Large Software Implementation Project: A study of software development and project management literature

UNIVERSITY OF TURKU
Department of Future Technologies
Computer Science
Maija Väisänen
September 2017

TURUN YLIOPISTO
Tulevaisuuden teknologioiden laitos

MAIJA VÄISÄNEN

Suuret kehitysprojektit: Kirjallisuuskatsaus ohjelmistokehityksen ja projektinhallinnan kirjallisuuteen

Pro gradu -tutkielma, s.56
Tietojenkäsittelytiede
Syyskuu 2017

Tämä tutkielma käsittelee isojen ohjelmistojen implementointiprojekteja, joissa kehitystyö tehdään valmiin tuotteen päälle. Tällaiset projektit ovat varsin tavallisia ohjelmistoteollisuudessa. Tutkielma pyrkii vastaamaan tutkimuskysymykseen: Kuinka toimittaa suuria räätälöityjä ohjelmistoprojekteja?

Suuret ohjelmistohankkeet tehdään pitkän aikavälin investointeina ja yleensä kehitystyö vie huomattavasti pidempää kuin pienissä hankkeissa. Suuret hankkeet edellyttävät projektinhallinnan erityisosaamista, ylemmän johdon tukea, investointeja, sekä IT:n ja liiketoiminnan yhdistämistä. Isot hankkeet ovat yleensä myös monimutkaisia ja niissä on paljon riippuvuuksia. Tutkielma pyrkii myös selittämään mitä ongelmia ja riippuvuuksia projekteissa yleensä ilmaantuu, sekä kuinka vanhat järjestelmät ja datan migraatio vaikuttavat projektin kulkuun. Tutkielmassa käydään läpi IT-järjestelmien eri aikakaudet ja miksi IT järjestelmiä pitää kehittää. Tutkielma esittelee myös lyhyesti vesiputousmallin ja ketterän kehityksen perusteet ja taustan. Scrum ja SAFe mallit esitellään lyhyesti esimerkkinä ketterästä kehityksestä.

Avainsanat: Perinnejärjestelmät, datamigraatio, ohjelmistokehitys, projektinhallinta, COTS, ketterä kehitys, vesiputosmalli

UNIVERSITY OF TURKU
Department of Future Technologies

MAIJA VÄISÄNEN

Large Software Implementation Project: A study of software development and project management literature

Master's Thesis, 56 p.
Computer Science
September 2017

This study focuses on large scale software delivery, where development is done on top of an exciting system or parallel to it. This thesis aims to answer to the question: How to implement a large scale custom solution?

Large scale projects take longer than smaller projects to implement and usually they are done in more than in one release. The application's life-cycle is also planned to last up to decades. Large projects also need special project management skills, executive support, internal investments, strategical vision as well as alignment between IT and business. Large projects are usually complex and have several dependencies. This study also explains what issues projects usually have and what are the constrains of legacy systems and data migration. Different eras of IT systems are also presented as well as reasons why companies should invest to IT solutions. Waterfall model and Agile methodology fundamentals and background are presented shortly. From Agile methodology Scrum and SAFe frameworks are presented as examples.

Keywords: Legacy system, Data Migration, Software implementation, project management, COTS, Agile development, Waterfall

Table of Contents

1. Introduction

Executive level managers are often frustrated by companies' information systems because they have difficulties getting information of how business is running. Due to this, they cannot analyze bottlenecks and solve problems beforehand (Cummins 2002, pg. 3). According to Capers Jones, software costs are often heavily impacted by poor quality, marginal security and other chronic issues. Therefore, software projects are difficult to control. Software cost also forms major part of the corporate overall cost structure (Jones 2010, pg. 1.)

When heavy cost structure and executive level frustration are put together, it is no wonder that IT investments are often labelled as failures (c.f. Standish Group, 2013). Although business is becoming more and more dependent on IT-systems as well as infrastructures, and the field is constantly re-inventing itself with new innovations like *Internet of Things* (IoT), *Cognitive systems* and *Big Data* there is still no common agreement how software development project should be managed.

Software projects vary a lot in complexity, size, time, value, scope, service level agreements as well as contractual obligations among other issues. Development and management methods that work for a small project might not scale up to big projects – and *vice versa*. For example, a large project might require more detailed documentation, because otherwise, after some time, it is impossible to track down what has been agreed upon. However, at the same time a small project with limited scope can be done with minimal documentation.

This study focuses on large scale software delivery, where development and new features are done on top of an existing system or parallel to it. In large development projects, there are several specific project attributes, which make the research of the large development projects interesting. The most distinct feature is usually the length of a project. Development of a large scale software applications, can take

up to five years. In addition, deployment and customization can add another year to the project's schedule.

After deployment, the application's life cycle is remarkably long, up to 25 years or even more. Over the long lifecycle, various enhancements and defect repairs will occur, including restructuring of the application, changing file formats, and perhaps even converting the source code to a newer language. (Jones 2010, pg. 39.) If work such as this is carried in iterations; five years of development would require a minimum of 30 sprints, and each sprint would only last 2 weeks. If there are parallel and longer sprints, the development effort is even larger. Waterfall method is still used, but also Agile software development methods are raising popularity among the practitioners (11th annual State of Agile survey pg. 2). Scaled Agile Framework (SAFe) has recently emerged to answer the needs of larger software projects and IT portfolio management, that wish to work in Agile ways but must answer to corporate regulations.

Installation of a software to personal computer via CD or DVD, or using it through Software as Service (SaaS) model, where applications are run in service providers' server and no actual installation to user's computer occurs, is remarkably different compared to the deployment and installation of a large application. Deployment or installation for large mainframe applications – like a telephone switching system, large mainframe operating system or an enterprise resource planning packages – can take even more than year. They usually require significant amount of customization to match the local technical and business needs. In addition, training user the use of large application might require classroom teaching, customized documentation, maintenance personnel, customer support, and other ancillary staff. Often, large applications have different kinds of users in various parts of the organization; therefore, also a variety of user manuals and classroom training is needed. More than a year might pass from the day when software is delivered to

the day when large scale usage by all users has actualized. (Jones 2010, pg. 12-14.)

The demand for IT systems is increasing annually; IT has changed from a supporting function to a strategic investment. This phenomenon is visible in several industries. From engineering point of view IT systems consist of technical structure from hardware to software components. More holistic view of IT systems includes also socio-technical aspect, which means that IT systems are not only technical components, but they involve interacting with human and organizational elements as well. (Sommerville et al. 2012, pg. 73.) In this study IT systems are considered from socio-technical aspect.

Use of Agile methods have become part of mainstream development toolkit. According to the 11th annual State of Agile survey *"enterprise agility is increasing throughout organizations and across almost all industries at an accelerated rate." (* 11th annual State of Agile survey pg. 2 *).* This has also changed the scenery not only for IT vendors, but also for clients as well. Clients are demanding fast response to business needs and fast development. Fast adaption and development is easy to apply for example to mobile application development, where expected life cycle of the product is shorter, which it is rarely possible for complex projects.

Enterprise software has different features than software behind electronic games, mobile application and websites. Enterprise software needs to adapt to the management pattern, business process and enterprise culture of the target organization. Groups of users are specific and system is easily integrated with other systems and platforms.

Following issues often emerge during software's life cycle (Yang & Jiang 2011, pg. 1):

1. Unclear and frequently changing requirements and objectives. This usually happens when the project group is unable to capture clear, detailed and specific requirements from users. The project's scope and quality measurements are set by the project group, which easily leads to the situation where the complete enterprise application is unable to reflect the business functions and workflow.
2. Cross-platform and multi-system structure. All applications must interact with existing software and hardware. This is notably challenging if the existing systems are heterogeneous.
3. High risk and uncertainty. There are multiple uncontrollable factors that might lead to failures in software projects. High uncertainty comes from unexpected risk, which, if realized during development, usually has an impact on the ability to deliver in budget and deadline.

This thesis aims to answer the following research question: How to implement a large scale custom solution? What are different methods used for developing large scale software solutions.

The rest of this thesis is structured as follows. The next section takes a look why software projects are so complex and on the reasons why a corporation should invest into IT. Section 3 presents project management literature and the fourth section different software development methods. The thesis is concluded in Section 5.

2. Complexity of IT

This chapter presents why IT systems are so complex by nature and also why companies should invest to them. To understand the current IT systems that are in use or under development, one must understand that IT-systems have growth exponentially an era after another. Huge growth has generated massive amount of code, legacy systems and not to mentioned data. Regardless of the amount of code and systems, not all components are reusable and can response current and future business requirements. Aligning business and IT is the motivation that companies have for renewing their IT systems.

2.1. Motivation

If renewing or creating a new IT system is so complex and risky, then why should companies invest to them? Companies seek to streamline their processes and to find growth potential. Many industries are moving towards more service-oriented solutions, and IT systems are shifting from being a part of support functions to being a part of service offerings. When companies are digitalizing their business, they naturally have higher demand for IT functions, which usually means updating current IT portfolio and investing into new IT systems, hardware or other resources.

Studies from leading companies have shown that electronically based business initiatives, like virtual doctor, integrated CRM (customer relationship management) and ERP (enterprise resource planning) or real-time data analysis, require high capability from the IT-infrastructure. Before a company can leverage their IT and speed up the go-to market time, they must have balance between business application development and infrastructure investments. Usually infrastructure needs to be upgraded prior to application development, as parallel development easily leads to fragmentation. (Weill et al. 2002, pg. 64.) Modern IT

has tremendous amount of calculating power compared to old hardware, but still even more investment must be made for infrastructure and hardware, to maintain a massive backend and data repositories.

Research by Weill and Aral (2006) found four different IT investments categories: transactional, informational, strategic and infrastructure. Transactional investments are made for cutting cost or increasing amount of data processed through same workflow. Informational investments serve the purpose of providing information to accounting, reporting, compliance, communication or analysis. Strategic investments are made for gaining competitive advantage, by new product, service or business development and entering new markets. IT infrastructure investments are shared applications and services. (Weill & Aral 2006, pg. 40.)

Transactional, informational and infrastructure investments are the reasons why companies traditionally invest on IT, they also present IT as support function. In modern society, IT has enabled and forced some very traditional industries to transform their way of making business and offering services. For example, services like Airbnb has entered markets that big hotel chains used to dominate by enabling people to rent their houses for a short term and to find a place for rent. Airbnb has leveraged mobile platforms as well as websites for finding and connecting customers, and their whole services model is not only supported by IT but enabled by it. Another example would be services that are used for money transaction. Not only banks and credit providers are offering money transaction or are handling clients' money. There is a completely new market for offerings like PayPal, which focus solely on online transactions.

Traditional IT vendors like IBM and Microsoft are facing competition from industries that were not originally in the IT business. For example, Amazon, which started as an online store is now the largest cloud platform provider, leaving the

second largest far behind. This trend of new providers entering traditional business by leveraging IT is showing that IT should always be considered as a strategic investment, not only as a support function. Although transactional, informational or infrastructure investments do not disappear either, as many industries also require the support functions as well. As mentioned earlier, clients' expectations are very high. Personal devices such as smart phones and tablet computers have widen the consumer markets and therefore also the business clients are also expecting enterprise software to have easy access, fast response time, mobility and visually appealing front-end. It is easy to forget that enterprise applications usually have massive amounts of data and the infrastructure is layered over the years. Although the building of a new system might start from scratch, required features might still come from legacy systems.

2.2. Five eras of information systems

Majority of the code that larger applications use today is built on top of an old code base. Leading software companies still have several products, which were originally developed in the 1980 and 1990s that are still being developed. Building on top of old hardware and software components can be challenging. Performance of the systems has increased tremendously and therefore also the performance requirements have changed, whereas size of the machines and physical components has gone smaller. Expectations are, that hardware is small with high performance. End users of the systems are accustomed for using computers and other devices for personal usage, so demand for easy to use interface and fast response time is increasing. Old components are not necessary originally designed to support complex integration, modern data storage, user friendly interfaces and high performance. Throughout different eras systems have been designed to fit the purpose of each era. Requirements and capabilities in different eras have been fundamentally different and therefore modifying them to match new requirements can be challenging.

According to Petter, DeLane and McLean (2012, pg. 343) there are five eras of information systems: 1) Data Processing Era, 2) The Management Reporting and Decision Supporting Era, 3) The Strategic and Personal Computing Era, 4) Enterprise System and Networking Era, and 5) The Customer-Focused Era. The Data Processing era was between 1950s-1960s. During that period, computers were used in military and financial sector by a small group of trained individuals. The computers were more calculators and only used in a small segment of the industry. The success of the system was measured just with speed and accuracy. (Petter & DeLane & McLean, 2012.pg. 343-345).

The Management Reporting and Decision Support era was from 1960s to 1980s. The calculation power of computer was harvested for monitoring and controlling production and automating administrative work. The early systems could produce structured information for routine decision-making tasks. The managers and researchers noticed that they were unable to process all the new data that was provided. Lack of understanding the data was a blocker for better decision-making. More and more employees were also exposed to computers, but the regular users were still limited and educated group of employees. The success of information systems was also evaluated by other factors than speed and accuracy, like profitability and improved decision making. (Petter & DeLane & McLean, 2012.pg. 343-345).

Strategic and Personal computing era started around 1980s and lasted until 1990s. Organizations started to realize that information systems can be used for achieving strategic goals and that information systems should be aligned with company's overall strategy. The use of information systems moved from back office usage to front office and more focus was put in to user friendly interface. Computers were used at home as well as at the office. (Petter & DeLane & McLean, 2012.pg. 345-351).

Enterprise System and Networking era was approximately from 1990s to 2000. During this era, it was possible to share data trough applications and access to data was not limited to one user or a computer. The number of users increased even more from previous era and information systems were used across the organization. This was also the era when outsourcing the IT and operations was introduced. (Petter & DeLane & McLean, 2012.pg. 345-351).

Customer- Focused era is the last one from five eras. Customer-Focused era starts from 2000s and continues to the future. During this era it has been possible to provide customized experience and tailored solutions based on customer's interest. The customer's interaction is shifting from employees to information systems. The governments are enabling citizen to access to decision making via information systems. Individual users as well as organizations are generating and sharing knowledge via information systems. (Petter & DeLane & McLean, 2012.pg. 345-351).

2.3. IT alignment and Outsourcing

According to Shpilberg, Berez, Puryear and Shah (2007) IT alignment means *"the degree to which the IT group understands the priorities of the business and expends its resources, pursues projects and provides information consistent with them."* Inadequacy of alignment can make IT irrelevant. Each growth strategy should be matched to IT spending. Underperforming IT is often rooted in the organization, not just misalignment. Each business unit might have IT systems that serve their individual needs, but do not serve the company's business as whole. Each business unit having their own, layered systems creates unnecessary complexity to IT infrastructure and application management. (Shpilberg et al. 2007, pg. 51-52.)

IT can be a strategic investment, therefore it should not see only as a support function for companies' business units, although IT also usually serves the support function role as well. Misalignment can be rooted also in this double role of IT: supporting other business units and leading development in others might lead to a conflict of interests. Direct profitability of IT department, resource allocation and problems on agreeing which department is responsible of covering which cost can cause friction between teams, departments and business units.

Alignment trap does not fix itself, not even when an IT organization learns to focus on aligned projects rather than less aligned ones. Dedicated resources in a wrong place might cause more harm than good. For example, if one business unit decides to develop new features on top of the old system, ignores the standardization and upgrades the legacy system, they will end up creating more complexity. Forthcoming systems enhancements and improvements will be even more problematic to implement, not to mention that benefits of scalability will be lost as well. (Shpilberg et al. 2007, pg. 53.)

Replacing the IT department will not help the organization to navigate from alignment trap. New staff will most likely run it to same issues as the old; only difference is that the new staff will lack knowledge of the existing systems. The right direction for the IT department is to start reducing complexity. Reduction means creating company wide standards and replacing legacy systems when possible, no more customization or layering on top of old. In the beginning, this approach requires significant investments. The cost savings and scaling benefits will occur in the long run. (Shpilberg et al. 2007, pg. 54.) Reducing complexity and standardization cannot be only lead by IT department. Like all strategic and companywide standards and investments, also IT investments and portfolio needs to be managed from top to bottom. C-level management needs to take concreate actions towards the alignment that business units will follow.

IT and business must both have accountability to deliver expected results on time and budget; this requires organizational change. High performance comes from centralization and simplifying IT functions, which might mean giving up customized department specific applications. IT governance needs to cross the organizational borders. Indication of being caught in the alignment trap is the ratio between maintenance and new product development. If more money is spent on maintaining the current system rather than creating new products, it can be a warning signal for misalignment. (Shpilberg et al. 2007, pg. 56-58.)

When companies arrange their IT function they often ponder the options of outsourcing. Outsourcing can be divided into two types: it can mean either sending work to offshore vendors or buying pre-packaged solutions. Outsourcing is nearly always cheaper than in-house development, but when applications are strategic or critical to competitive differentiation, it makes sense to develop them inside the company. Also, offshoring requires profound understanding of the project requirements from the client, because otherwise it can be difficult to track progress and hold the vendor accountable for performance and cost. Gathering enough information for offshoring might take the same effort than building the knowledge inside the companies own IT department. Routine tasks and less strategic parts of IT, that will not require lot of management or customer interactions are easier to outsource than complex systems development. Nevertheless, when a company decides to outsource some or all of IT functions, it should revisit that decision regularly, because business and house capabilities change over the time. (Shpilberg et al. pg. 55-56.)

Outsourcing to lower cost countries such as India, China and Eastern Europe usually significantly drops cost per-hour; however, overall project costs might be higher if challenges in communication, coordination and delays are counted in (Deemer, pg. 1). Outsourcing IT can be also done by buying consultant services. This is a good choice when work is project-based and demand for specialists is not

consistent. Most consultant houses offer specialist with a wide range of services and the specialist can be located to client facilities. It is also not uncommon that, for example, IT infrastructure is bought from a vendor. Infrastructure services can include leasing laptops and other devices as well as having dedicated servers from server farm. The outsourcing can be also variety of combination of all previously mentioned.

2.4. Software reuse

Software engineering is shifting from custom development of unique applications towards to building generic applications from certified reusable components (Jones 2010, pg.37). Custom solutions do not provide competitive advantages anymore, so companies are seeking to exploit more commercial-off-the-shelf (COTS) applications. Target is to reduce maintenance cost and have application vendor to carry the burden of incorporating technological advantages. Streamlining and continuously improving business process for gaining competitive advantages causes companies to avoid long, expensive and risky projects. Changes in IT infrastructure must be supported by adjustable integration framework; in addition, framework must proceed incrementally and rapidly. (Cummins 2002, pg. 3.) Core elements of COTS product are cost savings, ease of integration and extension, reliability and capability. However, COTS products are developed usually for general audience and therefore they are quite rigid for adapting special business rules and logics as well as challenging to customize without major development investments. (Ahmed & Kumar & Kumar 2017, pg. 2.)

Benefits of the COTS solutions makes them usually good candidate for large software implementation project. Enterprise software and information systems usually are not made from scratch, but build with existing solutions and customization. Rarely other than IT companies have capabilities and maturity to

research and develop completely new IT products for enterprise scale, unless they are trying to enter the new market and are ready to make huge investment to it. IT investments are rather done to widen the range of existing products and services and gain new markets by leveraging IT, not aiming to license new IT products. In COTS products, the R&D has already done by vendor. In large software projects the products are customized to match clients' needs, when implementing COTS products, the vendor carries the cost of developing the product and licensing and the client the cost of customization and adaption.

Object-oriented technology has boosted the development of reusable components. However there still are major blockers for building new applications or enhancements for COTS products. Sharable components require a standard environment where to operate and well-defined protocols to interact with other components. In addition, it should not be forgotten that the outcome must meet the business needs and must be adapted easily. (Cummins 2002, pg. 14.) Since IT industry has been expanding to several other industries, new solutions are created fast and market is changing rapidly. However, there also exist a risk of going live with already outdated technology or solution that does not meet anymore the business and market criteria.

Reuse of software may occur in the following way: 1) Purchasing a COTS product. 2) Use of the same system in different organizations. 3) Arrangement of shared services. 4) Use of a shared component. 5) Use of shared specifications. An assumption, however, usually is that it is more economical reuse software artefacts than build from software from scratch. Nonetheless, cost and benefits should be calculated carefully, because the cost of reuse might offset the benefits. (Cummins 2002, pg. 36.)

When purchasing a COTS solution, the enterprise shares development costs and support with other customers. The solution is compatible with related systems and

it meets the enterprise's business needs. Several business functions are supported same way in most companies, e.g. accounting is consistent for most of enterprises of a similar scale. Complications with COTS usually arises when an enterprise is expected to maintain unique ways of performing its business functions, or when an interface with other systems require peculiar adaptations. When an application is customized for one enterprise, it will lose value as a shared application. Vendor might also be unable or reluctant to resolve issues of nonstandard implementation. With new version updates, the enterprise will need to implement same adaptations once again and to realize this, it requires hiring and keeping the people who understand the application and unique business needs. Also, each time updates or enhancements are made, the quality, reliability and performance of the application are at risk. Whenever something goes wrong or a business process changes, there might not be anyone with sufficient knowledge to provide a timely solution. (Cummins 2002, pg. 37.)

Shared systems permit enterprises to gain competitive advantages by implementing its unique business process to all sites while sharing development and maintenance effort. Use of the same system in different locations provides data consistency, common interface, and common practice. However, implementation at multiple locations requires that operating environments are consistent across all sites. The implementation of common practices and computing environments might turn out to be very challenging for decentralized organizations. Arrangement of shared services is also one common service model; it can be applied to COTS applications as well as to custom applications. Different users can invoke the service at different locations, but execution happens in one system at one location. User will access systems by remote connection. This model offers a good cost reduction for development, maintenance and operation of systems in large enterprises, but on the other hand it provides very little adjustability for needs of different environments or related systems. (Cummins 2002, pg. 37.)

Shared components are exquisite parts of software, such as a function call or remote procedure calls in which a caller waits for a response. Usually shared components require compatible application architecture and supporting infrastructure to communicate with other components of the same application. Shared components have been an industry goal for many years. Compatibility between different implementations of the same component allows service providers to upgrade systems without replacing the whole system. Yet on the enterprise level, sharing components is not as simple as picking a component from an existing system and reusing it in another. It requires special analysis and understanding of the overall design to make components function in different contexts. In addition, how upgrades impact on hosting service must be taken into account as well as how support is arranged around components and hosting services. Inconsistency of components may become very expensive if compatibility is not considered in the design phase. (Cummins 2002, pg. 38.)

Use of a shared specification mean capability to generate applications from specifications done to metamodels and tools. Unified modelling language (UML) is one of the industry *de facto* standards for describing specifications. Combining standard component environments, common modelling language and standard technology mappings, enables development of tools for creating application from specifications. For enterprises that would enable code generation for a target environment each time when technology changes or when business process needs updates system changes can be implemented on specification level. (Cummins 2002, pg. 39.)

2.5. Legacy system

Adenakan Dedeke (2012) defines legacy systems as technologies that where implemented in a prior generation or era of innovation. All solutions including software, hardware, languages, standards, codes and technologies will all

eventually become a part of a legacy system. If a system has been used over 10 or 20 years, then it has spent more time in legacy phase than in development phase. (Dedeke 2012, pg. 38.) The reason why legacy systems tend to live longer than planned is that the costs and risks involved in changing the system are high enough to cause bankruptcy if they realize (Matei 2012, pg. 92).

If a system has been used 10 to 20 years, it has also generated massive amount of data, which is required to be migrate when systems are upgraded. Data migration is not light process itself, when the amount of data is large and complex, or when the legacy and new system are not directly compatible. Data in the systems is usually a valuable asset to the company. Planning and executing data migration should be done carefully, because the value of a new system cannot be realized unless it has data to work with.

Poor data quality impacts the company on three levels: operational, tactical and strategical. On operational level, poor data quality has in impact to customer satisfaction, operational cost and employee job satisfaction. Customers get easily dissatisfied if details associated to their data are not correct and for example their order or invoice is sent to wrong address. Operational cost increase when employees are detecting and correcting errors in customer data. Tracking errors and dealing with dissatisfied customer easily leads to lower employee satisfaction. On tactical level poor data quality compromises decision-making, the decision can be only as good as the data which they are based. Same applies to decision making on strategical level. Lack of relevant, complete, accurate and timely data– about customers, competitors, technologies, and other relevant features of the strategic landscape hinders developing successful strategical decisions. (Redman 1998, pg.80-82.)

One option is to buy a COTS system with the ability to support required business process and availability for customization. The risk in COTS products usually lies

with maintenance of the systems and with cost of customization. There is also no guarantee that the new system works better than the legacy system. Benefits are that the software is ready to use, if no customizations are needed, and the service level agreement (SLA) can be established immediately. (Matei 2012, pg. 93.) If a company has systems that are in legacy state, it is unlikely that the new systems are immediately ready to use, even with COTS products. There needs to be a planned sunset for the legacy systems, and before that all user must be already engaged with the new systems. The enterprise infrastructure and architecture should match the new system and the data must be migrated from the old system to the new, before any end users can access the system.

Current information systems are the outcome of the evolutionary development of business and technology. Information systems solve specified tasks and problems to achieve productivity improvements. The goal is to avoid manual work and improve response time. Several systems might exchange information with the same database, but in many cases, databases are not sole repositories for the data they contain, and similar data might be stored in other systems and databases as well. (Cummins 2002, pg. 2.)

In a modern business environment, the requirement for an IT system is that data can be combined from several systems. For being able to do better business and investment decision, companies combine structured and unstructured data and build predictive models from that data. The results, after they are deployed to production, can be accessed with multiple devices, such as tablet computers, smart phones and possibly even smart watches or medical devices.

When creating new systems, the latest technology is usually used. Old systems cannot be continuously redeployed with newer technology; they require considerable amount of rework and redevelopment to be compatible with latest technology. If a current system provides adequate business functionality, there is

no clear reason to update that system. After period of time there are major changes in technology and business operations and old systems become more difficult to operate and outdated. (Cummins 2002, pg. 2.) Managers and executives should consider several scenarios before legacy systems can be replaced. All decisions carry uncertainty and unpredictability. Companies have already invested large amounts of intellectual and financial capital to the existing systems. The companies are not willing to prematurely abandon systems before its development costs are recouped. (Dedeke 2012, pg. 39, 41, 42.)

In some cases, qualified employees are not willing to learn to work with a new system. Therefore, an IT issue can quickly become a human resource issue. The new system might degrade employees' domain knowledge and cause them to lose their work-related self-identity. Managers might have to make a decision for reallocation of the staff working with the legacy system. This might potentially cause a conflict between managers and employees. (Dedeke 2012, 39, 41, 42.)

The risk of employees not willing to adapt the new system can be mitigated by involving end users in the design and testing process. Group of key users are usually selected from group of end user. Key users can also include system matter and domain experts. When users are involved with development, they are also responsible for the business benefits and have opportunity to impact the outcome. Involvement usually leads to better commitment. Although sometimes the client's end users raise defects, even when the system is working as designed. This happens if the tester expects certain outcome and system delivers something else. Users also might make very direct comparisons between the old and new systems, even when the workflow is designed to be different.

Based on Dedeke's (2012) theory, business value can be defined with six factors. They are: 1) *Competitive advantages*, can an organization find new opportunities with the new system? 2) *Impact on profitability*, can the new system be taken into

use within budget? 3) *Growth potential*, how scalable the new system is, can it hold changes in business requirements and how a vendor support is arranged? 4) *Standardization and compliance*, does the system meet all existing and appearing standards and regulations for the system, data and platforms? How the licensing is arranged, can components be reused? 5) *System interdependency*, how much organization relies on legacy systems and how many applications are dependent to legacy system data? 6) *System security*, does the legacy system exposes the organization to security threats? (Dedeke 2012, pg. 40.)

All six factors should be taken into consideration when upgrading legacy systems. However, the profitability might be impossible to measure beforehand, because project cost may vary depending the complexity of the systems as well as how actively system is in use. Growth potential and competitive advantages are part of strategic IT investments, finding new opportunities and having systems that scale up requires a strong vision, both in business and IT. As was mentioned earlier in Section 2.3., standardization is needed for avoiding fragmented systems, without standards companies cannot gain any benefit of centralized IT. It is high risk factor if systems and its data are business critical, but system itself is already in legacy state. Business critical legacy systems might be very dependent on specific resources and skills as well as the technology used in them might be outdated. System failures are difficult to patch and security threats are more difficult to block. Nevertheless, upgrading a business-critical legacy system is not a trivial task, as it might require extensive data migration with minimum data freeze periods and system outages.

2.6. Data migration

Data migration is a process where data is transferred from an original system to a target system. The legacy and new systems are both usually structured differently on conceptual and technical level. (Matthes, Schulz & Haller 2011.) Data in the

legacy system is often from multiple sources and designed to use different data modelling tools or to be interpreted under different semantics. Understanding the legacy data requires understanding of all data sources, data constraints, interrelationships across different data sources, and availability of the data. Legacy systems might also suffer from inaccurate, incomplete, duplicate or inconsistent data. Improving data quality for the target system is often very expensive and time consuming. Furthermore, several data migration tasks such as data profiling, validating and cleansing must be executed iteratively, but parallel to project execution. (Thalheim & Wang 2012, pg. 260, 261.)

According to Razavian and Lago (2013), requirement for a new IT asset is that it can be easily distributed across organizational boundaries, whereas the assets in a legacy system are frequently a part of a large monolithic system. One owner also often controls all assets in the legacy system. There are also functional overlaps, data model, architectural and platform inconsistency between new and legacy system. (Razavian & Lago 2013, pg. 141.) As we have so many systems built in the 80's and the 90's still in use, data migration from legacy systems grows even more relevant. Developing a new system or managing an organization's IT portfolio starts from understanding the systems already in use. Migrating data from a legacy system to a new system is difficult mainly because it requires sufficient understanding of both systems. If the new system is under development when migration takes place, it might be challenging to align the development and migration schedules.

Thalheim and Wang (2012) present three migration strategies for executing data migration. 1) *Big Bang*: All data from the legacy system is taken in to use in the target system at one time. There are two ways to go-live with Big Bang. First option is to clean up the data in legacy system and map into the new system. Other option is to do *vice versa* and start mapping data into the new system and clean it up later. However, cleaning up data afterwards carries a bigger risk of system

failures, because of longer clean up processes. 2) *Chicken Little*: The legacy system is dived into modules; the goal is to have as few dependencies as possible between different modules. Migration takes place module by module. 3) *Butterfly*: Legacy data is transformed to the new system step-by-step, while the legacy system is under data freeze. The first thing to transform are read-only data sources, followed by temporary data storages. The difference between Chicken Little and Butterfly is separation of data sources in each step of the process. (Thalheim & Wang 2012, pg. 272-273.)

Thalheim and Wang (2012) suggested in their article that the reason why data migration often runs out of budget is because lack of a well-defined methodology that could help to sort out the complexity of data migration projects. (Thalheim & Wang 2012, pg. 260.) Razavian and Lago (2013) remind that reusing knowledge, experience and processes from other similar data migration projects is a key for achieving efficiency in future projects. The reuse can be achieved by recognizing patterns and similarities between different data migration projects and creating processes out of those patterns and similarities. Basic project elements that are carried out in several projects should be extracted from project specific ones. When elements are separated, it is easier to follow what drives the migration: business goals or to-be architecture. (Razavian & Lago 2013, 142,165.)

The data migration itself might not be technically challenging, but it is usually the data freeze periods that cause problems. When migration is in progress, the data cannot be modified, meaning that the data is frozen. The client might want to abandon the legacy system at a certain timeframe, therefore the new system must be ready and data migrated by a certain date. New data records are only created to the new system. A client cannot create new data to old systems as the migration is in progress, but the new system has not gone live yet, because some features might be under development. Depending on the project, the data freeze period might be quite long, which is frustrating to end users.

3. Project management

This chapter describes what issues and constraints projects have from the project management perspective. This chapter also discusses what skills and competences are needed from project managers. Executive support is mentioned as one of the biggest risks that leads to failing projects, by both project managers and executives (Liu, Zhang, Keil & Chen 2009, pg. 345-347). Therefore Section 3.4. describes role of project executive managers. Sections 3.5. and 3.6. describe issues and risks that projects have during their life cycle and what is project escalation.

3.1. Project management constraints

Different software projects experience different failures and not all software projects can be managed in same way. Although, according to studies many software projects tend to do the same mistake repeatedly (Nelson 2007, pg. 70). Projects might encounter their issues because contractor failures, poor requirement determination, ineffective stakeholder management, research-oriented development, poor estimation or insufficient risk management and in worst-case scenario, with all of them.

In a case of failure, the management needs to assess carefully what went wrong and what was done well to avoid the pitfalls in future projects. In addition, some mistakes form patterns and appear more often than others appear. Recognized patterns might help the management not to repeat same mistake again. (Nelson 2007, pg. 70, 72.) Underestimating the project management effort easily leads to budget overruns. According to study by Ahonen et al. (2015), they noticed that if customers are not willing to pay for the project management effort then no project manager hours are sold or reported by supplier, however the effort does not disappear.

The project management triangle is often used to describe what are the measurements and constrain of a project. The triangle is illustrated in Figure 1. Each side of the triangle (scope, cost, and schedule) should be balanced for a successful project. When that balance is achieved, the quality of the delivery follows. If any of the sections start deviating, it has an impact to other sides of triangle as well. Each of the constrains are often also competing, for example tight schedule usually increases cost and reduces scope. An increase in the scope usually leads to increased time and cost as well. The discipline of project management is to find tools for the project team and stakeholders to manage these constrains.

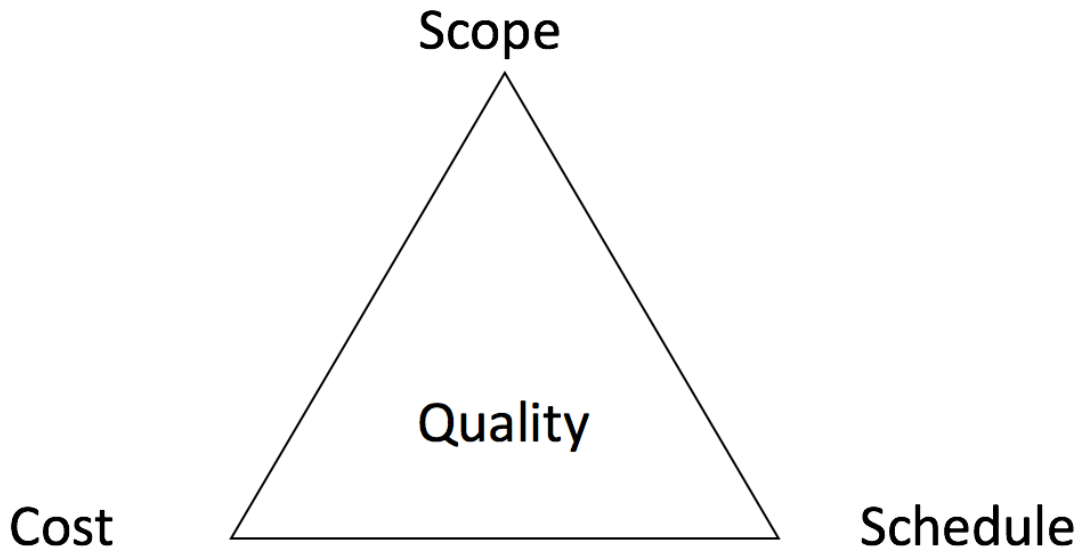

Figure 1. Adaption of Project Management triangle

As stated by Ahonen (2015), *"Project management effort is one of the necessary types of effort required for successful completion of software development projects."* In smaller projects, management of activities is easier to arrange. Furthermore, the communication structure can be simpler than in large projects. A project manager can have meetings with the whole team and directly communicate with all team members with easy. When the team size is bigger and project is more complex, performing project management activities takes more time. Larger teams

require formal communication channels and reporting structures to follow project status. (Ahonen et al. 2015, pg. 206.)

When issues emerge, the management tends to do same bad counteractions to tackle the known issues. Most common issue is that project is behind schedule. The attempt tackle impact of the delay is usually mitigated by either adding new resources to project, which temporary adds workload to existing resources, or by compromising project quality by reducing testing effort. Another common pitfall is, decision to upgrade software immediately after the new version comes available. Third common issue is that team dynamics are ignored by management. One of the key resources might be irritating rest of the team and management waits until end of the project before releasing him or her. (Nelson 2007, pg. 70.)

Project management training is required for students to understand projects' complexity. There are comprehensive lists and readings on different management approaches, competencies and skills, but very little guidance how to obtain, develop and practice those skills. How to recognize which technique to apply and when? The current approach for the project management education is linear: students are taught to solve linear problems, while this does not prepare students to face unexpected difficulties or unique situations and apply non-liner problem solving. An outcome might be that students will try to apply linear thinking to solve nonlinear problems; they are unable to adapt new and challenging situations and rather prefer to stay in their comfort zone. Small changes can develop in to huge and unexpected events. That is why project managers need to pay attention to relationships at all levels and recognize that organizational actions are not usually planned, but emergent. (Thomas & Mengel 2008, pg. 304-308.)

Focus of the project management training should be diagnosing different situations and how to adopt proper tools and techniques to each situation. Training requires learner to develop intense self-knowledge, emotional skills to coach and motivate

others and ability to adapt changes. Each problem solving instance should start by reviewing the situation holistically and not linearly. People feel most comfortable in a project environment where they feel that they are in control. When problems occur, they will try to avoid outside exposure. Approaching problems analytically will help us to understand what other team members think and how they perceive the dynamic of where the project is going. Meaningful communication, listening and aiming to mutual understanding will help the project team to create holistic shared goals, where all project aspects are taken in consideration. (Thomas & Mengel 2008, pg. 311.)

According to Thomas and Mengel (2008), three things are required from training that helps project managers to prepare themselves to constantly changing and complex environments: *"Flexibility to fit into the work life of senior practitioners. Develop a learning community that encourages questioning theory and practices. Sufficient duration to give the students time to reflect on actions and then apply that reflection in action to close the learning loop."* An IT project can succeed only when employees have sufficient skills. Management skills are not developed without practice and as much like developing any other skill project managers need repetition. Each team performs as combination of their skills. A less experienced project manager can have enough support from senior architect and the team can perform in sufficient level. This of course works *vice versa* as well, e.g., junior architect can find the needed support from experienced project manager.

3.2. Classic mistakes and how to avoid them

Nelson (2007) has divided classic project management mistakes into four categories: People, Process, Product and Technology. His findings in human capital show that frustration and dullness have larger impact on productivity and quality than any other factor. Second to motivation is working relationships between team members and skills within the team. Surprisingly one reason for

complaints is manager's lack of dealing or taking actions with problematic team members.

Nevertheless, the most classical mistake is to add resources when project is behind schedule; adding resources might end up taking productivity from existing resources without adding any to the new ones. Process related mistakes are found in both in management processes and technical methodologies. First mistake is to focus on the "fuzzy front end" before approval for budget and schedule. If a proper governance and focus are not in place, the project might end up moving from no schedule and scope toward very aggressive schedule where actual development is compressed to a short period of time. Humans have habit of making overly optimistic schedules and have poor estimates on delivering the scope. Poor estimates on scope, schedule and insufficient planning also build pressure on team performance, leading to issues in team's morale and productivity. Risk management is often overlooked area on project management, but losing the understanding of the risks is sign of losing control over the things that might go wrong. Common risks, for example, are changes in stakeholders, lack of sponsorship, scope creep and contractor failure. (Nelson 2007, pg. 70-71.)

Basically, all software projects encounter challenges with product dimensions. Product size has direct linkage to project schedule and therefore it has effects to time and cost. Requirement gold-plating happens when the product size is not correctly estimated and irrelevant features are planned into front end. Even when the requirement gold-plating is avoided, there is still around 25% change in requirements over the system's lifecycle. Sometimes developers are so excited with the product and technology that the project is using, that they end up trying out new features rather than creating what project really requires. This is called developer gold-plating. (Nelson 2007, pg. 71-72.)

The last one of product pitfalls is research-based development where engineering limits are exceeded in several areas. In technology category, classic mistakes involve usually misuse of modern technology, commonly known as silver-bullet syndrome. New technologies or practices are expected to solve all problems and eventually project team sets themselves for disappointment. Organizations can very rarely implement major changes in way of working in fast schedule, no matter how good new tools, methods and technologies are. For each new thing, there is expected learning curve before its start having positive impact on productivity. New practices also present new risk, which are discovered only by practice. Therefore, it should be carefully evaluated if presenting new methods or tools will bring any cost savings. In addition, benefits of switching tools during a project should be estimated against learning curve and possible rework, mistake taken with new tools usually cancel benefits. Although in some cases upgrades needs to be done within a product line. (Nelson 2007, pg. 71-72.)

To avoid and overcome mistakes, project needs to understand what went right and what went wrong. A foundation of a good project is in scheduling, scoping and in rational estimations. A project needs to have a calendar schedule that takes resource availability, business cycles and technology acquisitions in consideration. If estimations are done accurately less mistakes are made, less overtime is needed and there is less pressure on team to perform under tight schedule. A project manager also has better control over non-development tasks, which all leads to better budgeting. To plan well and maintain a plan, a project needs to establish a project management office that keeps track of project data. The project must be also divided to smaller parts, because smaller pieces are easier to estimate. Good work breakdown structure will help to understand the scope and size. Also, all project stages need to be measured against the original baseline to see where estimates went wrong. This is done in order that bottlenecks can be recognize for future estimates. (Nelson 2007, pg. 73.)

Managing stakeholder expectations and resistance plays important role on a project's success. If the key stakeholders are not satisfied, it does not matter how technically sound the product is. That is why the key stakeholders should be recognized as early stage as possible and make sure that they are included into progress reviews throughout the project. Another thing that is often neglected is identifying and keeping track of the projects risks. When the complexity of the projects increases, so does the amount and severity of the risks. If the risks are not identified, tracked down, mitigated and monitored the project has major vulnerability. (Nelson 2007, pg. 75.)

When a project is behind the schedule often testing and training are cut short. This might happen for two reasons. First, the team itself starts cutting corners by only doing minimal testing without planning. Second, traceability or the implementation phase is delayed and the management does not allow any flexibility on scope or schedule. In retrospectives, teams recommend automated testing and daily smoke test, which are well-known tools in agile methodologies. Continuous testing improves progress visibility, quality and mitigates risk of unsuccessful integration. (Nelson 2007, pg. 76.) Time and resource management leads issues in testing as well. If the project does not have enough resources, the developers are forced to test their own code, which is not considered as best practice for software testing.

Resourcing and personal relationships have impact on delivery and wellbeing of a team. Issues in the team *i.e* lack of commitment or underachieving should be dealt immediately. It is very common that a team is distributed and from different cultural background, which might cause time zone barriers, lack of face-to-face interaction and language barriers. It is recommended that at least part of the project team is co-located, even if it requires sending staff to foreign country for long period of time. (Nelson 2007, pg. 76.)

Resourcing and personal relationships are easily overlooked in a professional work environment. There are multiple reasons that might cause disturbances in the project delivery that are handled in the operative level of management. Leave of absence, sick leaves and employees leaving the company have impact on the project delivery and usually the project manager is responsible for finding replacements and on-boarding new team members. Regardless of the skill, there is always some leap time that is required for a new project member to get familiar with the project. Getting the new team members up to speed usually also takes effort of the existing team members and therefore the performance of the team usually drops. Drop in performance and effort required to train the new resources generates more cost, which is especially bad for a project that has a tight schedule and budget. Fear of cost overruns is the reason why project managers are also reluctant to change underperforming team members, if they are still performing on tolerable level and not burdening other team members too much. However, the project manager must also keep in mind that underperforming team member might cause other team members to lose their motivation, if so the performance of the whole team will drop.

3.3. Role of project manager

It is often mentioned that a project manager needs sufficient background or technical knowledge on the context of the delivery, but it is not specified what is the measurement for sufficient knowledge. In matter of fact, several researches emphasize other skills over the technical or context related knowledge. A project manager's technical and domain knowledge are important, but according to Medina and Francis (2015) not as essential as other characteristics such as efficiency. (Medina & Francis 2015, pg. 91.) Also Nelson's (2007) findings show that a project manager should be a people and process manger first; adding just technical expertise is seldom sufficient action to bring a project back on-schedule. Nelson also claims that if the project managers in his study had paid more attention

on scheduling, proper estimation, stakeholder and risk management they could have enhanced their project's success. (Nelson 2007, pg. 73)

Overall, a project manager should be sociable person who takes interest in the project team members and respects them. They can balance work issues with social talk. In stressful moments, a manager stays calm and engage the team to handle crisis. The team is involved in decision-making and planning, but the project manager carries responsibility of decision and outcome. (Medina & Francis 2015, pg. 91.) Successful project managers know how to leverage both formal and informal communication channels to solve problems, even when there are processes in place. Being familiar with organizational politics should not be underestimated; managers that are politically perceptive will not get that easily stuck into hierarchical and organizational boundaries. Mostly project managers need to help their team members to find and maintain a strong sense of purpose, belief and values, which will enable better team and individual performance. (Thomas & Mengel 2008, pg. 308-309.)

Acquiring the skills of an advanced project manager, one must have solid foundation of best practices, but also ability to learn beyond them. The project manager must be able to tolerate changes and recognize how situations emerge. In order to have capabilities to handle complexity, project managers also need ability to thrive on change and to be able to develop new methods on fly. Besides, being aware and having competence on traditional as well as new management methods, project managers need to apply them when correct circumstances present themselves. This requires critical reflection. (Thomas & Mengel 2008, pg. 308-310) According to Thomas and Mengel (2008) critical reflection entails: *"Questioning assumptions taken from granted notions embedded in theory and practices. Recognize the process of power and ideology inherited in institutional practices, procedure and institutions. Exploring the hidden agendas concealed by*

claims of rationality and objectivity. Working towards realizing a more than just working environment."

3.4. Executive support

Whereas a smaller software project might be easily managed by just project managers and team leads, large projects are often in interest of senior and executive level managers. The executive level support has following roles (Jones 2010, pg.74.):

- Approving project's ROI (return of investment) calculations;
- Providing project funding;
- Assigning executive roles, governance and project director roles;
- Reviewing status reports, risks, costs and milestones; and
- Following if ROI is reduced under targets because of overruns and delays.

Nonetheless, problems and failures can still appear, even if executive managers perform all their responsibilities. A fundamental base for good executive level business decisions is accurate status reports. Executive managers cannot reach good business decisions if they are provided with out-of-date or inaccurate project information. If problems and issues are hidden from the upper management, it will limit their possibilities to make decisions, prevent further delays and expenses or even to terminate projects that are out of control. (Jones 2010, pg.74.) Getting the executive management supports for a project is a critical success factor. Executive level sponsorship should be established from the start of the project and followed up so that the project sponsor is not departed midstream (Nelson 2007, pg. 76).

Both project managers and senior executives have raised that a lack of senior executive support is one of the biggest risk that easily leads to failing projects. Project managers tend to focus and identify lower-level risks, while senior executives pay more attention on high-level risks. Therefore, project managers

should also follow higher level risks and escalate to the senior management risks that are beyond of their normal concerns, like technology, user and requirement related risks. The risk of ignoring the needs of user is usually highlighted in literature, although risk of ignoring the needs of senior executives might have bigger impact to the project than ignoring the users. Unhappy users, of course, damage the project, which might lead to cancelling the project. However, only one unhappy senior executive is enough to call off the whole project. Senior executives should not underestimate the importance of user acceptance, but rather set up an example and be the driving force of embracing new systems and organizational changes. Both project managers and senior executives must understand their unique role to avoid failing to either get support or to provide it. (Liu, Zhang, Keil & Chen 2009, pg. 345-347.)

Managers also may find themselves in a situation where an executive management feels that they are supporting the project, but the project manager considers otherwise. In these situations, support might be different than the project manager wanted or requested. Sometimes also senior executives highlight how strategically important project delivery is, but deny all requests for additional resources or extensions to budget. Project managers might conclude this as a lack of support. Senior executives also sometimes like to avoid micromanagement and therefore do not actively seek to discuss with project managers about the daily operations on the project delivery. (Liu et al. 2009, pg. 347.)

Project managers and senior executives must have shared understanding what are project's high-level targets. Daily operation and strategic vision cannot be performed properly if there is no shared understanding of project's goals. Although a senior executive manages the projects high-level delivery and the client executives, the project manager oversees daily operations and gets delivery operations up and running. There needs to be trust and solid communications between both parties. If an upper senior executive does not involve the project

manager to high-level planning, how can the project manager handle the daily operations and deliver according upper management visions? Furthermore, if the project manager avoids informing and escalating problems to senior executives, there is no opportunity for them to do any mitigation actions or offer help.

Ideally, trust between the project manager and the executive manager should be so strong that the project manager can easily share issues with the senior executive, but keep the autonomy of the daily operations. The project manager needs to understand, that senior executives need to manage several stakeholders that might not have any or very little visibility to daily operations or their visibility might come from only one perspective. Pleasing several key stakeholders might sometimes cause turbulence in daily operations as well, in these situations senior executives need to communicate to their own project management why certain actions were taken and project managers needs to adapt to the new situation and support the decisions.

3.5. Managing delivery issues

The project performance is often evaluated by checking how the outcome meets the project specifications (Lai 1997, pg. 174). Techniques and tools do not manage projects; people do, but use of proper tools and techniques can speed up the job. However, it is still the people, who either make the project successful or contribute to its failure (Lai 1997, pg. 175). Proper tools also guarantee that there is audit trail on the project events.

Shortening the projects calendar time will increase costs and *vice versa*, both have negative impact on performance. If a project decides to go on a fire-fighting mode to enhance performance, it will also increase hours spent and, therefore, also the project's costs. In some cases, a project can also lower their performance standards to get the project back on schedule, although this action easily leads to extend

testing period and a higher number of bugs and extensive maintenance. Eventually, allowing too much slack in performance will lead to delayed delivery, decreased productivity and increased cost. Adding new resources usually does not scale down the task duration or boost productivity. Often it is the opposite, new tools or people might not be able to integrate to existing teams and therefore influencing the project team's overall performance. (Lai 1997, pg. 174.)

When there is slipping and delays in the project's schedule, it will take extra effort to get the schedule back on track. Team members should have autonomy and desire to manage their own work and make decisions without waiting for the manager's approval. The team must move flexibly from one part of the project to another and not concentrate only on line items. It must be shared responsibility to deliver on time and save costs. Quality cannot be frosted on top of the project delivery; the quality is a result of the commitment of the team members. To achieve better quality, the team must be self-regulated, self-directed and be able to change their way of working as soon as the tasks cannot reach the required level of quality. Project managers' job is to ensure that each element is executed adequately and that the team is working towards common goal. (Lai 1997, pg. 175, 177.) Lai calls her approach to "Synergistic", and it has many similarities with Agile and Lean methods. Already in 1997, she wrote about shared responsibility, self-managing teams and quality as output for highly motivated team. Although Agile methods have just recently increased in popularity throughout organizations, the practices have been existing for a very long time, but they have not been labeled being Agile.

3.6. Project escalation and terminating projects

In some cases, IT projects might take life of its own, which might lead to significant financial losses. 'Project escalation' means moving tasks that team is not able to complete or solve in time and budget to a higher level of management to be prioritized. Usually, the task has strategic impact on overall project delivery

and that impact must be evaluated and mitigated. After the team or the project manager has escalated the task, the higher project authority has a responsibility to found and implement an effective resolution. The purpose of the project escalation is to ensure that a higher project authority should address issue that have or might have significant impact to project delivery schedule, time and budget.

Meeting a very aggressive budget and schedule without specific targets communicates a vague goal, which is more difficult to aim than a specific goal. The difficulties of setting targets for budget and schedule are in the nature of work carried out in software projects. It is extremely difficult to make accurate estimates for software delivery, which suggest that it might be unwise to make irreversible decisions based on initial budget and schedule. However, estimations should be done and used as a reference point to see whether the project is on or off the track. (Lee, Keil & Kasi 2012, pg. 53-55.)

A common contribution to project failures is project managers' and the teams' inability to escalate troubled tasks and issue. The managers' and teams' willingness to escalate issues also depends what type of a relationship they have with the executive management. If there are no trust and respect between each other's, the escalation process might not serve either of the parties. When the project manager escalates something, it is a request for assistance. The higher project authority must offer support and include the project manager and team into decision-making. If the higher project authority isolates the manager and the team from decision-making, it implies that their input is not valued. The project manager might choose not to escalate things to avoid disturbance and to maintain authority over the delivery.

4. How to choose a development method?

This chapter presents the basics of software development and project planning regardless of the methodology. It also gives introduction to basic and background of Agile and Waterfall delivery. Due to raising popularity of Agile, Scrum and SAFe are presented as examples of Agile methods.

4.1. Software development

Yang and Jiang (2011, pg. 1) define that methods are process and techniques for completing tasks, whereas methodologies are patterns for solving problems with a set of processes, sub-processes and strategical decisions. Software projects are a set of processes that aim to accomplish a set of tasks that are in relation between each other. A software project is considered successful when quality, time and cost are accomplished at the same time.

There are many different development methods available. Just the number of existing methods prove that software development is complex and there are no straightforward answers what works and what not. Capers Jones (2010) suggests that selecting a software development method should start from examining benchmarks for applications that used various methods. A method or methods that outputs the best outcome for specific size and type of software project should then be selected. (Jones 2010, pg. 59).

In general and in granular level, all software projects consist of requirement gathering, design, development and testing. However, the type of the software project may vary a lot. The project can be a large ERP system implementation that might not require lot of programming, but can still be very complex by its design. An IT project can be data analytics that is done by using modelling tools like SPSS or by programming language like Python or it can be business intelligence that

focuses on analyzing business data. Web-development and mobile application development are also software development projects. Thus, there are multiple different pre-settings for a project and there is no one solution that fits for all methods.

According to Capers Jones (2010), an average requirement change is around 2 percentages per calendar month. Failure of understanding and managing the scope creep leads to severe overruns and damages the management's credibility and morale. (Jones 2010, pg. 49.) Schedule and costs correlate directly with the application size. Large applications are usually divided into multiple releases from 12 to 18 –month intervals. Understanding the overall size, and then the sizes of individual features makes it easier to plan a release strategy for the three to four consecutive releases. By knowing the size of each release, accurate scheduling and cost estimating is easier to perform. (Jones 2010, pg. 49.)

In 2009, more than 80% of software projects were modifications to existing products rather than completely new developments made from the scratch. The work tasks performed during a large software development project will vary depending, whether the application will be developed from the scratch, or will it consists of modifications to a package or a legacy application. Following list consists of process activities that effective large scale development process requires. (Jones 2010, pg. 61-62.)

1. Requirement gathering
2. Requirement analysis
3. Requirement inspection
4. Data mining of existing similar application to extract business rules
5. Architecture
6. External design
7. Internal design
8. Design inspection
9. Security and vulnerability analysis

10. Formal risk analysis
11. Formal value analysis
12. Commercial off the shelf package analysis
13. Requirements/package mapping
14. Contacting package user association
15. Package licensing and acquisition
16. Training of development team in selected package
17. Design of package modification
18. Development of package modification
19. Development of unique features
20. Acquisition of certified reusable materials
21. Inspections of package modifications
22. Document of package modifications
23. Inspection of documentation
24. Statistical analysis of package modifications
25. General testing of package modifications
26. Specialized testing of package modification (performance and security)
27. Quality assurance review of package modification
28. Training of user personnel in package and modification
29. Training of customer support and maintenance personnel
30. Deployment of package modification

Each above listed high-level activity usually contains from 150 to more than 1,000 tasks and lower level activities. A manual approach for managing each task is too cumbersome, therefore leading companies use management tools such as Artemis Views, Microsoft Project, and Primavera or similar. Because of the 2% requirement change per calendar month, each project activity must be performed in a manner that changes are easy to accommodate during development; this usually requires iterative development approach. In fixed schedule cases, it is also often contractually agreed and mandatory to develop in multiple releases. After certain point, all features must be frozen and no new features or changes are allowed. Changes after the freeze are to be scheduled to up-coming releases. (Jones 2010, pg. 61-62.)

No matter which development or project management method is chosen, the successful outcome must include following fundamental goals: Project planning and estimating must be excellent and accurate. Quality and change control must be excellent. Progress and cost tracking must be excellent. Measurement of result must be excellent and accurate. Navigating through major software project resembles going through a maze. There are more paths that end up in delays, disasters and dead ends than to successful outcomes. In addition, paths that lead to successful outcome are not the same for small and large projects or neither to all different software applications. (Jones 2010, pg. 10-11, 36.)

4.2. Project planning

As mentioned earlier, in 2009 more than 80 percent of software applications were not new and developed from scratch. For some of the legacy systems, written specifications might not exist anymore or they are out of date. Despite of missing and outdated documentation, legacy systems still contain business rules and algorithms that need to be transferred to new systems. Requirement analysis should therefore include not only new requirements but also data mining of the legacy code to extract hidden business rules and algorithms. (Jones 2010, pg. 70).

Modern business world is too dynamic for requirements to stay completely static over the development of a large application. External changes, for example tax laws, changes in corporate structure, business process re-engineering, or merges and acquisitions may prompt changes in software requirements. It is not realistic to think that corporations can freeze all its business rules while new software is under development, after all large application can take several years to be deployed. (Jones 2010, pg. 71). What applies to different development methods applies to different design and architecture methods as well, there are large number of techniques available and no the best practice has yet materialized.

In large software development projects, design and architecture has a major role to play, because features are often complex and expensive to develop. The enterprise architecture must match to corporate business needs including sales, marketing, manufacturing and other possible needs from other business. Enterprise level software may have more than 5,000 different applications. Software design describes functions and features available to a user, and how a user can access them. Internal design level documentation describes how different features and functions are linked and how they share information. (Jones 2010, pg. 76.) When project is large and complex, it is challenging to keep track of all activities and documentation. Easily it is thought that project documentation will decrease the productivity. This might be true in some cases but in most cases documentation and activity tracking is a part of the quality assurance. It is also a part of sharing knowledge and making sure that project activities are recorded somewhere else than in employees' head. It is also easier to on-board new resources when project data and knowledge is easily accessible. Improving productivity start from improving quality, because finding and fixing bugs is overall more expensive than software development. As said, *"quality leads and productivity follows"* (Jones 2010, pg. 7).

Quality and performance are not only driven by the software supplier. According to Capers Jones (2010), effort that is required from an expert of the client domain is on average about 20 percent from the overall effort of the software technical team. The percentage of the user involvement can vary from 5% to 50%, but if in a large scale software project it is under 5%, it will raise a risk of poor user satisfaction when the end product is finally delivered to the client (Jones 2010, pg. 7). Keeping the numbers in mind it can be concluded that software products are not delivered as ready to implement packages, but they require active management and ownership from the client side as well. According to Jansma (2004), a software vendor must proactively reach out to the customer and not wait for the customer

to come to them. Focus should be serving the customer and deploying products, not defining processes and generating assets (Jansma 2004, pg. 15).

4.3. Agile and Waterfall delivery

Software development methods are usually divided between Agile and Waterfall. Traditional software delivery is often called as a Waterfall model, because the workflow proceeds from top to bottom, just like the flow in waterfalls. Agile is wider concept, because it is an umbrella term for several different software development methods. Most distinctive difference between Agile and Waterfall delivery is that in Agile methods work is carried out in iterations. Similar top down model than Waterfall was introduced by Herbert D. Benington in 1956, the model was cited 1970's by Winston Royce, but as an example of non-working approach. Neither Royce or Benington called their model Waterfall. In its most common illustration, the model consists five stages and the workflow moves from top to bottom, like a cascading waterfall (Figure 2). Once the stage is completed, the model does not allow returning to previous stages.

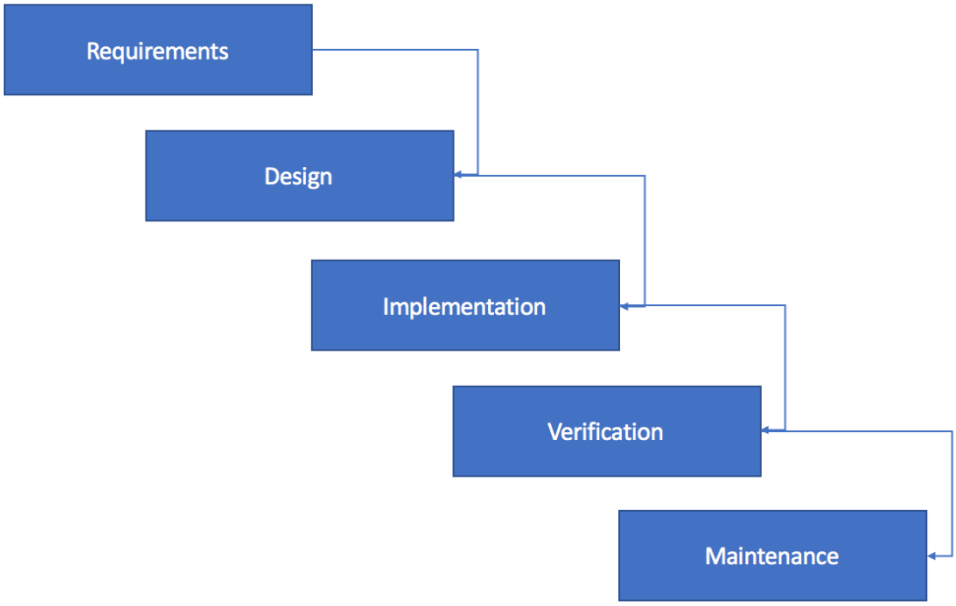

Figure 2. Adaption of Waterfall model

There are also other methods used for software delivery than Agile and Waterfall. One example is Lean software development, which is adopted from Lean manufacturing and Toyota production system (Poppendieck 2003, pg.1). Lean methods are widely accepted in Agile community and they partly overlap with Agile principles, but they can be still seen as their own category. In addition to the methods, there are tools that are used and accepted by Agile practitioners, but are not limited to Agile methods or do not origin from Agile methods. For example, Kanban is a very popular progress and task tracking practice, which can be used in equally in both Agile and Waterfall delivery. Kanban originates also from Toyota production system. There are also numerous variations to all methods and very often some methods and tools are used as mixture.

Traditional software delivery where planning is done predicatively and workflow is sequent from requirement gathering to testing, is not successful in today's complex project environments. In the so-called Waterfall approach, working software is delivered in the very end of development cycle, which delays feedback,

learning and potential return on investment. Releasing a software on very late in a project's life cycle will cause lack of transparency; it will reduce flexibility and ability to improve as well as increases technical and business risks. Alternative approaches, such as Agile methods have been available for practitioners for decades, but not been so widely adapted as the traditional approach. (Deemer, Benefield, Larman & Vodde 2012, pg. 3.)

The increasing popularity of Agile methods has made it even more difficult to draw distinct lines between methods, tools and best practices. There seems to be no common agreements what can be sold under Agile umbrella. However, some Agile evangelists have managed to bundle some methods, tools and best practices together and provide certified training. For example, Scrum is one of the most widely adapted delivery methods.

4.4. Scrum

There are several different Agile methods to choose from, but in this study, we focus on Scrum due to its popularity amongst practitioners. 58 % of practitioners use Scrum (VersionOne 11th annual State of Agile report). Against common belief, Agile methods, particularly Scrum can be very disciplined. In Scrum, there are roles and ceremonies just like in other methods. The overall development approach is very structured. Most distinctive difference to Waterfall delivery is that work is carried out in short iterations (see Figure 3).

In Scrum, each iteration or cycle of work is called as a Sprint. Sprints are time-boxed meaning that they end at a specific date regardless whether the work is ready or not. Sprints are usually 2-4 weeks long, and they should never exceed over four weeks. The team set themselves a target for each sprint by choosing items (i.e., requirements) from a prioritized list. No items can be added into sprint after it starts, because each sprint should hold the focus on a small, clear and stable target.

New items will be added to the future sprints. The team meets up daily to discuss the progress and possible obstacles. At the end of the sprint, the team presents the progress to stakeholders and collect feedbacks that can be exploited in upcoming sprints. The goal of each sprint is to have a shippable product that is completely tested, integrated and documented. (Deemer, Benefield, Larman & Vodde 2012, pg. 3.)

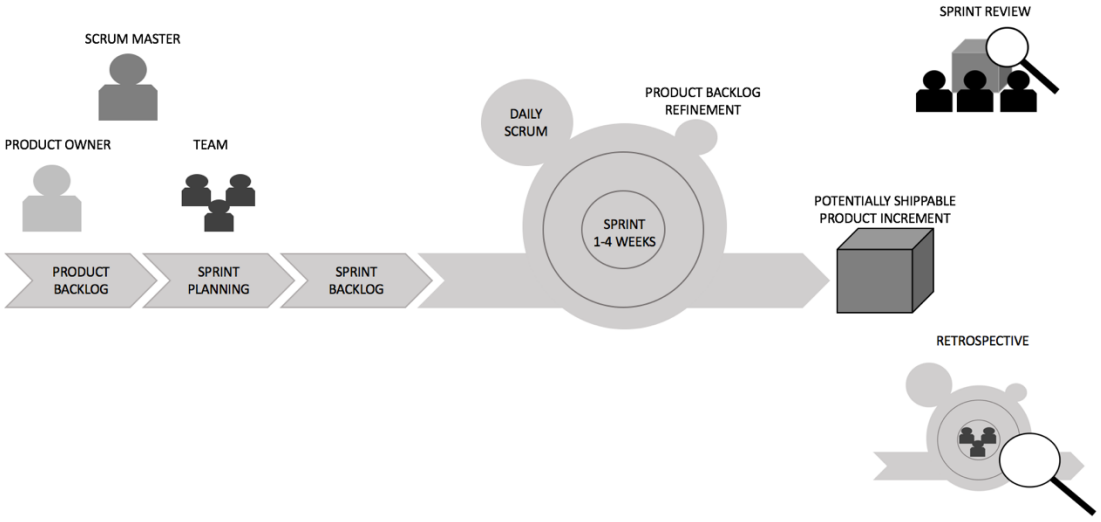

Figure 3. Adaption of: Scrum overview Deemer, Benefield, Larman & Vodde 2012

The Scrum team consist of Product Owner, Development Team and Scrum Master. In external development projects, the product owner (PO) role is recommended for the client to fill. PO is responsible of return on investment (ROI), recognizing product features and prioritizing requirements into a list. In organizations that deliver products, the Product Manager role is like Product owners role in Scrum teams. The difference is that Product Owners review the result of each Sprint, they actively interact with development team and other stakeholders and constantly prioritize the items in the backlog.

The Development Team is cross-functional team that has required skills to deliver product, that Product Owner request. The team has high autonomy as well as accountability over the delivery. The Team decides how many tasks they can deliver in each Sprint. The size of the team is usually from 6 to 8 full time employees. The productivity stays high when the whole time is fully allocated and committed to one product and Sprint.

The Scrum Master can be described as a coach or a teacher. Their role is to remove obstacles from other team members and guide the Product Owner. Scrum Masters makes sure that team members and organizations follow Agile practices, but they are not preventatives or managers of the team. Product Owner and Scrum Master should not be performed by same person, because the roles are fundamentally different. The outcome of combined roles leads easily to Product Owner that micro-manages the team. (Deemer, Benefield, Larman & Vodde 2012, pg. 4, 5.)

Scrum focuses moving from project-oriented work towards to a product and application. The teams are cross-functional and self-managing, therefore there is no project manager role inside the Scrum team. As a method, Scrum leaves lots of activities outside the actual scrum team, because it operates on team level. For example, financial planning, resources allocation, sales, contracting, application support and management after the releases are not addressed in Scrum framework. It is not described how and from where Product Owner will create the product backlog (i.e., a list of features that he wants to the product) and where the program level guidance comes. It could be argued that a traditional requirement analysis is done prior the development work to create the backlog, just like it is done in Waterfall development. Many of the portfolio and program level activities that are not described in Scrum are described in Scaled Agile framework (SAFe).

4.5. Scaled Agile Framework SAFe

Scaled Agile Framework is a method for scaling agile development across the organizations portfolios, value streams and programs. According to VersionOne 11th annual State of Agile report, it is the most popular method for scaling Agile and 28% of the practitioners who answered to questionnaire use SAFe. SAFe combines Agile principles and methods with Lean product development and system thinking as illustrated in Picture 4. The mindset of SAFe is captured in Agile manifesto and House of Lean. The purpose of the SAFe is to offer organizations structured way to manage their development pipeline and synchronize it with business requirements. In Agile development, non-development related tasks are often considered outside of scope, in SAFe they are built in to the framework, but not executed in the team level. SAFe can be used by smaller organizations as well, but the full framework is aimed for large companies with large projects.

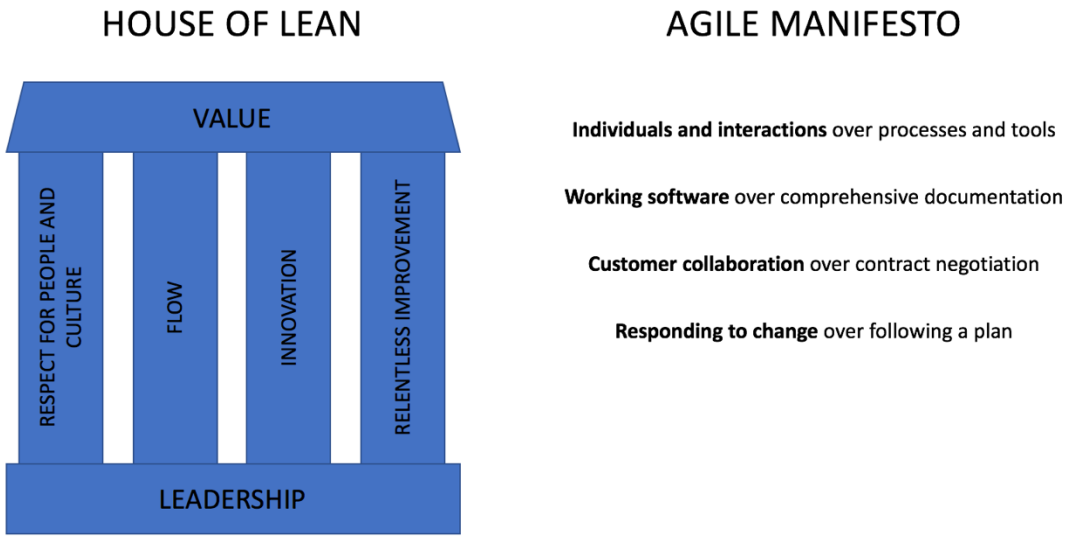

Figure 4. Adaption of: Aspect of Lean and Agile mindset, SAFe 4.0 Introduction 2016

The SAFe Lean- Agile principles are described in nine fundamental values (SAFe 4.0 Introduction 2016, pg. 6-8).

1) *Take economic view*. Supplier must have fundamental understanding of the client's mission for delivering best value and quality in the sustainably shortest lead time. Each decision should be made in reasonable economic context.

2) *Apply system thinking*. Challenges that occur in the workplace are result of a series of complex interactions between workers and systems that are used to conduct the work. System thinking is applied to organizations, system development as well as on how systems operates in its end user environment.

3) *Assume variability; preserve options*. Several requirement and design options are simultaneously kept in development pipeline. Focus is narrowed by using empirical data.

4) *Build incrementally with fast integrated learning cycles*. Systems are built incrementally in short iterations. Target of each iteration is to result in integrated increment of a working system. 5) *Base milestones on objective evaluation of working system.* It is suppliers as well as clients shared responsibility to ensure that new solution will deliver economic benefit. Frequent review in each integration point provides a milestone to evaluate if the investment is meeting the expectations. (SAFe 4.0 Introduction 2016, pg. 6-8.)

6) *Visualize and limit the WIP, reduce batch sizes, and manage queue lengths. "Visualize and limit the amount of work-in-process so as to limit demand to actual capacity"*, *"Reduce the batch size of work items to*

facilitate reliable flow though the system" and *"Manage queue lengths so as to reduce the wait times for new capabilities."*

7) *Apply cadence, synchronize with cross-domain planning*. Synchronization helps for understanding and resolving cross-domain issues, which are raised and identified in regular cadence.

8) *Unlock intrinsic motivation of knowledge workers*. The Lean-Agile team consist of knowledge workers. Providing teams with common goal and autonomy over their work, leads to higher employee engagements, which results for better overall outcome for company.

9) *Decentralize decision making*. Both types, centralized and decentralized decision making still occur in SAFe framework. However, decentralized decision making reduces delays, offers channel for faster feedback loop and improves product development flow. (SAFe 4.0 Introduction 2016, pg. 6-8.)

SAFe operates on three levels: The Program and Team level, The Portfolio level and The Value Stream level. The Program and Team level is brought together by Agile Release Train (ART), which is formed from self-organizing and cross-functional Agile teams. The size of each ART is from 50 to 125 team members. The teams plan, commit, execute together and move ideas from concept to deployment. Each team has shared goal from program backlog and the work between teams is synchronized. The Portfolio level controls the process of funding and governance for the product, services and solutions. The Portfolio level connects the portfolios with enterprise business strategy, larger business has multiple SAFe portfolios, whereas smaller business might just have one value stream and one ART. The Value Stream level in SAFe is optional. The organizations that have independent systems, which can be built with few hundred

employees, might not need additional process that Value Stream level operates. The purpose of the Value Stream level is to help organizations to manage the challenges that they face when building large scale multidisciplinary software, which is cyber-physical as well as high-assurance. All the ARTs in the value stream are synchronized in the The Value Stream cadence. (SAFe 4.0 Introduction 2016, pg. 8-20.)

5. Conclusion

The aim of this study was to answer the research question: How to implement a large scale custom solution? As mentioned previously, the most distinct feature of large scale implementation is time. The application development can take several years and applications life cycle can be up to 25 years. Schedule and cost of the project correlate directly with the size of application. Longer projects are also more complex, because usually the team size changes and skill levels deviate during the project life cycle. The scope is dived into several releases; project documentation and tracking tools are layered for each release. A large scale development process requires around 30 high level tasks that each has from 150 up to 1000 lower level activity, therefore understanding the dependencies between each task and managing the delivery is more complex than in smaller projects. When the team size is grows, a project require more formal communication channels and reporting structure, completing formal activities requires more project management effort.

This study is limited to reviewing existing literature, empirical research is not part of this study. Open Source solutions, which means that software including source code is free for studying, changing and distribution, are not presented in this study. Therefore, reviewing Open Source literature and doing comparison between COTS and Open Source in the development of large scale software solutions could be a part of future studies. Conducting empirical research with organization that have experience with large software projects would also offer more insight and perspective to this study.

Taking new system in use does not only mean installing software packages to dedicated hardware. Before that can happen there needs to be understanding what are the software and hardware solutions being installed, development effort concluded, system is tested and documented, users are trained and organization is ready to convert to use the new system. (Kakkar 2006, pg. 1.) According to

literature reviewed for this study, for organizations to leverage better their IT, they must aim to reduce complexity and have strategical and companywide standards. IT should not be seen as support function, but more as strategical investment, therefore like all strategical investments also major large scale development project should start only when C-level is supporting the project.

According to Capers Jones (2010), 80% of software projects are modifications to existing products. This implicates that one of the most used practices for large implementations is to use COTS products. Reviewing the literature for open source software was not part of this study. As mentioned earlier, in COTS the vendor carries the cost of developing the product and licensing and the client the cost of customization and adaption. Product support is arranged by the vendor and bugs in the core product are solved by the vendor, also further product development is carried by the vendor organization. This brings reliability that product features and security are maintained as longs as product is supported and client takes care of installing the fix backs and upgrades. Standard products with support and service level agreements standardize IT functions and reduce complexity.

Upgrading the legacy system is risky, because failures might paralyze the business or lead to loss of data, which will have major economic impact. However, the bigger risk of losing business because outdated systems and misalignment in IT, usually leads to decision to renew IT systems. Data migration is usually part of renewal, because companies do not want to abandon the data. Data migration adds complexity and work effort to a project, when old and new systems are structured differently on conceptual and technical level. Data migration also often runs out of budget due to well defined methodology. Reusing knowledge from similar projects is mentioned by Razavian and Lago (2013) as the best practice to tackle migration issues, this implies that project team needs to be well experienced with data migration projects before starting the project.

As it is mentioned in this study there are several development methods and tools available. Both Agile and Waterfall methods are presented in this study, but there is no direct comparison which methodology is better for delivering large software projects. Regardless of the development method the delivery should be well planned and structured, team and stakeholders motivated. Project should be well documented, risks and issues followed regularly and testing done continuously to avoid late stage defects.

When issues occur, they are in nature similar for a large and small project, but in large projects they tend to multiply. In small projects blocking issues can be caught when they are only risks or small issues, mitigation actions can be put to motion fast and most likely informally. In larger settings, it takes longer to have proper audience aware of the issue and to mitigate it, therefore formal communication channels and status reporting must exists throughout the project life cycle.

In literature, the distinctions between vendor and client organization was hardly done. However, both of these organizations are part of performing the project and both require set of individual but also overlapping skills. As no clear distinctions was made it is concluded that both vendor and client share the responsibility of making the project successful, by offering the needed resources and executive support, following the best practices and understanding the high risk and uncertainty. As mentioned earlier project have average of 2% requirement change per calendar month, both organizations need to understand that changing requirements are part of project uncertainty and tools to mitigate the impact is to have effective change management process in place. Also, both client and vendor need to have shared goal for project closing.

When it comes to comparing large projects with lot of implementation and integration on top of an old system, to, e.g., a webpage made from scratch it is like comparing apples and oranges. In a large project with exciting data, it is impossible

to estimate how much complexity exactly there is before actually starting the work. Smaller projects without the chains from existing data and system can adapt to changing requirements much faster than larger projects. Some large projects also have components that have not been build or integrated earlier, this adds complexity. Clients and vendor should equally understand that creating something, that has not been done before does not usually happen without incidents. Most common incidents are that project is not able to stay in the budget and deliver on time. It might be wise for project from time to time re-visit their budget, scope and delivery schedule. Especially clients should understand that even in fixed price deliveries changes in scope, schedule budget does not only impact the supplier, but the client itself. Most of complex deliveries require major input from the client organization as well, including joint office space for project team, system matter experts, testers, financial follow ups, steering group and other project committees.

It is nearly impossible to predict all the possible outcomes from all the decisions, but rarely staying still is either an option. There is still much uncertainty in large implementations and in software projects in general. Therefore, no method, technical knowledge, experience or other assets that project might have, do not fully prepare project for all obstacles that project will encounter during its life cycle. This uncertainty is important for a client to understand, when they are acquiring new information systems. It is also evident that large software delivery projects are not just software deliveries; they are organizational transformation. Organizations cannot change their IT systems, without having an impact on how the work is executed.

References

Ahmed, Z. & Kumar, V. & Kumar, U. (2017). Managing critical success factors for IS implementation: A Stakeholder engagement and control perspective. Canadian Journal of Administrative Sciences. Wiley Online Library.

Ahonen, J. & Savolainen, P. & Merikoski, H. & Nevalainen, J. (2015). Reported project management effort, project size and contract type. The Journal of System and Software 109. pg: 206-213. Elsevier Inc.

Cummins, F. (2002). Enterprise Integration: An Architecture for Enterprise Application and System Integration. John Wiley & Sons. United State of America.

Dedeke, A. (2012). Improving Legacy-System Sustainability: A Systematic Approach. IT Pro January/February, pg: 38-43. IEEE.

Deemer, P. The Distributed Scrum Primer. Version 1.0. http://www.goodagile.com/distributedscrumprimer/

Deemer, P. & Benefield, G. & Larman, C. & Vodde, B (2012). A Lightweight Guide to the Theory and Practice of Scrum. The Scrum Primer. http://www.scrumprimer.org/

Jansma, P. (2004). When Management Gets Serious About Managing Software. December 10, pg: 1-15. IEEE.

Jones, C. (2010). Software Engineering Best Practices: Lesson from Successful Projects in the Top Companies. McGraw Hill. United State of America.

Kakkar, S. (2006). Implementation Aspects of Software Development Projects. India Conference, 2006 Annual IEEE.

Lai, S. (1997). A synergistic approach to project management in information system development. International Journal of Project Management. Vol. 15, No.3, pg. 173-179. Elsevier Ltd and IPMA.

Lee, J. & Keil, M. & Kasi, V. (2012). The Effect of an Initial Budget and Schedule Goal on Software Project Escalation. Journal of Management Information Systems vol. 29, no.1, pg: 53-77.

Liu, S. & Zhang, J. & Keil, Mark. & Chen, T. (2009). Comparing senior executives and project managers perceptions of IT projects risk: a Chinese Delphi study. Information Systems Journal 20, pg: 319-355. Blackwell Publishing.

Matei, C. (2012). Modernzation Solution for Legacy Banking System Using and Open Architecture. Informatica Economicâ vol. 16, no.2, pg: 92-101.

Matthes, F. & Schulz, C. & Haller, K. (2011). Testing & Quality Assurance in Data Migration Projects. 27th IEEE International Conference on Software Maintenance, pg: 438-447. IEEE.

Medina, A. & Francis, A. (2015). What Are the Characteristics That Software Development Project Team Members Associate With a Good Project Manager? Project Manager Journal. October/November 2015, pg: 81- 93.

Nelson, R. (2007). IT Project Management: Infamous Failures, Classic Mistakes and Best Practices. MIS Quarterly Executive. Vol. 6 no.2, pg:67-78.

Poppendieck, M. & Poppendieck, T. (2003). Lean Software Development – An Agile Toolkit. Addison Wesley.

Redman, T. (1998). The impact of poor data quality on the typical enterprise. Communications of the ACM, 41(2):79–82, 1998.

Petter, S. & DeLane, W. & McLean, E. (2012). The Past, Present, and Future of "IS Success". Journal of the Association for Information Systems vol.13, pg: 341-362.

Razavian, M. & Lago, P. (2013). A lean and mean strategy: a data migration industrial study. Journal of Software: Evolution and Process 26, pg: 141-171. John Wiley & Sons Online Library.

SAFe 4.0 Introduction (2016). Overview of the Scaled Agile Framework for Lean Software and System Engineering. A Scaled Agile Inc. White Paper. http://www.scaledagileframework.com/introduction-to-safe/

Shpliberg, D. & Berez, S. & Puryear, R. & Shah, S. (2007). Avoiding the Alignment Trap in Information Technology. MITSloan Management Review. Vol. 49, No.1. pg: 51-58.

Sommerville, I. & Cliff, D. & Calinescu, R. & Keen, J. & Kelly, T. & Kwiatkowska, M. & McDermid, J. & Paige, R. (2012). Large-Scale Complex IT Systems. Communication of the ACM. Vol. 55, No.7. pg: 71-77.

Standish Group (2013) The CHAOS Manifesto: Think Big and Act Small. The Standish Group International.

Thalheim, B. & Wang, Q. (2012). Data Migration: A theoretical perspective. Data & Knowledge Engineering 87, pg: 260-278. Elsevier.

Thomas, J. & Mengel, T. (2008). Preparing project managers to deal with complexity – Advance project management education. International Journal of Project Management 26, pg: 304-315. Elsevier Ltd and IPMA.

VersionOne The 11th annual State of Agile survey. https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2

Weill, P. & Sinan, A. (2006). Generating Premium Returns on Your IT Investments. MITSloan Management Review. Vol. 47, No.2. pg: 40.

Weill, P. & Subramani, M. & Broadbent, M. (2002). Building IT Infrastructure for Strategic Agility. MITSloan Management Review. Vol. 44, No.1. pg: 57-65.

Yang, L. & Jiang, C. (2011). Research on Enterprise Application Software Project Implementation Model. Management and Service Science (MASS), 2011 International Conference. IEEE