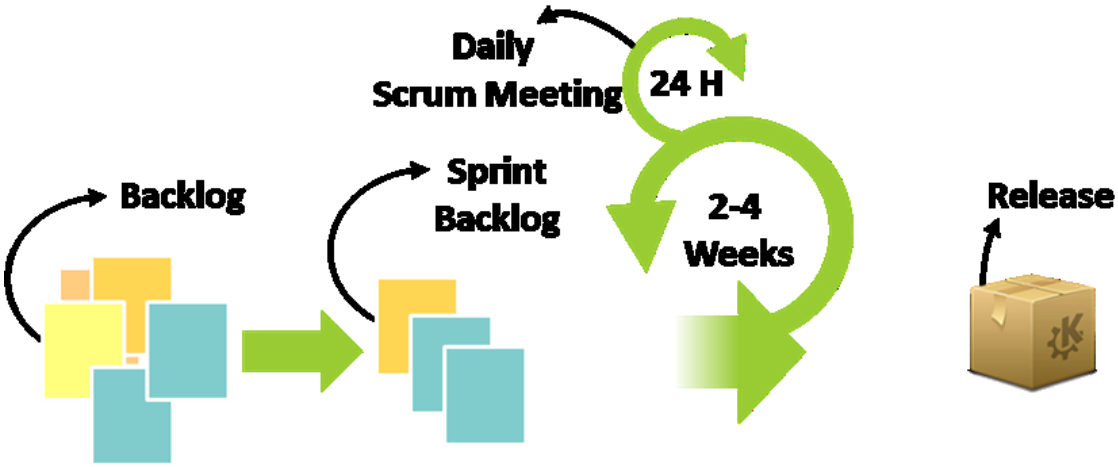


Figure 3 Scrum, one of the agile software development process's

Agile software development and enterprise architecture complete each other. Agile software provides fast results for business problems whereas enterprise architecture pro-

vides longer vision for the enterprises. Architecture means essentially the same with enterprise architecture and agile architecture, but the mentality is different. To achieve the best from the both, one should be pragmatic: pick the best tools and methodologies that suits for the situation (Bente, Bombosch, & Langade, Collaborative Enterprise Architecture, 2012).

2.3 Enterprise application integration

One of the most critical functions of IT organization is to integrate systems and organizations parts with each other in order to achieve better efficiency. In general, there is a concept called Enterprise application integration (EAI) that is architectural pattern that describes how systems can communicate with each other. EAI doesn't exactly define how to implement communication but rather defines core capabilities of such tools for instance mediation, messaging process and service orchestration. The basic role of the EAI is to bring two or more applications or systems together (Dahl, 2002).

In the beginning of the century EAI was divided on four different levels: user interface, method application interface and data. In data level integration, data is transferred between data storages and application code is intact. For example, applications database tables or views are queried directly with external program that transfers data to some other database such as datawarehouse. The advantage of this method that it doesn't require testing of application but the on the other hand data level integration requires knowledge on data and its purpose. Application interface level refers to data and process type of integration that once was done between packaged applications such SAP and Lotus. Although the interfaces were very different from application to application this method has proven successful. Nowadays many SaaS type of software, platform and even companies provides API's to enable a deeper integration with other business applications (Figure 4).

Method level integration shares business functions that are used to for example book an appointment while user interface level practically means some sort of widget-based system. The widgets would be configurable and enables the integration to the background systems. The widgets should be reusable across platforms (Paulheim & Probst, 2010).

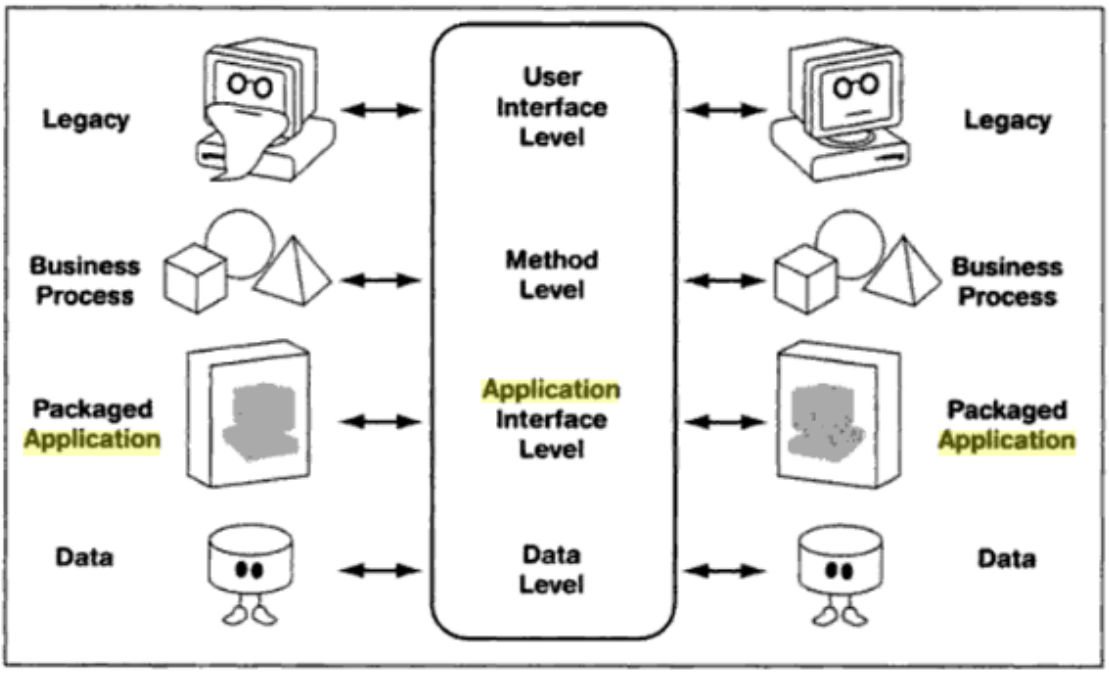


Figure 4 Types of EAI (Linthicum, 2000)

Basically, there are two ways for applications to communicate with each other. In synchronous mode application is waiting for the reply from another system before it proceeds to the next phase. In asynchronous mode application is not depend of arrival of message. This means that application can perform other operations before the reply is received whereas in synchronous mode application cannot perform other operations before the reply has arrived. Asynchronous implementation enables better user experience and scaling of the systems. The other way to describe system is request and event driven model. In broader context event driven model can be described as event-driven architecture.

2.4 Event-driven architecture

Event-driven architecture means a type of IT architecture where business event is captured by the IT system. An event can be for example an order in eCommerce site or when user views a web page. The event is change of state. Each business or system event can trigger processes in background systems. In case of the eCommerce order there could be an order processor that sends indication to warehouse system as system or business event to pick up the item for delivery. In another scenario where user is browsing a web site, there could happen analysis on user actions such as if the user is fraudulent or if the user is interested in a product. When user browses the web site, for example views different product pages or different images, each action user does produce an event, usually system event. However, the business events occur as well such

as ordering a product. The outcome of analysis of the system events could produce additional business events such as invoking a chat agent that could aid in purchasing.

Event-driven architecture has many related fields that overlap each other. Such terms as stream and log processing, real time analytics, big data, event and complex event processing are closely related to each other. The terminology varies a bit, but the basic principles seems to be the same: to be able to provide data and turn it into knowledge faster. Event-driven architecture emphasizes the transportation of data while big data focuses on data itself. Event-driven architecture can be divided to three mechanism:

1) Simple event processing in which notable event occurrence, such as eCommerce order is received, initiates action performed by different actors
2) Stream event processing broadcasts series of events to organization continuously.
3) Complex event processing combines different events and data sources to create new events (Overbeek, Janssen, & van Bommel, 2012).

Event-driven model can be described also as publish-subscribe pattern. Publisher is the source of the event and it is completely separated from the receiving end that is called subscriber. Designing the content of message must focus on characteristics of the event in business context. For example eCommerce order events content should describe information related to the order such as name of person who did the order, identification of product and the amount of the items just to name a few pieces of information (Pienwittayasakul & Liu, 2014).

In request driven model, application calls another application or service and a network of calls is created. The applications and services are tight coupled which means that if one service fails the whole request fails (Figure 5).

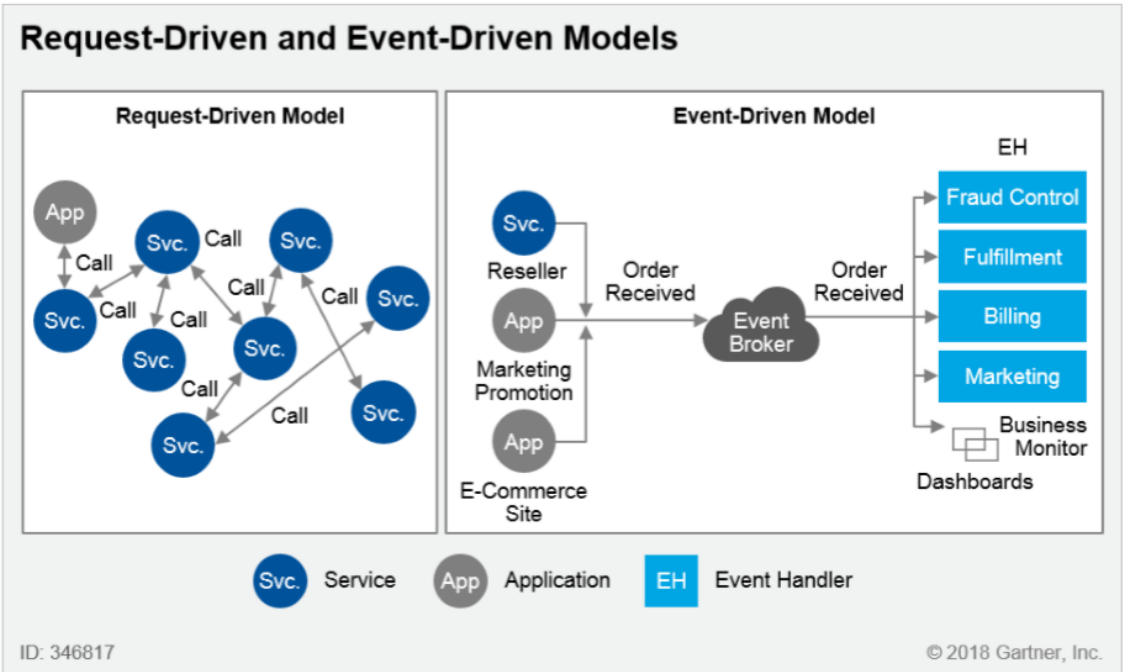


Figure 5 Request-driven and event-driven models (Cearley, Natis, Walker, & Burke, 2018)

Event-driven architecture can be hard grasp to since request-driven model has been prevalent. To be successful, event-driven architecture requires careful and domain driven planning. The architecture provides scalability and enables loosely coupled systems where if a system goes down the other system can continue its operations. Event-driven architecture

2.5 Event-driven architecture implementations

Event-driven architecture can be implemented many ways. Traditional enterprise implementations of event-driven architecture such as queues focus to guarantee the delivery of the message while in modern systems the focus has shifted on efficient delivery on mass of data. In modern systems such as Apache Kafka the guarantee of the delivery must be produced otherwise and might require different type of implementation at the consumer of data than with the queue-type of implementation. Apache Kafka main implementation case is to provide streaming event processing, however it has capabilities to support other types of event-driven architecture (Kreps, Narkhede, & Rao., 2011). In practice technologies that enable event-driven architecture are used to support different types of event-driven architecture such as simple event processing, streaming processing and complex event processing.

More traditional approach to implement event-driven architecture is to use queue software such as RabbitMQ or IBMMQ. They require specialized client libraries to consume

and produce messages since internal communication is often done with using binary protocol. However, technologies such as webhooks try to generalize communication protocol with streaming and queue software by using http. This makes easier to create client software with downside of being less efficient by adding more overhead in each message than binary protocol (Biehl, 2017).

2.6 Event driven reference architecture

The reference architecture is a way to capture knowledge from existing architectures and their realizations. They facilitate reuse and thereby saves the costs through reduced cycle-time and improved quality. However, there are few open issues regarding reference architectures such as who is the intended audience, how to manage the life-cycle of architectures and how to define the domain of architecture (Cloutier, et al., 2008).

Event-driven architecture leads to redundancy of data because same data is distributed to different systems. Centralized control of data is challenging to achieve. On the other hand, duplication enables faster development cycles since two or more teams can use the same data instantly and develop their own services at their own pace. In terms of supply chain management this is called simultaneity.

Event driven reference architecture consists of several components and can be divided in to three layers: integrated resources, enterprise integration backbone and event processing engine. Integrated resources consist of for example applications, events, services and data repositories. Enterprise integration backbone is a layer between event processing engine and integrated resources. Event processing engine consists of multiple components such event development and management tools, event processing rules, event specifications, event data, caching, transportation and service invocation (Figure 6 Event driven reference architecture).

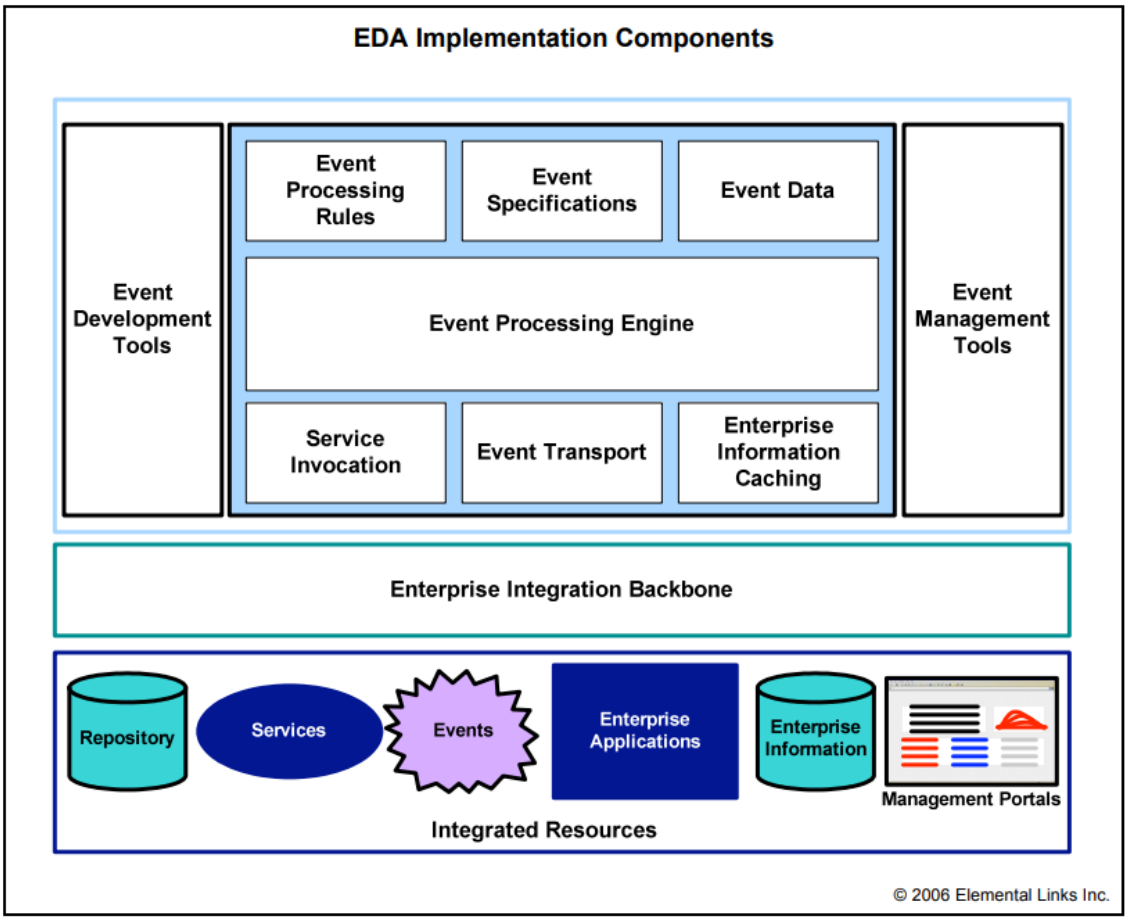


Figure 6 Event driven reference architecture (Michelson, 2011)

2.7 Microservice architecture

Recently trends have resulted a software development approach called microservices. Instead of doing big monolithic software that is tight coupled, microservice emphasizes smaller software component that are independent, loosely coupled and the data management is decentralized (Cuesta, Navarro, & Zdun, 2016). Event-driven architecture complement microservice architecture since event-driven architecture enable decentralized data management and loosely coupled architecture.

Microservice architecture has clients such as mobile application which acts as user interface for the end-user. Front end serves the content for client. Behind the front end are different kind of services both internal and external services. Internal services consist of application components that is usually generated by the team that builds the solution. External services are usually services that are maintained by other than then the team.

However, it is important to note that microservices benefits and requirements are not directly related to event-driven architecture. Event-driven architecture is one of the pat-

terns applied in successful microservice architecture. Main benefits of microservice architectures are evolutionary design, infrastructure as code and service-oriented approach. Evolutionary design enables both functional scalability and operational scalability. Functional scalability means that new features can be added or removed, and operational scalability means that microservices can be scaled easily depending on the number of requests to the service (Figure 7 Scaling microservices.

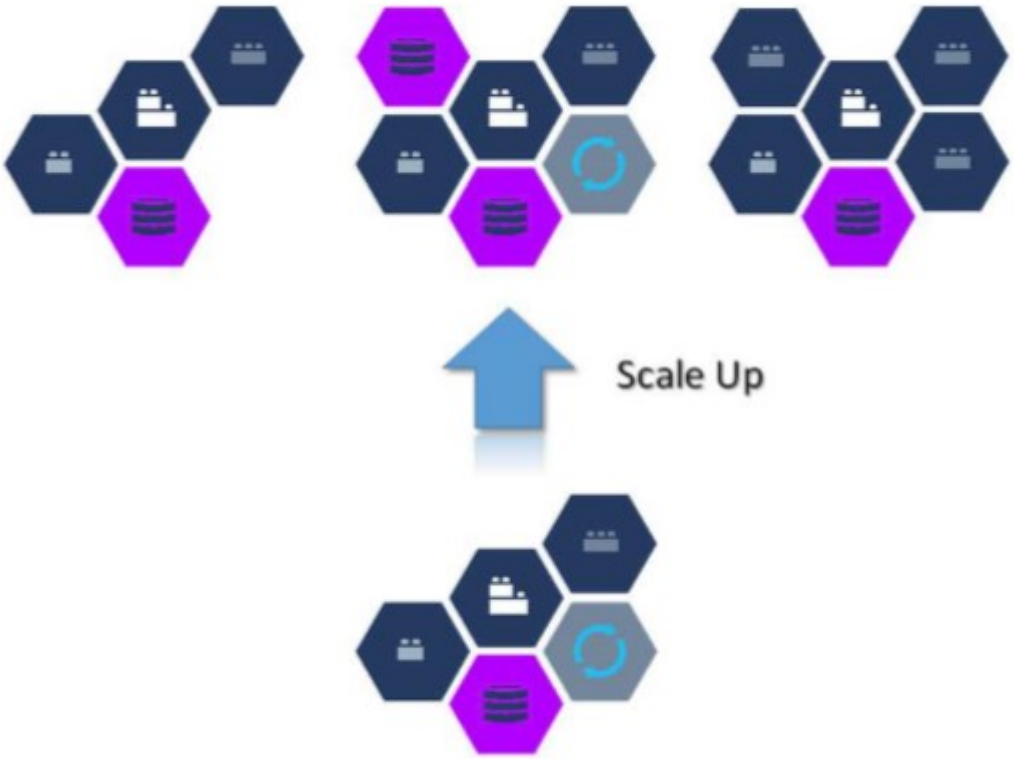


Figure 7 Scaling microservices (Cuesta, Navarro, & Zdun, 2016)

Service-oriented approach means that each service has defined boundaries meaning that its purpose has been defined in relationship of other services. Event-driven architecture enables communication between microservices and other services so that the services are not tightly coupled.

In microservice environment business events occur from outside of microservices such eCommerce order. There might be a microservice that is dedicated to order handling that receives and persist the order business event, enriches the event data with some other information and delivers the enriched business event to other systems for further processing. Communication between microservices can be done by sending more system events.

Infrastructure as code addresses issues where activities such as installing databases, servers, network-components and software requires manual labour. Using tools that automates previously mentioned activities eases deployments and makes infrastructure scalable (Cuesta, Navarro, & Zdun, 2016).

2.8 Summary

Event-driven architecture can be viewed from several perspectives for example from enterprise, integration or application architecture point of view. Each of these domains have something common in terms of event-driven architecture. Thus, there should be common vocabulary and longer-term visioning how the architectures develop within the company.

Event-driven architecture doesn't require any specific technology. Some implementation relies on concept of queue. In this context queue can be message queue or web service endpoint.

Event-driven architecture eases scaling, timeliness and agility but testability and development are challenging due to asynchronous nature of the pattern. The contract between producer and consumer must be established and maintained to overcome issues on the development and testability (Richards, 2015). Relying on open formats, such as XML and JSON, on data transport and schemas such as Json schema and Open API eases the development since data is in plain text format so user can use common text editors to inspect the messages.

Apache Avro, protocol buffers and apache thrift could be used as well on data transport. These technologies use binary protocol which makes the data transfer efficient yet harder to inspect since it requires special type of editors. However, it seems that in future these types of technologies that uses binary protocol will be gain more ground because of their efficiency.

From the business perspective, event-driven architecture is continuum theory of processing large amount of data in more real time. Supply chain management theory emphasizes for simultaneity, meaning that necessary information is sent immediately to all relevant functions for simultaneous processing instead of sequential. Event-driven architecture enables simultaneity by distributing same message to different subscribers or consumers (Cohen & Roussel, 2005).

Event-driven architectures brings value in many areas. Since it enables to take immediate action it enables just-in-time processing for example automatic exception handling, proactive handling, highly personalized services by extending publish-subscribe systems and timely notification for users (Etzion, 2005).

3. LITERATURE REVIEW OF EVENT-DRIVEN ARCHITECTURE

Next chapter consist of a literature review on what are criteria for successful event-driven architecture. First section describes research methods, sources and categorization of the results. In summary is concluded the results of the literature review.

3.1 Research method

The research on successful event-driven architecture implementation is done as literature review. Since the event-driven architecture is not new paradigm, the study takes account also older literature that was published before year 2010. The research focuses on case studies that have been published as part of books or journals articles on the topic. The literature is gathered from online scientific journals, online and traditional books that are accessible from university's library and Google scholar service. Queries that are related to event driven, service oriented, API and microservice architecture were used to gather information. In addition, publicly available implementation guidelines for microservice and event-driven architecture are used to have real life implementation reference. The goal is to produce a summary on basis of literature on what requires implementing successful event-driven architecture.

In the first phase of the research is gathered a long list of possible criteria. In second phase the list is narrowed down and finally analysed. Similar kind of method is done on other thesis on event-driven architecture but more focus on general quality criteria of software development (Klusman, 2019).

3.2 Sources

The sources emphasize different kind of things in event-driven architecture. Since the architecture can be understood different ways, some sources represent architectural components such as business rules and some architectural principles or management type of aspects such as distributed data management. The items of the event-driven architecture are gathered and divided by type.

Since event-driven architecture can be implemented various ways and with different terminology, the selection of sources is done by selecting only those that clearly mention event-driven architecture. By doing this kind of selection the scope is narrower.

The criteria are grouped together by categories and the characteristics of the category is described. The category maps together different criteria and tries to find common criteria in source material. Creating categories and mapping categories can on the other hand lose some vital information on the subject and some of the categories can overlap. Naming categories and mapping can also vary depending on who names the categories and does the mapping. Thus, the source material is provided so that the process can be tracked by another author.

3.2.1 Service oriented architecture

Service oriented architecture was mentioned in each source thus it needs to be considered as important success criteria. Service oriented architecture is a software architecture pattern or framework which evolved in the beginning of the century that focuses on the business-oriented service contracts and integrations between information systems. The pattern makes a layer above technology-oriented entities such as database rows or text documents (Krafzig;Banke;& Slama, 2005).

Service oriented architecture refers to large systems which is composite of multiple applications which are loosely coupled via infrastructure layer called enterprise service bus. Enterprise service bus mediates the communication between applications by providing data transformations and routings. Enterprise service bus is logical layer that consists of multiple specialized applications and components (El-Sheikh, Zimmermann, & Jain, 2016).

In the context of event-driven architecture, service-oriented architecture is both technological enabler and as mindset that enables organization to structure the architecture. When the organization grasps the essence of the service-oriented architecture, the organization can practice event-driven architecture.

Service oriented architecture can be described with many different metamodels that describes a model of model (Figure 8). Service metamodel described how to implement a service, process metamodel describes how implement a process.

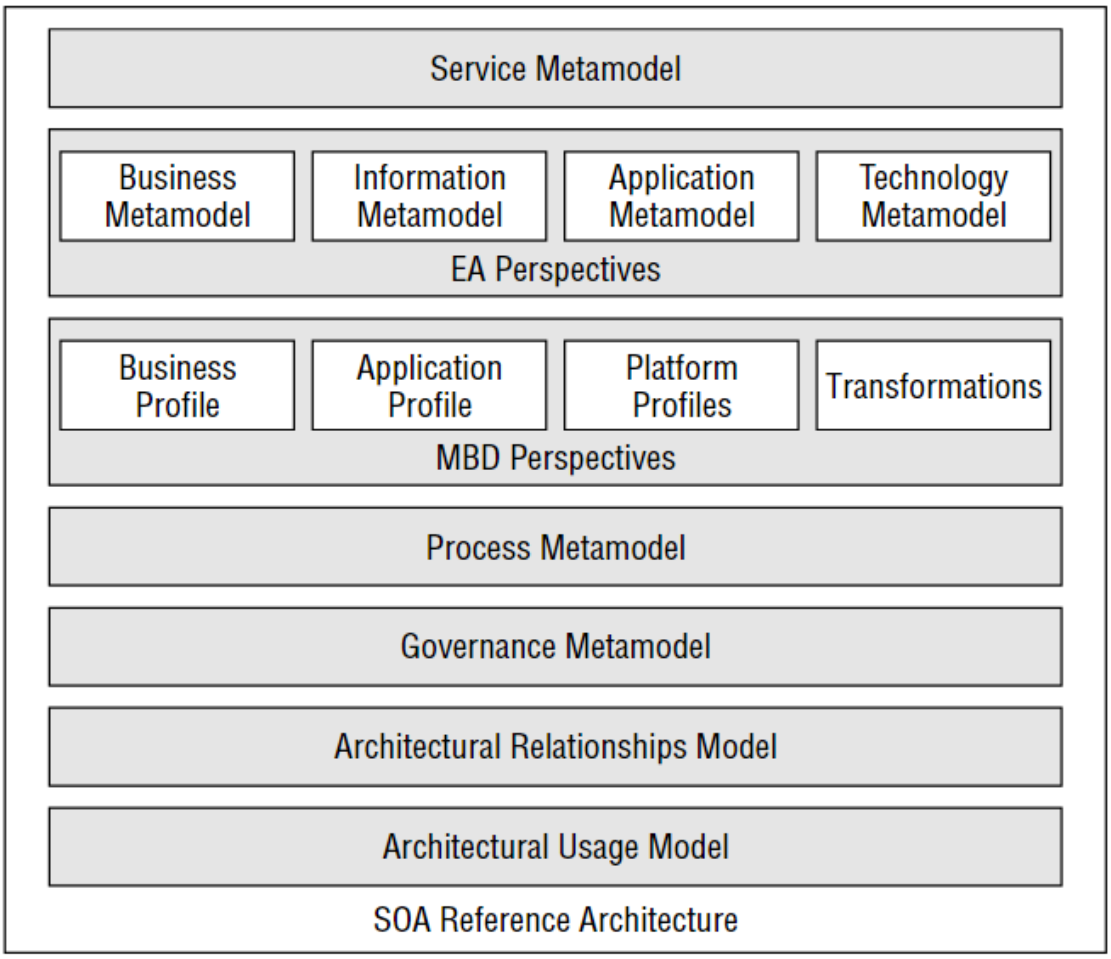


Figure 8 Example of SOA reference architecture

Successful event driven implementation requires many things from the underlying architecture. To achieve simultaneity, the state of business objects should be maintained in distributed manner. This requires that there are direct flows between applications. The enterprise application setup is usually heterogenous thus successful event driven implementation requires different kind of components. For instance, Microsoft and Java stack have its own implementation of queue-systems which are one way to enable event-driven architecture.

Since the events and data are distributed it is necessary to be able debug the data between all applications that uses and refines the data. Usually this can be implemented as so-called audit-log. Finally, an ability to evolve and configure services and applications without long application development cycle (Ghalsasi, 2009).

3.2.2 Governance

Governance itself is broad topic and its full extent can cover the full event-driven architecture solution and concept in organization. In addition of IT and SOA governance, event-driven architecture requires specialized governance. Governance of the event-

driven architecture requires guidelines for designing as described in chapter 3.2.6 since the messaging, technologies and processes can be chaotic if it is not managed properly. In addition, there should be guidelines who can produce and consume every type of events, either business or system. Without the knowledge of the participants, in case of changes in contract, there can be breach of contract that can result defunct features in existing systems (Phillips;Yochem;Taylor;& Martinez, 2009).

To have scalable solution, the governance should also scale. This means that participants of event-driven architecture should be automatically registered and do the necessary workflow to consume or produce the business events. The workflow could contain steps such as accepting the producer to produce the events or consumer to consume the events. Automating governance is time-consuming and makes innovation harder. When governance has been automated correctly, nobody notices it while automating it incorrectly makes frictions.

Governance makes design and development process more formal and thus can be achieved efficient way to utilise even driven architecture in larger scale in organizations. SOA enables systems to evolve independently if the contract or interfaces between systems doesn't change (Mankovskii, 2009).

3.2.3 Loose coupling

Loose coupling and extreme loose coupling are terms that describe the nature how services communicate with each other. Loose coupling is the opposite of tight coupling and extreme loose coupling is an extension for loose coupling. SOA encourages for loose coupling and in the context has been introduced term extreme loose coupling to emphasize difference. Coupling concepts are not only used in IT architectural context but is found used describe organizational theorems how different parts of organization can cooperate and how external pressures fosters organizational departments to decouple i.e. work separately from each other (Sauder & Espeland, 2009).

Tightly coupled systems are small, high performing software components that have internal dependencies for state, data or function. The trade-off is the upgrading and maintenance is more difficult than in loosely coupled system. (Mankovskii, 2009). Loosely coupled services have less impact on the other services, are easier understand and reuse. Loosely coupled service is typically created by using middleware software or API. Middleware software is technology between services to enable the loose coupling but on the other hand creates possible single-point-of-failure since in case malfunction of the software, it might break multiple systems that are using the services that are routed

through the software. Middleware software can be for instance integration or API management platform.

Extreme loose coupling is architectural pattern that encourages to mitigate the dependencies even further. In extreme loose coupled architecture, the producer of the information doesn't know how the information is subsequently processed (Michelson, 2011). Loose coupling enables also systems to evolve independently and on the other hand enables systems to be more isolated. In case of denial of service attacks, only one system is affected, and the rest of the infrastructure stays intact.

3.2.4 Extensibility

Extensibility means how easy is to add new features to the system. This however needs to be further elaborated since extensibility can be subjective. Agile principles encourage to build the simplest architecture that can possibly work. By using this principle, it should be easy add new features since unnecessary features lacks from the architecture (Leffingwell, 2011).

Extensibility is needed when new requirements arise for the system and there is need for a change. Evaluating extensibility of a solution can be done only once the solution has been implemented and changes to the system has been applied. In the context of event-driven architecture, following current best practices for the solution, ensures the extensibility. Some of these best practices are REST-architectural principles and publish-subscribe pattern (Moilanen, 2018).

Minimum viable product is an agile approach where there is vision for a product and its features and they are clearly communicated, not necessary explicitly defined. This approach is inherently extensible since it requires close collaboration with stakeholders of the product. The approach encourages to release the product or feature as soon as possible, gather feedback and extend the solution as per feedback (Moogk, 2019).

From technological point of view, the choices should match to requirements of the case. However, in organizational point of view there should be balance between emergent architecture and intentional architecture to enable innovation and scale benefits. For each case there shouldn't be new sets of technologies since onboarding new technologies can be both risky and time-consuming. Thus, there should be mutual understanding in organization when to try new technologies and when to stick with the tried and tested technology (Leffingwell, 2011).

Extensibility can be divided to changes where is needed changes in client side and changes where there is no need for changes in client. Both approaches have their pros

and cons. When extending system, information about the change needs to be communicated to clients, implement and test. If there is need for a change in client system, the implementation and deployment of change, needs to be closely coordinated between two parties. If there are even more parties, coordination requires more effort on each party. On the other if preferring method where there is no need for change in client systems, it can lead unnecessary complexity in system and reduce the extensibility. To help with coordinating change, SOA helps with governing (Rytter & Jørgensen, 2010).

3.2.5 Technology

The sources mention that event-driven architecture requires specific tools for processing events, agents and rules engine. The sources did not have any common tools or technologies to manage the event-driven architecture. The tools were mentioned as high-level architectural enablers. Some technology vendors such Tibco and IBM were mentioned.

Event processing tool can be full solution that handles many aspects of event-driven architecture such as monitoring, processing and event rules. Rules engine is instead specific component in the architecture.

However, the emergent event-driven architecture has created various tools that are not mentioned in the sources. Business oriented social network Linkedin has created distributed log management system called Kafka that helped Linkedin to scale their services. Since introduction of Kafka similar tools has been introduced to market for example Amazon webservices (AWS) has Kinesis, Microsoft Azure event hub and the original developers of Kafka created commercial version of the tool called Confluent (Kreps;Narkhede;& Rao., 2011).

Event cloud was mentioned in one of the sources. The event cloud was described as backbone for message processing that can be understood as something that Kafka enables. However, event cloud could be interpreted as bigger environment such as AWS, Azure or Google cloud that has other tools that supports event driven and other architectures.

On organizational level, choices for the technology needs to be done and components of event-driven architecture needs to be defined and their role must be mutually understood.

3.2.6 Design

Event model was mentioned in many of the sources. Event model describes the content and information of event. Event modelling is an activity within design phase in which the

information and the content of event is described. (Dunkel, Fernández, Ortiz, & Ossowski, 2011) divided event information to five layers such raw sensor, domain data, problem, cause and action events (Figure 9). In their case study of traffic monitoring system, raw sensor events form the most of events in number. Raw sensors events are too fine-grained to be create suggestions for actions in traffic. Domain data level events are gathered to domain level concepts such as orders or blocks in traffic in some segments. Domain data is a basis for problem and cause events that classifies possible reasons that might cause the problem. For each classification there is several actionable events that shows some information to end user. End user in this case could be road user or customer that has just bought something from eCommerce site. Actionable event could be an email to Ecommerce site user or indication to road user that there is traffic jam coming in 5 kilometres.

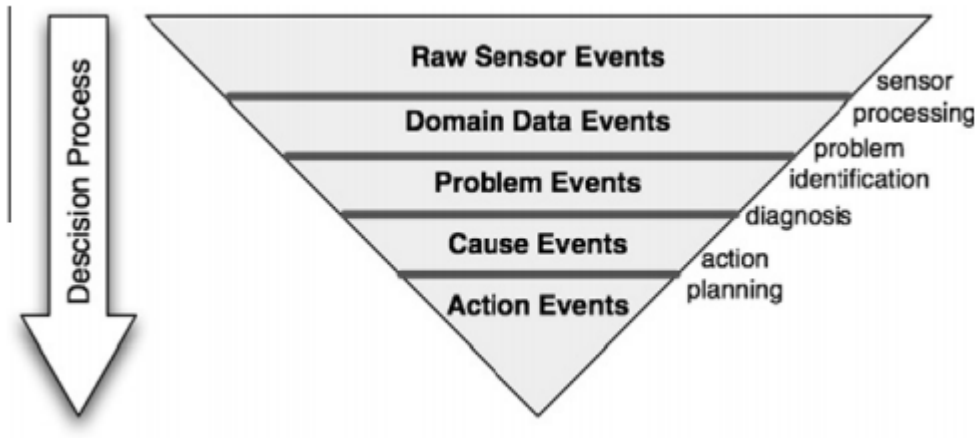


Figure 9 Event types (Dunkel, Fernández, Ortiz, & Ossowski, 2011)

Each event should contain metadata such as event id, event type and event timestamp and the actual payload that varies between event type. Action events are most human readable while single raw sensor events are hard to grasp since they contain only technical data that is understandable only when it is combined with other event data (Dunkel;Fernández;Ortiz;& Ossowski, 2011).

While modelling the event data and schema many companies has established an API styling guide that enforces consistency among the API's (Murphy;Alliyu;Andrew;Kery;& Myers, 2017). Fashion company Zalando has used similar approach for event data where there is agreed model on metadata what each event should contain (Zalando, 2019).

Modelling events can be done with various ways depending what technology is used. Technologies such as Apache Kafka or AWS doesn't require formal structure such as

JSON or XML. However, to gain event-driven architecture's other benefits such as loose coupling and extensibility, formal structure is required.

3.2.7 Development

Successful implementation requires that development of the services should be simple. Requirements management for event-driven architecture should not vary from other kind of architecture or software development process. The requirements phase emphasizes real-time data processing. Design phase can differ depending on preferred technology. Implementation or development phase is heavily dependent of the choice of the technology and the actual implementation. The development should speed up since event-driven architecture requires separation of business components and forms formal contracts such as API's for the data. This enables multiple teams to work simultaneously with the same data without interfering with others. Formal contracts enable easier testing since test data can be produced independently by testing personnel.

The event-driven architecture enables to easily create a concept called audit log which can be used during testing or maintenance phase. Audit log, audit trails or logs are immutable lists of records. This feature allows to create applications for example to ensure transactional integrity of order transactions.

In addition, statistical applications such as machine learning cases require a lot of history data to make predictions. Event-driven architecture and audit log feature enables another way to collect data. In traditional integration platforms the changes in integration required a full development cycle in the integration platform before the change was ready. In event driven platform the change can be applied as own development cycle.

In the style guide of Zalando can be seen that request-driven and push-driven API's have much in common while defining operative aspects of API design such as which date or currency format should be used when developing API's. This is implementation of description where event-driven architecture is an extension for SOA. On organizational level combining two similar kind of concepts can have efficiency benefits by making learning curve smaller and communication more efficient (Zalando, 2019).

Zalando has published their style guide in public Github-page that is available for everybody. This eases the communication to suppliers and Zalando can gather feedback from suppliers and other interested parties and this way improve their processes.

3.3 Summary

Success event-driven architecture criteria as per literature was narrowed to seven categories (Table 1). The categories interlay with each other and it is impossible to implement category by category in organizations. Naming categories is difficult because the terms overlap and there can be different wording for similar terms. Organizations must understand the essence of categories to successful implement event-driven architecture. In this chapter is displayed part of characteristics of event-driven architecture based on selected literature. The list is not comprehensive and there can be different interpretation from the same literature. The categories describe the areas of interest for event-driven architecture as per literature. Each category was further categorized as to characteristics or as activity. Activity means an action that organization must perform to achieve event-driven architecture. Characteristics describe distinctive or typical features of technology or system that event-driven architecture requires or enables. Type of each category further defines each item in order to make item more understandable.

Table 1 Summary of event driven categories based on the literature review

Category	Type
Service oriented architecture	Principle
Modelling	Activity
Development	Activity
Loose coupling	Principle
Extensibility	Principle
Technology management	Activity
Governance	Activity

Service oriented architecture (SOA) is mentioned in each article thus it should be counted as important criteria. SOA is a requirement for event-driven architecture since it gives guidelines to architecture that can be extended to event-driven architecture. Loose coupling and extreme loose coupling are subject to much interest now because they enable isolation of systems which contributes to increased speed of development and increased security of the systems. SOA introduces term loose coupling and event-driven architecture emphasizes coupling even more by introducing term extreme loose coupling while also giving guidance how to implement such systems. When having multiple IT systems in organization, the vendor of the systems tries to create the best possible solution for client and try to prevent clients for looking for alternative solutions. With the help SOA each system has its role specified and thus the overlapping features should

appear less. This leads on the other hand to increased need for integrations between systems (Mankovskii, 2009).

Extensibility is one of the benefits that is expected from event-driven architecture. Extensibility is combination of governance, technology and SOA. By having implemented these, the architecture should be extendable. In agile context, extensibility means that not all the things can be pre-planned and accepting the change.

The choices of technology should be fitted to the use case since event-driven architecture can be implemented many ways. However, successful implementation requires that the needed components are named, their role is described and understood in organization.

(Ghalsasi, 2009) dedicated an article to identifying success criteria of event-driven architecture. This review included other sources as well and brings extended view on the topic. The sources did not mention new paradigms such as serverless, cloud platforms or technologies such as Kafka or Kinesis and how to leverage such technologies with event-driven architecture. This might indicate that there needs to be further academic study on how to combine these tools and technologies with event-driven architecture.

What event-driven architecture is trying to achieve is to send information about something that has happened from one party to another. To understand the information, there must be contract, formal or non-formal, between two parties. Since technology enables to deliver the same information about the event to multiple parties, each party must be able to interpret the message. Thus, is needed governance that defines for example the message format, type and how to receive and send the messages. This helps in both design and development phase and in addition there should be clear distinction between event types. Event have different purposes that changes according who is receiving or sending the event. Currently internet-of-things trend tends to increase the number of devices that are to send events. However usually these devices have limited computing power, are energy-aware and are fit for one specific purpose which means that the devices are very optimized. Thus, the devices are not able to compute vast amount of information and are able to produce limited amount of information and send it via protocol that doesn’t require much computing power. This means that the information must be correlated and curated somewhere else. Combining event types (Figure 10) and data pyramid (Figure 1) gives news perspective how to differentiate event type from each other (Figure 10).

Figure 10 Combined event types and data pyramid

Event-driven architecture has many criteria for success. Instead of organizations of implementing criteria one by one, the path to event-driven architecture is rather evolutionary rather than revolutionary. Event-driven architecture is extension to SOA and organizations must understand preceding concepts before they can implement event-driven architecture.

4. USE CASES OF EVENT-DRIVEN ARCHITECTURE

In following chapter is described two sample use cases where event-driven architecture can be applied. The use cases are based on real life cases that have been in discussion within the company. Based on the use cases and literature review, a blog post is done that is posted in company's internal website. The blog post is used as a basis for interviews. The blog contains internal information thus it is not part of the thesis.

4.1 Case descriptions

Cases are described by its business case and what value solution should bring to both organization and to customer. Architecture is described by its domain model, use case and system context. Domain model describes the artefacts of problem area such as cars, persons or customers and their relationships (Lano, 2017). Use case describes complex behaviour of systems and how the systems interact with each other's and how their actors interact with systems. Use cases are derived from user stories that describes what the system needs to for the user. System context diagram describes the sources that contributes to whole system (Leffingwell, 2011).

Other options to describe the architecture would be to use TOGAF-type of division in business, application and technology architecture. In this thesis business cases represents informally business architecture in terms of TOGAF. The Application architecture is described with architecture, system context, domain model and as use cases. Both formal and informal ways are used to describe the architecture, the thesis emphasized however informal methods to keep the thesis scope in line.

The illustrative figures are done by using Archimate notation since it provides way to describe enterprise architecture. Archimate is standard that consists 6 components including framework, modelling concepts and language semantics. However, only small subset of Archimate's modelling concepts is used to keep the scope of the thesis limited (Lankhorst, Proper, & Jonkers, 2009).

The vocabulary while naming the fields of the schema and objects, for instance for first name and last name for the customer data, is used from the schema.org. The schema.org gives common vocabulary for the terminology for free of charge and the terms are available online thus it is easy make references. Vocabulary also standardizes the date, language and currency formats. Other option would be to be use TMForum's

specifications but since the material is not publicly available it is not utilized in this thesis. TMForum is a non-profit organization that helps communication service providers by providing collaborative environment which contains for example data model and API examples and vocabulary (TmForum, 2019)

At the core of the event-driven architecture is formal way to describe the structure of events information. For each event a sample of event message is described (Table 2). Each event type has similar structure which consist of metadata that describes common properties of each event and the actual message of event. Description-field tells the purpose of the event in short sentence. Name gives global unique name for the event. Owning application describes the application that sends the event. Schema describes the actual payload for the event, and it is different for each event type. CreatedAt and updatedAt describes when the event is created and updated accordingly and are given in ISO 8601 date format.

Table 2 Metadata of event type

Field	Description	Example
Description	The description of the event	Event for updating customers sta-
Name	Unique name of the event	customer.credit-score.update
Owning application	The origin of event	Credit-scoring-service
Schema	The actual payload of the event	
createdAt	Time when event was created	2015-05-28T14:07:17Z
updatedAt	Last update time for the event	2017-04-12T12:15:37Z

4.2 Case 1 - Credit scoring

Credit scoring is a method of transforming relevant data into a numerical representation that helps make decision on giving credit. Scoring requires only those variables in decision making that correlate with repaying history (Abdou & Pointon, 2011).

4.2.1 Business case

Business has a hypothesis that instead of relying on 3rd party credit score providers, the company has enough information that they can produce credit scoring themselves. The case has potential to lower the costs of scoring per client. The company would not have to rely on 3rd party for providing the score since 3rd party charges on each scoring request.

Business must be able to adjust their risk depending on market situation by selling products to reliable or non-reliable customers. The problem lies how to provide such information in real-time to all the necessary. The business case is not about how the credit risk is calculated but rather how the information is provided in real-time.

As descripted above business case can be divided following user stories:

- As business user, I can create automatic rules that determines monetary amount that client can buy at once
- As business user, I can deliver automatically monetary amount to various systems
- As developer, I can change the rules set by the business user
- For system reliability other systems should be loosely coupled

4.2.2 Architecture

The scoring is done by analysing the payment behaviour of each customer and giving a score from 0 to 100 depending how much the company can rely on client for selling products on credit. 0 means that customer is not reliable for giving credit and 100 means that according to algorithm is very reliable. Business can adjust their risk by selling products to customer that have lower or higher credit score.

The company has many systems that can sell items. Each system is different, they have different data model, different user interface, different customers. This poses a challenge on how to provide a credit score for everyone.

The targeted sales system had already concept of credit limit. Credit limit is monetary limit that enables client to purchase only items within the limit. So, if the credit limit is for example 200€ then client can purchase items which total sum is below credit limit. Instead of providing a credit score it made more sense to calculate monetary credit limit that are based on credit score. This approach has several advantages: the credit score is hidden from the sales personnel and they are only able to see credit limit as they are seeing it before. Before the change each client had fixed credit limit.

4.2.3 Domain and data model

Domain model consist of three objects: customer, credit score and credit limit. Customer object details all necessary information about the customer such as customer identification number. Credit score object details a credit score that are associated with customer. Customer can have exactly one active credit score at any time. Credit limit object is associated with credit score and is dependent on credit limit business rules. Business rules defines how credit score is transformed to credit limit (Figure 11).

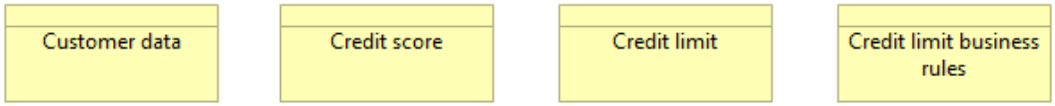


Figure 11 Credit score domain model

Data model consists of data credit score service and point-of-sales system. The integration is described as “interface 1” in (Figure 12). Data model is rather simple, it contains 7 fields (Table 3). Interfaces 1,2 and 4 are not described to keep the scope of the work narrower.

Table 3 Credit-score event type data model

Field	Example value
description	Event for updating customers credit score
name	customer.creditScore.update
owningApplication	credit-scoring-service
customerId	007
creditScore	98
dateCreated	2015-05-28T14:07:17Z
dateUpdated	2017-04-12T12:15:37Z

4.2.4 System contexts

Request based system context consists of three actors and two relationships between actors (Figure 12). Customer data has all the necessary data of customer for credit score service to form the credit score decision. Credit score service generates the score once every night. Point of sales system gets the customer score from the service once every night.

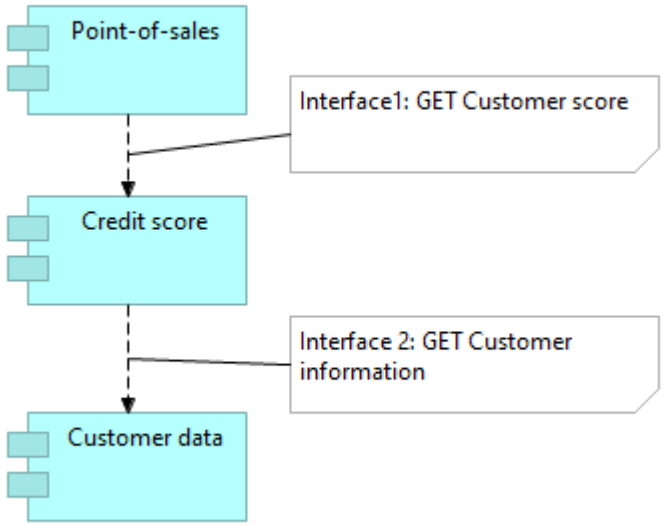


Figure 12 Request-based system context

Event-driven system context consists also of three actors and two relationships between actors (Figure 13). The actors are the same but instead of actors requesting data, actors will push data forward. This enables that each service and system can react instantly when the data arrives. The difference between request and response model is that in response model once the data changes it will calculate new credit decision. Decision is a formula that uses many different data points that are related to the user. It is vital to recognize the most important data that affects to the credit scoring and new calculate credit decision when there is a change in that data.

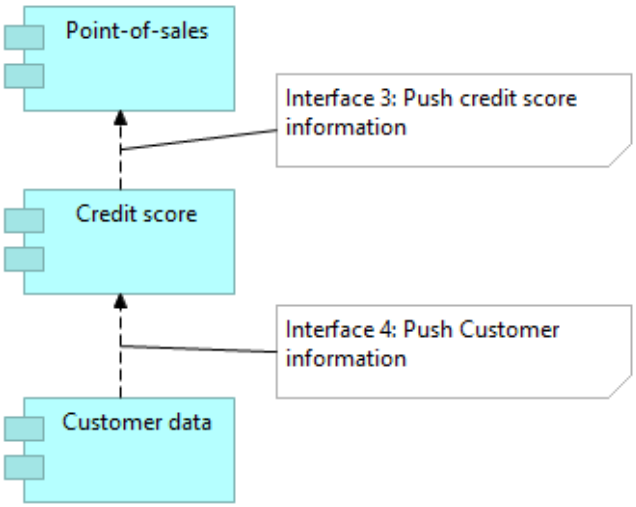


Figure 13 Event-driven system context

4.3 Case 2 - Real time account information

4.3.1 Business case

Account and customer information is scattered around in different business supporting systems and CRM systems. The business supporting systems and CRM systems are responsible for account information that are related to their businesses. The account information is gathered to single data repository where the data is harmonized, and the data is used for marketing purposes. However, there is not clear process on how to provide the information to other systems, the current method is perceived slow in terms of real-timeless and it is not clear how to scale the process. Scaling in terms that how multiple systems can provide real time changes to account or customer information. There is no agreed single point of truth for account information nor it is available for systems that needs the information.

In future there is need to provide the account information to and from various systems in real time. Multiple teams should be able to communicate the information via uniform and scalable way.

4.3.2 Architecture

Current process is to load the data to datawarehouse, to uniform the data, create views from the data for the consumer needs. This means that there are at least three teams and systems are involved while transferring data: producer such as CRM-system, middleman such as datawarehouse and consumer such as web portal (Figure 14). Current process is perceived slow since while transferring data from CRM systems all data is transferred, not just customer data. Once the data is transferred, data needs to be transformed multiple times: first to datawarehouses internal model then to the model that consumer requires. In addition, when bringing new data objects to datawarehouse, process requires resources for modelling the data from each team.

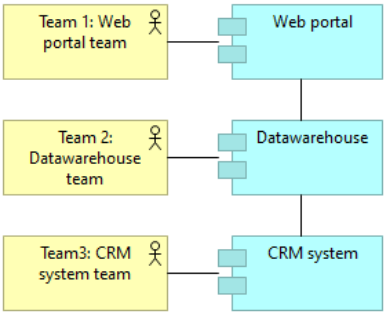


Figure 14 Current system responsibilities

While datawarehouse is necessary component in today's enterprise architecture, it may complicate the architecture and slow the data processing. To complement enterprise architecture, event driven components needs to be introduced. While planning the solution datawarehouse team has responsibility to create the data model and provide it consumers. The data producer, such as CRM system doesn't have much responsibilities. Team 1 and team 2 has the discussion about the data and the team 3 is consulted if necessary. There might be issues since the team 2 doesn't own the data and doesn't necessarily understand the semantics of the data.

On the organizational level, having middleman between different parties enables to create governance on the data model, data usage and data generation. In the current scenario every time there is a requirement for different data model from the consumer, it requires effort at least from datawarehouse-team and they might have to consult CRM-team as well, thus the datawarehouse-team might become bottleneck. On proposed solution datawarehouse and web portal systems acts as event consumers and CRM system as event producer. The solution requires that CRM system has technical ability to produce events and both datawarehouse and web portal consume the events.

The event model needs to be agreed between CRM system and web portal since web portal is the primary use case. As described in Figure 9, the event should be actionable, meaning that the information is as complete that consumer of the information can continue processing when it receives the information. Consumer doesn't have to do additional processing for the information.

The term customer data itself is problematic since it is ambiguous; different people understand different data as customer data. While much of the data relates to customer, it can be difficult to decide what is the master data and what is related data. In addition, organizations may have pieces of customer data such phone numbers and names that might have some significance in some context but cannot be considered as master data in company level.

The main problem with master data management is that the perceived benefit is that it provides a single point of truth for each need regarding the information domain such as customer or product data. However, since not all needs can be fulfilled with one solution, it requires complementary solutions. The amount of data is growing rapidly thus there needs to different kind of solutions to ingest and publish the data. Master data management deals with more static data and event-driven architecture deals with transactional-

data means that with master data management it is difficult to achieve real time data processing and distribution (Smith & McKeen, 2008).

Since the data producer cannot know for which purpose the data will be used, there should be categorization for each event message. By categorizing the message, producer can give a hint of intendent purpose of event. There should be also information about the domain of event such as is it related to customer, product or to some other agreed domain. In addition, in case of customer data or personal, general data protection regulation (GDPR) must be taken in account.

The problem can be divided to two concepts: master data management which addresses data harmonization from multiple sources and providing single source of truth. The second concept is real time data processing that can address data transfer in and out to the master data management system. Master data management strategy can be divided to three types virtual data integration, datawarehouse loading and mixture of previous (Bernstein & Haas, 2008). Datawarehouse loading means that the actual data is loaded to centralized location acts as point of reference for the information. Virtual data integration means the data remains in original system and acts as point of reference. The point of reference can be implemented as an Application Program Interfaces (API).

Since customer data is scattered around, there is a need to determine the master data management process. During the creation of master data management process there should be establish also ways to deliver the data.

4.3.3 Domain model

Customer data's domain model consists of the data that resides at the different BSS and CRM systems. Domain model is kept rather simple; it consists of customer data that resides in different CRM's and BSS's and the aggregation of this information is simply called customer data (Figure 15).

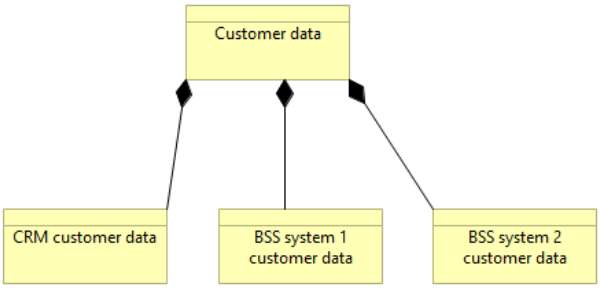


Figure 15 Customer data domain model

The target data model that each BSS and CRM system tries to send the message is described in Table 4. In case it is not feasible for the BSS to generate the message in target format, the Datawarehouse formats the message to target model. The downside is that while datawarehouse combines the data, sending the data is slower because there is additional logic.

Table 4 Customer data model

Field	Example value
eventDescription	Event for updating customers credit score
eventName	customer.information.update
owningApplication	CRM-system-1
customerid	The actual payload of the event
givenName	Teppo
familyName	Testaaja
socialSecurityNumber	010101-123d
addressCountry	FI
addressLocality	Tampere
postalCode	00101
streetAddress	Kalevantie 20
birthDate	1985-05-01
deathDate	1995-01-06
email	teppo@testaaja.fi
gender	Male
dateCreated	2015-05-28T14:07:17Z
dateUpdated	2017-04-12T12:15:37Z

4.3.4 System contexts

Master data management systems relies heavily on centralizing the data. Event based system enables distributed data management. Request-based system doesn't determine where the data is located. Event, request-based and centralized master data management systems complement each other, and their role change according to viewpoint.

Event platform centric approach expects that producers can produce events that match agreement. On the other hand, some of the business logic transferred to consumers such as duplication detection. This is not desirable since same business logic must be

duplicated in each consumer. In case of introducing new consumer, same business logic must be in new consumer (Figure 16).

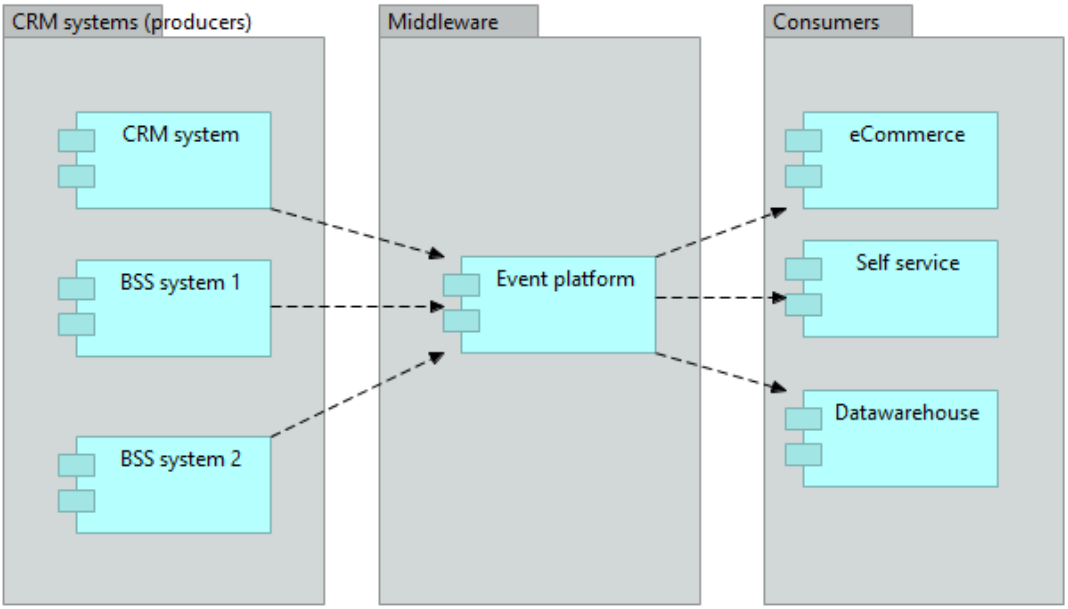


Figure 16 Event platform centric approach

Master data system centric approach doesn't enforce the producers to produce data in agreed schema. However, this slows the data transfer. In this scenario business logic is written only once and consumers receive data without need for additional processing (Figure 17).

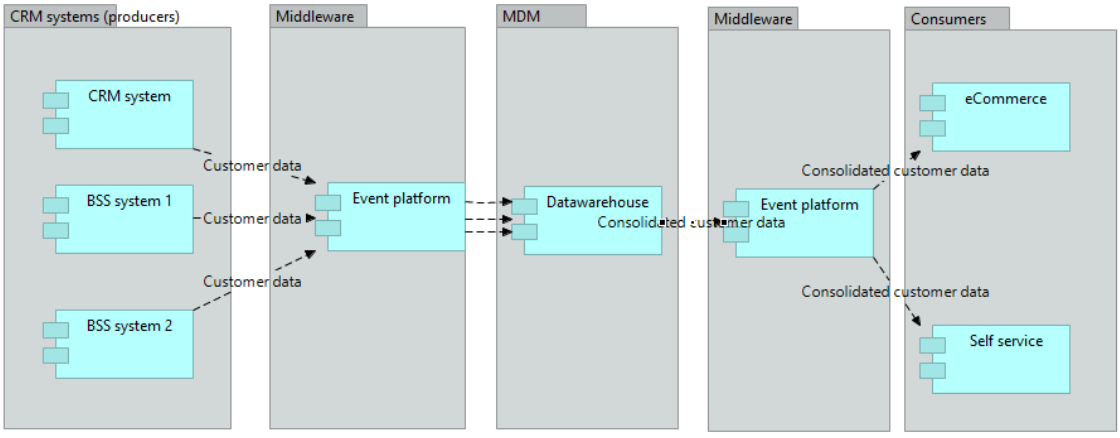


Figure 17 Master data system centric approach

4.4 Summary

Master data management and event-driven architecture are concepts they complete each other. Master data management focuses on business process of data while event-

driven architecture focuses on data transfer and what are the roles of each components. Master data system be one consumer or producer, depending on viewpoint.

Managing credit risk information is interesting topic in many contexts. How to combine information to create credit risk profile is interesting topic itself. However, this case is not covered detail in this thesis but in general level. How to distribute the credit risk profile once the profile is created is covered.

5. EMPIRICAL INFORMATION GATHERING

5.1 Research method

The research method for the gathering empirical information was chosen to be quantitative with focus on open questions. The research area is wide, and it is not practical to gather qualitative information due to schedule and the scope of the thesis. To gather qualitative information would require different type of approach for the research question and the target group.

Interview as research method tries to understand the interviewees in various ways. The experiences and the meaning of what the interviewees say. Interviewer is part of measurement and in this kind of cases, it might be problematic if interviewer has deep knowledge on the subject. The challenge is addressed by studying the related literature.

The interviews are recorded but since the focus of the interviews are not on interaction between interviewee and interviewer, the interviews are not transcribed. The recording of the interviews helps to analyse the interview-situation and acts as an aid for recollection. The purpose of the interviews is not track individual opinions or views and while composing the analysis the anonymity of the interviewee should remain (Ruusuvuori & Tiittula, 2005).

The form of the interviews is semi-structured. There are common questions that act as themes for the interviews but at the same time, the questions may be altered during the interview to try to reflect the situation of the interview.

5.2 Interviews

Interviews aim to gather knowledge from the interviewees. While the event-driven architecture as a concept might be new to interviewees, they might have knowledge on the subject due to prior experience on similar topic. Hypothesis is that if interviewee has participated software development progress and has particularly been involved with data and integrations, there is good chance that he or she is able to relate event-driven architecture even though he or she hasn't heard about the concept before. Thus, interviewees can give further knowledge on the subject. For the stimulation the interviewer describes the business cases as mentioned in chapters before. The business cases are expected to spark some discussion.

The purpose of the questions is to evaluate the proposed solutions and what sort of building blocks is required for event-driven architecture and what sort of building blocks company might already have and are either underused or in good shape. The results are used as material on next chapter to create event driven capability. A Framework for conducting the interviews was created (Table 5). The purpose of framework is to help to structure the interview and to ensure that each topic was covered during each interview. During the interviews the framework is loosely used since the style of the interviews was conversational.

The interviews have also educational aspect in terms that the interviewees learn about the concept more and can reflect their own knowledge and know-how to match on the subject. Form organizational point of view interviews prepare interviewees for possible changes.

Table 5 Question framework

Theme	Question	Objective
Background	Tell about yourself and your background	Understand about interviewee
Event driven warmup		Set the context of the interview
Business case 1 & 2	Was the business case clear? What would like to know more about it?	
Business case 1 & 2	In the context of the event-driven architecture, how would solve this issue? What issues should be addressed?	
Business case 1 & 2	Could name components with basis on reference architectures.	
Common	How do you see relationship between master data management or datawarehouse and event-driven architecture? What sort of enablers do you think our company would need to be event driven company?	
Common	What sort of benefits you see for our company? Your department and for your work?	

Common	What kind of obstacles and possibilities do you see with the approaches	

5.3 Target group

The target group for the interview are the architects or people who work in such area. For some people of the target group the concepts of event-driven architecture and cases are known since they have worked on these areas. The target group were given a brief introduction of event-driven architecture in the form of blog post in company's internal collaboration space.

The target group consist of three men working in IT department in roles of domain, integration and information architect. They have been chosen for the interview because it is recognized that they have related experience on the event-driven architecture. It is assumed that interviewees have knowledge that can contribute on creating event-driven architecture capability.

Other stakeholders for the interview are enterprise architecture team, chief enterprise architect, other architects, IT department and other departments that are close to IT development. The chief enterprise architect represents the customer in the interview.

The relationship between interviewer and interviewee are collegial. Everybody works in the same company, they may or may not have superior position with each other. Interviewer has known most of the interviewees for year and half.

5.4 Assumptions

Since the interviews are not just isolated events in corporate environment, there is continuum before and after the interview. There are certain assumptions that has emerged before interviewing due to the discussions between participants and interviewee that has happened during the normal interaction between colleague and during the more formal meetings.

Among the target group there has been discussion how to document enterprise architecture artifacts such as descriptions of architecture. This might raise some questions during the interviews for some participants. The orientation for the interviewees is important since this kind of questionnaires are not common among target group. The presented architecture might conflict with the beliefs on the good architecture. Especially distributed data storage and delivery might have different opinions since the discussions

before the actual interview have shown different opinions on the subject. The term good architecture is subjective, but it has some commonly defined principles that each interviewee looks from him or her own point of view. The commonly defined principles are not written down and mutually understood.

There have been discussions about the master data management (MDM) and master data system (MDS). It seems that there is a need for clarification for the roles of MDM and MDS. There have been discussions about a notification service. It has been recognized that notification service might have something to do with the event-driven architecture, but the notification service concept has not been defined properly.

The interviewees are technically oriented and have common interest on architecture through their role. Interviewees can be somewhat familiar with the cases and might have preliminary ideas how to solve the issue. This might distract the purpose of the interview and lead the focus on how to solve the issue on local context and not thinking on the company-level. On the other hand, it is important to solve issue on local context. Interviewees are provided material such as overall description of event-driven architecture and two business cases where the architecture could be adopted, before the interview. There is a risk that interviewees don't have enough time to go through the material thus the material is tried to keep minimal.

5.5 Interview analysis

On following section is analysed the interviews and gathered a summary of interviews. Based on these interviews and summary a capability of event-driven architecture is recognized.

5.5.1 Interview 1

Event-driven architecture was recognized as a software pattern that interviewee had experience in past in large retail case where point-of-sales system which was required to be functional in case network outage. The past technologies did not match to the current technologies that have come across with big data. Also, in past the networking capacity was more limited than currently that led to situations where data receiving party had to build additional logic to handle the traffic. The debugging was more difficult since it requires low-level packet sniffers that were difficult to use.

There was experience that event-driven architecture can accumulate mistakes quite rapidly due to speed and scale. For instance, wrong price information for the products in large retail store chain can have unwanted impacts such as customers will buy product in wrong price. For the company impact is that they will not get the planned profit. While

designing system it must be considered if it is important to share always all products and their information or should there be only the changes of the product. In past there was not enough technical capacity to update all information but now the situation might have changed.

When discussing about the customer data it was noted that traditionally the customer data has been centralized which means that event-driven architecture has not be utilized fully in this context. On case of credit scoring, it seemed important case but the implementation in event driven context was hazy.

Since event-driven architecture relies heavily on technology there should processes to manage the technology since technology obsoletes rapidly. Since there are many development teams there should be a way to communicate about the preferred choices in technologies and tools. There should be some mechanism to evaluate which technologies should be used in new and old projects. Decisions should be made whether old projects will be upgraded to use new technology. However, for the upgrading there should be clear business need to utilize new technology. In addition, there should be focus for interoperability since in big organizations there will be technologies from different vendors and in different life-cycle phases that should be able exchange data in common way.

When discussing about the master data management and its relationship with event-driven architecture it was noted that as technological solutions both can be lightweight. Meaning that there doesn't necessarily have to be big upfront investment in some event driven or master data platform, instead the architectural patterns should be understood and what are their benefits and downsides. There should be evaluation how such pattern could be implemented with existing tools.

5.5.1.1 Summary

Interview enforces theories that event-driven architecture is not new thing and such pattern has been used previously in critical use cases. Technology and portfolio or project management was recognized as one of the enablers for the event-driven architecture. It should give answers what technology to use in which project.

5.5.2 Interview 2

The main responsibility of the department where interviewee works has been as data processing, data warehouses and master data management. The department has recognized that there is direction towards events or business objects, but events has not been their focus. It was predicted during the interview that in future there should be event hub that will provide events to any interested party.

Event is seen as change in business object such creating a new lead about customer. In terms of integration there has been discussion between process and data integrations. It was discussed that data integrations have been there for long time. Receiving the data in format of event, the process integration is enabled, and other parties doesn't need to poll the source to receive the data or information. This streamlines the development and releases resources. One notes of interview was that event-driven architecture requires schema where is described each field and fields type such numeral, textual or boolean. This could be called as schema management capability.

In past while integrating the systems, the order of the messages was considered critical. However nowadays the importance of the order has reduced because architecture has been evolved and is evolving to that direction where the order isn't that important.

The business case about account and customer lifecycle matter most if the customer is already formed in systems. The master data type of approach was complementary to event streaming.

5.5.2.1 Summary

The interviewee is focused on data platforms such as datawarehouse, master data management systems and real-time analytics. The concept of event-driven architecture was familiar to interviewee and he has insight how to further develop the architecture with an example of concept called event grid. It was recognized that there are many event types such as data events and business events. The departments scope has been on system events not on business events thus the event-driven architecture has not been topical. However, the department has both technological and organizational enablers to further develop the architecture.

5.5.3 Interview 3

Interviewee was a bit skeptical about the approach but after initial material, finds event-driven architecture as a viable pattern for some cases.

Data sharing requires special care and coordination how to resolve conflict requires special care such as if same information is updated in multiple places, how to determine which is the clients will. Interviewee stated that it doesn't make sense to have multiple master for the data that can update data.

Governance model must be though thoroughly. It should be known that what data resides where and who has which data. For instance, in case of customer data each data consumer should take care of privacy related issues. In addition, configuration management should be taken care of.

Adopting event-driven architecture can be difficult because of cultural issues. People tend to stay in old habits, and it can be difficult to adopt to new kind of thinking what event-driven architecture requires. There was discussion that there should be company-wide rules for events and API's but implementing those can prove to be difficult. There are many products in company and asking vendors or integrators to provide event data or API's that comply to company's standard will be difficult

There could be at least two type of event processing engines such as enterprise and application level. Enterprise level will provide information that is commonly agreed in company, it will be used in wider context and its quality will be better. Consumer should receive only so-called enterprise level event data because this way there is no need to interpret from the data that what is for example customers name. Instead enterprise level event should normalize this terminology and data. This leads also to fact that the event data should be discoverable from API portal or other such place with good documentation. API portal is a web page where is listed all API's that are published. By having API's and event listed and properly documented preserves time while implementing application that uses events or API's. What is proper documentation and what is API needs some more specification.

When discussion about the use cases for the event-driven architecture, real-time inventory came across. Physical goods stock levels are usually stored in ERP system. Since there is a lot of queries for the stock data, ERP can become a bottleneck in some case. Stock data could be provided as events in order to ease the load of ERP system.

5.5.3.1 Summary

The interviewee was responsible of integrations. The integrations being done in his department is more request type integration as opposed to event driven. However, there was found many similarities in both approaches. The data is essential in both approaches and when viewing the subject from more strategic level, the direction of data is just a detail. However, the changing the direction of data has much implications in implementations and different type of things must be considered.

The governance related issues were raised such as in company level there should be knowledge who is using which data. Or which systems consume which data. The discussion revolved around technical topics which was understandable. However, there was discussion about the culture that it might create barriers because personnel have used to request-type of behaviour. Other topic was that there is purchased software products with different type of contracts, and it can be difficult to make changes in their systems. Some changes can be done depending on contract and on level commitment from the partner side. Telecommunication industry has formed some standards around

the API's so that should ease the changes if the proposed changes are aligned with standards, business requirements and products own roadmap.

5.6 Summary

The atmosphere on interviews were pleasant. The interviewees were able to familiarize themselves with the material provided before. Interviewees could relate themselves with the material. Two of the interviews were facilitated through remote meeting via Skype-software and one were in person. There was reserved one hour for the interviews but for each of the interviews the interviewees indicated that they would like continue discussion after allocated time. This was encouraging and can be interpreted that interviewees find the subject interesting.

The interviewees were technically oriented, and the material was built for technical oriented people. To have such interview for more business-oriented people can be more challenging and would require different kind of approach.

During the interviews it became apparent that the company is on early stages on event-driven architecture. However, there is good grounds to make advances in this topic since within the interviews there seems to be interest and motivation to co-develop further the architecture and issues around it. During the interviews there wasn't discussion about the advanced topics issues that will arise while implementing the event-driven architecture. Such advanced topics could be software patterns such as Command-query response separation or event sourcing. For these reasons, while advancing with the event-driven architecture, there should be careful consideration.

5.6.1 Business cases

Business case about credit check seemed quite trivial case for most of the interviewees. However master data management and event-driven architecture sparked some very good conversation. It was perceived that producing same information, for example customer information can create conflicts. It certainly can create conflicts but since the discussion did go too deep in use cases, interviewer was left with a feeling that there might be some true behind the thoughts, but it requires more thoroughly cases to be able concretize the problem. When discussing about the customer data for instance, the concept is so ambiguous that it doesn't reveal what exactly is customer data or what are the use cases.

The business cases gave some concrete topics for discussion. On the other hand, most of the interviewees knew a lot about the problem area so they could give insights on

topic. However, were there an interviewee that wasn't too involved in subject, the business case might seem too trivial or as topic they cannot relate to. The customer master data business case seems to good fit for the interviewees since it sparked discussion

5.6.2 Organizational

Traditional organizations may have obstacles since communicating about the technology requires significant knowledge sharing for each level of personnel. And as pointed out in interviews, event driven approach requires learning in personal level. So, in this perspective to utilize fully event-driven architecture, one must answer the question how to achieve learning organization and at the same time must be admitted that event-driven architecture is just another architectural approach for some certain problem areas. However, to achieve learning and flatter organization structure serves companies in other architectural or organizational challenges in addition to event-driven architecture.

From IT organization perspective event-driven architecture considers many departments. Operations departments needs to if there is disruptions or abnormalities within the flow of events. Data and integration related departments are mindful about the data schema the integrity of the information and discovery of information. On the other hand, business is considered how fast they build new features. If all these aspects are not somehow addressed, the event driven initiative will not be successful.

5.6.3 Technological

On technological point of view, there are many products that enable event-driven architecture. Apache Kafka is one of most prominent because of its ability to store and distribute data. Kafka position in enterprise architecture can be seen replacement or complement in so called middle-ware layer where in traditionally has been such tools as Oracle service bus and IBM datapower. Kafka and IBM streams were one of topics of discussion in interviews. Kafka and its related products Kinesis are currently in use within the company. Thus, Kafka and is important for the company since there is already knowledge about it. Cloud services such as AWS and Azure are providing Kafka and other event driven components as Software-as-service model. However, since technological innovation is moving rapidly and when some architectural patterns gain more traction, it will further increase innovation. Thus, there should not be too deep commit to certainly technology and there should be an ability to change the underlying technology easily. This means that the event drive architecture implementation must be clarified in each organization.

When moving closer to the source systems such as databases, there is a requirement to get the data from database tables or views. Change data capture (CDC) concept helps

with the issue. The implementation is effectively scanning database tables and views and providing the results to some endpoint such as Datawarehouse or some event driven component. This enables transforming legacy database-based architecture to more real-time architecture.

One of the puzzling issues is how to know who is receiving and producing events and is it business or system event. In case of confidential data such as customer information this is crucial and must be solved because otherwise there is greater risk that confidential data end up to participants who are not entitled to that data. From technological point of view event driven platforms provide some solutions how to authenticate and authorize the participants. The example of this are such as secure-socket layer type of authentication.

During the interviews was discussion how to further develop the event-driven concept. Example of this could be Azure's event grid that is implementation of web hook protocol. Webhook is event-driven implementation on top of http-protocol. However, webhook is not clearly defined or standardized and since it uses http-protocol, it limits its use cases. Http-protocol has more latency compared to for example tcp protocol. On the other hand, Http protocol can be easier to implement thus enables wider adoption of event-driven concept.

During the interviews it came clear that roles of different departments and teams are not clear in context of enterprise architecture. Teams have similar kind of capabilities, but teams might have different kind of tools in use. This creates issue where similar things are done a bit differently in different part of organization without coordination. This causes inefficiency from organizational point of view. In context of event-driven architecture this sort of issues could be for instance different format of data, one team might use JSON and another XML. Solving the issue requires coordination with management layer since it might be that teams themselves cannot solve these issues.

6. EVENT DRIVEN CAPABILITY

Following chapter defines a framework how to build capability that is used in next chapter to build event driven capability and describes capability-based planning. The framework gives a strategic tool for business to plan and follow the execution of the strategy.

6.1 Capability

Capability is organizational ability to achieve a goal. Usually capability is defined in terms that necessary stakeholders can understand. Capability-based planning provides value as a link between business requirements and IT by using high-level language so that both parties can understand. Capability defines something organization does but not how it does.

Capabilities can be divided to ordinary and dynamic capabilities. Ordinary capabilities are needed to for managing normal operational processes such as selling static products and services. While dynamic capabilities are those that make differentiation and make the competitive advantage (Teece, 2014).

Another perspective for the dynamic capabilities is that they are existing assets of company that can be reconfigured to another purpose. Resources alone such as busses, drivers or routes in context of bus company doesn't alone make the dynamic capability but the ability to fast reconfigure routes with different busses because of changes in market contribute to a dynamic capability (O'Reilly & Tushman, 2008).

Capability can be both horizontal and vertical in context of organization. Improving horizontal capability usually means that organization enhances its business functions while vertical improvement focuses on processes (Aldea, Iacob, Van Hillegersberg, Quartel, & Franken, 2015).

Capabilities can be domain specific such as information technology capabilities which can be further divided to personnel, management and infrastructure capabilities. Personnel capability defines how well personnel can use the knowledge and tools in order to achieve strategical goals. Management capabilities focus on utilizing internal and external resources while infrastructure capabilities manage physical assets (AlHarbi;Heavin;& Carton, 2016). Recent agile movement focuses on other hand people, process and technology.

6.2 Capability-based planning

Capability-based planning is an activity that focuses on planning and delivery of business capabilities. On high level the steps consist of mapping, assessing and planning the capabilities. Capability-based planning can be utilized also linking the strategy to implementation planning. Capability-based planning has its root in defence industry but has gained recently interest in other domains as well. It provides an alternative to criticized resource-based planning (Azevedo;Iacob;Almeida;& van Sinderen, 2015).

Capability-based planning can be divided to five stages: identifying capabilities, creating a capability map, link capabilities, gap analysis and planning.

6.2.1 Identify capabilities and building blocks

Existing capabilities should be identified in order to see to which capabilities organization already has. Organizations have also capability building blocks such as strategies, processes that should be identified. Identification and naming the capabilities and capability building blocks gives common vocabulary for the company which eases the development and leading. Once the capabilities and capability building blocks have been identified they should be grouped to different domains or capability areas such as product development, client interaction or similar. There should be only one domain per one capability. The outcome of the stage is a capability map (Figure 18).

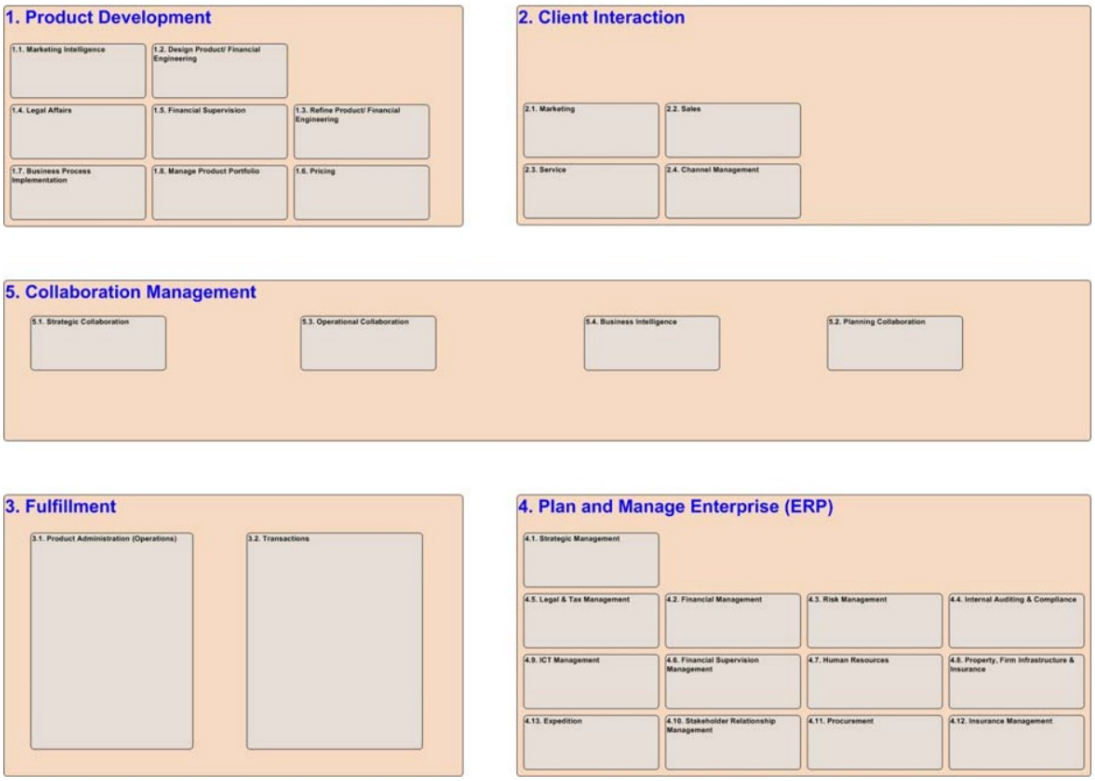


Figure 18 Capability map example (Beimborn, 2005)

6.2.2 Link capabilities to strategy

After identifying the capabilities, the capabilities should be pointed out which contribute to the selected strategy. Capability-based planning can start after company has selected strategy, KPI and other business metrics to steer the development. The outcome of this phase is next version of capability map where capabilities that are linked to strategy are highlighted (Figure 19).

Figure 19 Capability map with linked capabilities (Beimborn, 2005)

6.2.3 Analyse gaps

Analysing gaps is done by comparing current performance levels to the desired performance levels. Quantitative analysis gives concrete and easily trackable goals. Each capability should be attached with metric that is aligned with strategy. Gap analysis can be used also to compare performance to other similar processes. For instance, if there are issues with customer data quality in some certain customer segment, gap analysis can be used for benchmarking data quality in other segment in order to provide realistic goals and give insights on process. The outcome of this phase is a list of to be developed capabilities, their domains, selected metric, current performance level and desired performance level (Table 6).

Table 6 Example of outcomes of gap analysis

Domain	Capability	Metric	Current perfor-	Target perfor-
Client interac-	Customer information manage-	Data quality	50	98
Collaboration	Business intelligence	Predictive model accu-	0	80
Fulfillment	Packet delivery	Predictability	50	98

6.2.4 Plan the increments

At the final stage, the capability's maturity is raised to another level by executing the plan. The execution can be done whatever method businesses are using for example Kanban, scrum, waterfall or else. The execution of plan can be done in parallel in case there is synchronization between the executive functions such as agile release trains. There should be appropriate steering mechanisms for executing the plan in order to achieve the planned results. The outcome of this can be a project plan or items in teams' backlogs.

While planning the increments there should be used metrics to measure the business value and the size of the work. Depending on the maturity of the company, the measurements can be either expert opinions or jointly agreed. The unit for the measurement can be for example Fibonacci sequence of number (1,1,2,4,6,8 and so) or some other logarithmic scale. The focus on estimating the work is not how many days the effort last but rather jointly creating the plan within team (Leffingwell, 2011).

6.3 Capabilities for implementing event-based system

Following chapter focuses to summarize the findings of the interview and literature review by using capability framework described in chapter 5.6.1. While in interviews and literature reviews, the capabilities were not necessarily described but rather activities, policies and such. These are linked in into capabilities in order to create actionable plan.

6.3.1 Identify capability building blocks

As basis of the capability domains is used IT capabilities that focuses on people, technology and processes. Recognizing the actual capabilities of the organization is beyond the scope of the thesis. Thus, is used a framework from literature to illustrate the end results (Table 7). In addition, it is recognized that chosen view on the capabilities is not comprehensive, but it enables demonstration how to adopt capability-driven development framework introduced in chapter 5.6.1.

Table 7 Adopted IT Capability building blocks (Donnellan;Sheridan;& Curry, 2011)

Category	building block	Description
Strategy and planning	Alignment	Define and execute IT strategy to influence and align to business objectives
	Objectives	Define and agree objectives for IT
Process management	Operations and lifecycle	Source, operate and dispose IT systems to deliver objectives
	IT-enabled business process	Create provisions for IT systems that enable improved outcomes across the extended enter-
	Performance and reporting	Report and demonstrate progress against IT-specific and IT enabled objectives, within the IT business and across the extended enterprise
People and culture	Adoption	Embed principles across IT and the extended enterprise
	Language	Define, communicate and use common language across IT, other business units and partners to leverage a common understanding
Governance	External compliance	Contribute to industry best practices
	Corporate policies	Enable and demonstrate compliance with IT and business legislation and regulation. Require accountability roles and decision making across IT and the enterprise.

6.3.2 Link capability blocks

After recognizing capabilities, critical capabilities that contribute to the strategy should be highlighted. Another way would be to analyze the stage where company is currently by analyzing each category. In Figure 20 Figure 20 Event-driven capability building blocksis recognized a map of event-driven architecture capability building blocks and

after following chapter is described each capability building block and how they contribute to event-driven architecture

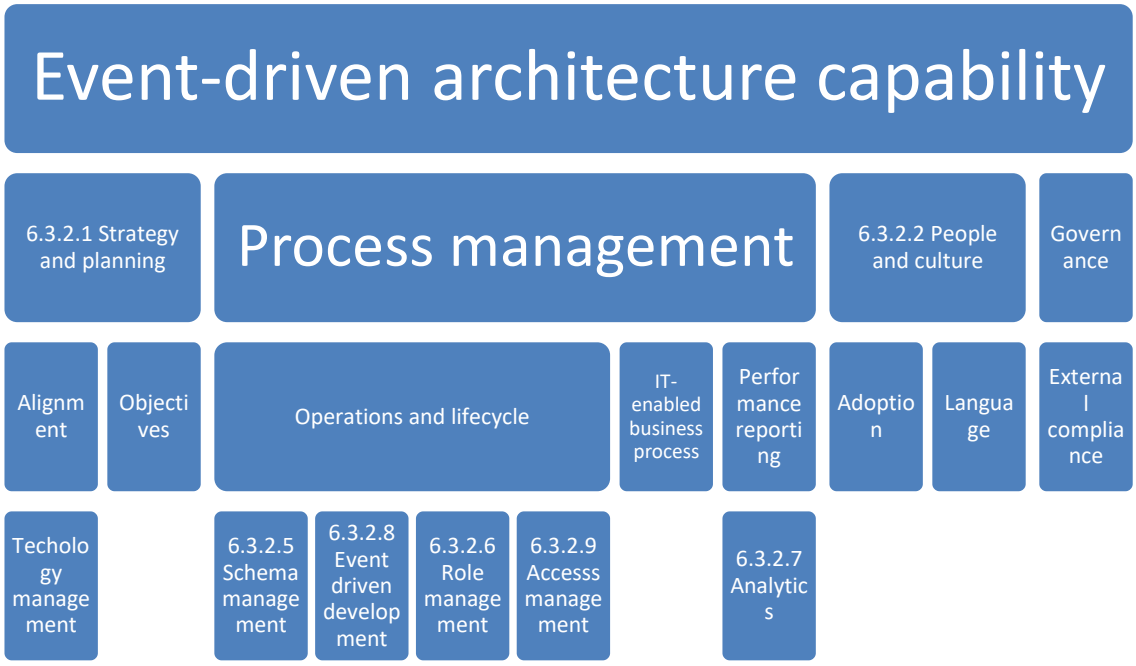


Figure 20 Event-driven capability building blocks

6.3.2.1 Strategy and planning

Event-driven architecture provides simultaneity and more real timeliness these topics should emphasize and be aligned with other departments. The commercial side of company might not see event-driven architecture as important topic since it doesn't have direct relationship with cash flow. However, event-driven architecture has direct correlation with the user experience since it matters how fast user receives information and how much there is capabilities to process information in parallel.

6.3.2.2 People and culture

Since it is recognized that event-driven architecture is in its early stages within company, People and culture should be emphasized. This is under assumption that there is alignment with IT and business objectives and since event-driven architecture can contribute to many business cases, it is under assumption that alignment is in place. The conversation with interest group should be held in their terms. For example, the emphasis with the software developers should be different than with development managers. To have event-driven architecture a compelling language should be developed around the subject. There should be something familiar and yet something new to gain attraction.

6.3.2.3 Governance

There should be a way to have audit who are consuming and producing events. In addition, there should be a mechanism to know what sort of business or system events are generated and consumed and what are their contents.

6.3.2.4 Technology management

Ability to transit from technology to another with minimal disruptions in operations. Such transit would be to change the event driven middleware software. Event producers and consumers should have also technical capabilities to produce events. Since event-driven architecture is developing field, there should be also understanding how the architecture could be further enhanced with such technologies as event hub or other emerging technologies.

6.3.2.5 Schema management

Ability to evolve the information model to serve as many as possible cases. There should be agreed rules for creating event models, make new versions and retire them. The rules could be in form of written document and/or some component that validates event model automatically according to formal rules.

6.3.2.6 Role management

Ability recognize who should produce, consume and deliver events. There should be also clear roles between departments. For instance, who has operational responsibilities of event driven platform and who develops the platform.

6.3.2.7 Analytics

Ability to tell how many schemas there is, how many of those schemas are reused. How many consumers and producers there is? The analytics is used for instance measuring the success of event-driven architecture initiative.

6.3.2.8 Event driven development

Ability to develop events and systems that produces and consumes event. In addition, there is development for the necessary architecture around the concept.

6.3.2.9 Access management

Ability to control who can see which events and each event consumer and producer is known. There is ability to revoke access for consumers and producers in case misbehaving.

6.3.3 Analyse gaps

In this chapter is analyzed is capability and its state and envisioned a target stage. In Figure 21 is described an event-driven architecture heat map. The color on capability name indicates the criticality of the capability. The criticality is based on the interviews and literature review.

- Red indicates that capability is more critical than those that are marked with yellow or blue.

- Yellow indicates that the capability is more critical than those that are marked with blue.
- Blue indicates that capability doesn't contribute significantly this time or in this case

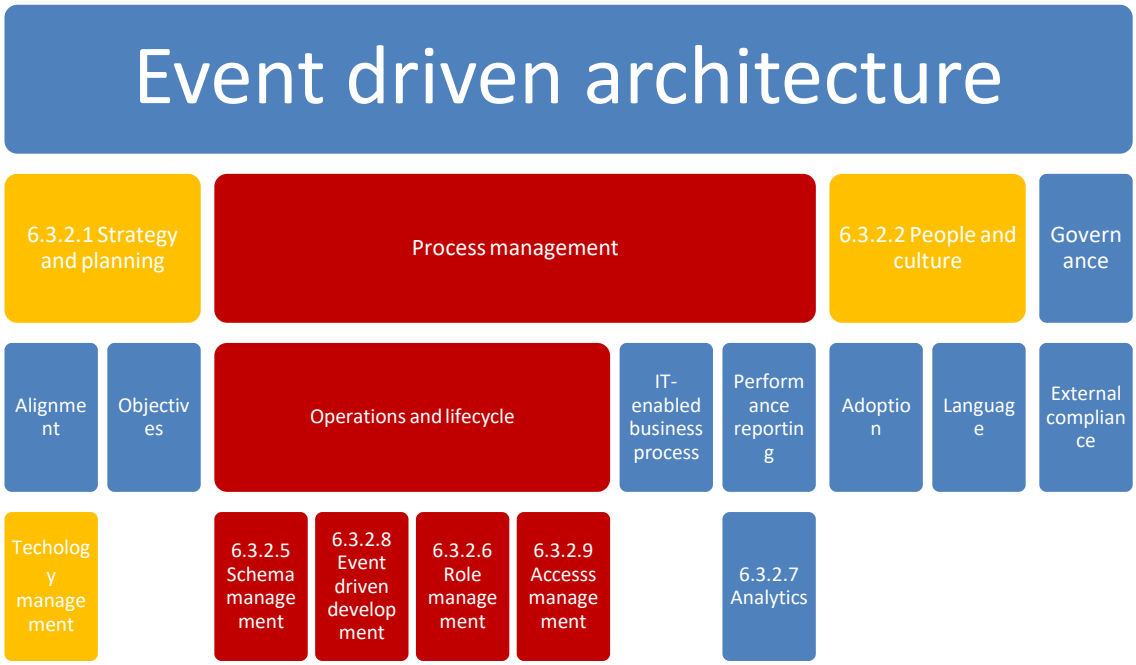


Figure 21 Event driven heat map

6.3.3.1 Technology management

Currently there is no centralized function or process to manage IT technology. Each department and platform owner manage their own system. In target stage there should be a described a general process and responsibilities should be clear. There should be mutual understanding and visibility how the technology evolves. After the general technology management principles are decided, specialization of event driven technology management can be formed.

6.3.3.2 Schema management

Currently there is some local rules for API schema management. In target stage these rules should be combined and enforced. In target schemas are shared and reused. Schemas evolve with collaborative method.

6.3.3.3 Role management

Since event-driven architecture is relatively new term, there have not been much discussions of the roles. In target stage the roles have been identified and each participant understand its role.

6.3.3.4 Analytics

Currently there is a registry of API's and their consumers and producers. In target stage same kind of ideology is reused for event driven management. Analytics should provide

both operational metrics and business metrics. On operational level there should be tracked such things as abnormalities while business metrics could provide information on reuse or things that are relevant to strategy.

6.3.3.5 Event driven development

Currently there is ability to develop event driven software. However, since the concept and enabling features are not in use, the development features are not full used. Ability to develop events and systems that produces and consumes event. In addition, there is development for the necessary architecture around the concept

6.3.3.6 Access management

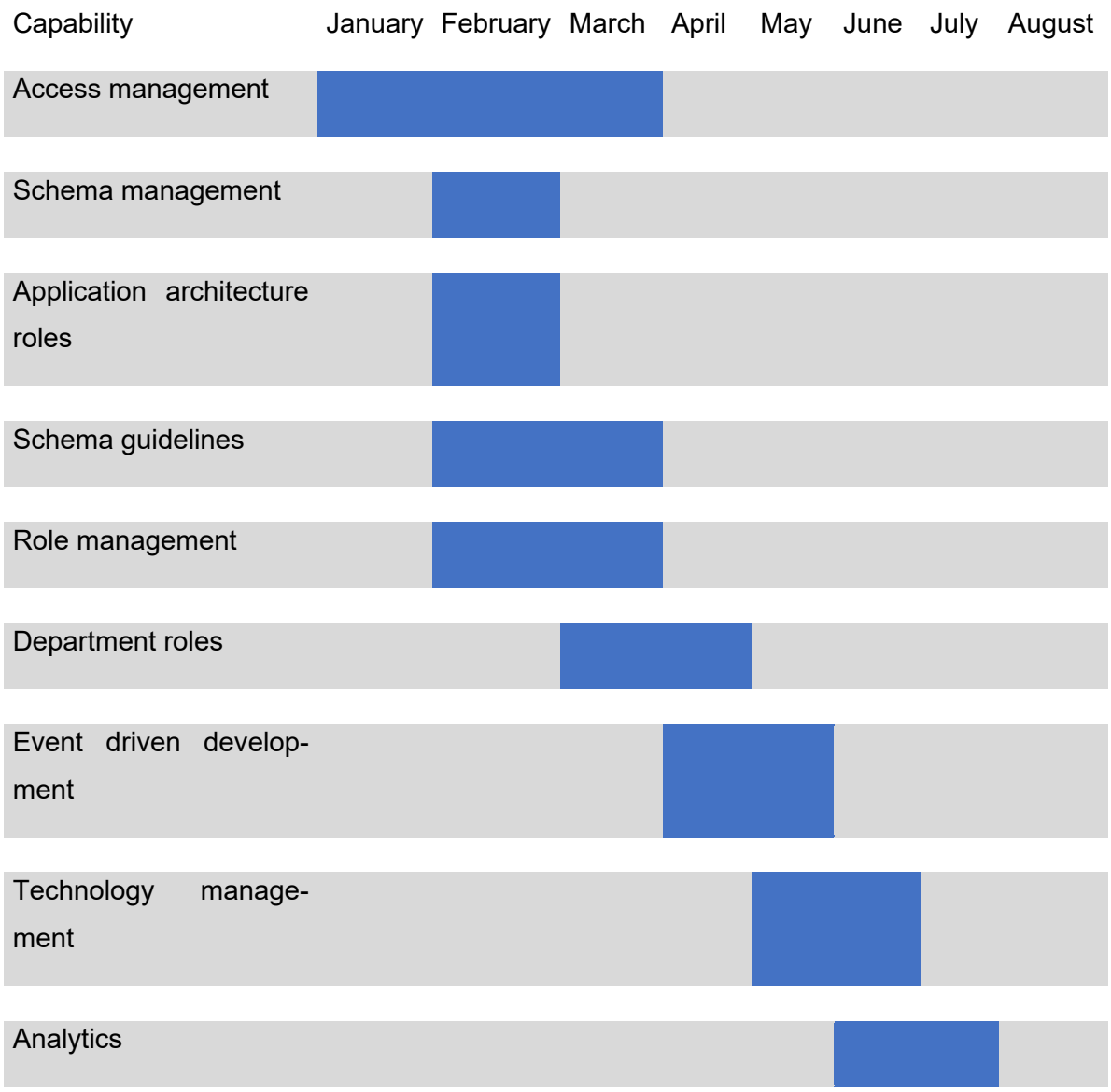
Currently the access management requires additional components. In target stage each existing and upcoming consumer and producers is utilizing access management.

6.3.4 Plan the increments

On last stage is created plan how to achieve target stage and what are tasks dependencies between each other's and their business value. The detailed definition of the target stage is not in the scope of this thesis, but this gives rather a framework how start building the capability. The business value is subjective in this case and reflects the values of writer in this case.

Capability	Business value	Reason
Access management	9	Security and hard to change later
Schema management	8	Start for common evolvable rules
Application architecture roles	7	Clarification
Schema guidelines	6	Start for common evolvable rules
Role management	5	Clarification
Department roles	4	Process for changing the roles must formalized
Event driven development	3	Process must be formalized
Technology management	2	Process must be formalized
Analytics	1	Business and operational metrics must be agreed

Once the business values are defined a concrete plan should be formed where is planned what happens, when and what are the dependencies.



6.3.5 Summary

In this chapter was presented a roadmap how to build an event driven capability to Telecommunications organization. The roadmap is based on literature review and to empirical interviews.

7. SUMMARY

In following chapter is summarized the findings of this thesis.

7.1 Main results

Thesis addressed three questions and respective responses are presented below:

1) What event-driven architecture is

Event-driven architecture is pattern to create software and systems that are loosely coupled. Loosely coupled means that software and system are built in manner that for example when a participant in the system, such as software or server, experiences malfunction other participants can function normally. Event-driven architecture focuses how to coordinate designing of system or systems where requirement is to transfer data as fast as possible to trigger some further activity and where it is required to combine multiple events to create new information.

Event is a significant change of state and event can be seen in the different context to have different meaning. In this thesis is discussed about business events and system events. Business events means when customer makes order from eCommerce platform for instance new cellular phone or changes his/her information in a self-service portal. In these case order and change his/her information produces a business event. The business event is transferred to eCommerce platform System events are events that happens between systems such as logging systems activity.

To sum up: Event-driven architecture is approach for creating loosely coupled applications and that enables simultaneity in business processes. Event-driven architecture can evolve to more real-time data transfer and it completes batch-type of data transfer by providing alternative approach.

2) How to design, implement event-driven architecture through business cases

Designing event-driven architecture requires several things to consider on technical and organizational aspects. Event-driven architecture relies heavily on principles of service-oriented architecture thus grasping principles of SOA is requirement for organization to implement event-driven architecture. While implementing event-driven architecture to organization should be considered both technical and organizational aspects. On organizational level the business benefits should be evaluated, and event-driven architecture

should be a part of organizational objectives. Not necessary as a separate topic but as an accepted principle. People and culture should be considered as well to create common vocabulary on the subject. Managing event-driven architecture development process requires own focus for operating and measuring to meet the objectives.

Event driven business cases usually have characteristics where it is necessary to create quickly new applications or modify existing and there's a need to make direct event flow between services that have been done with different technologies and with different internal structure. Such example would be to receive an order event to eCommerce platform and further send it to messaging component which delivers the order confirmation to customer.

To sum up, for implementing event-driven architecture was provided a capability driven approach that is based on the literature review and interviewing subject experts on DNA.

3) What kind of obstacles and possibilities, both technological and organizational, a company may have?

Third question aims to find insights from the company on how to enable event-driven architecture. DNA has established IT governance and organization that supports other organization and drives technological initiatives with cooperation of business. This is an enabler and possibility on organization's event-driven architecture journey.

On technical point of view organization have established integration and data practice so event-driven architecture is an additional pattern for these practices to learn and further develop. On the other hand, organization structure is perceived complex and follows agile principles and this might cause an obstacle on transfer knowledge on the organization.

To sum up, DNA has already knowledge and technological capabilities to achieve basic level on event-driven architecture. More advanced level requires steps that are described in chapter 6

7.2 Practical implications

DNA could benefit from event-driven architecture. Suggestions based on this thesis to DNA for implement event-driven architecture are following:

- Test event-driven architecture with small business case
- Follow event driven capability roadmap as depicted in chapter 6 to further enhance the event driven capability

- Raise organizational awareness

DNA already have technical capabilities for implementing event-driven architecture. However, event-driven architecture is not coordinated strongly now. Event-driven architecture might benefit if the concept is bundled with other topical concepts such as microservice or API architecture since these concepts overlap and benefit from each other's.

Organizational awareness of obstacles and benefits and topical use cases that would benefit from event driven approach. Thus, the concept would have a validation that it is viable solution DNA.

7.3 Limitations

Like any work, this thesis has limitations such limitations on data collection methods, time, sample size and related to the research environment.

The thesis was done within a year of period while the author was working fulltime. There was limited amount of time daily to spend on thesis. The thesis had one author. Having multiple authors might improve the results since the work could be reflected with others.

Data collection was done by interviews and reading the articles and books that were available for the university's library. The results of the literature review are subject to authors interpretation. Based on readers background and experience on subject, the literature review can have different results.

The interviews can have issues on objectivity since the results depend on interviewer's interpretation. The personal relationships between Interviewer and interviewee has some contribution to the results and there has been discussion on the topic before and after the interviews which effects on the results.

The interviews consisted of 3 people from the same department with similar background, so the sample size was small. To widen the perspective more interviews could be conducted with wider target group and more heterogeneous background.

Research was conducted only on one company thus the results might be difficult to generalise and implement in some other environment. These issues were mitigated by having literature review and thus gathering background information on the topic.

7.4 Conclusions and future research

Event-driven architecture provides many benefits for the companies. Recent emergent technological changes such as microservices, big data and real-time analytics combined

with business requirements such as faster time-to-market and high availability, makes event-driven architecture topical.

To gain the enterprise-wide adoption, there must be focus on education on such topics as service-oriented architecture and API-first thinking. Different individuals and teams are on different level on their journey towards event-driven architecture journey and the question is that how to support them so that each enough support. If thinking from team point of view, for example IT integration team and teams that focus more business-sides of things, need different kind of support. IT integration team might need more hands-on facilitation for example on how to agree on general rules regarding event driven API's while business teams need to be aware that such rules exists.

During this thesis a baseline of knowledge on event-driven architecture was introduced to the company through in form of internal blog, formal and informal discussions and giving implementation guidelines on such cases that was thought that could benefit from event-driven architecture. The event-driven architecture might seem as complex and laborious for some business users with minimal benefit. For the company that this thesis was done, event-driven architecture provides value by enabling near real-time data transfer.

Event-driven architecture must be translated to business language to have better impact. In the end event-driven architecture is just another architectural pattern for organize to adapt which answers to some topical business problems. Since business drivers changes and new technologies emerge that improve current architecture, the question is that should organization invest event-driven architecture or some bigger concept. The event-driven architecture itself affects to only small part of the organization so at the organization level, it might be more beneficial to invest on topics such continuous learning and system thinking. However, event-driven architecture in context of IT architecture is important concept and can foster adoption of other related things such as API first thinking and more timeliness message distribution in company.

Continuous learning or lifelong learning are concepts that are topical in current corporate context. These topics emphasizes thinking that learning doesn't happen only in institutions such as schools or facilities dedicated to learning but rather in normal everyday life. Systems thinking is a theory that dates to the ancient times and have been further developed by in the fields of philosophy, mathematics, ecology, engineering and psychology.

From organizational point of view, the ideal structure to support event-driven architecture should be studied. Current event-driven architecture literature emphasizes technology

and on the other hand event-driven architecture tries to solve business problems. Additionally, event-driven architecture requires at least two systems to be integrated and it gets most of its potential when a large amount of systems is using it and sends messages to each other. Recent discourse on agile emphasizes multi-disciplined team. Thus, in context of event-driven architecture, team should at least consist on technical and business-oriented personnel who both can understand the concept and implement it.

From organizational point of view the most important questions is how to enable learning in order to event-driven architecture gain more traction. Some solutions are to make organization flatter. However, that makes some other problems to be solved and requires careful planning especially in bigger organizations.

Event-driven architecture needs to prove itself in corporate context and it needs support to succeed. Thus, actual implementation of such architecture needs to accomplish in order to gain feedback and to learn in specific context. The existing architectural components and teams focusing on the components, helps for testing the hypothesis.

8. REFERENCES

Abdou, H.;& Pointon, J. (2011). Credit scoring, statistical techniques and evaluation criteria: a review of the literature. *Intelligent Systems in Accounting, Finance & Management 18 (2-3)*, ss. 59-88.

Aldea, A.;Iacob, M. E.;Van Hillegersberg, J.;Quartel, D.;& Franken, H. (2015). Capability-based Planning with ArchiMate Linking Motivation to Implementation . *In Proceedings of the 17th International Conference on Enterprise Information Systems (ICEIS-2015)* (ss. 352-359). SCITEPRESS.

AlHarbi, A.;Heavin, C.;& Carton, F. (2016). Understanding the Characteristics of IT Capability in Delivering a Customer-Focused Strategy: The case of Saudi Bank. *25TH INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS DEVELOPMENT (ISD2016 POLAND)*, (ss. 197-211).

Azevedo, C.;Iacob, M.-E.;Almeida, J. P.;& van Sinderen, M. (2015). Modeling resources and capabilities in enterprise architecture:A well-founded ontology-based proposal for ArchiMate. *Information Systems*, 235-261.

Becker, J.;Matzner, M.;Müller, O.;& Walter, M. (2011). A Review of Event Formats as Enablers of Event-Driven. *Proceedings of 5th International Workshop on event-driven Business Process Management*, (ss. 433-445).

Beimborn, D. &. (2005). Capability-oriented Modeling of the Firm. *Conference: 2005 IPSI Conference.* Amalfi/Italy.

Bente, S.;Bombosch, U.;& Langade, S. (2012). *Collaborative Enterprise architecture.* Elsevier.

Bente, S.;Bombosch, U.;& Langade, S. (2012). *Collaborative Enterprise Architecture.* Morgan Kaufmann.

Bernstein, P. A.;& Haas, L. (2008). Information integration in the Enterprise. *Communications of the ACM*, 75.

Biehl, M. (2017). *Webhooks – Events for RESTful APIs.* API University.

Bughin, J. (2016). Reaping the benefits of big data in telecom. *Journal of Big Data* .

Cearley, D.;Natis, Y.;Walker, M.;& Burke, B. (2018). *Top 10 Strategic Technology Trends for 2018: Event-Driven Mode.* Gartner.

Cloutier, R.;Muller, G.;Verma, D.;Nilchiani, R.;Hole, E.;& Bone, M. (2008). *The Concept of Reference Architectures.* Wiley.

Cohen, S.;& Roussel, J. (2005). *Strategic Supply Chain Management.* McGraw-Hill.

Cuesta, C. E.;Navarro, E.;& Zdun, U. (2016). Synergies of System-of-Systems and Microservices. *SiSoS@ECSA '16 Proceedings of the International Colloquium on Software-intensive Systems-of-Systems at 10th European Conference on Software Architecture.* New York, NY, USA: ACM .

Cummins, F. A. (2009). *Building the agile enterprise.* Elsevier.

Dahl, O. (2002). *Enterprise Application Integration - Applying Patterns to the Process of.* Växjö: Växjö University.

Donnellan, B.;Sheridan, C.;& Curry, E. (January/February 2011). A Capability Maturity Framework for Sustainable Information and Communication Technology. *IT Professional Magazine*, ss. 33-40.

Dunkel, J.;Fernández, A.;Ortiz, R.;& Ossowski, S. (2011). Event-driven architecture for decision support in traffic management systems. *Expert Systems With Applications 06/2011, Volume 38, Issue 6*, 6530 - 6539.

El-Sheikh, E.;Zimmermann, A.;& Jain, L. C. (2016). Evolution of Service-Oriented and Enterprise Architectures: An Introduction. Teoksessa E. El-Sheikh;A. Zimmermann;& L. C. Jain, *Emerging Trends in the Evolution of Service-Oriented and Enterprise Architectures* (ss. 1-3). Springer.

Etzion, O. (2005). Towards an Event-Driven Architecture: An Infrastructure for Event Processing Position Paper. *Rules and Rule Markup Languages for the Semantic Web* (ss. 1-7). Berlin, Heidelberg: Springer.

Garlan, D.;Allen, R.;& Ockerbloom, J. (2009). Architectural Mismatch:Why Reuse Is Still So Hard. *IEEE Software*, 66-69.

Ghalsasi, S. Y. (2009). Critical success factors for event driven service oriented architecture. *ICIS '09 Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, (ss. 1441-1446). Seoul, Korea.

Goerziga, D.;& Bauernhansla, T. (2017). Enterprise architectures for the digital transformation in small and medium-sized enterprises. *11th CIRP Conference on Intelligent Computation in Manufacturing Engineering - CIRP ICME '17*, 543.

Hristov, D.;Hummel, O.;Mahmudul, H.;& Janjic, W. (2012). Structuring Software Reusability Metrics. *ICSEA 2012 : The Seventh International Conference on Software Engineering Advances*, (ss. 421-429).

Klusman, M. (20. 12 2019). *Master Thesis Computing Science: Event Driven Architecture in software development projects.* Noudettu osoitteesta Radbound universiteit: https://www.ru.nl/publish/pages/769526/maxime_klusman.pdf

Kotusev, S. (April 2018). *Fake and real tools for enterprise architecture*. Noudettu osoitteesta British Computer Society: https://www.bcs.org/content/conWebDoc/59399

Krafzig, D.;Banke, K.;& Slama, D. (2005). Enterprise SOA: Service-oriented Architecture Best Practices. Prentice Hall Professional.

Kreps, J.;Narkhede, N.;& Rao., J. (2011). Kafka: a Distributed Messaging System for Log Processing. *Proceedings of the NetDB*, 1-7.

Lankhorst, M.;Proper, H.;& Jonkers, H. (2009). The Architecture of the ArchiMate Language. *BPMDS 2009, EMMSAD 2009* (ss. 367-368). Berlin, Heidelberg: Springer.

Lano, K. (2017). Agile Model-Based Development Using UML-RSDS. Teoksessa K. Lano, *Agile Model-Based Development Using UML-RSDS* (ss. 6-7). CRC Press.

Leffingwell, D. (2011). Agile Software Requirements. Teoksessa D. Leffingwell, *Agile Software Requirements* (s. 247). Boston: Pearson Education, Inc.

Linthicum, D. (2000). *Enterprise Application Integration.* Chicago: Addison Wesley.

Luckham, D. C. (2011). *Event Processing for Business : Organizing the Real-Time Enterprise.* John Wiley & Sons, Incorporated.

Mankovskii, S. (2009). *Encyclopedia of Database Systems.* Boston: Springer.

Michelson, B. (6. February 2011). *5th Anniversary Edition – Event-Driven Architecture Overview.* Noudettu osoitteesta Elemental Links: http://elementallinks.com/el-reports/EventDrivenArchitectureOverview_ElementalLinks_Feb2011.pdf

Moilanen, J. (2018). Tyyli ennen kaikkea - API Design guide. Teoksessa J. Moilanen;M. Niinioja;M. Seppänen;& M. Honkanen, *API-Talous 101* (ss. 145-154). Alma Talent Oy.

Moogk, D. (4. 7 2019). *Minimum Viable Product and the Importance of Experimentation in Technology Startups*. Noudettu osoitteesta Technology innovation management review: https://timreview.ca/article/535

Murphy, L.;Alliyu, T.;Andrew, M.;Kery, M. B.;& Myers, B. (2017). Preliminary Analysis of REST API Style Guidelines. *PLATEAU'17.* Vancouver.

Nandico, O. F. (2016). A Framework to Support Digital Transformation. Teoksessa E. El-Sheikh;A. Zimmermann;& L. C. Jain, *Emerging Trends in the Evolution of Service-Oriented and Enterprise Architectures* (s. 120).

O'Connor, R.;Elger, P.;& Clarke, P. (2016). Continuous software engineering—A microservices architecture perspective. *Continuous software engineering—A microservices architecture perspective. Journal of Software: Evolution and Process*.

O'Reilly, C.;& Tushman, M. (2008). Ambidexterity as a dynamic capability:Resolving the innovator's dilemma. *Research in Organizational Behavior 28*, 185–206.

Overbeek, S.;Janssen, M.;& van Bommel, P. (2012). Designing, formalizing, and evaluating a flexible architecture for integrated service delivery: combining event-driven and service-oriented architectures. *Soca*, ss. 167-188.

Paulheim, H.;& Probst, F. (2010). Application integration on the user interface level:. *Data & Knowledge Engineering*, 1104-1105.

Phillips, L.;Yochem, A.;Taylor, H.;& Martinez, F. (2009). *Event-Driven Architecture: How SOA Enables the Real-Time Enterprise.* Addison-Wesley Professional.

Pienwittayasakul, C.;& Liu, Y. (2014). Comparative Study on Service-Oriented Architecture and Event-Driven Architecture. *Proceedings of the International conference on Computing Technology and Information Management.* Dubai, UAE.

Richards, M. (2015). *Software Architecture Patterns.* O'Reilly Media, Inc.

Ross, J. W.;Weill, P.;& Robertson, D. (2006). *Enterprise architecture as strategy: Creating a foundation for business execution.* Harvard Business Press.

Ruusuvuori, J.;& Tiittula, L. (2005). *Haastattelu - Tutkimus, tilanteet ja vuorovaikutus.* Osuuskunta Vastapaino.

Rytter, M.;& Jørgensen, B. N. (2010). Independently Extensibile Contexts. *Software Architecture: 4th European Conference , ECSA 2010, Copenhagen, Denmark, August 23-26, 2010, Proceedings*, (s. 327).

Sauder, M.;& Espeland, W. N. (2009). The Discipline of Rankings: Tight Coupling and Organizational Change. *American Sociological Review 74*, 63-82.

Smith, H. A.;& McKeen, J. D. (7 2008). Developments in Practice XXX: Master Data Management: Salvation Or Snake Oil? *Communications of the Association for Information Systems*, ss. 63-72.

Smith, P. (8. March 2009). *Wikimedia commons.* Noudettu osoitteesta File:Waterfall model (1).svg: https://commons.wikimedia.org/wiki/File:Waterfall_model_(1).svg

Teece, D. (2014). A dynamic capabilities-based entrepreneurial theory of the multinational enterprise. *Journal of International Business Studies volume 45*, 8–37.

TmForum. (16. 08 2019). *About us*. Noudettu osoitteesta TmForum: https://www.tmforum.org/about-tm-forum/

Zalando. (2019). *Zalando RESTful API and Event Scheme Guidelines*. Noudettu osoitteesta Zalando: https://opensource.zalando.com/restful-api-guidelines/