European Journal of Futures Research (2017) 5: 16
https://doi.org/10.1007/s40309-017-0123-7

ORIGINAL PAPER

Future software organizations – agile goals and roles

Petri Kettunen [1] · Maarit Laanti [2]

Received: 31 July 2017 / Accepted: 5 December 2017 / Published online: 16 December 2017

Abstract
Digital transformation is rapidly causing major, even disruptive changes in many industries. Moreover, global developments like digital platforms (cloud) and IoT create fundamentally new connections at many levels between objects, organizations and people (systems-of-systems). These are by nature dynamic and often work in real time – further increasing the complexity. These systemic changes bring up new profound questions: What are those new software-intensive systems like? How are they created and developed? Which principles should guide such organizational design? Agile enterprises are by definition proficient with such capabilities. What solutions are the current scaled agile frameworks such as SAFe and LeSS proposing, and why? In this paper, we aim to recognize the design principles of future software organizations, and discuss existing experiences from various different organizations under transformations, and the insights gained. The purpose is to systematize this by proposing a competence development impact-mapping grid for new digitalization drivers and goals with potential solutions based on our agile software enterprise transformation experiences. Our research approach is based on the resource-based and competence-based views (RBV, CBV) of organizations. We point out how most decision-making in companies will be more and more software-related when companies focus on software. This has profound impacts on organizational designs, roles and competencies. Moreover, increasing data-intensification poses new demands for more efficient organizational data processing and effective knowledge utilization capabilities. However, decisive systematic transformations of companies bring new powerful tools for steering successfully under such new business conditions. We demonstrate this via real-life examples.

Keywords Digital transformation · SAFe · LeSS · Agile enterprise · Systems thinking · Value streams

Introduction

Digital transformations are a cause of rapid and even disruptive change in a majority of companies and future competitive environments. Fundamentally novel models for organizations and businesses (like Uber-type) are emerging, and traditional companies as well must consider their structure and their roles in achieving new business goals. Both the software producer organization and the customer viewpoints should be understood via comprehensive sense making. Companies now begin to focus on software – either following strategy, or ad hoc, when forced by competitive pressure imposed on them by digitalization.

We view future competitive companies as agile and sustainable, as well as more fundamentally software-based with respect to both their outcomes (products and services) and operations. For industrial-age companies such new principles will require new organizational roles and goal setting. Systemic changes (digital transformation) are complex to achieve, but can be steered through by employing holistic resource- and competence-based views.

When digital elements and data become increasingly incorporated, more and more software is included both in existing and totally new processes in different organizations. These questions are increasingly imperative for software development organizations to comprehend, requiring new capabilities. Is the only problem we are solving how to achieve faster time-to-market by improving flow, or are there also other aspects to consider? In this paper, we disentangle this question from the organizational resource-based view. Software development organizations have the added need to not only understand the new systems to be developed, but also be able to create

✉ Petri Kettunen
petri.kettunen@cs.helsinki.fi

Maarit Laanti
maarit.laanti@nitor.fi

[1] Department of Computer Science, University of Helsinki, Helsinki, Finland

[2] Nitor Delta, Finland

and evolve the strategy, software and solutions to create them. The strategy should impact on organizations, software systems development and human resource management (HRD).

Agile software methods (typically referred to as 'agile methods' or 'Agile') have been utilized ('agile transformation') for a long time in many software development organizations. Essentially, they realize in software development what agile enterprises in general aim towards. Modern large-scale agile methods and frameworks expand the basic agile software development principles to the enterprise level. Consequently, it is prospective to assess how these methods and frameworks would support future companies when they strive to become software-intensive.

Our primary focus is businesses in private sector, but also many public sector organizations have similar considerations when they digitalize their services.

Research propositions

In this paper, we operate with a dualistic view of software organizations, and by *software organizations* we mean the following:

1) Software firms / IT companies (or internal software R&D units / IT departments) with new software production as the core purpose
2) Software-intensive customer organizations of those software producers, including traditional companies who replace parts of their systems or solutions with software and need to understand what challenges that brings to business

We operate from the premise that increasing digitalization will cause most – if not all – companies to become software companies [1–3]. Consequently, they turn into software organizations. Naturally the timeframes for such transitions vary across industries, but for example in music and media businesses, it has already taken place. Future changes for the financial sector could be even disruptive in the short-term (less than 5 years) stemming for instance from the European PSD2 directive commencing in January 2018, while for example the construction industry and healthcare sectors may expect longer-term evolutions due to their different nature. However, even entire current industry sectors and boundaries (e.g., energy) are currently under digital reformation.

It follows that software use and its production will be more and more intertwined, making future software organizations interconnected in complex ways. Moreover, software systems will be more dynamic and under continuous evolution, particularly in Internet of Things (IoT) environments.

Following this line of thinking, we posit the following research propositions:

1. Digitalization broadens software use and software use cases and creates new ecosystems. This will add on to the complexity of software systems, systems interconnectedness, and the value of software becomes more intrinsic.

- The complexity of the system increases (e.g., IoT).

2. Software companies must be prepared to master such new concerns in order to be able to serve their customers successfully.

- Software organizations may (have to) realize digital transformations internally.

3. New kind of structures and roles / competences may be needed to support agile and flexible development needs and goals.

- There is a need to organize for (real-time) continuous software evolution (architecture, organization).

In this paper, we develop and elaborate our initial ideas presented in [4]. We compile a set of competence development and resource impact mapping grids to address those research propositions. The key idea is to define what (goals), why (purpose), and how (roles) future successful software organizations perform. Our research approach is design science. The purpose of this design-scientific work is to construct actionable artefacts (the grids) for future software organizations facing digital transformations. We validate them by informed arguments and our real-life empirical experiences. Different industrial companies can then compare and relate their situational conditions and transformation circumstances towards software-intensity accordingly.

Well-known scaled agile frameworks such as SAFe (Scaled Agile Framework) and LeSS (Large-Scale Scrum) provide some directions, but not a complete answer. How are SAFe and LeSS tackling these systemic problems, and why have these approaches been selected? The SAFe structures are incorporated in the grids. This reflects how (if) they currently provide support in achieving the future performance traits of software organizations in digitalization.

Drivers and needs

Complexity and speed

Complexity is inherent to modern software organizations. This stems from two main reasons:

1) Multiple people and roles are needed to create total customer value end-to-end.

2) Software product (system) use environments (in particular IoT) grow larger and more dynamic with many different interconnected systems and actors.

Moreover, the software products themselves are often increasingly technically complicated. While complicated systems are not necessarily complex, their development and management may impose complexity on the software organizations. In particular, if the internal product architecture does not flexibly support the necessary evolution of the customer value-in-use for instance because of tight modular coupling, the technical dependencies may require several different organizational actors to coordinate their decision-making and consequent actions in complex ways. Furthermore, software issues like technical debt and legacy system components may exacerbate this.

In principle, there are different archetypes of software systems as characterized in Table 1. It is crucial for software organizations to understand their basic properties to be able to develop and manage the software solutions accordingly. Realizing their fundamental competence of is particularly important for future software organizations in the digital economy of the Internet era [5–7].

Notably, the division (A-C) in Table 1 is in practice not as clear-cut. Even in a mostly well-defined software project there may be some less clear (uncertain) parts. Moreover, in large long-lasting projects, the primary type may even change over time [8]. This requires additional sensitivity and dynamic capabilities from software organizations.

Following that line of thinking, it is crucial for each software organization to understand the complexity level of their customer environment in order to be able to develop matching levels of capabilities to deal with the complexities. Otherwise there is a risk of producing a complexity gap, which may over time even cause the total failure of the company [9]. Consequently, the internal complexity of the software organization should not be excessive, in order for it to be able to cope with the external complexity with matching speed.

Scaled agile frameworks SAFe and LeSS extend agile with systems and complexity thinking

Basic agile software development methods such as Scrum have been in common use for more than a decade [10]. However, when entire product development organizations adopt them, there are additional needs for larger-scale models to take into account the aspects of organizational complexity and speed [11]. The most well-known of such recent models are LeSS and SAFe.

Organizational complexity, software complicatedness, and product development speed are interrelated. In general, the larger required development organization, the more complex it becomes, which may even lead to a combinatorial explosion of complexity [12]. Rising complexity of communication scales with the size of large development organizations, and so larger software organizations tend to suffer delays in decision-making. These delays lead to less flexibility and agility,

Table 1 Software solution natures and development principles

	Customer Space Problem Type	Software (System) Solution Nature in Theory and Practice	Strategic Approach
A	Well-defined	Goals and requirements known in advance, solution fully specifiable	• Plan-driven software engineering and management • KEYS: *Time-to-market economically with optimal solution*s
		Except ill-defined boundary conditions Example 1: Calculation of interest in banking software (mathematically defined, ill-defined boundary conditions such as number of needed calculations per second or accuracy as number of digits in output. Example 2: Implementation of a new law, e.g. in banking.	• Optimize for development risk. Example: Implement first most crucial parts of the law (ones with most penalties). Implement less risky parts later, closer to deadline in order to avoid or minimize the risk of paying penalties.
B	Ill-defined	Success criteria and requirements uncertain, multiple possible solutions.	• Agile software development (accommodating uncertainty and change iteratively) • KEYS: *Stepwise shaping following customer feedback to converge for a mutually satisfactory solution*
		Example: developing new type of application	• User-experience driven; • Example: Measure user retention and service usage and develop or pivot accordingly.
C	Wicked	Problem changes over the life-cycle influenced by the software solution interacting with the users and the system environment.	• Evolutionary software development with continuous experimentation and feedback • KEYS: *Continuous value definition and assessment (assumptions), core asset development and management, platforms and ecosystems integration (co-creation)*
		Example: New product development, e.g., new fitness device platform enabling customized software updates	• Surveying users and the ways they use the product and insert new software

preventing organizations from communicating their objectives and outcomes efficiently. This is increased by organizational and software technical dependencies (new built functionality; technical, testing, defect, integration debts).

SAFe and LeSS approaches differ on this matter. SAFe suggests a number of roles whose responsibility is to manage the communication, e.g., Product Owners and Product Managers who should communicate on backlog contents at least twice a week. Another example is Product Integration (PI) planning, where Agile Release Train personnel gather together to discuss what they should develop in next 10 weeks time.

This is where the LeSS and SAFe approach differ. In LeSS, the aim is to scale down and try to manage with the smaller number of development teams. This way, the approach allows the organization to emphasize communication while scaling down the sheer amount of it.

Automation can be used to make communication, and therefore development faster (e.g., test automation for developmental software defect detection and correction to decrease the lead-time from detection to correction, and thereby reducing the risk for defect debt). In all, with end-to-end transparency across the organization, the complexities (possibly rigidities) with software and organizational dependencies should become apparent. The systems dynamics for the overall speed can then be realized. The use of automation, Continuous Integration and DevOps are largely accepted in the agile software community, and these techniques are also part of SAFe and LeSS.

So while both SAFe and LeSS approaches pursue the mastering of development speed and complexity, the strategic approach is left for the company.[1] Their answer to different spaces in Table 1 is similar – scaled agile frameworks can be used in all three cases. In business environments, software organizations have multidimensional needs for speed [7, 13, 14]. Therefore, each software organization should understand what their particular external speed requirements are, and how the internal speeds contribute to that in total. This is what a software company does when adopting some of these frameworks, such as SAFe or LeSS.

Value flow

We maintain that the principal measure of software organization is the customer value(−in-use). Consequently, the overall performance objective of the software development and delivery chain is to achieve and sustain that value. In business contexts, this must be done economically considering also the exchange value and production costs.

[1] However, SAFe advocates that a company should create an Economic Framework that states how a company uses agility to enable its strategy.

There is no universal measurement of customer value, and even for a specific customer the value is relative and may change over time [15]. However, our premise is that the customers will be satisfied when they experience (customer experience, CX) value from the supplied products (user experience, UX) and services considering the benefits and costs. The goal of the supplier company is then to provide that value, optimized with regard to the economy of the company. Customer experience stems from the perceptions of all the cognitive and emotional touchpoint encounters (i.e., products, services, information exchanges, personnel interactions, etc) which may then affect their future behavior (e.g., loyalty, repurchasing).

The role of software in the customer value creation process (value-in-use) varies, depending on the customer solution type. However, the role of software is growing, even in many traditional physical products with more and more embedded software (e.g., automotive electronics), “smart” devices, and in more general product / digital service bundles. Software organizations should realize the impact that this expansion in the roles of software has in their specific digital transformations in order to be able determine and assess the future customer value constellations – which may be radically different from the traditional ones [16, 17].

The size of the software organization matters [18], i.e. in a small organization the entire software development and delivery chain may be performed by a single (co-located) team. For instance Scrum has only three roles in order to create flow. There are no separate testers and coders in Scrum, because everyone does what has to be done, in order to get the Sprint release ready. In larger organizations, typically multiple people and different roles are needed to create an end-to-end value stream like illustrated in Fig. 2. For each particular software organization, it is revealing to map the value stream flow backwards (upstream) from the focal point of the customer software use to realize all intermediate steps and exchanges, since the customer value (−in-use) is only created when the customer(s) can actually consume and use the software.

Like discussed above, the speed of the software organization depends on its complexities. Consequently, they affect the overall time-to-value. It is thus essential to realize the level of complexity in each software organization. Unnecessary complexities may then be reduced systematically (e.g., organizational dependencies of structural complexity).

Furthermore, keeping the software solution up-to-date and continuously (even in real time) improved requires looping flow incorporating the customer use feedback and other potential sources of inputs [19, 20]. Achieving and maintaining such continuous value flow requires end-to-end software organizational capabilities.

Scaled agile frameworks aim for creating a value flow

Basic agile software development methods such as Scrum have been in common use for more than a decade [10]. However, when entire product development organizations adopt them, there are additional needs for larger-scale models to take into account the organizational flow aspects [11]. Scaled agile methods SAFe and LeSS embrace that line of thinking.

Large-Scale Scrum (LeSS) aims towards creating an organization that is able to optimize Value Flow through that organization by enabling a number of Feature teams to work together using the Scrum method. Figure 1 illustrates this.

The Scaled Agile Framework (SAFe) aims towards creating a value flow and an economic framework for a company. The key concept in SAFe is the Agile Release Train (ART), which creates end-to-end value for the customer. For SAFe transformation, it is essential to identify the value chain within the company that creates the systems or solutions. The Agile Release Train or Release Trains are then used to optimize those chains so that all the people who work in that value chain learn to work together. In a way, ART is a team of teams-structure, enabling the people who create the same systems to communicate and synchronize frequently at regular intervals. Figure 2 presents such a team of teams-structure.

Organizational capabilities

In sum, we have recognized the following needs for future software organization capabilities:

- When smart products and services interconnect to other such entities, cloud services (in particular, data), other systems and people, organizations need certain new competencies and abilities to utilize them:

  Specific product / service design competences
  The ability to combine them (including data analytics)
  Holistic system design competence (including knowledge and human factors)

- Business competence must be developed accordingly in order for the organization to be able to fully utilize the software-intensive design competences to gain total large-scale business benefits in new digital environments and economy.

  One of the key consequent questions is how to select and measure appropriate business key performance indicators (KPI) (e.g., the contribution of embedded software in the product sales and export).

- Each software organization should first and foremost comprehend what types of software it is developing in what environments (c.f., Table 1). The competences and capabilities of the organization should then be fitted accordingly:

  The assessment of matching speed requirements

Particularly in the environments of the new Internet era, these capabilities (type C in Table 1) entail new principal traits for software organizations. As external systems complexity increases, which is mostly uncontrollable (e.g. IoT), systems thinking, analysis and engineering capabilities are needed to cope with it. Internally, there is a need to organize for continuous (real-time) development and continuous delivery (CD) with flexible architectures (technology, organization) guided by the new overarching software paradigms [6]. Basic engineering approaches need to be augmented with capabilities to accommodate mathematical and social complexities [21].

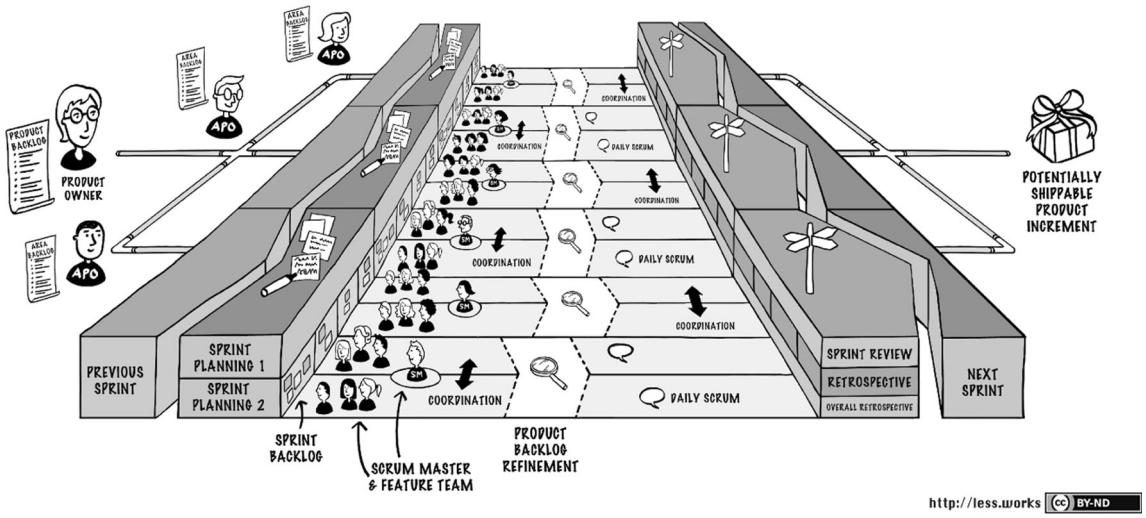

Fig. 1 Software organization design framework of LeSS

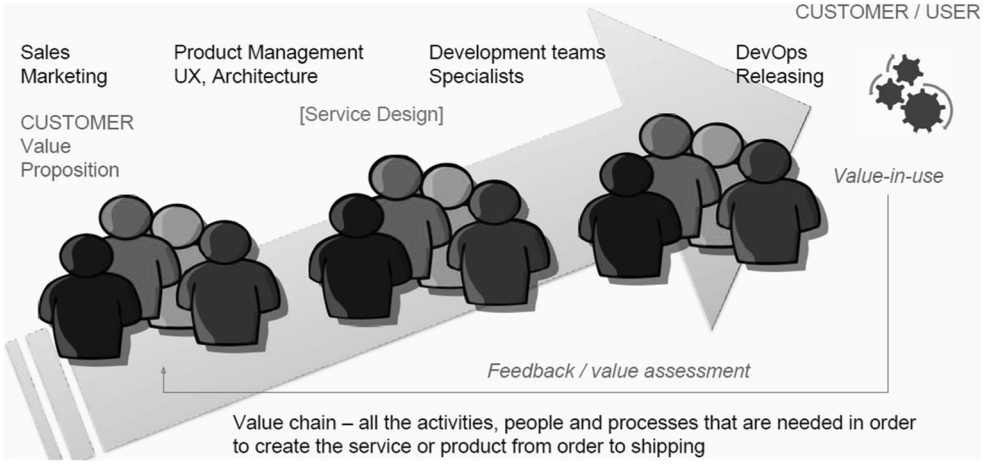

Fig. 2 Multiple people and roles needed to create end-to-end value flow in larger organizations

Vision of future software organizations

Matching speed matters

There are two dimensions of externally observable speed of software organizations: outbound and inbound (c.f., Fig. 2). The former concerns the value delivery to customers from the starting point (customer order or opportunity identification), and the latter spans from the input recognition (e.g., customer complaint or environmental signal) to the information processing and responding appropriately.

Considering development and the release speed in business contexts in general, every feature and product or service has a window of opportunity – if you are on the market at the right time, you will generate more revenue than those who enter the market later. If people are happy with your product / service, you can use the revenue to improve the service / product and keep the competitive advantage – it will be more difficult to enter the market later. Lean and agile methods both aim towards rapid cycles of development, fast returns on investments, and minimal inventories of work in progress.

Taking the stance that *time-to-value (−in-use)* is the overall performance goal, there are different needs and means for speed in different types of software described in Table 1 (A-C). Following that line of thinking, Table 2 outlines the key aspects of speed for each archetype. With type A the main goal is fast execution based on known specifications while in type B uncertainties should be reduced as soon as possible, in order to converge towards an acceptable software solution. However, in type C there is no definite end criteria and the software organization must continuously stay in sync (even real-time) with the software in its use environment.

Consequently, fast software development and release speed has multiplied enabling roles by providing time-based competitive advantages:

- Facilitating more effective value capture (type A, B) – potential first-mover advantages
- Incorporating more and faster feedback (type B, C) – better responsiveness (agility)
- Supporting continuous experimentation (type B, C) – more experiments possible within limited time periods

Proficient software organization designs achieve speed by continuous development flow and avoiding complexity hindrances

Proficient large-scale agile software development coaches also advocate the following rules in order to gain development speed and effectiveness. These rules follow the value flow view outlined in Fig. 2 and the consequent needs and means for speed in Table 2.

1. Information must flow as efficiently as possible to all needed people; preferably broadcasted at regular intervals – to avoid communication debt i.e., explosion / missing information.

- This also applies to arranging regular demos in order to spread information on the latest updates in the system under development to the whole organization.

2. New software code must be integrated as fast as possible and be immediately available to all people – using automation (and continuous integration (CI) systems) to keep the amount of currently ongoing changes in the code small.
3. Testing must be planned and executed parallel to development, feedback should be as fast as possible, and the amount of open defects should be kept into minimum to avoid the accumulation of testing debt.
4. Defects must be fixed as soon as discovered, in order to avoid error debt.
5. Architecture should be as simple and modular as possible, in order to avoid technical debt.

Table 2 Needs for speed in software organizations

	DRIVERS for Speed	NEEDS for Software Competences and Resources / Roles
A	Lean (streamlined) product development and delivery flow end-to-end	• Stable software bases (e.g., manageable technical debt) • Efficient development and delivery automation
B	Fast absorption of feedback and consequent adaptation	• Flexibility in software technical design (architecture) and management • Organization design for rapid learning loops
C	Continuous sense-making and action in emergent environments with interactions of various objects and actors	• Sensing and responding with intelligent use of heterogeneous data from diverse sources • Continuous software-oriented business processes (e.g., ecosystem and asset strategies)

- This enables faster integration (2.) and better testability (3.)

6. Keep documentation up-to-date – to avoid documentation debt.

- This facilitates information flow (1.)

7. Systems that are used to build the systems must be updated to latest versions and kept up to date in order to keep competitive / (interface) compliant.

Interestingly enough, speed targets can be achieved and improved at least partially by avoiding delays. Both SAFe and LeSS promote Feature teams and removal of waste for maximizing flow. However, in practice the Feature team composition requires multiple overlapping skills for the development teams as well as in-depth knowledge of all the developed code. Thus using only Feature teams may not be a practical solution.

The Feature teams are a preferred solution in agile software development for gaining speed in order to avoid dependencies between components and thus between development teams. While certain dependencies are intrinsic in large-scale software organizations, the key to speed is to avoid unnecessary, excessive (even combinatorial explosion) dependencies cause for instance by structural complexities in organizational decision-making hierarchies. The following are some organizational design patterns to accomplish this:

- Work organized as component teams: The traditional way of managing work is to split complex solution problems into (fixed) architectural layers and develop each layer separately. This may cause integration problems and non-fitting components.
- Work organized as feature teams: The new way is to speed up by developing functionality to all layers simultaneously – and to integrate frequently and manage the technical dependencies during development with end-to-end real-time transparency and visibility supported by design and integration automation.

Key measurements

Is the only problem we are solving how to achieve faster time-to-market by improving flow [22]? Our contribution is to advance from this basic question in two ways for future software organization performance analysis and improvement as highlighted in Fig. 2. Considering the time-to-market, we are stressing time-to-value(−in-use). With respect to the internal development flow we extend outside the focal organization to the business environment where the customer value is actually determined, created and assessed in real time.

Following this line of thinking, we suggest two principal measurement categories (KPI) for software organizations:

1) VALUE ECONOMY:

- *Right customer value delivered economically (quality, usability of service / product)*

2) REAL-TIME BUSINESS:

- *Responsiveness to customer feedback and software use data*
- *Sensitivity for new value-creation and business opportunities*

For those principal measures we need to define expressive indicators, in order to be able to tell whether the software organization is really capable of attaining those performance targets.

A “People first”-attitude is characteristic for any Agile or lean organization. Many organizations, such as Valve, let employees decide which project they want to work on, thus leaving the choice of which is the right project to the employees. Others try to inquire this from the market, e.g., by releasing alpha or beta releases to specific markets that forecast the market behavior on larger market areas. For these reasons it is typical for game software developers to release in Canadian market before entering the global market. Even in large companies, the employees may be in the frontline knowing and serving the customers they interact with, and so we introduce

additional metrics tailored especially for service businesses (VALUE ECONOMY):

3) EMPLOYEE SATISFACTION:

- *How happy the employees are in the current company*
- *How proud the employees are of their product / service*

4) CUSTOMER SATISFACTION:

- *Would the customers recommend our services / products (UX, CX)*
- *Brand value*

Metric 3 leads to the acquisition of the best talent in market – which in turn will help boost performance as the best employees deliver an order of magnitude times more than an average (less-motivated) person, Metric 4 helps in mastering the networking effect, i.e., be the recommended service or product vendor. It has been studied that in the software industry the network effect is key to winning over the market [23]. The network effect is one of the reasons why typically one software vendor gains a monopoly position, along with a secondary "overflow" vendor, while other minor vendors only receive small fractions of the market [24].

When a market is volatile, it is generally better to use leading indicators that forecast market behavior, than lagging indicators (REAL-TIME BUSINESS). This is especially important when a company starts new business or new activities or when a company is new (i.e., start-up). When software deliveries are of a continuous nature, an additional major leading indicator is:

5) CUSTOMER RETENTION:

- *How many of the customers return to service*
- *How often the customers return to service*

For a continuous service business it is also important to monitor the availability of the service. Service organizations may even use the big data and forecasting to detect errors, e.g., if an internet shop is implemented using micro services. And thus we need additional metrics:

6) SERVICE AVAILABILITY:

- *How many customers are currently using the service compared to the typical number in respective conditions*
- *Service interruptions, speed of operations*

Such data could be used even to decide if the service provider should roll back to previous version of the software. With DevOps practices more and more companies are able to have automatic integration, deployment and service level control. With automation, companies also typically would like to go for smaller and smaller release sizes to reduce the within each new release and thus make thousands of releases per day, like Amazon, Google or Facebook. With these mini-releases it starts to become also more typical that companies also invest in automatic rollback of the previous versions, if something goes wrong with the release.

Notably none of these proposed measures (1–6) are strikingly new nor unique to software organizations. However, what is foundational is the realization of new factors and rules for them caused by software. For instance customer satisfaction (4) has been a traditional business KPI for years but now the sources of (dis)satisfaction are often more and more software-based or software-enabled (e.g., driver assistance systems in automotive). Moreover, many traditional cost structures may change radically when physical product components and manufacturing operations are replaced by software elements. In particular, the economies of scale are fundamentally different when it comes to software. Also the time scales may become vastly shorter, and supply chain management rules change when physical, but connected products can be field-updated with software even remotely and customer feedback data can be collected from many sources automatically.

Capabilities

Established organizations base their business typically on one established business innovation or idea. This is known as exploitation [25, 26]. A typical established organization can try to streamline its value chain and cut costs by using lean methods, yet keeping the same quality as before. Similarly, established large software organizations can streamline their development operations using scaled agile frameworks.

Organizations can try to create new revenue streams with the help of innovation. Quite often bringing new innovations to market per se is not possible, but market space needs to be carefully explored before, in order to find out what kind of products and services could fit in the market. An established organization may try to create directly radical innovations, or employ a rapid innovation-exploration cycle with the help of automated continuous integration and delivery systems in order to explore the market needs and available opportunities. An established organization may also try to avoid costs by replacing a part of the value chain with software.

Novel software start-ups typically start from the exploration phase, using lean startup methods or user experience driven agile software development. Successful software startups focus on following customer satisfaction and customer retention as key metrics (c.f., chapter Key measurements). As start-ups grow and mature, the brand value, and the ability

to scale up the start-up-like operational mode become keys to maintaining successful operations.

Figure 3 presents a summary of these discoveries. In essence, it is a grid of key strategic capabilities for software organizations in different business situations. The strategic moves are based on certain software capabilities like described above, and – consequently – possible capability gaps may prevent organizations from realizing such moves. It is thus crucial for different companies transforming towards software organizations to acquire and develop such new software capabilities in order to be competitive in the future.

With increased competition, the challenge for established companies is to move from the exploitation-only mode to exploration [26, 27]. Buying a few promising startups and integrating those into established companies has not been proved to be a good strategy, as most of these kind of company acquisitions fail. Modern companies try to implement agile methods to create a corporate culture within the company that would better support exploration and software-enabled innovation, but their challenges lie in what kind of future organizational structures, ways of working, management and investment models and calculations could support this [17].

Following this line of thinking, we expect the following core capabilities for software organizations (c.f., Tables 1 and 2):

1) High-performing (speed) software development and delivery engine (technical and structural)
2) Rapid (continuous) innovation with software
3) Proficiency at dealing with software-intensified customers / users
4) Scaling with Internet-era software-based systems and assets

Notably our suggestions in the chapter Key measurements indicate those traits.

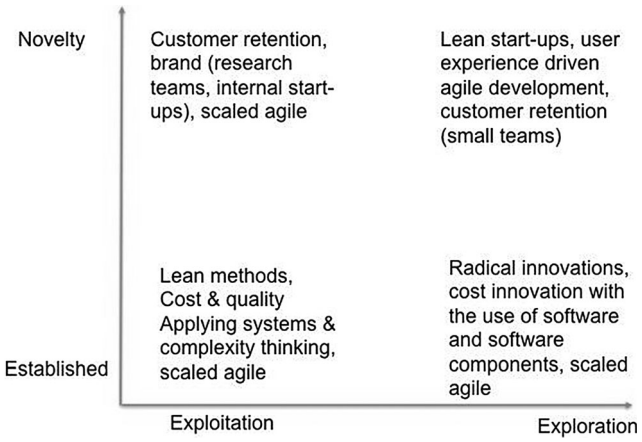

Fig. 3 Software organization strategic capabilities grid

Transforming

Digitalization changes

Digitization, digitalization, digital transformations and the general concept of digital, are more prevalent and omnipresent (e.g., cyber-physical systems, CPS). Strikingly, this phenomenon does not concern just developed countries and economies, but also developing areas (e.g., mobile technology in Africa) – often more fundamentally.

For most companies, if not all, this causes needs for organizational changes towards software organization to various degrees. For current IT companies / departments, this could be developmental or transitional change, while Industrial IoT for instance brings even radically disruptive, transformational changes to software organization for many companies in traditional industries. The role of IT is to be both a driver and an enabler [9].

Consequently, each (new) software organization should be able to deal with such new factors as digital / (smart) products / services and digitalization in product marketing, design and production (manufacturing). Furthermore, even fundamentally new, software-enabled business models based on for example open APIs will be possible. For instance brick-and-mortar shops and other physical infrastructure and CDs as software distribution media have been costly. Delivering digital (or physical) goods via Internet costs less, and subscriptions can happen more often (e.g., newspapers) [28].

Table 3 categorizes those traits for future software organizational changes. It follows that there are needs for new competences and resources to develop and implement the related core capabilities introduced in the chapter Capabilities (1–4).

Digitalization enables

Future software organizations have digital innovation opportunities both internally and externally like outlined in Table 4. However, in order to be able to realize them, there are needs for new competences and resources to develop and implement the related core capabilities in the chapter Capabilities (1–4).

In addition, overall customer / user experiences can be developed with the guidance of external data and internal real-time measurements [5]. Customer retention is one of the key consequent business performance objectives (c.f., chapter Key measurements).

The meaning of future software organizations in digital transformations

The overarching consequence of digital transformation is that more and more software organizations will be

Table 3 Factors of future software organizational changes

Changes		NEEDS for Software Competences and Resources / Roles
Digitalization changes former physical goods	… into abstract form.	• What is the role of software going to be in Your products / services [3, 5]? • CAPABILITIES: 2)
In physical form, the domain knowledge has been essential.	Domain knowledge becomes hygiene; knowledge on meta-level is essential (who builds the platform).	• What software will contribute to Your customer value creation [6, 29]? • CAPABILITIES: 4)
Old players have a challenge to use the digital channels.	Automation reduces transaction cost. Delivering digital (or physical) goods via internet costs less, and subscriptions can happen more often.	• How does software change Your demand / supply chain [9, 30, 31]? • CAPABILITIES: 3)
The market could change or extend because of digitalization.	New players (having software background) have been able to enter to digital market.	• What software companies will be Your prime competitors / collaborators and what are their key competitive advantages [3, 28, 30, 32]? • CAPABILITIES: 1)

created and formed in the future. Hardly any industry sector will be totally unchanged. On the contrary, many traditional industries are already facing radical changes with software, and new software-enabled cross-industry networks are further blurring the boundaries (e.g., smart energy transition).

It follows that companies in their particular digital transformations should reflect on themselves as software organizations. In Tables 3 and 4 we have presented archetypal software-related questions to consider. We stress that each company should be able to give conscious answers to those strategic considerations in order to be able to develop the consequent critical software capabilities for successful transformations. Moreover, they should also be frequently revisited, because digitalization is continuously shaping competitive environments, while the development of organizational capabilities may take time.

Results and experiences

In this section, we present a real-life case of software organizational change. The case is about extreme speed and service quality needs of the Yle Graphics Team, which reflects many future software organizational aspects of our present research.

The example organization impacted by digitalization is Yle Graphics design. They are responsible for providing variety of graphics like logos, animations, drawings and any kinds of graphics that are needed in various broadcasted and internet productions, such as news, television series, webpages etc.

Digitalization has transformed traditional manual work and brought in many new tools and also new techniques. The ease of making new graphics has resulted in an increase of graphics in both traditional broadcasted media and also services available online, thus transforming the nature and scope of the work.

Table 4 Opportunities of future software organizations

Enabling	NEEDS for Software Competences and Resources / Roles
Cost innovation: • Same service or product can be implemented, provided and serviced with a lesser cost but similar quality. • Moving digital data costs less than moving physical goods. • Software component in goods allows reduction of the production costs compared to physical / hardware components. • Shared computational services / more IP-addresses enable to enlarge the networks and computational capacity – and e.g. new business with shared services. • Thinking whole chain from production to consumer some steps can be streamlined for business opportunities.	• How could software change Your company (industrial engineering and operations management) [2, 32, 33]? • CAPABILITIES: 1) 4)
Business innovation: • smart products and services • new business models in digital economy	• What is the role of software in Your industry going to be [3, 5, 30, 34, 35]? • CAPABILITIES: 2) 3)

Managing this kind of rapidly changing work and needs (customer requests and requirements) is extremely challenging even while serving only internal customers within the same company. New requests pop up daily, and changes in needs happen constantly. Most of the needs have a strict deadline, and if the graphics are not available on time of the scheduled broadcast or show, the opportunity is lost.

Before lean and agile methods were taken into use (i.e., before agile transformation) in this organization, the majority of the projects were made by one person only, and lasted less than a day. On the other hand, a small amount of the projects were repeated on a weekly basis, which continued for years, like producing graphics for a series of productions.

The challenge was to weigh how much and which projects could be done with internal people, and where and how there was a need for external freelancers. Another challenge was the number and variety of different tools used and new tools emerging and developing constantly. Designers were familiar only with their own relevant tools and techniques, and none were masters of all.

Figure 4 represents a specific Kanban board based on Agile and lean thinking that we helped the graphics team with to visually manage and schedule the work assignments. It enables the team to see and discuss all the upcoming assignments and to discuss who are interested in which assignments.

The board brings all information needed to manage the work together with the people that do the work into one place. Having visibility and an open discussion enhances cooperation and teamwork. After taking this board into use, most of the assignments are now done as teamwork, which is more fun, more efficient, and allows cross-using different tools and techniques. Because of this change the graphics teams have decided that they will no longer do any project alone as one-man tasks but will do all future work as a team.

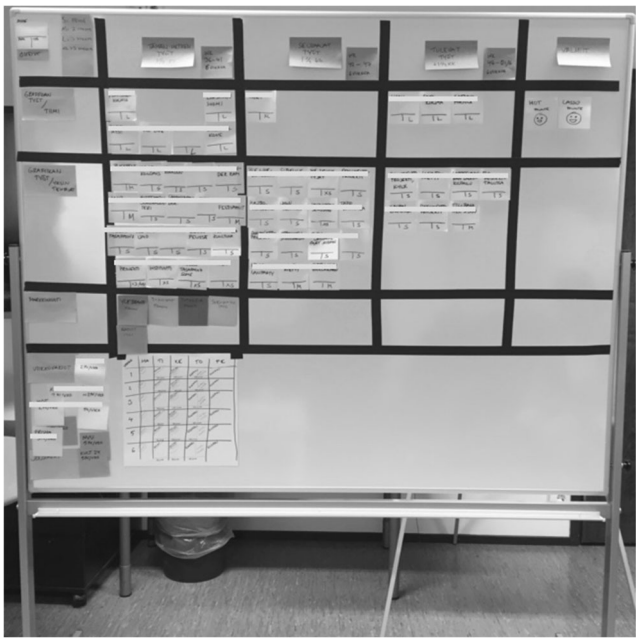

Fig. 4 Board for flexible work/teams allocation in Yle Graphics Design, following lean and Agile principles

The Yle Graphics organization is an example of an organization that has already changed due to digitalization, and can thus be used as an example of future organizations. Main existing traits are:

1. It is responsive, flat and cross functional with frequent communication.
2. Traditional mechanisms would be too slow for managing it – only visual management adopted from lean and agile software development culture enables responding to customer requests with enough speed – even though it is not developing software, nor is it an IT organization.
3. Regarding our chapter Key measurements, the key metrics are geared towards customer satisfaction – tier 1 (immediate internal customers) and tier 2 (users of broadcasted and internet media and services).

Discussion

Strategic changes cause a need to change value streams and the organization

Like illustrated in Fig. 2, we envisage that the overarching organization design principle of future software organizations is to organize and optimize for developing and delivering software-enabled customer value (in-use) economically. Consequently, software-intensive value streams become key vehicles requiring new software-oriented value stream management capabilities (c.f., Table 2):

- A *fluid* organization should be organized around the value stream.
- An *adaptive* organization should be able to quickly align itself along the new value / revenue stream:

  Cause minimum disturbance in organization and team structures and allow the organization to self-morph into new needed state.
  Allow learning of new (needed) competences.

- A *flexible* organization is extending and relying on 3rd party suppliers.

Moreover, with such new capabilities, the software organization can impose new strategic changes with modifications in the value stream. Value streams may be combined or changed in particular based on software platforms. New startups and ecosystems create new value streams, which could build on

open platforms of incumbent companies (with open application programming interfaces, APIs).

One of the key questions of such future software organization design is whether to base on self-organization or imposed structures. While traditional organizations rely on imposed structures, we have had some trials for self-organization. Organization can be a subject of continuous evolvement, and people who do the work can propose and accept changes to structures.

Furthermore, when products become more software-intensive, the value streams are increasingly reliant on data / information. That is, when future companies become more software-intensive, they will need more efficient internal data processing and effective knowledge utilization capabilities (incorporating data science). More input data (feedback) can be received from the products/services-in-use (even real-time), and various new external data sources (e.g., IoT) should be scanned to stay in sync with the environment and its business networks. External outputs of the value stream may be, not only products and service features, but increasingly also data. All this changes the organizational dynamics of the value stream towards real-time functionality, as data (information) transfer can be fully digital.

Insights and foresights

Software-intensification is a megatrend. The use of software components transforms both the existing products and goods, but also changes how work is done more cost-efficiently with the help of software (like in our Yle Graphics case, chapter Results and experiences). This will lead to:

1. Use of lean and agile software management methods in other than software organizations due to increased speed. 11th state of Agile survey (Version One, 2017) already reported a growing number of non-software companies using agile methods [10]. In fact, only 36% of all companies were software-only companies. Also many traditional industrial companies (e.g., automotive) are increasingly looking for them when they are becoming more software-intensive [3].
2. More networked organizations for additional flexibility
3. Software is common. Creating new software products or services becomes more challenging.
4. Digitalization continues and changes former local businesses to global, increasing competition

Following this line of reasoning, we can elaborate our vision (chapter Vision of future software organizations) consequently as follows. This puts our stated needs for future software organization capabilities into a larger context, thereby rationalizing them further.

Traditional management methods emphasize quality and price, and aim towards markets of economy (i.e., the larger the customer base, the cheaper the single product price is). Traditionally, organizations are managed as projects that are measured for being on time and within budget.

When companies face the challenge of digitalization, the market is being disrupted. This enables new players to enter the market place. This is visible e.g. when examining how digitalization has changed the travel business. New companies that operate only in network have entered to the market while some old players have vanished. Using software platforms as a mechanism to offer their service, new digital players are able to offer better or more focused services with less costs and with significantly less human workforce involved. The key in how to be successful in digital business is to focus on having the right service with better usability in place. The old project mode is replaced by the continuous offering of the service. The same trend is visible also with mobility and IoT disruptions as these businesses are also software businesses.

When digitalization has fully happened (like it has in the travel business today), the value chains are already very cost efficient with the help of created software. The markets have been re-distributed, and new digital brands have been created. New efficient players are able to dominate the markets by offering services with lesser prices. While at first glance, software, internet and digitalization seem to enable globalization and fewer but global players (such as Amazon offering books throughout the western world and Google offering services in multiple areas from data to books and mobile phone operating systems) it is not yet clear if the future will be dominated by a few global players only. The internet is also enabling long tails on businesses, i.e., enabling niche groups to reach special services and a wider variety of offering than what we have seen ever before. Current provider ranking services rate companies and their services based on price, but it could be of equal possibility that we start to rank companies by a multitude of values, and select those providers whose values match ours. Few digital services, like Zero Waste listing companies, who offer products that create less or no waste or various fair trade shops are examples of this possible future.

Figure 5 presents a summary of these findings of how digitalization changes businesses and the needed capabilities within organizations as well as their business models (c.f., Tables 3 and 4). In effect, it outlines strategic road mapping towards future software organizations reasoning our submissions in the chapter Research propositions.

Many challenges lie ahead, as many companies have trouble understanding the actual impacts of digital disruption for their businesses in the future complexities of the software world. When software is increasingly embedded and ubiquitous, people connect ever more with systems and with each other on many different levels (socio-tech-economic-environmental). Moreover, the connections are often real-time

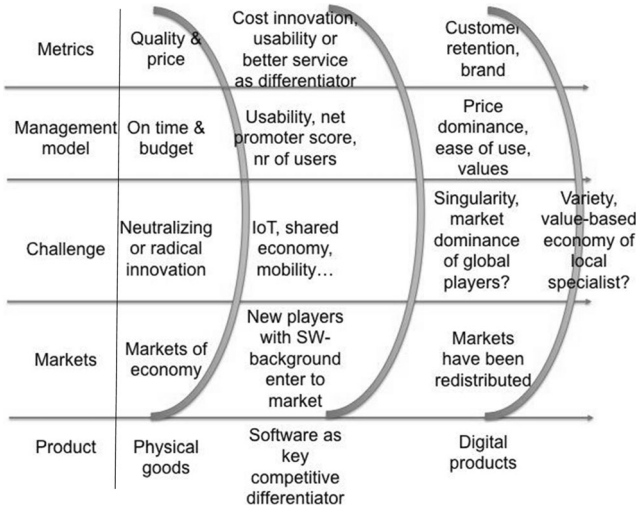

Fig. 5 How digital disruption changes companies, their business models and needed organizational capabilities

(Internet of Everything, IoE). These trends bring up totally new opportunities but also challenges for software development organizations, which have traditionally designed specific IT systems and separate software products.

It is hard to forecast where these changes will lead. While some people state that the future will lead to a singularity of some rare giant global players, learning the new management rules may help great local players and value-based economy if the consumers start to value more than just the price. With the use of software the whole value chain can become transparent – and thus the consumers may refuse to buy unethically produced goods or services. Equally, what will the future of work in software organizations be like, and what shall next-gen employees expect from their employers – given the considerable lack of competent software professionals?

From our point of view, future research interests include visualizing software organizations like depicted in Fig. 2. How can the invisible and intangible nature of software be illuminated, particularly in traditional companies transforming towards software organizations (c.f., Fig. 4)? Future research will benefit from including the measurements from chapter Key measurements. Further cases (chapter Results and experiences) would strengthen the validation on our design artifacts proposed in this paper. Future research may also include comparative studies with scaled agile frameworks (SAFe) [36, 37].

Implications

Based on the projected strategic changes value streams and the organization need to change, as well as the envisioned core capabilities (chapter capabilities) for software organizations. Table 5 suggests overall organizational change strategies for different types of organizations towards software organizations, which imply needs for new software competences.

It is imperative to realize that the particular context of the company affects the nature of the software organization. For instance media companies (like our Yle case) with new digital multichannel productions, power distribution companies offering real-time customer web displays of their electricity distribution situation, and paper machinery manufactures

Table 5 Change strategies for software organizations

Goals	Means	NEEDS for Software Competences and Resources / Roles
• If you are a big company: How to make your structures lightweight, and increase flow through the system?	By **fluidity**: • Decouple product architectures and teams. • Reduce organizational layers and move to flat organization(s). • Brake new product / service initiatives into smaller chunks (e.g., microservices). • Develop using smaller batches.	• Flexibility in software systems architecture and organization design • Feature teams, cross-functional teams • Unanimous prioritization of work
• If you are a traditional company: Understand the opportunities of cost reduction and innovation	By **adaptability**: • With the use of software • With modularity and cross-use of components / systems of systems • With integration of new software capabilities into existing systems	• Software-based value determination, software as the key enabling technology (KET) • Enlarging to new software-enabled business opportunities, new markets, new areas / technologies • Forming new alliances / co-operation (e.g., with software houses
• In ecosystem (digital economy):	By **flexibility**: • Build and streamline value streams for cost efficiency and new value. • Create efficient subcontracting and supplier networks. • Determine the (software) platform strategy (create platforms and/ or use available ones). • Data systems (e.g., big data, APIs)	• Value co-creation based on software core assets • Value creations based on wide offering (ecosystem) • Value creations through better software-based service (customization, predicting service interval, informatics, accumulated data from system or users) with value networks

offering remote condition monitoring services may each require new capabilities and consequent competence profiles compared to their current organization designs and ways of working. Notably many traditional competence descriptions and role titles may have to be specified (e.g., software project manager, business analyst).

Moreover, there are needs for new overarching and integrative digitalization competences and organizational capabilities:

- T-shaped competence
- Bridging capabilities

For example, the Industrial Internet of Things (IIoT) R&D requires new multidisciplinary approaches combining industrial automation, computer science, data communications, instrumentation, mechanical engineering and process technologies.

In general, IT capabilities are essential parts of digital transformations [9]. However, we posit that future software organizations need not only information technology, but also higher-level software capabilities for deep transformations. While developmental and transitional organizational changes could be achieved by merely shaping the current structures and processes, truly transformational changes require revisiting and challenging the fundamental business assumptions and organizational culture possibly stemming from the industrial age. Failing to recognize the new software-driven changes in the business rules, and what level of agility is needed in different industries may lead to poor competitiveness [28]. It follows that agility is a strategic organizational design decision [11, 38, 39].

Conclusions

In this design study paper, we have examined what particular capabilities (competences, resources) future software organizations need and why. Based on this, we presented our vision for such organizations and outlined the transformational path towards it. The real-life case of Yle Graphics illustrates some of those aspects in practice. In conclusion we have discussed what future changes companies should be able to foresee in order to be able to develop the necessary future capabilities when most companies become software companies.

Modern scaled agile frameworks SAFe and LeSS offer certain solution schemes for realizing those future software organization goals. The essence is to understand what particular key activities should be conducted in such organizations and how. That is, rather than stressing certain roles, we elevate the strategic design of the organization to a higher level, to comprehend how they contribute to the overall performance goals of software organizations.

Industrialism created organizational hierarchies and the benefit of scale. The Internet brought networks, and the benefit of connectivity and digitalization. We refrain here from presenting future scenarios, but it is striking to compare and contrast different industries of today. For instance, some media business companies are currently struggling with transforming their traditional businesses and operations to the new digital business models, and many manufacturing companies and industrial equipment providers are increasingly extending their offerings with software-enabled services. More generally, *deep tech* digital increases, as digital workforces and digital humanities are profoundly shaping both the software organization internal, as well as the external customer worlds. Sapient leaders for future software organizations are called for, like we have aspired in this paper.

References

1. Meijer E, Kapoor V (2014) The Responsive Enterprise: Embracing the Hacker Way. CACM 57(12):38–43
2. Chew K (2015) Digital Organizations of the Future. In: Collin J, Hiekkanen K, Korhonen JJ, Halén M, Itälä T, Helenius M (eds) IT Leadership in Transition – The Impact of Digitalization on Finnish Organizations. Aalto University publication series SCIENCE + TECHNOLOGY 7/2015, Helsinki.
3. Accenture (2016) Technology Vision for Industrial & Automotive. https://www.accenture.com/us-en/insight-industrial-automotive-retooling-digital-advantage
4. Laanti M, Kettunen P (2017) Future software organization − Agile goals and roles. In: Futures of a Complex World Conf. Book of Abstracts, pp 20. https://futuresconference2017.files.wordpress.com/2017/06/fcw-boa1.pdf. Accessed 11 Dec 2017
5. Porter ME, Heppelmann JE (2014) How Smart, Connected Products Are Transforming Competition. HBR November
6. Taivalsaari A, Mikkonen T (2017) Roadmap to the Programmable World: Software Challenges in the IoT Era. IEEE Softw 34(1):72–80
7. Holmström Olsson H, Alahyari H, Bosch J (2012) Climbing the "Stairway to Heaven" – multiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In: Proc. 38th Euromicro Conference on Software Engineering and Advanced Applications pp 392–399
8. Walker D, Lloyd-Walker B (2016) Understanding Collaboration in Integrated Forms of Project Delivery by Taking a Risk-Uncertainty Based Perspective. Adm Sci 6(3). https://doi.org/10.3390/admsci6030010
9. Korhonen JJ (2015) IT in Enterprise Transformation. In: Collin J et al (eds) IT Leadership in Transition – The Impact of

Digitalization on Finnish Organizations, pp 35–43. https://aaltodoc.aalto.fi/handle/123456789/16540. Accessed 08 Nov 2017
10. Version One (2017) 11th Annual State of Agile Report. http://stateofagile.versionone.com. Accessed 31 July 2017
11. Laanti M (2016) Miten ketteröitän ison organisaation? TIVI October. http://www.tivi.fi (in Finnish)
12. Hatch MJ (1997) Organization Theory. Oxford University Press, Oxford
13. Ahokangas P et al (2015) Need for Speed Strategic Research and Innovation Agenda. DIMECC Oy. http://www.n4s.fi/en/documents/articles/. Accessed 08 Nov 2017
14. Fitzgerald B, Stol K-J (2017) Continuous software engineering: A roadmap and agenda. J Syst Softw 123:176–189
15. Kettunen P, Ämmälä M, Sauvola T, Teppola S, Partanen J, Rontti S (2016) Towards Continuous Customer Satisfaction and Experience Management: A Measurement Framework Design Case in Wireless B2B Industry. In: Abrahamsson P et al (eds) Proc. PROFES. Springer, Berlin, pp 598–608
16. Kettunen P (2013) Bringing Total Quality in to Software Teams: A Frame for Higher Performance. In: Fitzgerald B et al (eds) Proc. LESS. Springer, Berlin, pp 48–64
17. Teppola S, Kettunen P, Matinlassi M, Partanen J (2016) Transparency Of Information To Improve Continuous Innovation Experimentation Performance. In: Proc. CINet Conf
18. Dingsøyr T, Fægri TE, Itkonen J (2014) What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development. In: Jedlitschka A et al (eds) Proc. PROFES. Springer, Berlin, pp 273–276
19. Terho H, Suonsyrjä S, Systä K, Mikkonen T (2017) Understanding the Relations Between Iterative Cycles in Software Engineering. In: Proc. of the 50th Hawaii International Conference on System Sciences, pp 5900–5909. doi: https://doi.org/10.24251/HICSS.2017.710
20. Tyrväinen P, Saarikallio M, Aho T, Lehtonen T, Paukeri R (2015) Metrics framework for cycle-time reduction in software value creation. In: Oberhauser R, Lavazza L, Mannaert H, Clyde S (eds) Proc. of the Tenth International Conference on Software Engineering Advances (ICSEA). IARIA, pp 220–227
21. Aaltonen M (2010) Emergence and Design in Foresight Methods. EFP Brief No. 180. http://www.foresight-platform.eu/brief/efp-brief-no-180-emergence-and-design-in-foresight-methods/
22. Reinertsen DG (2009) The Principles of Product Development Flow: Second Generation Lean Product Development. Celeritas Publishing, Redondo Beach
23. Gallaugher JM, Wang Y-M (2002) Understanding network effects in software markets: Evidence from web server pricing. MIS Q 26(4):303–327
24. Katz ML, Shapiro C (1994) Systems Competition and Network Effects. J Econ Perspect 8(2):93–115
25. Brown SL, Eisenhardt KM (1998) Competing on the Edge: Strategy as Structured Chaos. HBS Press, Brighton
26. Laukkanen S (2012) Making Sense of Ambidexterity. Dissertation, Hanken School of Economics, Finland
27. Power B (2014) How GE Applies Lean Startup Practices. https://hbr.org/2014/04/how-ge-applies-lean-startup-practices. Accessed 31 July 2017
28. Kusek D, Leonhard G (2005) The Future of Music. Berklee Press, Boston
29. Day GS (1994) The Capabilities of Market-Driven Organizations. J Mark 58:37–52
30. DDI (2015) Digital Disruption of Industry. http://www.aka.fi/en/strategic-research-funding/programmes/programmes-20152017/disruptive-technologies-and-changing-institutions/ddi/. Accessed 31 July 2017
31. Collin J, Eloranta E, Holmström J (2009) How to design the right supply chain for your customers. Supply Chain Management: An International Journal 14(6):411–417
32. Porter ME, Heppelmann JE (2015) How Smart. Connected Products Are Transforming Companies, HBR October
33. Overby E, Bharadwaj A, Sambamurthy V (2006) Enterprise agility and the enabling role of information technology. Eur J Inf Syst 15:120–131
34. Tahvanainen AJ, Adriaens P, Kotiranta A (2016) Growing Pains of Industrial Renewal – Case Nordic Cleantech. ETLA (The Research Institute of the Finnish Economy) Reports 58. https://pub.etla.fi/ETLA-Raportit-Reports-58.pdf
35. Martinsuo M et al (2016) Future Industrial Services. Final Report, DIMECC Oy http://hightech.dimecc.com/results/final-report-futis-future-industrial-services
36. Paasivaara M (2017) Adopting SAFe to scale agile in a globally distributed organization. In: Marczak S et al (eds) Proc. ICGSE. ACM, New York, pp 36–40
37. Luhtala K, Korhonen JJ (2015) Case RAY: Playing It Digital. In: Collin J et al (eds) IT Leadership in Transition – The Impact of Digitalization on Finnish Organizations, pp 109–116. https://aaltodoc.aalto.fi/handle/123456789/16540. Accessed 08 Nov 2017
38. Kettunen P, Laanti M (2008) Combining Agile Software Projects and Large-scale Organizational Agility. Software Process: Improvement and Practice 13(2):183–193
39. Kettunen P (2007) Extending Software Project Agility with New Product Development Enterprise Agility. Software Process: Improvement and Practice 12(6):541–548