Whitepaper

SCRUM AND ERP – DO THEY GO TOGETHER?

An ERP system is no less than the backbone of a company. As a reflection of current business processes it is intended to support work flows and help keep them efficient. Seen from a functional point of view ERP systems offer countless possibilities. But it is also a fact that they rank among the most sluggish and, at the same time, most cost-intensive architectures in the IT landscape of a company. Should a company wish or have to adapt to changes on the market, the highly branched ERP giants often move very slowly: if one screw is turned, it turns countless others with it – effect doubtful.

CIOs and those in charge of IT are the people responsible here: caught up between cost reductions, growth plans and innovation pressure it is their job to make ERP systems essential value adding factors. Many IT specialists regard agile management frameworks positively, but ask themselves whether these can also cope with the complex adjustments of ERP systems. The answer is quite simple: agile management frameworks, in particular Scrum, were developed for the exact purpose of enabling successful execution of large and complex projects.

THE THREE STUMBLING BLOCKS IN ERP PROJECTS

It is not only directly attributable costs that arise with the implementations or modifications of an ERP system. Every change in an ERP system is a disruption – and this gives rise to costs. The disruption can be limited temporarily to the phase of the change (e.g. when an employee is stopped from carrying out other important work), but in the worst case its impact can be far more extensive – if the result of the implementation or adjustment of the system does not match the reality of the user at all. The more often the processes within a company change, the more expensive the ERP system becomes. The magnitude of these costs depends on technical, human and organizational factors.

Technical pitfalls
The ERP system should fit the system and not the other way round. That is why it is important to involve future users as soon as possible, in order to avoid development passing by practiced processes. Many of those involved in a project agree with this, but nevertheless we are often confronted by three extremes in everyday practice:

1. **One for all.** The same – and simplest to implement – solution is imposed on all divisions and departments; the reality of the workflows is not taken into consideration. Result: the ERP solution becomes an expensive, but de facto useless toy.

2. **All for one.** (All) wishes of the departments are taken into consideration; a small change is made here or there "on the side" – suddenly nothing at all works any more.

3. **All for all.** Because providers offer any number of attractive add-ons for standard products, something new keeps being added without any kind of coordination. The long-term consequence: a technical juggernaut that causes costs to explode and destabilizes the system.

In order to keep control of the costs, the modifications have to be integrated as soon as possible in such a sensitive construct. This is the only way that the implications can be recognized in time. To do this however, it is necessary to maintain an overview of all changes to the system and to prioritize modifications.

Organizational pitfalls
Depending on the size of the company, an ERP project including preparation can take over a year – or it can become a long-runner that eats up resources. One reason for this: if projects are handled in line with traditional procedural models, lengthy integration and acceptance tests are not carried out until the end. So at the end of the development the detected errors and change requests pile up and delivery is delayed. More costs!

The human aspect
Changing over to an ERP system or the modification of modules always requires the willingness to accept change from those involved. In most cases the colleagues in the various departments cannot assess from the technical specifications what impact the proposed solution is going to have on their work. In the very worst case employees have to work with software that does away with existing processes from one day to the next – resistance is virtually inevitable. And the other way round, many employees do not understand what effect their own particular wishes have on the functioning of the project as a whole. So some people use political means to get what they want.

For this reason the implementation or adaptation of ERP systems should be undertaken within a framework that

- promotes **communication and coordination** between the IT department, other departments, the employees involved and where applicable external consultants in the spirit of achieving a common goal.
- makes **intermediate products** visible. Whether programming or customizing: both concern the creation of new functionality. The sooner the customer or user sees the results and can try them out; the more useful the result will be thanks to the feedback received. And not only that: people accept new solutions more readily if they are allowed to work on them actively and are involved in the decision making.
- **ensures quality** and thus **saves costs.** An iterative cycle of programming and testing prevents detected errors piling up at the end of the project, by which time they have become almost insurmountable.

HOW DOES SCRUM WORK?

Scrum is a management framework for iterative and incremental – "agile" – development: whereby a product is divided into cycles from the vision onwards and refined step-by-step. People know what they want, but accept the fact that the product cannot be specified down to the last detail in advance.

In order to reduce complexity, a Scrum team works in individual steps of maximum two to four weeks – so-called "sprints". To start off with the key functionalities are determined and then developed further or rejected in each sprint on the basis of customer or user feedback. Unlike classic project management, the customer does not have to wait to see the result until the end of the entire development, but instead is involved in every step of the development. The organizational principles of Scrum:

- **Small, self-organized, self-responsible, cross-functional teams** – consisting of the ScrumMaster, product owner and development team.
- **Working in compliance with the pull principle:** the development team decides on the number of functions (that have been prioritized by the product owner based on the business value) it will develop in a sprint and it decides how it will carry out the related tasks.
- **Clearly limited time intervals (timebox):** the aim is to actually finish the selected functions within the sprint.
- **Useful business functionality:** at the end of each sprint the team makes a delivery, which complies with the standards, directives and specifications of the project.

A key factor of success is intensive communication: The team members confer on the current state of the work in short daily meetings (daily Scrum). They work on the joint conception of the functionalities to be developed in estimation meetings (backlog grooming) and in the sprint planning meetings 1+2 specify what, when and how development is to be carried out. In the review meetings at the end of a sprint the team talks with the customer about the developed product increments and retrospectively assesses its work in order to improve this where necessary.

AGILE ANSWERS TO THE MOST FREQUENTLY EXPRESSED CONCERNS IN THE ERP ENVIRONMENT

The challenges mentioned at the beginning concern ERP projects that we have experienced ourselves or have been told about by customers and colleagues. Agile management frameworks promise to solve these challenges - nevertheless the field of ERP in particular is not penetrated by agile methods to the same extent as classic software development.[1] We are well aware of the reasons for this from our experiences in practical consulting: the opinion still stubbornly persists that ERP systems cannot be part of agile development due to their special position in the company. We give answers here to the most frequently expressed reservations from the agile perspective.

[1] Status Quo Agile 2014 www.status-quo-agile.de

"Why should agile methods be used for standard ERP implementations? This does not normally involve any kind of development, just adapting software to the existing processes."

ERP implementations intervene in the way many employees work, whether standard or in-house development: changes give rise to uncertainty. You must be aware that employees have got used to working in a certain way – and this way of working will now possibly be done away with altogether or else changed considerably. So do not be surprised if you are confronted with – active or passive – resistance.

So those responsible for implementing ERP should treat the fears, concerns and working methods of those involved with all due respect. Fear of what is new can be subdued or disappears altogether if employees are involved as part of the project from the very outset and if they can participate in decisions concerning what path should be taken to achieve the goal. In agile management frameworks such as Scrum, great scope is given to communication with the user by means of regular interviews. The users try out the product increments and give feedback as to what works well and what does not.

"Agile development is very difficult in ERP systems, because all requirements must be known from the very beginning. The data model and specialist concept can then be created on this basis."

In principle: concepts should have a calming effect. Project managers especially feel more secure if the path to be taken is defined specifically. Customers and users on the other hand often simply wave through these concepts without even grasping their basics. The reason: excessive demands.
A document with data models and at most some graphic suggestions for the user interface provides no basis for the future user upon which he/she can assess whether the product will actually support his/her work. In addition users usually have a clear idea of what they want – which is in fact not always what they actually need. The user can only find out what really helps by trying it out.

In agile development the user regularly tries things out, because new parts of the product are delivered regularly. The experiences made by the users while trying out these parts in turn help developers to deliver the ultimately "right" product. Instead of getting stuck in the strict – and possibly unsuitable – specifications of an irrefutable concept, reality flows into the further development. But continuous evaluation has another benefit as well: adjustments to the system are effected as efficiently as possible. In this way it becomes clear whether simple customizing of the standard module actually suffices, for example, or whether a self-written code (e.g. in ABAP) gives a better depiction of the business process. Expensive adaptations often become obsolete if users can already try out the standard in the first iteration.

"With the exception of some small adjustments, an end-to-end functionality can rarely be developed in two weeks. Modern business processes are too complex for anything tangible to be developed in such a short time."

In agile software development requirements are divided into parts that are as small as possible. These parts constitute added value for the customers and can be accomplished in one sprint. A so-called minimum viable product can also be engineered in ERP projects using this basic principle[2]: for example a core process, a basic functionality, a main transaction or quite simply the "80%-case" of the operative business.

[2] Robinson, Frank: Minimum Viable Product https://en.wikipedia.org/wiki/Minimum_viable_product

This minimum viable product can usually be developed with relatively little effort and can therefore be implemented even end-to-end within a few days to weeks. The process is subsequently modeled and refined step by step in order to support the user in the way he/she needs to be supported. A first breakthrough can be achieved quickly with a minimum data feature set even for the implementation of interfaces between different systems or ERP modules. The complete data structure of the interface is then designed and implemented in the subsequent sprints. Important in all cases: the actual users verify the desired process as quickly as possible. This is the basis for its continuous adaptation.

"The possibility of delivering and going productive with finished software with additional functionality every two weeks is very limited in ERP systems. There is the danger of existing functionality no longer operating correctly and that operative processes are blocked."

Companies are often confronted with the problem of a system framework that has expanded due to an uncountable number of sometimes random requirements that pushes up maintenance costs.[3] In the case of agile development in an ERP environment the right development practices are also absolutely essential. No giant system develops if you consistently ensure of the quality of the source code. Continuous delivery as well as assurance of existing functionality is possible with automated tests.

[3] IDC: Maintaining ERP Systems – The Cost of Change http://www.unit4apac.com/products/agresso/idc-cost-change-erp

Continuous integration and delivery systems are used in the agile environment for these tests that take place after every change. Intelligent and optimized development environments (such as ABAP Eclipse support) test automation suites (eCATT and CBTA in SAP), version/build management and mocking frameworks (MockA) are gaining ground even in the field of ERP and support the business requirements of a software that has gone productive permanently.

“How can agile methods provide an overview of the numerous small requirements of the many different departments? Isn't it enough to simply use to-do lists and processing lists?”

Transparency and visualization are among the most important levers of agile software development. This is an enormous help for requirement management – especially in complex areas like ERP systems. The requirements are clearly shown in a product, company and/or team backlog, so that all those responsible maintain an overview. They can exchange views on the basis of the required functionalities, agree a comprehensive prioritization, detect dual implementations early on and resolve the associated conflicts of interests.

Even more important and more effective is the visual support of this in large projects, in which many different departments must be involved in the business processes to be depicted. If every department follows its own interests, this can soon lead to uncontrolled changes – the scope of which grow continuously. If on the other hand, product owners are responsible for the backlogs and their prioritization, then the really important interests of the various stakeholders are taken into account specifically. Functionalities are implemented one after the other and immediately go productive, in order to deliver added value to the customers and to receive feedback.

“To be able to configure and adapt the diverse and complex ERP components we need specialized experts. That makes work in cross-functional teams more difficult.”

It has also proved effective in ERP projects to unite different skills in one team: ERP consultants work with ERP developers, CRM experts with MM experts, while business analysts or system architects also enhance such teams. In this way the requirements are viewed from different perspectives and the exchange of ideas within the team brings aspects to light that the individual alone would not have detected – this again ensures that the "right" product is delivered. In addition intensive cooperation – e.g. pair programming – promotes the exchange of knowledge within the team and within the company. This leads to employees developing a T-shaped profile: they have their specialist knowledge, but expand this with a broad basic knowledge in other disciplines. This promotes the personal development of the individual, who can then give his/her support for more diverse tasks.

3 good reasons why Scrum and ERP go together

1. Accurate fulfillment of customer needs

Continuous feedback means that the right software for the user is developed in the course of the project. The better the system matches the business process, the better market opportunities can be exploited.

2. Speedy delivery

Complex requirements can be implemented in short intervals with Scrum and agile development practices. The project with regard to its deliveries is transparent for all those involved – from the development team to the management. Changes flow in while the project is in progress and disruptions are kept within limits for all those involved.

3. Motivated employees

Cross-functional teams help employees to develop outside of their field of specialist knowledge and to find solutions together with other experts. The common goal promotes cooperation, significantly fewer handovers saves valuable time.

Real life example: Cross-system SAP project with cross-functional teams

How can complex business processes be modeled across several SAP systems and at the same time an uncountable number of departments be integrated in the modeling? This was a key question for the customer in one of our projects. An operative business process that was used by several hundred employees in very different departments had to be reproduced across two SAP systems and a third-party software.

All necessary knowledge in the team. Two closely linked, cross-functional Scrum teams were brought together to address this initial situation in an optimum manner. Each team consisted of ERP developers as well as SAP consultants from both SAP systems, a representative from the manufacturer of the third-party software and a tester with knowledge of eCATT. This meant that each Scrum team was able to implement functional user stories end-to-end and ensure these with automated tests.

Early integration. Focusing on the minimum viable product enabled early integration of all systems. The interface between the two SAP systems was already implemented in the first sprint using a minimum data set. In the following sprints the interface was expanded step by step and the third-party software was integrated.

Communication. In order to integrate the many - especially operatively affected - persons, the managers of the departments had regular info-exchange meetings with the product owners of the two teams that took the form of backlog groomings. In order to integrate users “key user groups” were formed that covered a representative cross-section of the users. The key users not only took part in sprint reviews, they also worked regularly with the Scrum team during the sprint.

The Results. This project was completed perfectly on budget and in time. But even more important: all those involved recognized what they really needed thanks to the iterative approach and many test runs. Many of the requirements originally specified either disappeared completely from the "wish list" or were replaced by new ones. The risk of an incompatible interface also went down drastically thanks to timely integration of the modules. The department responsible for the project was so surprised and impressed by the fast and "correct" results that it intends to carry out all its commissioned projects with Scrum in future.

WHAT YOU SHOULD PAY ATTENTION TO WHEN STARTING OFF WITH AGILE METHODS

When introducing Scrum many companies concentrate simply on implementing the "Scrum handcraft" in the development teams. If you want to implement Scrum successfully on a permanent basis, you need a broader perspective. So keep an eye on the following three points in particular when introducing Scrum:

- **Involvement of the periphery.** Scrum does not only concern the development team. "Agility" is not a product; it is an attitude that everyone must be committed to. For this reason the management and the customers (i.e. the departments) must also be supported with committing themselves to agile methods and values. This is the only way to bring about a constructive atmosphere for the development of products and solutions.

- **Modern development practices.** Test-driven development and continuous integration are essential nowadays in order to manage the complexity of systems in small iteration steps. To achieve this development teams need the required technical infrastructures and possibilities – and especially the appropriate knowledge.

- **Self-organization needs leadership.** Agility is an interplay of leadership that provides the necessary framework and self-organization. Lateral leaders, such as ScrumMasters, but also product owners and managers, need support to enable them to develop a new self-conception. And they must know how best to handle the social dynamics.

The market for supporting organizations with the introduction of agile methods is still veryyoung. But it is growing continuously, because word of its benefits has spread far beyond just software development. There are nevertheless only a few specialist providers in the Germanspeaking area that have more than five years of in-depth experience with agile methods, both on the team level as well as in the scaled environment. There is even less experience in the ERP environment, because agile values are only just starting to gain ground in this field. So keep a critical eye open when selecting your partner for this path.

HOW DOES BORIS GLOGER CONSULTING WORK?

Agile product development is the ideal way to equip your company and its ERP system landscape for the dynamics of the market. So it is our aim to render ourselves unnecessary: you yourself and your employees must live agility – we cannot do that for you. But we can build the necessary foundation: from the very outset we use simple images, methods and clear processes to create a common understanding among all those involved – from the development team to the management - of what agile methods mean for the organization. Based on this common understanding we establish pilot groups that decide how they should start with the tasks necessary for the implementation and how they can go about solving these. This includes setting up a first Scrum team: this team runs through several sprints during which it recognizes together what we can improve.

We involve the management from the very outset, which then takes an active part. It makes the necessary decisions and itself learns what it means to work with agile methods. This integration is very important for us, because trust in the new methods must be built up on all levels. We integrate the appropriate development practices step-by-step during the work with agile methods, we also integrate your external partners and the entire organization starts to concern itself with the new way of working.

That is one possible way of doing things. We always work together with you to decide on the path that best meets the exact needs of your company – a path in line with your initial situation and your goals. We remain at your disposal during the entire implementation process.

Let's discuss your challenge!

For us mutual trust is the essential prerequisite of a cooperation. We show you who we are, how we work and what we can do for you in a non-binding talk. Just give us a call or write to us!

borisgloger consulting GmbH
Lichtentaler Straße 7
D-76530 Baden-Baden
T +49 (0) 72 21.398 737-0
M +49 (0) 151.54 40 12 61
F +49 (0) 72 21.398 737-10
office@borisgloger.com
www.borisgloger.com

About borisgloger consulting GmbH
The borisgloger consulting GmbH with its headquarters in Baden-Baden and Vienna was founded in 2008, has 20 employees and ranks in the DACH (Germany, Austria and Switzerland) region as one of the leading management consultants in the field of agile change management and agile product development. We focus on the management framework Scrum. borisgloger consulting also offers training and consulting for specialists and managers in the field of agile management. Our customers include among others the Scout Group, Roche PVT, otto.de, Deutsche Post and the Ergo Direkt insurance company.

More information at www.borisgloger.com