MANAGING ICT DEVELOPMENT:

Selection Between Agile, Hybrid and Waterfall Methodologies

midaGon

Success is a Choice

CONTENT

BACKGROUND AND PURPOSE

The Midagon white paper on Complex ICT and Digital Transformations addresses the question of whether the choice of methodology makes the difference between success or failure. The differences between the management methodologies, as well as their strengths and weaknesses, were addressed in the white paper Description of methodologies to manage ICT development.

The nature of the work being done determines the optimal methodology. Furthermore, one single methodology is not always enough, and sometimes it makes sense to use them in parallel within a development program or portfolio. The purpose of this white paper is to provide guidelines for the reader to select the suitable methodologies for different tasks and situations.

The company first needs to understand whether the ICT development is a one-off exercise or whether it is continuous by nature. A project serves its purpose in a unique change situation. A long-lasting or continuous evolutionary change requires more agile management methods, continuous delivery, and even managing the whole company as an agile innovation system.

This white paper is divided into three parts:

> **Part 1** presents guidelines for selecting the methodology based on the context and decision-making situation and summarizes this approach with a decision tree. Our purpose is to help the reader find the ideal methodology that fits the broader management context.
>
> **Part 2** presents, as an alternate or complementary approach to the above, guidelines for selecting the methodology based on priorities and constraints related to the traditional project management components. Our purpose is to help the reader choose between the options available in real-life situations of managing ICT development.
>
> **Part 3** presents aspects to consider in the selection of the methodology and different ways to ensure success in ICT development. Our purpose is to help the reader to utilize the strengths and to overcome the weaknesses of different methodologies. A special focus is in making the agile approach work, since it is currently receiving considerable hype. Turning the company more agile overall is a major transformation. The most important management practices addressed are those of leading people and financial management.

The success and failure factors for managing an ICT transformation are independent of the methodology chosen, which means that selecting a specific methodology is not a guarantee of success. However, the factors are mostly related to leadership and management. ICT development is not managed in isolation but within the management culture of the company. Thus, the management culture and practices need to be considered, when selecting and applying the methodology.

INTRODUCTION: RECAP OF THE CLOSED LOOP MANAGEMENT PROCESS AND ICT

In our previous white paper Managing ICT development – Description of the methodologies, we presented two frameworks relevant for the selection of the methodology. We also presented tables on how the basic methodologies of a waterfall project, a hybrid project, and agile development relate to these frameworks. The frameworks are:

1. The cyclical "closed-loop" PDCA process of management with the stages Analyze/Adjust – Plan/Propose – Decide/Do/Delegate – Check/Complete

2. The traditional project management components: Scope, Time, Cost, People, and Quality (to manage Risk and Change)

The figure below (*Figure 1*) combines these two frameworks by placing the components roughly into the PDCA cycle. The scope (or requirements) needs to be determined to detail the mission and to enable planning. The plan contains the people involved and timing, as well as the cost estimate and the arrangements to ensure quality. The work is done according to the plan or, if there are surprises, in another way that achieves the mission and purpose of development. The completed deliverables, their quality, and the cost are checked in order to adjust the approach or plan.

Business management, change management, and ICT development management are usually seen as separate domains with different processes and tools. Business management is considered cyclical and tightly connected to the company's rolling financial planning. At the same time, ICT implementations are traditionally considered technical and linear one-time efforts, that are mostly isolated somewhere within an ICT development function. However, it is useful to recognize the cyclical nature of all management. The essential element is communication – the transfer of information from the previous stage of the management process to the following one.

Connecting the four stages into a closed loop enables the production of the deliverable and the creation of the desired change, according to the Mission. Gaps between the stages lead to failure. For example, the user needs may be listed from the business process perspective, but the solution design documented from the application module perspective. It may be difficult for the customer to connect these two, which may lead to surprises and disappointments during testing. The check stage of development – testing – ensures that the solution technically works as designed. However, in order to align the development deliverables successfully with business objectives, we need to verify that the solution works with real users and measure if the expected business benefits were realized. This requires a feedback loop from users and operations – a part of the program or product level management cycle.

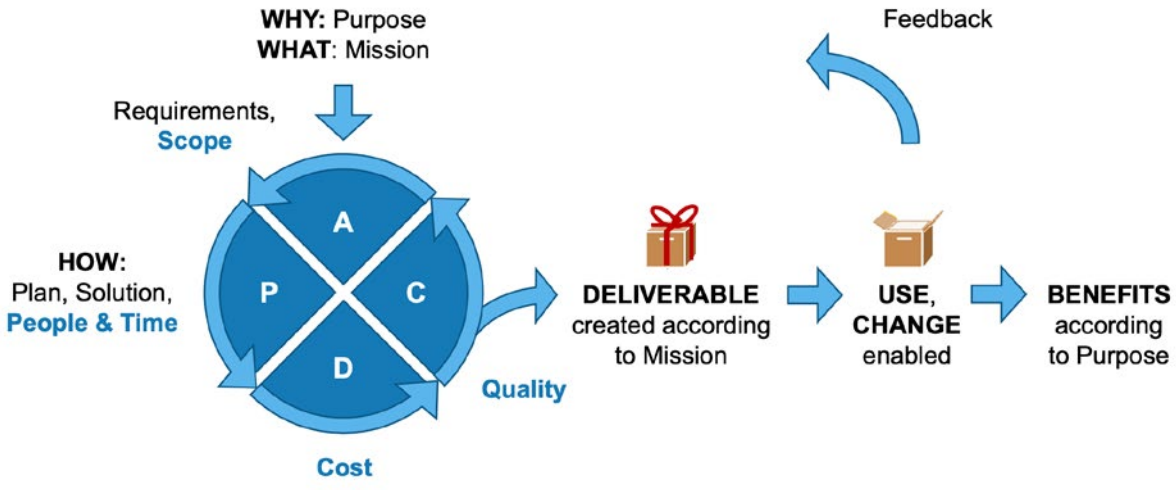

Figure 1: Management cycle and components

PART 1 – SELECTION OF METHODOLOGY BASED ON CONTEXT AND SITUATION

In the following chapters, we present guidelines for selecting the ideal methodology based on the context and decision-making situation – and summarize this approach with a decision tree.

Business context and the change required

In our white paper Introduction to methodologies, we compared the basic methodologies of managing ICT development. We also stated the simplest rule in selecting the methodology:

- Projects with their temporary teams are suitable for managing one-off changes.
- Agile development with more constant team allocation is suitable for continuous improvement.

Typical examples of one-off changes are mergers and acquisitions of companies with the following selections of ICT tools. An example of a goal requiring continuous improvement is the creation of continuous digital services for customers. This initial analysis based on the business context is often rather simple to make.

An initiative that contains both one-off changes and a process for continuous development work may require a program with different methodologies used in combination. The one-off changes may involve physical construction or otherwise contain decisions that are difficult to reverse. Software configuration related decisions, on the other hand, are typically easier to reverse, and the development can be more flexible and iterative.

An example is a transformation of a traditional Data Center to a Software Driven Data Center (SDDC) – a business renewal that includes facility renovation or construction, moving and setup of physical equipment, software configuration, and a transformation of processes to a cloud-based service for customers.

Methodology: A smaller scale renewal can be implemented as a hybrid project. A large transformation of this kind can be run as a hybrid program. The main phases of the program might be the following:

1. Physical construction of the facility and the physical move and setup of the data center – a traditional project of construction/renovation and network/data engineering:
   - Logistics, rooms, power supply, security, cooling
   - Transport and/or installation of the server and network equipment into their rooms and the setup of capacity and payloads

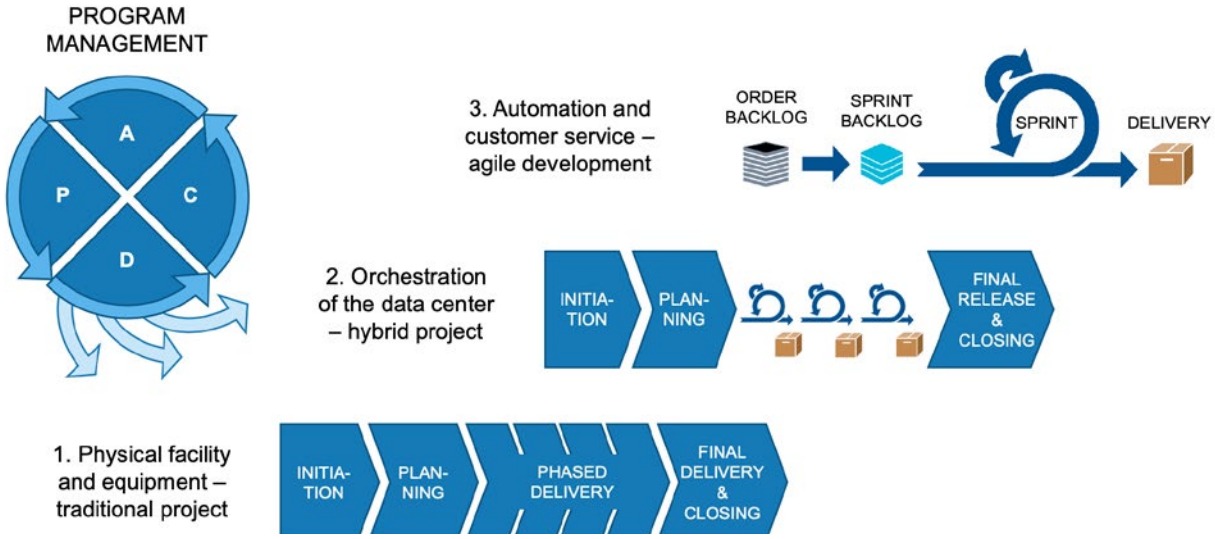

Figure 2: Hybrid program to implement a Software Driven Data Center

2. Orchestration of the data center, setting up virtual loads inside the hardware per customer need and request – a hybrid project including proper planning of capacity and the service initiation tasks in an agile mode

3. Automation of control and modernization of processes to form a Software Driven Data Center (SDDC) – sales of capacity, agile development for fast execution of customer orders, and continuous improvement.

The simplified figure on the previous page *(Figure 2, p. 5)* illustrates the hybrid program of renewing a data center. Change management on the program level is important, since changes in the customer requirements and software solutions may affect the scope of the facility and equipment project. It is sometimes difficult to recognize the need to combine methodologies, since the main debate in recent years has been between the pure waterfall project and the pure agile development.

Clarity of the mission

Sometimes the mission cannot be determined merely based on the business context. We, therefore, need more analysis to determine its nature. In the ideal Analyze/Adjust stage of the management cycle, the development organization receives a clear purpose and mission from the business management. The mission should contain the deliverables to be created and the change enabled by ICT development. The following list *(Table 1)* contains criteria that can be used to determine the clarity of the mission.

Methodology: If the mission is clear, and there is a need for a one-off change, a project is the natural selection. As a simple solution to meet simple or well understood needs, a waterfall project can be used. An example of this case is the implementation of a software product that contains all the functionality needed out-of-the-box and only requires configuration.

However, even if the purpose and mission are clear, the solution required may be large and include many dependencies and/or risks. In this case, it is necessary to plan the change and the ICT implementation carefully, make a thorough analysis of needs, and design the solution in detail. In order to avoid risk, it also makes sense to build the solution incrementally, if possible, and to apply agile methods to any newly developed applications or enhancements to software products. A hybrid project supports this approach.

If the mission is not clear, it makes sense to verify the initial idea with feasibility studies and prototypes, and to build the solution iteratively. Agile development or a hybrid project can be used for this.

Several approaches can be used in parallel. The business change may require several ICT streams with differences in the nature and pace of development. We can also manage a one-off task as a project and then hand over the deliverable to agile teams for maintenance and continuous improvement. Program management may be required to handle the dependencies. A program level is also useful, when following a

Clear mission:	Unclear mission:
• Existing business process	• New business process
• Minor change, not much to ask from business	• Major change, a lot to ask from business
• Change clearly described, e.g. business concept	• Change described as direction of improvement
• Valid quantitative metrics	• No quantitative metrics
• Solution initially described in a solution concept	• Only a vague idea of the solution
• Mature technology we are familiar with	• New technology we are not familiar with

Table 1: Criteria to determine the clarity of the mission

phased roadmap and managing larger-scale agile development.

If there is a large investment into development and there is a lot of work to do, the need for effective management increases. The frameworks presented so far may not be enough for effective decision-making and a wise selection of methodology. Therefore, we introduce one more.

Cynefin Framework

The feasibility of a methodology in managing change depends on the situation. As a third management framework, we present the Cynefin Framework by David Snowden et al. There are four types of decision-making situations: simple, complicated, complex and chaotic. Moving from a simple toward a chaotic situation means less predictability and more uncertainty. In simple situations, after the decision between pre-planned alternatives, action is straight-forward. Chaotic situations again call for straight-forward action to make sense of what is happening and to restore order. The complex and complicated situations require more sophisticated management. The following figure *(Figure 3)* summarizes the four different situations with their primary courses of action.

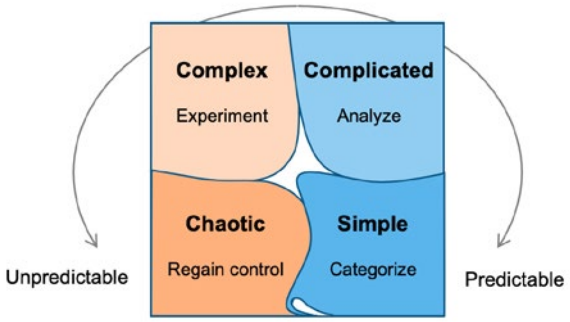

Figure 3: Cynefin framework

Simple situation ⇨ Various simple methods

In a simple situation, the causalities are known. We need to categorize the task at hand, choose the best practice approach and implement according to it.

An example is the renewal of an outdated application with mature technology by re-building the old functionality as-is. The work can be routinely handled according to pre-set rules and a “cookbook” approach. Most of the planning has already been done through the development of best practices. The following implementation is straight-forward and controlled by using the best practices.

Another example is the enhancement of a web shop point solution, where single features can be added, changed, or cancelled, as needed. An additional example is a customer service request that originates from the operations and is received by the development team.

Methodology: After the categorization of the case, the management chooses the appropriate method with the restrictions discussed in subsequent chapters. The one-off case of renewing an outdated application can be handled in a simple waterfall project. One needs to be cautious, however, about applying this approach to any sizeable changes or about initiating long projects. If the task is not small and cannot be accomplished quickly, it might not be so obvious and risk free after all.

Agile development is suitable for the continuous improvement of a web shop solution. More demanding customer service requests can also be handled with an agile team that deploys their changes into production continuously. However, developers are sometimes reluctant to handle simple service requests – they can also be handled by a customer service team as routine work orders.

Complicated situation ⇨ Hybrid project or program

In a complicated situation, the causalities are potentially knowable. We need to analyze the system using experts to understand all its dependencies and constraints. Alternative scenarios can be used to cover uncertainties.

An example of a complicated case is combining the application landscapes of two companies merged together. A thorough requirements analysis is done to understand the dependencies between different parts and the trade-offs related to alternatives. The needs and the solution required are truly unique, which sets the case apart from the simple ones. A considerable amount of time is spent on planning the implementation. Many possible good practices exist covering single parts.

Methodology: A large and complicated ICT transformation calls for a project to manage the dependencies. If the change contains new parts in the architecture requiring innovation, an iterative approach can be used. The combination can be handled by organizing a hybrid project that contains a thorough planning phase and a closely controlled execution phase using agile team management methods. The analysis and planning are often iterative. A rough solution concept is created and further elaborated into the solution design and implementation plan.

Using multiple methodologies in parallel allows us to acquire one-off deliverables as projects, while maintaining flexibility and learning as the work proceeds. However, one needs to be cautious about dependencies between the projects and agile development. Agile development alone is not well-suited to manage dependencies across teams and different development tasks. In the case of multiple methodologies, it makes sense to set up a program to collect the projects and other types of work under joint management. The Scale Agile Framework (SAFe) 3 includes a planning event before every program increment and a "Large solution" level for coordination. However, we still need to manage the scope well. A systematic hierarchy of requirements is needed to facilitate the necessary exchange of information between the teams.

Complex situation ⇨ Agile development or Innovation system

In a complex situation, the environment is in constant flux – understanding causalities comes only after observing the effects of action. We need to experiment with the customers – to create a hypothesis and a "minimum viable product" to verify it. It is necessary to test different versions with users to see which one works better.

A situation is complex, for example, with a new kind of software service or a new market, where the patterns of user behavior are not understood in detail and are constantly changing. The environment calls for an experimental "fail fast" style of management of a "Lean startup". The company makes assumptions on customer value and preference, and verifies them through designed experiments. A working system for gathering usage metrics and feedback is necessary to enable fact-based decision making. The critical decisions of either "pivoting", changing the approach, or "persevering" on the chosen course are done in the Adjust stage of the management cycle.

Methodology: Smaller-scale development of a new system into the landscape can be handled through agile development. A large-scale ICT solution with many uncertainties calls for an agile program. This means closing the management process with an effective feedback loop and steering the development through concept work and backlog prioritization on the program level.

Growing a company in a volatile business environment calls for an agile innovation system. Responding fast to market changes is enabled by accelerating the whole delivery pipeline. Agile development is connected to Operations through Continuous delivery – DevOps. New features or versions are developed fast and released to customers on demand. An efficient feedback loop enables learning and fact-based decision-making. Agile business planning again directs the development. Eskola & Töyrylä call this kind of an innovation system Business Development Operations – "BizDevOps". The following figure (*Figure 4*) describes the idea.

Figure 4: Business Development Operations (BizDevOps)

Chaotic situation ⇨ Crisis management

In a chaotic situation no causalities are known – the situation is completely unclear to us. We need to regain control: this is done by finding out what is happening and acting fast. The objective is first to prevent damage and then to reduce uncertainty – to turn the situation into a better known one.

An example of a chaotic system is a customer service failing without knowledge of the root cause. There is typically not much time for planning. We need to contact the customer and see what really is happening. In a chaotic situation, we are not controlling events through our action, or the effects of our action are not clearly known to us, because the feedback loop is missing.

Methodology: A chaotic situation calls for crisis management, instead of elaborate project planning or continuous improvement. It is important to act fast to prevent damage and to establish situational awareness. We may have to create an input to the system to observe the effect. This requires a working feedback loop with constant customer contact and/or metrics to be set up. At the same time, acting fast shows people that somebody is in control. Making it through a chaotic situation means entering a new one of a different type, with the problem possibly divided into several parts of different types.

How to recognize the situation

In selecting the methodology for large ICT transformations, the Complex and Complicated situations are the more interesting ones to consider. A company may find itself in a complex business situation but still have complicated ICT transformations that, by nature, are run as projects.

It is sometimes necessary to manage parallel changes in a portfolio of initiatives that use different methodologies. Building the management system for this combination is not a trivial task. Possible dependencies between the agile development and the waterfall projects may force the whole company to proceed at the pace of the slowest stream.

In any case, special emphasis must be placed on people management, synchronization of decision-making, sensible metrics, and financial management structures. The following table *(Table 2)* presents attributes that can be used to differentiate between a complex and a complicated situation.

Complex situation ⇨Agile development or innovation system	Complicated situation ⇨ Hybrid project or program
• Accelerating change in market needs – fast product changes are required	• Stable market, significant changes happen relatively seldomly
• Difficult to predict market/system behavior – assume and experiment to learn	• Data available to predict market/system behavior
• High level of risk acceptable in starting a new business	• Low level of risk acceptable in fine-tuning and protecting the old business
• “Failing fast” attitude to experiments – lower cost of failure through fast correction	• Experiments not accepted, and/or high cost of a failure – e.g. patient data mgmt system
• Low level of regulatory control	• High level of regulatory control to prevent risk or ensure quality
• Reversing decisions is possible/ cheap – e.g. software development	• Reversing decisions is very expensive or not possible – e.g. metro tunnel excavations
• Only a few stakeholders involved in an experiment – one provider, one customer	• High number of stakeholders involved – need to plan, commit, and coordinate
• After the first delivery/release, others follow with increased competence and emerging good practices	• Unique solution and one-time delivery – best practices utilized for parts
• Gradual development is possible with a Minimum Viable Product followed by enhancements	• The whole solution and full functionality are required at the Final delivery or one Release/Go-live

Table 2: Attributes that can be used to differentiate between a comlex and complicated situation

Decision tree based on the situation

The following figure (*Figure 5*) summarizes the thumb rules of selecting the methodology presented so far. It combines our understanding of the business context and mission with the rules of selecting the methodology based on the decision-making situation.

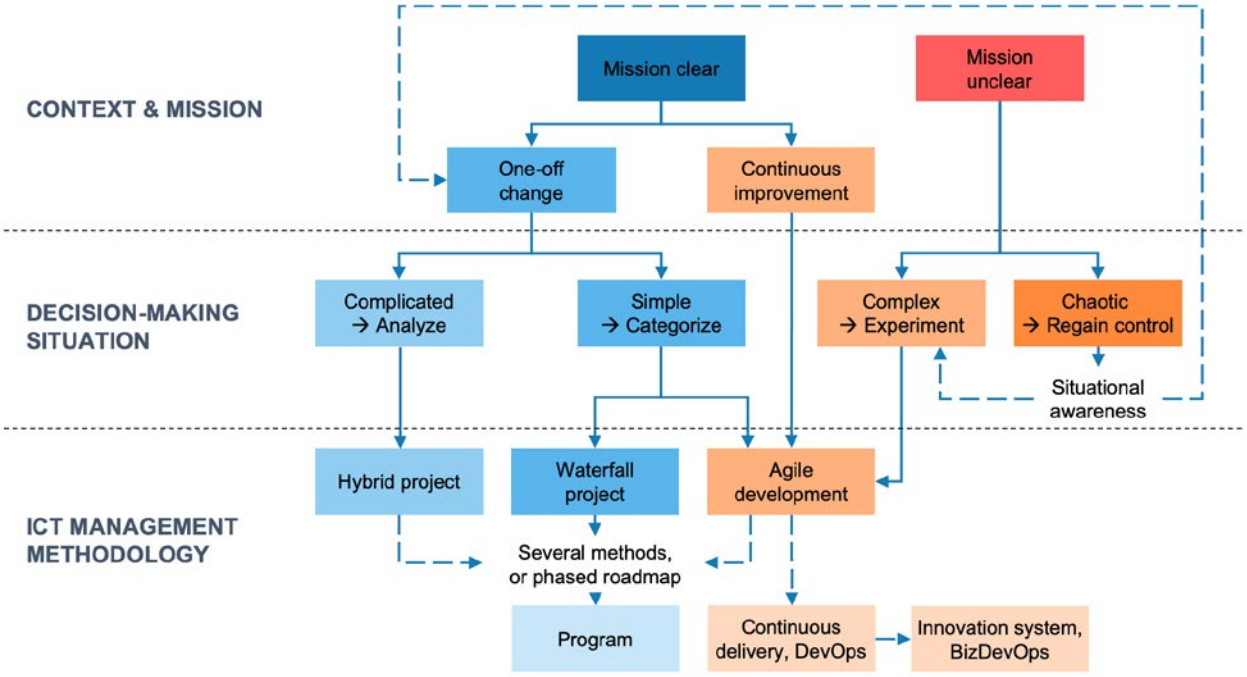

Figure 5: Selection of methodology based on context and situation

PART 2 – SELECTION OF METHODOLOGY BASED ON MANAGEMENT PRIORITIES

In the following chapters, we give guidelines for selecting the methodology based on priorities and constraints related to traditional project management. They can be regarded as a “sanity check” complementing the one presented above, or as an alternate approach to choose between the options available in non-ideal real-life situations.

Fixing the components

The choices in managing ICT development are often limited, with at least one of the components – Scope (S), Time/schedule (T), People (P), Quality (Q), or Cost (C) – prioritized and fixed. Thus, the flexibility and contingencies needed to control risk and adapt to change must be achieved by adjusting the other components.

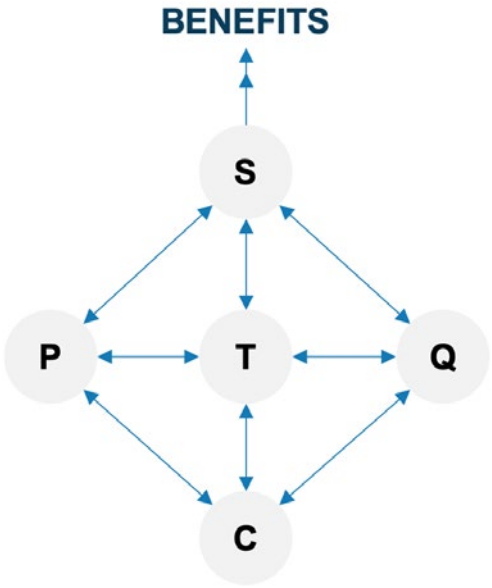

Figure 6: Management components

Components may also be fixed as a result of the choice of methodology. To summarize, a waterfall project fixes the scope up-front (through requirements) and then derives the demand for people, time, quality, and cost, based on the scope. Agile development fixes the people, cost, and quality up-front, time-boxes the work into sprints, and lets the scope vary. These effects are embedded in the methodologies. Therefore, in the following chapters, we focus on the selection of the methodology after the component prioritization decisions by the management.

The genuinely compelling reasons to fix a component are based on the environment, requirements, solution, or resources available. Here are some examples:

- **Legal/contractual** requirements or restrictions – e.g. the new ERP solution needs to be up and running by a certain deadline stated in a divestment agreement. ⇨ Time fixed
- **Large business case** – e.g. the possibilities in a new business area are so big and success there so important to the value of the company, that it makes sense to dedicate a highly competent team to develop the solution there. ⇨ People fixed
- **Large risk** – e.g. inadequate functionality in securing the privacy of patient data could prevent a healthcare system vendor from competing in Europe as GDPR becomes effective. ⇨ Scope and quality fixed, probably also time
- **Customer demands** – e.g. the customer has a tight budget, and only the vendors adapting to it have a chance of winning the contract. ⇨ Cost fixed
- **Technical limitations** – e.g. only a few integration technologies can be used with a certain old system. ⇨ Scope fixed

There are optional reasons to fix a component (but often considered compelling by the organization):

- **Assumptions** on client behavior, team performance, technological functionality or the like – not verified and thus not necessarily true ⇨ Scope and/or Time fixed
- **Promises** given by the management ⇨ Time fixed
- **Original business cases and plans** with benefits, schedules, and budgets we are unwilling to revise, although experience has proven them obsolete – often related to personal incentives and the need to report success upwards ⇨ Scope, Time, Cost, or People fixed

- **HIPPO** – highest income paid person's opinion that directs towards manager-based, rather than fact-based decision-making.

Fixing several components together, especially together with the scope, makes the management of ICT development more difficult. Therefore, it should be carefully considered whether the optional reasons for fixing a component are valid.

Fixing the components directly affects which methodology will work best. Therefore, a mission that contains fixed components limits the choice of methodology. On the other hand, the management often makes decisions on fixing some components based on their own judgement. The methodology selected naturally allows fixing some components and not others.

In the following section, cases of fixing components are presented with the resulting recommendations for a suitable methodology. Furthermore, the possibilities of maintaining flexibility and tradeoffs with the remaining free components are examined.

Scope fixed – Project

The scope can be fixed when there is a detailed understanding of the output (e.g. the functionality of an information system), the architecture, and the constraints. A typical example is the replacement of a system at the end of its life cycle, without implementing any new functionality. Another example is implementing functionality to enhance data privacy, according to the requirements of the GDPR.

Typically, also the quality is partly fixed. The demands of compliance with external regulations or architectural policies might be strict. Detailed requirements can cover this component, and there may be an architect team in the organization to coordinate planning and to review and approve the design. However, fixing the quality completely is usually not desirable because of the high cost.

Methodology: Here it is natural to define the scope precisely and completely with all its dependencies. This calls for a time-consuming requirements analysis, which is typical for a waterfall project. However, time-boxed sprints provide a better control of progress and the ability to deliver the solution incrementally. Therefore, a hybrid project is recommended.

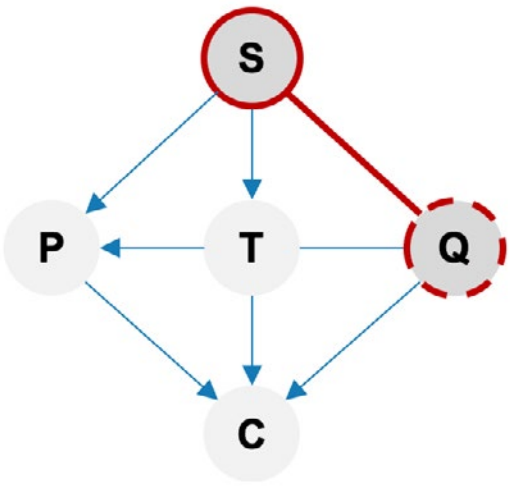

Figure 7: Scope fixed and partly Quality

Flexibility: Only partially allocated people would require more time, whereas fully allocated people could be faster with the same cost. Knowledgeable in-house experts could be effective, but their availability is typically a problem. More junior people require more time and control. Reducing the cost can mean lowering the velocity of the team.

Accelerating the work would mean increasing the velocity (#scope items /time) of the team. Resources can be added by increasing the amount of people or by increasing competence. An increase in cost will typically follow. Waterfall-style precise analyses and definitions help in introducing and briefing new people.

Scope and time fixed – Project

It is very common that along with the scope, the time of the change is also fixed. A typical example of fixing the schedule is a divestment situation with a fixed go-live date for having the acquired business unit out of the selling company's systems. Another example is the forced schedule of a system upgrade or replacement, due to the termination of support for the software version used.

Fixing the time can also be natural from the investment perspective: the profitability can be calculated as a net present value (NPV), which is affected by the time spent. Another reason to fix a schedule is to create a sense of urgency and a spirit of getting things done. If there are no target

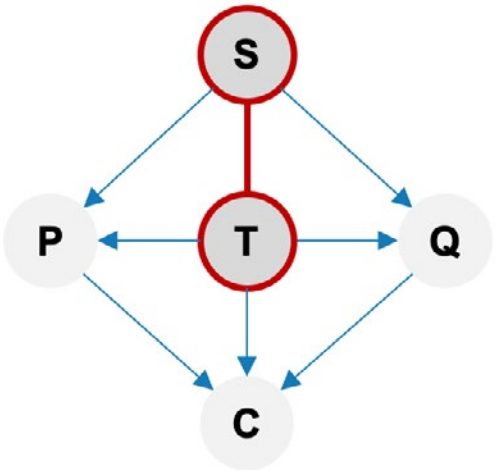

Figure 8: Scope and Time fixed

timelines, it is easy to give a lower priority to the task and to prolong things. The development unit might sometimes be forced to keep a schedule promised by the top management to customers, owners, or the media.

Methodology: Fixing the schedule brings an additional constraint. It may increase the risk of failure to meet the objective. However, having the scope known and fixed still leads in the direction of using a project. If possible, the risk related to a large fixed scope can be lowered by using a hybrid project – planning incremental deliveries and using agile style team management with daily check meetings. In the cases described above, the release of the solution may need to take place in large batches. Therefore, the increments would be built and tested in sprints, but deployed to production as a combined final delivery.

If there is minimal time for the analysis of requirements and design, the team can first implement a “quick and dirty” solution. This initial step can also be considered as the “minimum viable product” which forms the basis for further improvement releases in a hybrid project or through agile development as maintenance. A hybrid project with early incremental deliveries, that form intermediate deadlines, also helps in creating a spirit of getting things done.

Flexibility: If the time runs out after design and build, the team may have to at least temporarily sacrifice quality and to cut down on testing. A better response to the risk of a delay is to resource the project heavily with all the required competence and a strong enough team. Additional people can be found from consultant partners. Another alternative is to keep the same team and have them work overtime. Strengthening the team or working evenings naturally increases the cost. Even an agile team can be ordered to work overtime. However, this is not consistent with the agile principle of working at a sustainable pace and will probably raise resistance.

Should the timeline prove to be just too tight, it may sometimes be eased with money. An example is a divestment situation, where the selling company is paid for extra months of having the acquired unit in their systems. Careful evaluation is needed in that case – more cost and more time (with more trust on good quality) vs. more cost within the planned timeline (with less trust on quality).

Time fixed – Hybrid project or agile development

When the time is fixed but the scope isn’t, it is typically important to show results and to make a change quickly. A development unit might be given a mission to improve the profitability or efficiency in a business area. Resources are allocated to this task, and the course of action and solution requirements are left for the development team to figure out. Depending on the solution, the cost can vary.

The development management might fix the schedule themselves to raise the sense of urgency and ensure some results in a reasonable time. Alternatively, they might be forced to keep

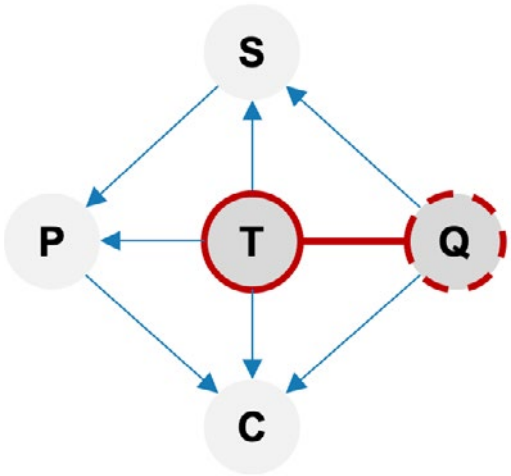

Figure 9: Time fixed and good Quality

a schedule promised by top management to customers, owners, or the media. Sometimes the schedule is simply fitted naturally into the company's management cycle, and the mission is to be attained within the normal strategic and financial planning horizon.

Methodology: A one-off change calls for a project with a temporary team. Since the solution scope is flexible, it should be a hybrid project. If there is an agile team available, the task can also be given for agile development. We can also continue as agile development after the initial results of a limited-time project. It is natural to start with a joint requirements analysis in order to collect a backlog of software features and to prioritize them for the best value in the limited time. The prioritization can be done, for example, using an estimation tool called Weighted Shortest Job First (WSJF)

Flexibility: If the schedule is tight and there is pressure to produce a quick win, it is advisable to avoid a lengthy solution definition phase. It is instead necessary to address the most urgent needs and to produce a Minimum Viable Product (MVP), which can then be enhanced in coming releases. MVP means delivering limited functionality with good quality, not wide functionality done badly. We can consider quality also to be fixed as good, which is natural for agile development.

Increasing the scope within a limited time means increasing the size of the team and the cost of development. However, implementing a broader functionality in a limited time may mean larger releases and risky "big bang" go-lives that are typical for waterfall projects. They often fail, and the original schedule is ultimately exceeded. Deciding on a large suite of functionality up-front might not be wise, if only a portion can be implemented in the give time frame. A software platform that allows us to add modules gradually or to build applications at will is desirable here. Cloud platforms are now making this approach more feasible than before.

Cost fixed – Agile development or project

Traditionally, fixing the cost has been done through calculating a business case or an investment plan and then fixing a project budget. A business case includes the expected benefits and at least a rough business concept and high-level ICT solution scope. In continuous improvement, a sum of money is allocated into the improvement of a business area – to be spent where it is most beneficial.

With the budget, we can decide whether to acquire a software product or to build a tailored solution. It makes sense to produce good quality, since the cost of bad quality is considered waste. We can fund existing development or form a new team, depending on the competence and availability of people.

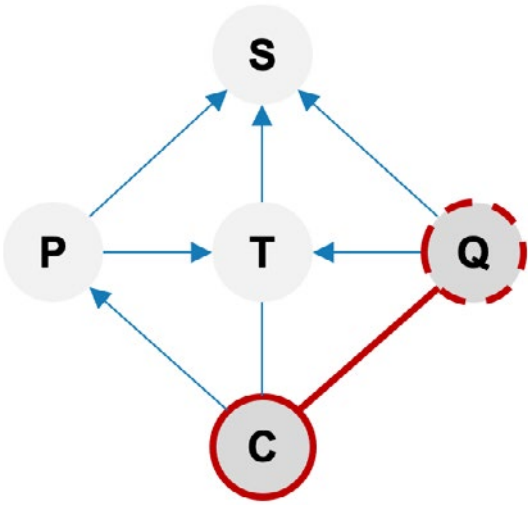

Figure 10: Cost fixed and good Quality

Methodology: Fixing the cost does not favor using any methodology. Instead, the decision is based on the solution or the people available. Since the people component is not fixed in this case, we either have a temporary project team or a temporary allocation of a permanent agile team through their backlog.

Traditionally, a detailed business case has led to detailed requirements and a waterfall project. If the solution is by nature better implemented through a "cookbook" approach and a waterfall project, that is what we should do. However, a large solution can be delivered in increments through sub-projects to reduce risk. We can hire an external team for a project and contractually tie them to the budget.

The alternatives of using an internal team are limited by their competence. Junior workers may require more detailed waterfall-style documentation. Agile methods direct to self-organization, but not all workers are capable of that. However, the close control of progress can be achieved by using a hybrid project approach that combines the detailed instructions of a waterfall with the visibility of agile sprints.

Agile development includes constant re-evaluation of the backlog that affects the scope of development and the value created. Cost per time unit is in control through the stable team allocation. We can stay within budget and release the results of the sprints frequently. When the money runs out, we have delivered functionality and can avoid the risk of having an incomplete solution at that point. A hybrid project offers the same advantage.

It is now common that the development partner is used to and prefers an agile methodology, especially when creating customized software. It makes sense to adapt to this. If the customer company itself is more accustomed to projects, a hybrid approach can be suitable.

Flexibility: Any problems with the implementation will require extra work. In a project, this needs to be handled either by consuming the contingency buffer or removing elements from the scope. In a waterfall project, the risk of exceeding the budget may be avoided contractually. In agile development, fixing problems in one area pushes the development in other areas further and thus lengthens the time required.

We usually need to adapt to the availability of people. A highly capable but expensive project team can deliver fast. A less capable and cheaper team – smaller, or perhaps including either junior or partially allocated people – will probably deliver with about the same cost but more slowly.

People and cost fixed – Agile development

The team may be fixed because we want to rely on internal people to retain them and to build lasting expertise on an area of operations or technology. It may also mean either avoiding a more costly external team or alternatively committing to a selected partner for close cooperation and high quality.

Naturally, a stable team is necessary when the development is continuous improvement by nature, as opposed to one-off exercises. This is typically the case with a company constantly providing new versions of software or digital service. However, the volume of development does not always justify a permanent team.

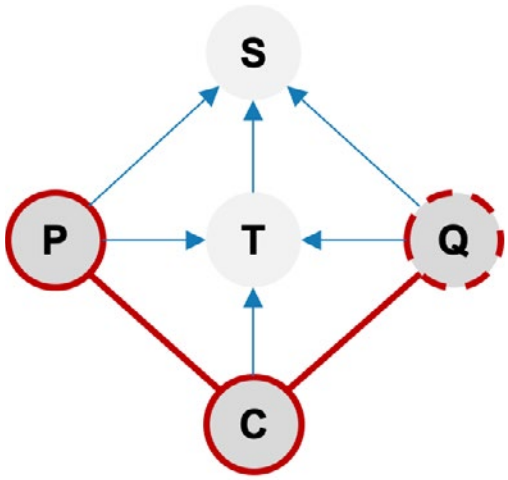

Figure 11: People and Cost fixed

Methodology: A way to ensure the availability of people and to simultaneously exercise cost control is to organize a “permanent” fully allocated development team. This fixes the labor cost per time unit. It directs toward agile development, as opposed to a temporary project team. When organizing a permanent team for a longer period, the company needs to take measures to ensure that the people stay in the company and remain engaged in the development the whole planned time.

When the scope is flexible, it again makes sense to organize the work as continuous agile development. The Product Owner or Product Manager forms the scope by prioritizing and selecting a feature backlog for the team, assisted by architects. The selection of people has a direct effect on time and value, since a highly competent team understands better and delivers faster. The cost per time unit may be higher, but the total cost of a development scope is the same. The business value may be received earlier. This happens with any methodology. However, the agile style of constant testing and demoing is a way to ensure that the fixed team delivers value.

Flexibility: Agile development is time-boxed but also continuous by nature. The delivery time of a feature or a solution depends on the requirements and the velocity of the team. With a competent team and with the close participation of the customer, a good understanding of the requirements should be built in. The team might be expected to work in increments and deliver limited functionality with good usability and technical quality. They might alternatively be expected to work iteratively and deliver a wide solution with

lower quality to be improved later. The real flexibility is provided by prioritizing the requirements from the backlog and selecting the sprint scopes. On a higher level, funding can be re-allocated between product streams, and the permanent development team can be directed to work on the solution area prioritized by the management.

Scope, time, and cost fixed – Project

Setting a project objective by defining the output and by further dictating all the means of creating it belongs to the waterfall project tradition. The actual output, the scope or the "product" is the natural starting point. Planning the project, in this case, includes the initial estimate of the people needed, the time spent and, therefore, the budget allocated. We also consider time and cost to be fixed.

Methodology: The fixed requirements in our scope lead us to set up a project. In order to prepare for changing circumstances, project management requires a "contingency", a buffer reserve for the cost and time. This is all natural for waterfall projects.

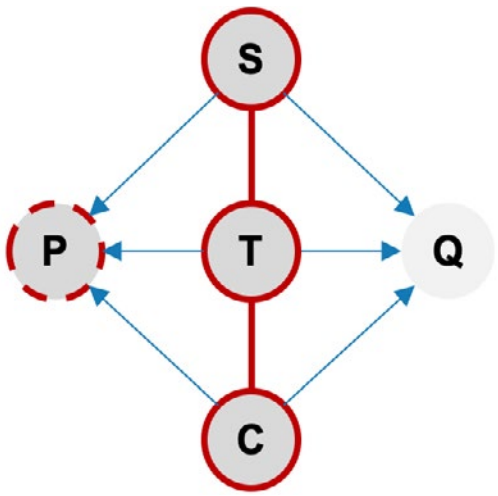

Figure 12: Scope, Time, and Cost fixed

To manage the project risks even better, we can dismantle a large delivery to smaller partial deliveries or at least features to demonstrate at milestones. Agile methods provide more frequent control of progress during the execution phase. However, a totally agile approach does not fit this situation. Pre-setting the scope and schedule takes away the flexibility related to agile development, and the development effort may be too one-off by nature to justify a more permanent team. According to the management culture of the company, either a hybrid project with sprint demos as the intermediate milestones for the team, or a program of small waterfall projects is recommended.

Flexibility: We may have a false sense of security, when expecting success based on this kind of a plan. Fixing all three components simultaneously is not sensible, considering the resulting rigidity and the limited capability to respond to changing circumstances. In many cases, the management maintains some freedom by prioritizing the three components of scope, time, and cost and by declaring that, if necessary, the third most important of these can be considered flexible.

Because of the cost estimate, we may also consider the people component fixed. However, the possibility of selecting a partner based on competing bids or to change persons later, provides some flexibility.

Since the possibility of adjusting the people component is limited, pressure is directed toward the remaining component – quality – to provide flexibility. In order to keep the schedule and budget without trading away parts of the scope, we may need to cut corners with the design, limit the testing or accept a larger number of low and medium priority defects. This way of coping sadly often goes together with the imperative of "sticking with the plan". Correcting the defects later requires more time and increases cost.

PART 3 – ENSURING SUCCESS AFTER THE SELECTION OF THE METHODOLOGY

In the following chapters, we present aspects to consider during the selection of the methodology and different ways to ensure success after the selection. We build on the differences between the management methodologies and their strengths and weaknesses, that were addressed in our white paper Description of methodologies to manage ICT development.

A special focus is on the attitude in leading people and the practices of financial management in an agile context. The current hype around agile methodologies is strong. However, the company, including the top management, is often not ready for them. It is impossible to effectively fit agile development into an environment that is built to support projects.

Management components as prerequisites

Before deciding on the methodology to be used, it makes sense to consider what the prerequisites are for the chosen methodology to work, and what kind of capabilities are required when using different methodologies.

To make a **waterfall project** work well, one naturally needs to have the requirements and scope well defined. Everything depends on it – the reliability of the effort estimates and thus the feasibility of the plan, the effectiveness of the execution, and the success of testing. The value to the users and the business can be measured against a business case, but that also requires clear requirements. Waterfall provides the opportunity for the customer to be absent between the requirements analysis and the user acceptance testing, in case their experts are needed elsewhere. Therefore, waterfall works with cases, where the needs are well understood. With one-off improvements, the experts that normally do not participate in development need to make an extra effort.

We can use a **hybrid project** or a **program** to implement a pre-defined scope in increments. This requires the representatives of the user community – the customer – to participate regularly, especially if parallel projects are proceeding with different schedules. If the partial solutions address needs in different parts of the organization, the same experts need not be continuously involved. The purpose of the project should always be clear in the beginning. This also determines the scope. The Agile approach of splitting development into pieces and demoing outcomes after each iteration works well when the customer can participate, receive demos and give feedback and guidance. Without their participation, there is not much improvement compared to the waterfall. Traditional project status meetings can also be held very frequently to ensure the control on progress. In a hybrid project, the agile meeting structure and the physical proximity of the team naturally support cooperation.

Agile development offers the opportunity to progress in complex situations where the needs are unclear. Although the requirements don't have to be fixed and detailed in the mission given by the management, the team needs to understand the purpose – the overall goal and value they should be striving for. For planning the releases or program increments and the sprints, we need a backlog of sufficiently well-described requirements and acceptability criteria provided by the Product Owner. Therefore, a skilled Product owner is a clear prerequisite for success. Without proper requirements, the team might not know what they are expected to do, and mere self-organization could slip into a "developer anarchy". In all the phases from planning to testing and deployment, close customer involvement is essential.

Another prerequisite for succeeding with agile development is the utilization of self-organizing teams with a suitable collection of competencies. In the absence of superior individuals, this means collecting people from different domains of expertise and supporting their cooperation effectively. A good team can be highly capable of creating superior value. In order to provide good leadership, the company still needs to have the suitable persons available to work as Product Owners, Scrum masters and in the other agile roles required. The management culture needs to favor empowered knowledge workers. This will be addressed in a later chapter.

A potential problem with agile development is the difficulty for product management to continuously confirm that the features they prioritize are of value for users and to the business. This is the case with a software company, when the customer as the end user is not a member of the development team. The prioritization might be politicized and based on guesswork. In order to ensure customer value, the company can start creating hypotheses and experiment with customers, as described in the HYPEX model by Olsson et al. Another way to ensure valuable features is to describe the requirements as measurable improvements that need to be achieved in a business process or customer behavior. The competencies in defining the requirements have been further discussed by Jantunen et al.

Agile development is most effective in the management environment of an agile company. Building an **agile innovation system** – BizDevOps – is a transformation itself. It requires investments into the information system platforms to enable an automated continuous integration and deployment process, release on demand, and a working feedback flow back to product management. It is best to start small, by first ensuring that one simple user story gets through the process, and then continue to improve the system.

General pitfalls to avoid

ICT development is not done in isolation but surrounded and enabled by the management culture and tools. Successful development requires that the development team communicates and cooperates effectively with the business customer. The table on the next page (*Table 3, p. 19)* lists the weaknesses and risks with the basic methodologies, that we need to avoid when planning ICT development.

The failure factors related to analyzing the requirements and scoping a waterfall project are well known. Planning may be difficult, even if the requirements were reasonably well-written. We might still have unrealistic expectations toward the feasibility of a software product, the capability of a service provider or an implementation partner, or the competence and effectiveness of the in-house people. The latter is often negatively affected by the temporary nature of a project team and the reluctancy of line management to allocate their best people. A very competent partner can support and cover the role of a missing customer expert to some extent, but not completely.

From an investment perspective, much of the effort in a waterfall project may be put into backward-looking cost control and hardly any on realizing and measuring the business benefits – e.g. exploiting opportunities to increase revenue. Furthermore, the long duration of a waterfall project might make it necessary to wait for the benefits for too long. They might even be diluted, as the business environment changes. This would effectively ruin the business case.

The more flexible approaches of **programs** and **hybrid projects** can help us avoid the above-mentioned weaknesses and risks of the traditional waterfall. However, the increased flexibility comes with the requirement of increased participation by internal experts and management. Without these efforts, the potential benefits of close control are lost. In a project with partial deliveries or a program with multiple projects, the financial decisions should be arranged to fit the more flexible roadmap structure. Otherwise, the decision-making will be almost as rigid as with the traditional waterfall.

The imperative of customer participation is most clear with **agile development**. Otherwise, the methodology just won't work. Another difficulty with this approach is that it is still new to many people. There is a lot of hype involved, and not all the service providers nor internal development teams are either willing or able to instruct and support the customer with the methodology. For people burdened by the heavy up-front requirements analysis and planning, the agile approach often seems to be a permission not to analyze or plan at all. As a result, there might be a backlog of requirements that contain just a couple of words and no acceptance criteria, or even simple calendar events for design meetings as "user stories". A proper business case or even a hypothesis of a business benefit might also be missing, which means that the development team gets no real guidance from the customer.

Financial management for agile development must also be flexible. The decision to choose an agile approach usually comes with a decision to build a relatively stable development capacity and to have a budget on this capacity or headcount. The capacity can be re-directed by re-prioritizing objectives, and the development team can be

Stage & outcome	Waterfall project	Hybrid project	Agile development
Adjust ⇨ Clear mission	Weakness: Requirements that are unclear or obsolete lead to a bad result.	Improved: Shorter iterations and less risk of obsolete requirements or even an obsolete mission.	Risk: Unclear business situation and customer needs can be managed, but large and complicated systems are hard to handle.
Plan ⇨ Doable & flexible plan	Weakness: Too detailed planning with guesswork estimates. Unrealistic expectations on simplicity and speed. Insufficient allocation and availability of people.	Risk: Flexibility through batch size reduction, but critical requirements still need to be analyzed and solution defined up front. Avoid budgeting 100% based on scope.	Risk: Selecting only a few agile practices may lead to not planning at all. Prioritization may be poorly justified. Financial planning must also be flexible.
Do ⇨ Reliable execution	Risk: Efficient when the tasks are clear, costly rework if they are not.	Improved: Close control and immediate progress with time-boxed increments	Improved: Close control and continuous progress with time-boxed increments
Check ⇨ Updated understanding	Weakness: Delayed testing, late feedback, late or uncertain correction of errors. Slow generation of value, backward-looking control.	Risk: Better control through testing initial results and earlier warnings about the need to revise the plan. Requires customer participation and clear feedback, which may be missing if the approach is new.	Improved: Better control through testing initial results. Requires customer participation and clear feedback.

Table 3: List of weaknesses and risks with the basic methodologies

granted autonomy in designing solutions. Therefore, the waterfall style of precisely defining deliverables and estimating workloads to produce them is no longer meaningful. The long-term estimation needs should be covered with a rough high-level plan and only the near future estimated in more detail.

Waterfall style control of deliverables and cost in combination with an agile development team may paralyze the decision-making and create frustration within the teams and poor effectiveness across them. We need to abandon the detailed project control structures and replace them with new ones more suitable for the agile methodology. The financial management aspects will be addressed more closely in a latter chapter.

An agile approach should not be permission for the team not to plan or to report on progress at all. Properly applied, it focuses on frequent management events: product planning, release/feature planning, sprint planning, daily stand-ups, demos and approvals, and sprint reviews. There are also excellent visual tools to check on progress, like burn-down charts and Kanban tables.

Making agile management culture work

Agile methodologies are closely related to the ideology of Lean thinking, which has been adapted to the software industry as Lean software development. The main Lean principles can be summarized into the following six, as is done in the "House of Lean" belonging to SAFe (© Scaled Agile, Inc.):

1. **Value**, as perceived by the customer, is the only justification of any activity. Anything that does not add value to the customer is waste.

2. **People** do the work, not systems. A company or a development team is not a machine but a group of thinking, feeling, and learning individuals. People have a need to be respected. Instead of telling them to follow orders, they should be trusted, supported, and encouraged to learn.

3. **Flow** of work is accelerated through reducing batch sizes and optimized to a sustainable level. Each requirement taken from the backlog queue is completed, instant feedback on the fitness for use received, and deployment to operation done.

4. **Innovation** is driven through making planned experiments, having the customers validate, and then deciding whether to "pivot or persevere" with the approach. People are provided with the time and space to be creative. Only the things that really matter are measured.

5. **Relentless improvement** encourages learning and growth through recurring reflection sessions built in the process. Lean tools are applied to determine the root cause of inefficiencies, followed by rapid countermeasures.

6. **Leadership** is a key enabler for team success. The ultimate responsibility for the successful adoption of the Lean-Agile approach lies with the enterprise's managers, leaders, and executives. Such a responsibility cannot be delegated.

Principle number 2 – respecting people as individuals, supporting their learning, and empowering their cooperation – stands behind the idea of self-organizing teams. As Peter Drucker says, "knowledge workers are individuals who know more about the work that they perform than their bosses". This leads to decision-making on the level where the best competence on the subject lies. Instead of demanding the team to follow a plan, the leader should provide a purpose (WHY), a mission (WHAT to accomplish), and the least possible amount of constraints. From there on, the job of the boss is to support the team by removing obstacles.

It makes sense to consider what the current situation of the company's ICT development is. At times there are desires to centralize services and decision-making to the company headquarters or a service center. Perhaps at the same time, there are competing desires to decentralize the decision power to divisions, countries, or business lines. How detailed is a manager in deciding on a plan and giving instructions? Or how detailed is an architect giving guidance? What is the balance between listening to the users' needs and complying with the company policies? Centralized decision-making promotes projects, and decentralization promotes agile methods.

Principle number 6, the imperative for Lean/Agile Leadership may be a new one for someone who considers business management, change management, and ICT implementations as separate domains. In SAFe, scaling the agile methodology anywhere above the level of a single development team requires the managers and executives to study and adapt a Lean mindset. Otherwise, their actions may prevent decentralized decision-making, learning, improving, and effectiveness. Hybrid projects contain a portion of agility within. An organization that uses a hybrid approach does not face quite as high a demand for the Lean principles, but it still makes a lot of sense for the management to study them.

Before jumping fast into an initiative of establishing Lean management or agile development into the company, one should again consider how far from the above principles the current management culture of the company is. Many "Lean" initiatives have, in practice, only

utilized the daily standout meetings and metrics to control people but not granted the people any power over their work.

The general order of events in a transition toward Lean are the following: Leaders show the way by their example and set the suitable structures for more agile development. The people learn by doing together. Eventually through success, the new habits stick, and the company culture changes by itself.

Military leadership applied into business

An essential question to test whether the company can provide a suitable environment for agile development is: Do we have an attitude of trust in our employees, and freedom of action? Can the person or group with the best knowledge and expertise in any given area act in a timely manner without asking for permission?

Surprisingly to many people, very close to the people-centric Lean principles is the military tradition of Mission command (originally “Führen mit Auftrag” in German) where the subordinate leader is focused on accomplishing the task/ mission, as opposed to executing a set of orders. The purpose of this approach is to accelerate decision-making and to increase the flexibility of the force in responding to changing situations. For mission command to succeed, it is crucial that subordinate leaders:

- understand the intent of their orders
- are given proper guidance and
- are trained to act independently.

Mission command does not allow the “leader on site” to disobey orders, but it does allow – and even demands – that he uses his judgement and considers the order details no longer binding, if they wouldn't be given in the changed situation. Essentially, the intent of the commander regarding the desired end state must be maintained. The leader on site is obeyed unconditionally, but once the commander's intent is understood, decisions must be devolved to the lowest possible level to allow front-line soldiers to exploit the opportunities that develop. To summarize, a leader does not micromanage, but instead:

- expresses intent in terms of WHAT to achieve and WHY
- grants autonomy in terms of what to do and HOW to realize the intent.

Stephen Bungay has applied the ideas of mission command to business. “The business environment is unpredictable and uncertain, so we should expect the unexpected and should not plan beyond circumstances we can foresee.” Everyone must have the skills and resources to do what is needed and the space to take independent decisions and actions, when the unexpected occurs. Of course, in order to act effectively in an autonomous role, the team needs to be able to trust the organization for support.

In the PDCA cycle presented earlier, a leader moving from the “Plan” stage to the “Do/Delegate” stage communicates the purpose and mission to the team. This is done by giving a briefing, which is very close to giving a military order. The main elements of a briefing are:

1. A description of the situation
2. A short statement of overall intent
3. Extrapolation of specific tasks implied by the intent (= missions to the team members)
4. Any further guidance about boundaries and support

It is important that the specific tasks and guidance contain only the amount of information necessary to effectively delegate the tasks and to divide them between team members or different teams. Selecting a project as the methodology does not force us to work in a Tayloristic and unified best practice way, but there is a risk of too much planning and instructing even in situations where that is not necessary. From that perspective, a hybrid project with an agile team looks more appealing than a traditional waterfall project.

On the other hand, self-organization by the subordinates has its limits. If the company faces a

crisis that makes the mission given earlier obsolete, the employees cannot be left alone to handle the situation. In the military, situational information is escalated – the commander re-evaluates the situation and gives a new mission that supplements or overrules the old one. In civilian companies, the leaders often face strong opposition by people mentally attached to the old situation and old ways. Managing a large transformation emphasizes the need for very clear communication of purpose and intent.

Financial planning and control in the agile context

Adopting an agile methodology in a company that is used to running projects can be a demanding change, since it requires the whole management system to be more agile. There are fundamental differences in the financial processes and structures required and natural for projects and agile development. The differences are well presented by Sirkiä & Laanti *(figure 13)*.

To summarize the difference, let us consider a conflict of interest between two professionals: The financial manager wants to know in advance which deliverables and benefits the investment will bring, in order to evaluate the business case. The manager of agile development wants to fund capacity in order to implement the backlog in a constantly revised priority order. A company with traditional cost center and project accounting trying to implement agile practices will face problems such as these:

- Dispersed cost center level priorities, instead of focus on a common prioritized backlog
- Insufficient resource reallocation methods in case of a delay or re-prioritization
- Long-reaching resource allocation plans and cost estimates based on pure guesswork, followed by the managerial overhead of monthly updates, cost overruns, and re-approvals
- Traditional metrics and performance targets not improving efficiency, as the people are not empowered to adjust the centrally approved plans.

The following approaches are worth looking into, when adopting a more agile approach in investing in and controlling the development of ICT solutions:

- Real options valuation
- Planning granularity
- Agile prioritization including the cost of delay
- Planning for a lower utilization to ensure flow
- Focusing on constraints/ bottlenecks to ensure flow
- Focusing on actionable metrics.

Traditional cost center and project accounting	Agile development accounting
• Expects a long horizon with detailed cost estimates that must be frequently updated	• Avoids detailed and long horizon planning
• Values planning accuracy and executes variance analysis against the original estimate	• Allows for uncertainty; the final content delivery and total cost may be impacted by the empirical feedback received
• Draws attention to budget overruns	• If the initial feedback proves to be positive, we want to allow and even encourage further investments. In larger enterprises the allocation of funds can be controlled via the hierarchy of backlogs – the teams allocate tasks flexibly according to the changing backlog priorities
• Requires a re-approval for any delays which increase the project budget. Rigid approval limits force a budgetary approval process that again may force the development team to wait and create further delays	• Delays place incomplete deliverables back into the backlog, and the teams can now re-prioritize them and choose the order of development more flexibly

Table 4: fundamental differences in the financial processes and structures required and natural for projects and agile development.

The more software-driven the company is or the more uncertainty there is, the more real options thinking needs to be allowed in planning the future. Real options valuation 10 adapts the techniques developed for financial options to “real-life” investment decisions, such as contracting, deferring, expanding, or abandoning ICT development initiatives. For agile development, the allocation of funds to different streams means prioritization on the top level of the backlog hierarchy. For projects, funding decisions can be done on the program or portfolio management level.

The principle of planning granularity used in manufacturing for decades can be used in ICT development: more accurate short-term resource planning on a sprint or release level, and only rough long-term planning on the project and program levels. If the headcount is relatively stable, the OPEX run rate can be planned on a higher (cost center) level. Furthermore, the different planning horizons can be taken into account, when calculating budget and latest estimate figures.

In prioritizing requirements and thus investment targets, one needs to consider the cost of delay. Waiting is waste for the customer and waiting for a highly valuable feature costs more in lost benefits, than waiting for a less valuable feature. Thus, the Lean-Agile style of prioritizing and scheduling the requirements based on customer value and using tools like Weighted Shortest Job First (WSJF) should automatically lead to a better financial result.

On the provider side, the old way of resource planning with a 100% utilization target leaves no room for unexpected changes – and on the customer side for that matter, no room for development or continuous improvement. Traditional controlling often tries to improve the efficiency by seeking maximum utilization, and hunts for waste with tools like activity-based costing. It makes more sense instead to plan for an 80-90% utilization to manage bottlenecks and surprises and to ensure the flow of work. The flow is further supported by applying Lean-style queuing principles and strict work-in-progress (WIP) control. This means arranging the development tasks in line and completing them one by one, as well as limiting the number of concurrent tasks for a team or person.

When controlling and improving the effectiveness of a process, a team, or a system, it is far more important to understand and maximize the output of the constraint (bottleneck) at each level or function. According to the principles of Constraint management by Eliyahu Goldratt, the bottleneck resource must never wait, as it dictates the throughput of the whole system. Instead, it makes sense to leave a queue of work, a buffer of “inefficiency”, in front of the bottleneck in order to ensure the undisturbed flow of work. As an example, this bottleneck resource might be an architecture team that, according to the company policy, must approve any significant change to the solution design.

Sensible metrics need to be in place to measure success. The agile development community favors “innovation accounting” introduced by Eric Ries in The Lean Startup. Only “leading indicators” or “actionable metrics” that test our assumptions and can predict the outcome of our course of action are useful. “Vanity metrics,” like a developer’s utilization %, can look good in a graph and be easy to collect. However, they draw our attention to secondary targets and away from what we really want to achieve.

Compared to traditional financial management procedures, applying an agile methodology contains the risk of too much freedom: A very independent development team may have their own preferences, and sometimes no sound financial arguments are presented during the creation and prioritization of backlog items. Furthermore, the management may neglect the need for traditional business cases, although they could be created with a reasonable effort and would be necessary in justifying the distribution of funds between different development programs or streams.

Closed loop program management

Management practices mostly underlie the success and failure factors in large ICT transformations, as presented by Eskola & Töyrylä. Therefore, it is necessary to examine how management on the program or business level can fit together with the selected methodology and ensure the success of ICT development. The purpose of the closed-loop management process is to ensure that information flows from the previous stage to the next, forming a sound basis for analyses and decision-making. Furthermore, the loop form contains the ideas of experimentation, learning from experiences, and adjusting the direction.

The logical starting point of the closed loop "PDCA" management process is not the Plan stage but the Adjust stage that includes understanding the purpose and setting or receiving the mission (WHAT to achieve). These two things are the most important points to communicate when leading people. The table on the next page *(Table 4, p.25)* presents management stages with their desired outcomes and common questions related to ICT development. It also presents simple thumb rules about fitting together with the ICT development methodologies, as answers to these questions based on the previous chapters.

An ICT development effort sometimes fails to reach its objectives with the methodology we are using and the plan we have devised. The approach we chose may not have been optimal for the task at hand, or changes in the environment have made our plan outdated. Program management or steering team level decisions are required to correct the approach.

Some real-life examples:

Example 1:

We have launched a waterfall project to implement an ERP software suite to support business processes with documented requirements that can be covered through configuring the software. However, during testing the business representatives are dissatisfied. Many requirements prove to be more complex than what was initially thought, and the software needs to be heavily customized. We need to adopt a more agile approach of new software development in several areas.

Example 2:

We are using agile development with rather independent teams to meet clearly limited and described requirements. However, along the way, we learn that the seemingly separate solution areas are connected to each other in many ways. Our agile teams are not used to coordinating the work between each other. A joint design and planning effort is required to manage the complicated solution.

Example 3:

A large software implementation starts as a replacement of old versions, with the objective of creating a basis for future improvements. There is no business case, since no substantial process improvements are expected in this phase. However, along the way, the business representatives learn about the possibilities of the new software and begin to demand new features to be implemented to enable the removal of some pain points sooner rather than later. This creates the situation where different process and solution areas are advancing at different speeds, and more operational model change topics need to be addressed.

Example 4:

A challenge that is very common in general and related to all the examples above: We start with a seemingly clear and limited mission and launch a simple project to accomplish it. However, the task at hand proves to be larger and more complicated than we thought. ICT development needs to be dismantled into parts with an incremental approach, with some of the parts implemented in an iterative way. All this requires a substantial increase in management capability and in the number of managers and business representatives.

For the program management and steering team level decisions, we need to make a revised analysis of the situation – we are in the Analyze/Adjust stage of the PDCA management loop. We can then apply the guidelines presented in this white paper to form a revised plan.

The following table (*Table 5, p. 25*) presents some program management problems in the form of questions. It also draws solutions from everything stated above and presents them as methodology related thumb rules. Therefore, it summarizes the learnings of this white paper into the closed loop management cycle stages.

Management stage	Desired outcome and questions	Methodological thumb rules
Adjust Analyze	⇨ **Clear mission** • Is the purpose (WHY) clear? • Is the vision inspiring? • Do we understand the situation? • Do we need to continue like before or revise our approach? • Is the mission (WHAT), the ICT requirements, and scope clear? • Are the expectations realistic?	• Always define a clear purpose! • Consider the decision-making situation and mission • If the mission is one-off and clear, set up a project • If the mission is unclear, set up an agile team and experiment • To develop and improve continuously, set up agile teams • To really agree on the expectations, have a business case, clear requirements and actionable metrics
Plan, Propose	⇨ **Doable and flexible plan** • Is the approach (HOW) clear? • Is the solution design of high quality? • Are the tasks well-organized? • Is the schedule realistic? • Are the resources adequate? • Are the risks under control?	• To outsource a simple task, set up a waterfall project • With limited availability of experts and managers, use a waterfall project • If the solution is critical, complicated, or highly regulated, set up a hybrid project and make a careful design • For decentralized decision-making and team autonomy, apply agile methods – make sure you have good Product Owners in place. • Have rolling planning and frequent control with any methodology • Avoid risk through incremental delivery
Decide, Do, Delegate	⇨ **Reliable execution** • Are we progressing as planned? • Were the tests passed? • Are the deliverables as expected? • What is the feedback? • Are we performing well? • Are there obstacles to remove? • Is the team up to the task? • What is the team spirit?	• To control closely and avoid surprises, use a hybrid project or agile development • Have frequent control with any methodology • Customer feedback tells us if we created a proper deliverable • To release frequently, use an agile team and continuous delivery • Give a clear mission when delegating, and grant enough autonomy with the solution – this also enables good spirit
Check Complete	⇨ **Updated understanding** • Did we succeed in the delivery? • Is the business outcome as expected? • Have we accomplished the task? • Are we creating value? • Is the customer satisfied? • Are the metrics reliable? Do we understand the meaning of figures? • Do we understand the root causes? • Are we likely to succeed?	• If you can validate the solution against design documents, a waterfall project works • To validate early and frequently, set up a hybrid project or an agile team • To have an innovative company, use agile development and continuous delivery, enable the feedback flow, and experiment with hypotheses • Validate the delivery and ensure the value with actionable metrics
Account for, Adjust, Analyze... Next round		

Table 5: Methodology thumb rules in the management process stages

CONCLUSIONS

The feasibility of different methodologies to manage ICT development depends on the business context, the mission given, the capabilities at hand, and other factors in the management environment. In this white paper, we have given frameworks that help determine the suitable methodology. As the success and failure factors of ICT development and especially of larger ICT transformations are mostly related to management, we have committed a substantial portion of this white paper to frameworks that apply to management in general.

Based on our experience, the discussion on methodologies in companies has focused on the comparison and choice between the traditional waterfall project and the agile approach that has grown to be quite popular. However, one must remember that the methodologies are tools and guidelines – they are not to be followed dogmatically, and they should always be adjusted to fit the company and the task at hand. In this white paper and the previous one, Description of Methodologies to manage ICT development, we presented an alternative that combines the careful pre-planning and design of a waterfall to the fast-paced and flexible execution of the agile approach – the hybrid project. A hybrid project is in many situations the suitable tool to accomplish the ICT development mission in an incremental and iterative way. We also addressed the program as a management level to combine and run several development efforts using different methodologies.

Lastly, we addressed the pitfalls to avoid, the pre-requisites to ensure success, and the management culture aspects to consider in making the methodologies work in real-life. The last chapters were mostly about what needs to change, in order to succeed in applying agile methodologies in a traditionally waterfall project culture driven organization. In these chapters, we referred to several important sources that the reader is invited to investigate as further readings. There is a vast amount of useful information available. We have tried to give an overview, point out the most relevant ideas, and direct the reader to search for more detailed recommendations of tools and practices, in the spirit of continuous learning and improvement.

AUTHORS & ACKNOWLEDGEMENTS

MSc Tech, MBA Juha Jouppi, Program Director, Midagon Oy

Dr Tech Ilkka Töyrylä, CEO, Midagon Oy

Other contributors at Midagon: Jarmo Hiljanen & Juha Asikainen on Agile methodologies, Paula Aaltonen on Project & Program management, Esa Haakana on SDDC and Hybrid programs, and numerous other colleagues that have shared their practical experience.

REFERENCES

Description of methodologies to manage ICT transformation, Juha Jouppi, Ilkka Töyrylä et al, 2019 – a Midagon white paper available in https://www.midagon.com/app/uploads/2019/10/Midagon-White-Paper-Description-of-methodologies-to-manage-ICT-development.pdf

A Leader's Framework for Decision Making, David J. Snowden & Mary E. Boone, Harvard Business Review, 2007 – an overview of the Cynefin framework available in https://hbr.org/2007/11/a-leaders-framework-for-decision-making

SAFe is methodology and a trademark by Scaled Agile Inc. – description of the Program and Large solution levels available at their site https://www.scaledagileframework.com/#

Why large and complex ICT and digital transformations fail? Framework for managing success and failure factors, Lauri Eskola & Ilkka Töyrylä, 2018 – available in https://www.midagon.com/app/uploads/2018/10/Midagon_Whitepaper_Why-large-and-complex-ICT-and-digital-transformations-fail.pdf

The HYPEX Model: From Opinions to Data-Driven Software Development, article by Helena Holmström Olsson & Jan Bosch – available in https://www.researchgate.net/publication/283625598_The_HYPEX_Model_From_Opinions_to_Data-Driven_Software_Development

Towards New Requirements Engineering Competencies, article by Sami Jantunen, Rex Dumdum, & Donald C. Gause – available in https://arxiv.org/ftp/arxiv/papers/1903/1903.10214.pdf

Many related articles on and around Mission command in Wikipedia: https://en.wikipedia.org/wiki/Mission-type_tactics

Art of Action, Stephen Bungay, 2014 – presentation available in https://portal.netobjectives.com/wp-content/uploads/2017/12/Bungay_ArtOfAction_ChapterRecaps_Article.pdf

Lean and agile financial planning, Rami Sirkiä & Maarit Laanti, 2013 – white paper available through https://www.scaledagileframework.com/original-whitepaper-lean-agile-financial-planning-with-safe/

Real options valuation explained in https://en.wikipedia.org/wiki/Real_options_valuation

The Goal: A Process of Ongoing Improvement, Eliyahu Goldratt & Jeff Cox, 2012

The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Eric Ries, 2011

midaGon

Success is a Choice

Keilaranta 1
02150 Espoo

Business ID: 2058234-3
www.midagon.com