Enterprise Transformation using a Lean and Agile Toolbox

Stefan Bente[1], Uwe Bombosch[2], Shailendra Langade[3]

[1] Cassini Consulting GmbH, Halskestr. 46, 40880 Ratingen, Germany
stefan.bente@cassini.de
[2] Platinion GmbH, Im Mediapark 4a, 50670 Köln, Germany
uwe.bombosch@platinion.de
[3] Tata Consultancy Services, Gateway Park, Road No. 13, Mumbai 4000093, India
shailendra.langade@tcs.com

Abstract. Despite large investments, a high percentage of IT transformations fail to live up to expectations. Amongst the reasons are lack of stakeholder buy-in, bureaucratic processes, and an ivory tower mindset. No wonder that some blame IT in general as heavy-weight, time-consuming, and inefficient.
A pragmatic antidote is to introduce lean and agile concepts. They have proven their ability to streamline complex processes in manufacturing and software development. A lean and agile toolbox for enterprise transformation uses an iterative approach, broad participation across hierarchy levels, and stripped-down processes.
Such an agile approach can be used for strategic IT planning, including the C-level decision makers. Transformation work is clocked by specific iterations, which are compatible with the TOGAF phases. The architecture models for complex IT transformations are created and controlled in custom-tailored "scrum of scrums" teams that makes sure all stakeholders are involved. An innovative interpretation of the TOGAF framework allows to use a lean Kanban approach in the EA work itself.

Keywords: Enterprise transformation, enterprise architecture, IT transformation, lean, agile, Kanban, scrum, scrum of scrums, toolbox

Enterprise Architecture – Where it works, and where it fails

Information Technology (IT) is pretty young as compared to many other industry sectors. It has been used in the mainstream business only for fifty odd years. Over the last few decades, it has made a startling impact on the way enterprises do business. As enterprises started relying more and more on IT to support their day-to-day business operation and to enable their long-term business strategy, the role and contribution of IT in the enterprise gained more and more importance.

With a continuous addition of new applications, prolonged retention of legacy platforms, and a growing number of system interactions, the enterprise IT landscape has gradually reached a remarkable level of complexity. Apart from incurring heavy costs, this complexity impedes business changes, and therefore causes frustration among the business people. For business, IT is an asset and a liability at the same time.

Enterprises want their IT to be stable, agile, adaptable and efficient. Unfortunately, these noble goals are a far cry from reality today. IT invariably remains brittle, sluggish, inflexible, and expensive. This IT nuisance, in most of the enterprises, is a compounding effect of the complexities originating from many sources:

- Complex business operations
- Technology changes
- Immaturity in software engineering
- An uncontrolled proliferation in the IT landscape (sometimes dressed as lack of organizational ownership in IT)

The task of Enterprise Architecture (EA) is that of a *hygiene factor* for the complex IT landscape. An enterprise architect must safeguard IT as a corporate asset. Every opportunity to eliminate needless complexity and costs in the IT landscape must be spotted and leveraged. In addition, the enterprise architect must make sure that every new addition or change to IT is justified by an agreed business purpose, and has a committed business value.

The grey reality

As much as the need for IT transformations in today's enterprise world is acknowledged, their execution often fails to meet the expectations set into them. In our jobs as architects and consultants, we have seen quite many enterprises from inside over the past years. We have seen a large firm, where the absence of an effective EA yielded shortcomings across all organizational units in touch with IT:

- Business managers who had no clear conception of how their business unit is utilizing IT
- Technical application owners who did not know which systems their application was interfacing with
- Developers who felt uncertain about which framework to use (and consequentially downloaded whatever they felt suitable from the internet)
- Operational staff who desperately experimented with re-starting a bunch of information systems in production, because they had no idea about their interdependencies

On the other hand, we encountered companies that, despite having a fully institutionalized EA in place, were in a state close to paralysis. The conglomerate of business, organizational, and technical dependencies had grown to a muddle that made reasonable changes impossible. As a consequence, they were still able to operate their existing IT assets for daily work – but unable to move in any future direction. They had usually gone through several not-so-successful application rationalization attempts over the years.

In one such case, the only possible way to find out whether an application was not in use anymore, and could therefore be actually ramped down, was to literally place a red phone beneath the hosting server and then shut down the server. If no angry calls would come through for a period of one month, the conclusion that no one needed the application anymore was considered safe. Everyone can draw his own conclusions on such an approach.

IT transformations being cancelled, a wrong portfolio of initiatives taken up, totally overambitious harmonization programs, anarchy due to ineffective governance – the list of possible IT blunders could easily be prolonged further. The reasons for such problems are usually manifold, of course, and cannot be pinned down to malfunctioning enterprise architecture alone. However, as we have seen in the previous section, EA is supposed to prevent such failures – so why do they keep happening?

Although EA has reached the mainstream, a skeptic undertone with regard to its effectiveness has always been there. In 2004, after completing many years of EA effort, the General Accounting Office (GAO)[1] reported a standstill in EA maturity within the US government agencies: "Of the 93 agencies that we reported on in 2001 and 2003, 22 agencies (…) increased their maturity, 24 (…) decreased their maturity, and 47 (…) remained the same." [9].

Adopting a harsh critic's perspective, one could claim today that:

- EA does not scale to create any visible impact in a large enterprise setup
- It is not equipped with the right approach and toolset to cover the entire scope of work, from strategy through implementation
- It fails to keep pace with the speed of change in modern business

Yet, according to literature and own experience, spectacular failures happen only in a minority of EA programs. As a general rule, case studies documenting the effect of an EA program – be it success or failure – seem to be rare. Especially we have not found any documented cases where EA programs have actually been cancelled, and the EA organization been disbanded. More often, EA seems to slip into a grey mediocrity: Fulfilling some promises, but failing others. An EA organization working in this mode is too lively still to be abandoned, but does not have enough strength to effectively set the course for the whole IT organization.

The lean and agile toolbox for enterprise transformation

EA is far from having the sustainable and sweeping effect that it promises. But how can we do better, and avoid these extremes?

The key success factor is to what extent a collaboration on EA between all stakeholders comes alive. The often recited mantra that the executive board's commitment to an EA initiative is the deciding momentum falls short of the mark. If

[1] The GAO is a part of the legislative branch of the US government. It also functions as the audit, evaluation, and investigative arm of the US Congress.

an EA initiative is running into a dead end, it is mostly the breach between strategic vision and ground level reality that makes it go astray.

The commitment of the CEO is a necessary, but not a sufficient condition for closing this breach. It fails to be a substitute for a living, self-dependent collaboration. With commanded EA, people must collaborate, because the chief is saying so. But as soon as high-level management attention goes away, EA again becomes stale, unless it is valued and supported by the ground level personnel being involved into designing, developing, operating, or simply using IT. Hence, collaboration is the leitmotif for a better way to practice EA.

But how can we elicit collaboration, and apply it as an antidote against a stale EA? A natural approach for unloading some ballast from EA can be found in the principles of *lean and agile software development*. Cornerstones of these methodologies are reduced process overhead, people empowerment, and a strong focus on deliverables. Lean and agile methods can help making EA more down-to-earth, efficient, light-weight, and flexible, without neglecting the traditional qualities, processes, and tasks of EA work. The essence of *lean and agile toolbox for enterprise transformation* can be expressed in two central maxims:

- Establish a lean set of processes and rules, instead of overloading the stakeholders with bureaucratic processes and unsolicited artifacts
- Adopt evolutionary problem solving, instead of blueprinting the whole future rigidly on a drawing board

To avoid one misunderstanding right from the start: We do not assume that all IT development projects are executed according to the lean or agile methodology. Our approach to enterprise architecture works with waterfall projects, just as well as with agile or lean projects. The proposal we are making here is to *apply lean and agile principles to the EA processes themselves*, irrespective of the methodology used in the downstream software development projects.

Practicing lean and agile methods means to welcome change, planning and executing incrementally, and focusing on structured human interaction instead of channeled reporting lines. These methods are an enabler for a reduced and efficient organization to deal with complexity – exactly what EA needs. By applying these methods to the creation of *architecture*, instead the creation of software, we follow the above two guidelines, when dealing with EA activities on a day to day basis.

The focus on *continuous delivery* avoids an EA that is just dreaming of the future, instead of realizing short- and mid-term benefits. *Synchronization points* (iterations, demos, frequent planning and feedback workshops) ensure that EA is aware of the demands of the stakeholders, and can easily adapt to changes in their requirements. The lean methodology, on the other hand, teaches us how to establish *lightweight processes*. Work-items are pulled by demand, instead of pushed under (often false) assumptions.

No.	Tool	Goal
#1	*Get rid of waste by streamlining architecture processes*	Values the sparse time of enterprise IT stakeholders by focusing on lean processes with as little management overhead as possible
#2	*Involve all stakeholders by interlocking architecture scrums*	Makes sure that all stakeholders are involved by focusing on structured human interaction as a main channel of information flow
#3	*Practice iterative architecture through EA Kanban*	Welcomes change, and favor iterative design over large-scale upfront planning, and supports this approach by tools and methods

Table 1. Lean and agile tool #1 through #3

The practical implementation of these concepts has been condensed into three concrete tools, as outlined in Table 1. We will discuss them in more detail in the following sections.

Tool #1: Get rid of waste by streamlining architecture processes

Lean management has its origin in the Toyota Production System (TPS). End of the 1940s, the company faced the challenge to produce inexpensive cars for a striving post-war society. The TPS, introduced by Taiichi Ohno [5], focused on optimizing the production efficiency by consistently eliminating "waste", unnecessary or even harmful steps in the design and production processes. At the same time, aspects of human motivation gained a high degree of importance, adding "respect for humanity" as an explicit value.

In the late 1980s, under the impression of the Japanese economic miracle of the past twenty years, Japan was hyped as a role model for industrial efficiency in the Western world. Against this backdrop, *lean* became a popular buzzword in management theory. The term was coined by John Krafcik [2], and made popular by the books of James P. Womack [10] and others. They transferred the lean management principles to general business organizations. Mary and Tom Poppendieck [6,7] eventually established lean thinking as a software development method by condensing the lean thinking into seven principles, adapted to the software creation process.

Waste removal from EA processes

Tool #1, *Get rid of waste by streamlining architecture processes,* takes up an essential element of lean development, the elimination of "waste". The essence of this tool is to analyze the EA processes for unnecessary bureaucracy, over-processing, and delays, and streamlining them by removing such wasteful activities.

Production waste is not only a phenomenon in manufacturing, but also occurs in the non-physical world of software development, IT processes, and the creation of enterprise architecture models. Waste in EA processes can be classified under the following categories[2]:

1. *Partially Done Work:* Information is not available, and architectural artifacts are not completed, but stockpiled in a semi-finished state
2. *Over-Architecting:* The wrong architectural issues are tackled, or the right ones are covered with too much detail
3. *Redundant Processes:* Essentially the right things are done, but with too much effort; inadequate, unnecessary or duplicate processes rendering the overall process inefficient
4. *Handoffs:* Friction occurs in the interaction of different actors in the EA processes, due to task handovers or communication problems
5. *Task Switching:* Enterprise architects and other participants in the EA process lose time and energy to switch between many concurrent tasks
6. *Delays:* People wait for information, feedback, or approvals
7. *Defects:* EA strategy, models, and documents contain flaws, or participants in the EA processes display erratic behavior

A detailed analysis of each EA waste type would exceed the scope of this paper. A comprehensive list for each type can be found in [1]. As an example, we list possible types of waste created by *Over-Architecting*:

- IT strategy derived from industry hype (SOA, WOA, Cloud, …), rather than business maxims
- Vendor-driven, technology-focused, or product-centric IT strategy with no correlation to what the business is asking for
- Overly complex solutions for simple problems (like “Do everything with SAP”)
- Modeling too detailed (e.g. for a mere strategy option, where a superficial model would suffice)
- Architecture view too formalized for an audience of non-architects
- IT transformations not justified by business needs or by IT strategy
- Up-front investments into technologies and platforms in anticipation of future needs
- Planning horizon reaching too far into the future
- Transition to EA standards, policies, principles, and guidelines that later turn out to be irrelevant, abandoned, replaced, rejected, or ignored
- Over-specification of corporate standards and guidelines
- Too much involvement of enterprise architects in the nitty-gritty details of IT projects
- Key architecture concepts (like adaptation, mediation, transformation) only introduced on project level because EA compliance requires it, not because they are suitable for the project

[2] The EA waste interpretation as outlined in this section is mainly derived from [4, 6, 7, and 10], These authors describe waste in lean manufacturing and product design. We have adapted their terminology and ideas to EA processes.

- Mitigating IT risks that that are improbable, too far in the future, or have already materialized
- Architecture as an overreaction to never-to-happen-again problems of the past

Analysis tools for EA waste detection

Even if we know what kind of waste EA processes can *potentially* contain, it is still required to analysis the EA processes of a concrete enterprise in order to spot opportunities for waste removal. In the lean methodology, such analysis is usually carried out by using a set of tools summed up as *value stream mapping*[3]. One of them, the *Process Activity Map*, we will take a closer look at in this paper.

The removal of waste is, in lean methodology, inseparably linked to the analysis where value is generated. In other words, the *EA value stream* needs to be analyzed. Trivial as it may sound, the first step in value stream analysis is always a clear definition of the term *value*. Any process step either adds value, or it does not. In mass production, a step is value-adding if it directly contributes to manufacturing of a work piece.

Painting a car body is value adding; storing it to dry is not. The latter might be an example for a step that does not have a value in itself, but is required by the process. Therefore, lean experts usually use the three-fold classification scheme below[4]. The goal of value stream optimization is to eliminate the *NVA* and minimize the *R-NVA* process steps.

- *Value-adding (VA):* Directly contributing the outcome of the process
- *Required but non-value-adding (R-NVA):* No direct value contribution, but required due to the technical nature of the production process, for instance coordination meetings
- *Non-value-adding (NVA):* Process steps that are not required by the process, and do not add value

The *Process Activity Map* is the central tool in value stream mapping. It depicts the flow of an EA document or model, through different kinds of processing steps. The notation is listed in Figure 1. Broad arrows symbolize major flow of information or intermediate work products, while the fine arrow stands for contributing information like review feedback. The process nodes can be:

- Start- or endpoints
- Operations (like *Generalize Requirements*)
- Other activities like review, approval, and handover between different actors[5]

[3] A comprehensive description of value stream mapping tools can for instance be found in [4].

[4] We use the definitions by [4]. Sometimes R-NVA is also abbreviated as NNVA (necessary but non-value-adding).

[5] The *Approval* and the *Handover* node are not part of the notation as described by McManus and other authors, but have been added specifically for EA processes.

- Temporary inventories where intermediate products wait for further processing

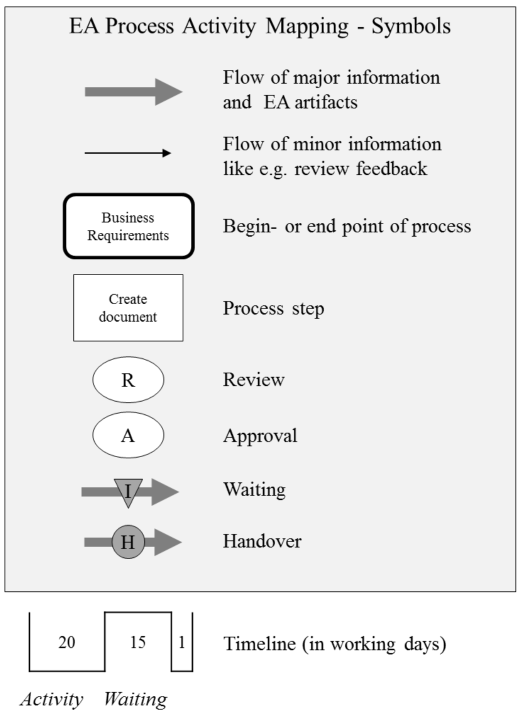
EA Process Activity Mapping - Symbols
Flow of major information and EA artifacts
Flow of minor information like e.g. review feedback
Business Requirements
Begin- or end point of process
Create document
Process step
R
Review
A
Approval
I
Waiting
H
Handover
20
15
1
Timeline (in working days)
Activity
Waiting

Figure 1. Process Activity Map notation for EA processes

Let's use an example to demonstrate the usage of value stream mapping in EA processes. Figure 2 shows a piloting project[6] to develop EA guidelines for developing Mobile applications. It was started in the context strategy given out by the CIO to bring the enterprise closer to its customers. One IT maxim in this strategy was *Full Multi-Channel Capability*, but at first the company did not have any standards in place at all that covered mobile technology.

Therefore, the EA team proposed a thorough technology and platform evaluation, targeting at the creation of compulsory standards and development guidelines. Not much mobile know-how was available in the IT organization. For that reason, the allocated budget for the project included an unprecedented luxury – the creation of small prototype, by an offshore team of the preferred implementation partner, managed by the IT organization.

Much to everybody's dismay, the project turned out to be a failure. It took a full year instead of the planned six months, overrunning schedule and budget by 100%. And yet, no one was really happy with the result. The IT crowd claimed that the

[6] The example itself is hypthetical, but is based on real experience by the authors.

guidelines were impractical and overly rigid, while the business was disappointed with the prototype's capabilities. Worst of all, due to the delay, several development projects could not wait for the guidelines to be finalized (or claimed they couldn't), and started with the implementation of their mobile applications – each using a different technology, of course.

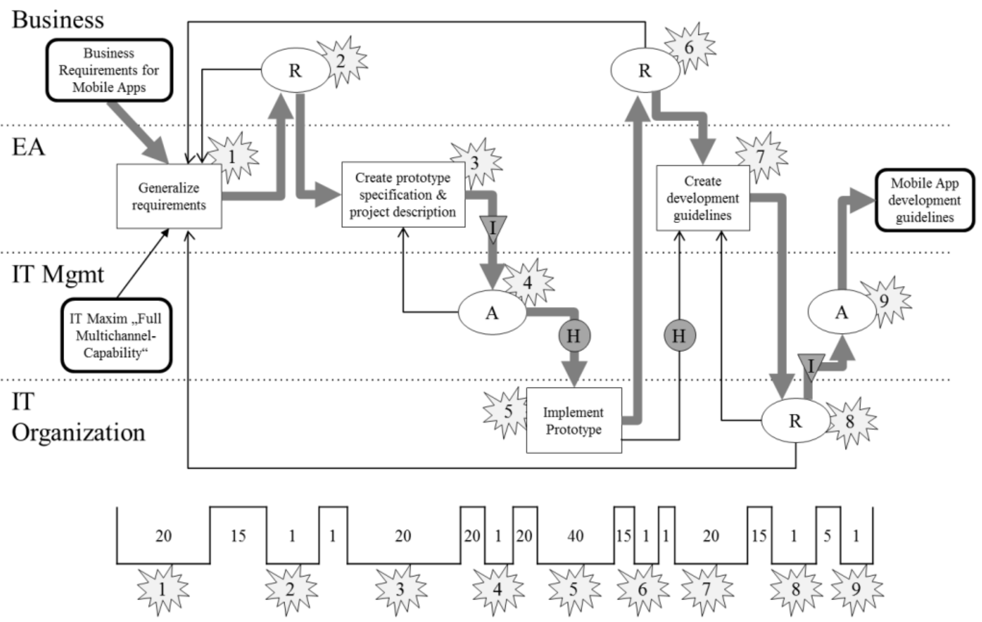
Business
EA
IT Mgmt
IT Organization
Business Requirements for Mobile Apps
IT Maxim „Full Multichannel-Capability"
Generalize requirements
Create prototype specification & project description
Implement Prototype
Create development guidelines
Mobile App development guidelines
R
A
H
I
1
2
3
4
5
6
7
8
9
20
15
1
1
20
20
1
20
40
15
1
1
20
15
1
5
1

Figure 2. Process Activity Map for an EA project developing Mobile Apps development guidelines (times given in working days)

From the description above, it has become evident that this piece of EA work has much room for improvement. But what can be concluded by specifically by drawing a Process Activity Map, as in Figure 2?

- *Long lead time before reviews.* The unwritten rules of the IT culture demanded that reviews need a lead time of roughly three weeks.
- *Each review triggered extensive rework.* The project had three formal reviews (two with business, one with IT). Each of them caused rework. In the second review (no. 6), where the finished prototype was inspected, gaps in the initial specification were detected. The third review (8) with IT architects and project managers was even worse: One requirement was discovered to be incompatible with the enterprise's security guidelines, and had to be changed in the initial specification.
- *Long waiting time before the prototype implementation.* Although the budget for the prototype implementation had already been approved, the interaction

with procurement, which had to approve the project order and select the implementation partner (3 – 4), cost a full four weeks.

- *Difficult handover to the offshore prototype team.* Another four weeks passed with the ramp-up of the prototype implementation team in offshore, and the knowledge transfer to this team via teleconference (4 – 5).
- *Poor knowledge transfer from the prototype team to the enterprise architect.* The architect in charge of the project had only limited access to the experience of the prototype implementation team when he wrote the final guideline document (7). The employees of the partner company had already been allocated to other projects, and were only available for a relatively short phone conference.
- *No requirements input from the IT organization.* Requirements were only discussed with the business departments, but not with the project managers, architects, and developers who would eventually implement mobile applications.

Besides the Process Activity Map, there are other value stream mapping tools that have proven useful in analysing EA processes. [1] provides an exhaustive overview on them.

Summary of tool #1

The lean quality of adopting a top-down, holistic perspective that is strictly goal-oriented helps in stripping off needless ballast from the EA processes. The lean approach has been tailored towards complex, repeating activities that are to be optimized.

EA has a lot of these. Lean principles and tools, like the Process Activity Map, can be used to streamline top-down decision making or design processes, like the creation of an EA vision, an IT strategy, or the design of a complex IT transformation.

Tool #2: Involve all stakeholders by interlocking architecture scrums

Tool #2: Involve All Stakeholders by Interlocking Architecture Scrums makes sure to include all stakeholders on a regular basis. It forms a kind of “clocked” factory approach to producing EA artifacts. Agile is an iterative way to design a system – be it a house, a production plant, a software system, or a whole enterprise.

Being iterative is the canonical approach for an enterprise architect, as architecture is exploratory in nature, especially due to immature and evolving nature of both business and IT today. This way, the enterprise architect is not overburdened by an up-front mental picture of the complete system in her mind. Instead, she can focus on the structure and problems of the more nearby deliveries, and trust in the ability of selective refactoring later on, in the sense of an *intentional architecture* as coined by Leffingwell [3]. The usage of sprints and architecture scrum teams introduce a regular heartbeat to architecture work, and allow for a more iterative work mode where information is shared regularly and early.

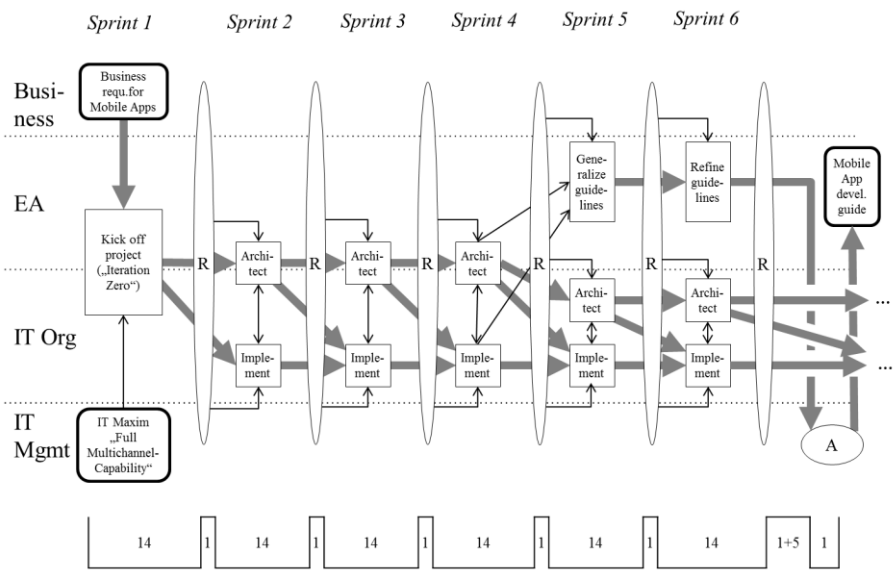

Figure 3. Process Activity Map for a lean and agile version of the EA project from Figure 2 (times in working days)

Figure 3 shows how the Process Activity Map for the EA project from Figure 2 would look like, if executed using agile sprints. It looks radically different from the original scenario. With this process model, it would be possible to finish in time. With six sprints of three weeks each, and a little more than a week for the final approval, the total duration is less than five months.

By introducing sprints, the review lead times are eliminated. There is no real waiting time in the process anymore, only the demo day is not directly productive (therefore a sprint is denoted as 14+1 days). Architecture and pilot implementation are handled within *one* project, but not merged up to a point where the architecture work is in danger to become swallowed up by many implementation tasks. Both are handled in separated tracks.

The generic word *track* has been used deliberately. There are several ways to implement this pattern. The simplest option is to handle architectural work items in the same way as implementation features, and to just tag them differently in the requirements management system. This might be sufficient our example. In larger programs or IT transformations, EA work should be handled by a dedicated architecture scrum team – we will describe possible setups for such a situation later in this section.

However, the main issue in the EA process redesign is the elimination of uncontrolled rework of architecture specifications. The stress of this statement is on *uncontrolled*: Of course the frequent reviews will provide feedback, and cause changes. But these changes are routinely incorporated into the next sprint planning – they are expected as a natural part of the process. There is constant interchange between those project members working on architecture, and those implementing features.

Major input from the architecture track is translated into implementation features only at the *beginning* of a sprint. There is a strict rule in the agile methodology not to accept additional requirements in the middle of a sprint (it can only be terminated if needed). This approach protects both architects and developers from uncontrolled ad-hoc changes and half-baked architecture decisions.

In sprints 2 to 4, the enterprise architect works as part of the project's architecture group. It allows her to learn and teach at the same time. She gathers first-hand knowledge on the technology, and can in parallel coach the project architects in their design decisions. It allows her to keep the option open to generalize certain features into an enterprise platform later.

From sprint 5 onwards, architecture is handled exclusively by the project architects. The enterprise architect uses her knowledge from the pilot phase of the project to write the guidelines document. That work is still handled as part of the project, and its result is presented in the demos. This ensures that experience, concerns, and ideas from the IT crowd are sufficiently embodied in the development guideline document.

Scrum patterns for EA work

In an agile project setup, teams are usually organized as *scrums*. A scrum team usually comprises six and ten members. A *scrum master* acts as moderator for the team. The scope of this role differs from that of a classical project manager; the scrum master is responsible for moderating, rather than managing, the team, and protecting it from outside disturbances.

Since the agile approach is all about managing change in a light-weight, efficient way, a continuous interaction with the project customer is a key to success. Therefore, another important role in the agile methodology is the *product owner*, who acts as a representative of the customer, and makes sure his voice is heard in all planning sessions.

The scrum notion has been created with software development in mind. Can it also be applied, as a mode of team formation and interaction, to the creation of enterprise architecture? Our answer is yes - we will see that the agile approach of continuous, clocked interaction helps avoiding silo effects of different groups in the enterprise.

Figure 4 sums up the notation used in this paper to describe scrum patterns. Pattern 1, as depicted in Figure 5 (left) is the team setup used in the EA piloting project we used as an example in this section. If the project is set up using agile processes, the enterprise architect would just join the architecture team and fulfill her leading, coaching, and supervising role there. If the project contains a large amount of architectural work, and has more than one architect, the introduction of a dedicated architecture scrum (indicated by a dotted line) might be advisable.

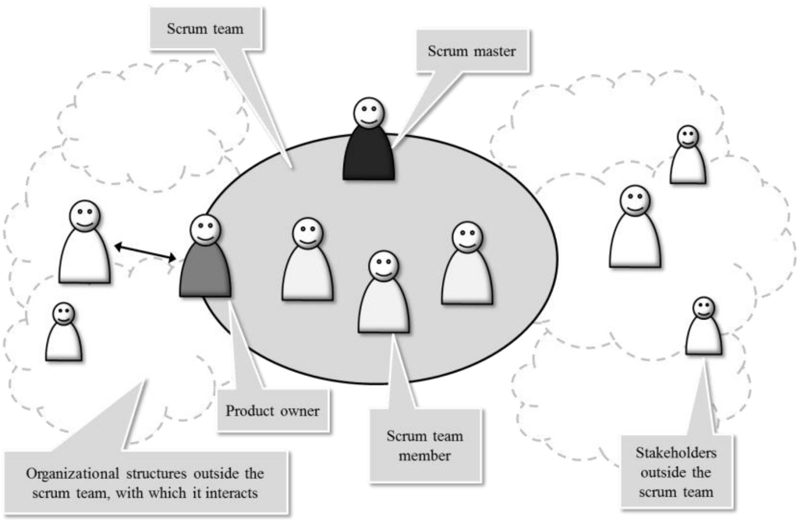
Scrum team
Scrum master
Product owner
Scrum team member
Organizational structures outside the scrum team, with which it interacts
Stakeholders outside the scrum team

Figure 4. Notation used for scrum patterns in this paper

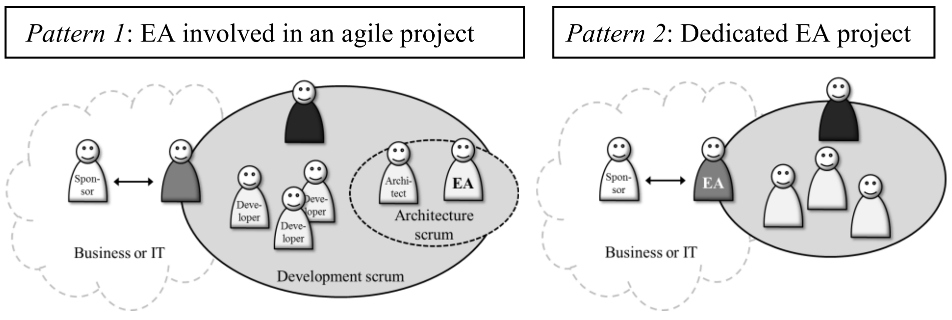

Figure 5. Left: *Pattern 1* (an enterprise architectect involved in an agile project, for instance as a technical lead). Right: *Pattern 2* (dedicated agile EA project)

Pattern 1 can be used for running any key IT project that is considered strategically critical, and therefore should be run under the technical guidance of an enterprise architect. If, however, the project is entirely dedicated to developing EA artifacts – for instance a proof of concept for a new technology, or development of an enterprise-wide platform component –, an agile setup like in Pattern 2 (Figure 5, left) is a useful option. The enterprise architect takes over the role of the product owner in this case, since she is the primary recipient of the project deliverable.

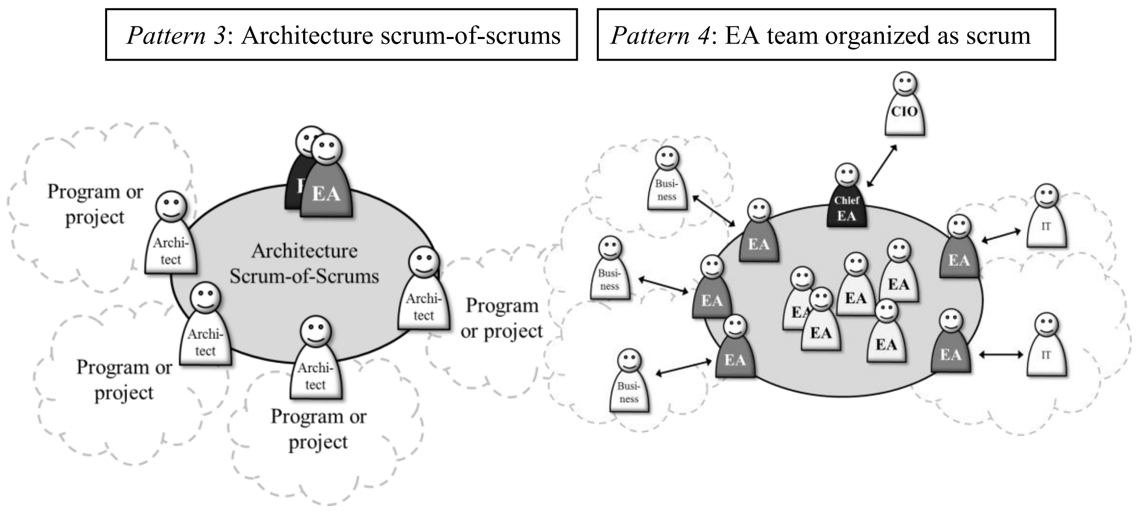

Figure 6. Left: *Pattern 3* (architecture scrum-of-scrums, for technical steering of a program or IT transformation).
Right: *Pattern 4* (organizing the EA core group itself as a scrum team)

The agile scrum patterns outlined so far work for a collaboration of enterprise architects with *agile* IT projects. But what about EA work in general, and especially the steering and monitoring of IT projects and programs that use the classical waterfall model?

The solution for the latter issue is an *architecture scrum-of-scrums* as depicted in Pattern 3 (Figure 6, left). It can be used for a variety of situations, from supervising an IT transformation consisting of several loosely related programs to the development of EA guidelines with an early buy-in from the IT community. This pattern does not require the projects or programs themselves to be agile. The lead architects from each project or program join a regular scrum-of-scrum meeting (for instance on a weekly base). The enterprise architect takes either the scrum master or the product owner role, or a combination of both.

Last but not least, Pattern 4 (Figure 6, right) depicts a model for the EA team itself. The Chief Enterprise Architect (or who else heads the EA team) should take the scrum master role. Unlike in agile software development, the EA scrum team develops more than one product, and pursues multiple different activities at the same time. This means addressing a variety of stakeholders both on business and IT side.

Therefore it makes sense to have *several* product owners in the team. Each product owner should have a deep expertise of the enterprise areas(s) and stakeholder(s) she represents, be it on the business or the IT side. This should not be a problem in a well-composed EA team, being a mix of members originating from both business and IT background.

In such a setup, the EA team should subscribe to a regular sprint and demo rhythm. This way, the main stakeholders for various EA tasks can become used to the enduring existence of an EA group (whether they like it or not). Strict adherence to a clocked rhythm for EA-related review and demo meetings helps establishing EA as a reliable key player in IT decision processes. In addition, the quality will improve due to the necessity to demonstrate the results.

Summary of tool #2

In our tools, agile principles come into the picture when EA is realized, in order to deliver timely, and to get everyone on board. To put it simply, agile is good in "doing". It has a strong focus on human interaction, which is a very important part of EA, and provides a number of practices how to structure that interaction. In addition, agile has been designed to welcome ad-hoc change. This helps in making EA more pragmatic and flexible.

The scrum patterns presented in this section offer a concrete advice how to implement an agile approach to EA. Patterns 1 and 2, with the direct involvement of enterprise architects in the practical development work, are a proper antidote against IT anarchy on one side, and EA detachment from reality on the other. With joint scrums, IT and EA are at least sitting in the same boat (even if they still might not row in the same direction all the time). That makes it harder to simply ignore each other.

Pattern 3, the architecture scrum-of-scrums, helps in all areas where EA has a supervising, monitoring, or evangelizing function. It can also be used as a way to organize an Architecture Board for technical approvals.

Setting up the EA group as a scrum team, as in Pattern 4, helps steadying the flow of EA work and decreasing waiting time. In addition, it ensures effective information flow by the regular stand-up scrum meetings within the team. Such setup is especially helpful for organisations prone to the ivory tower effect, where the constant interaction with the stakeholders is neglected.

Tool #3: Practice iterative architecture through EA Kanban

At first sight, one could consider an EA framework like TOGAF [8] an incarnation of the up-front-planning, waterfall mindset many IT professionals have lost faith in. However, the actual truth is that the process model TOGAF ADM bears many traits of an iterative work model. We will show how to combine TOGAF with the lean *Kanban* approach, in order to achieve a very flexible and transparent task planning system for EA tasks.

	TOGAF phase(s)			
Theme / Task	**ESA**	**SA**	**CA**	**Architectural Task**
1. Achieve 20% savings by application rationalization	A			Define KPIs for application rationalization
				Specify assessment methodology
	B			Define segmentation for *Segment Architecture*
	C			Revise application catalogue
				Revise data structure catalogue
	D			Analyze technologies due for retirement
	E			Create work packages for *Segment* level
	F			Estimate effort
	F			Prioritize Epics
2. Assess trading applications segment for rationalization potential		…		….
		B		Model business processes
		C		Assess application landscape using standard EA tools (e.g. segmentation according to Ward-Peppard, etc.)
		…		…
3. Explore option to slightly extend application X, in order to make application Y obsolete		…		…
		B		Re-model business architecture with hindsight to possible X extension
		C		Re-model application architecture with X extension and Y ramp-down
		…		…
		E		Create work packages *Capability* level
		F		Estimate efforts
		F		Prioritize Features
4. Specify enhancement of application X			A	…
			…	…
			H	…
5. Specify ramp-down of app. Y			A	…
			…	…
			H	…
6. Continue assess-ment with core banking systems		A		…
		…		…
		F		…
…				

Table 2. Typical application rationalization in nested TOGAF loops (*ESA* = Enterprise Strategic Architecture, *SA* = Segment Architecture, *CA* = Capability Architecture, according to the TOGAF framework [8]

TOGAF offers the option of nested loops through its ADM cycles. The top level refers to the *Enterprise Strategic Architecture*, where the program themes at an executive level are defined. Phase F of the ADM framework (Migration Planning) can be used to initiate architecture work on the next lower level, which is *Segment* Architecture. Trom there, enterprise architects can in turn can dive deeper into the *Capability Architecture*. This way, a nested loop of TOGAF ADM iterations is created.

Table 2 shows this for a prototypical IT application rationalization program in the banking context as an example.

- The theme *1. Achieve 20% savings by application rationalization* is part of an CIO mission statement. The enterprise architects go through the TOGAF phases A – F on Enterprise Strategic Architecture level to define KPIs and methodology. Then they define a segmentation of the application landscape, and create work packages on the next level.
- They start with assessment of the trading application landscape on Segment Architecture level. This requires two loops. In the first, defined by *2. Assess the trading applications segment for rationalization potential*, the initial target application landscape is drawn. Then, the EA team discovers a time- and money-saving alternative: If they slightly enhance the functionality of application X, they might be able to completely ramp down application Y. As a consequence, they lauch a second loop through phases A – F on segment level, and re-model the target architectures (3).
- Following this Segment Architecture, a deep dive needs to be started. Two architecture tasks can be worked on in parallel: *4. Specify enhancement of application X* and *5. Specify ramp-down of app. Y*. These loops through the Capability Architecture will now cover the full cycle A – H, including the actual implementation. One can imagine how much complexity this will involve. For the sake of simplicity, these tasks are not shown in detail in Table 2.
- In parallel to assessing the trading application landscape, the EA team needs to go through several other segments as well. This is only hinted at in Table 2, by *6. Continue assessment with core banking systems*.

EA Kanban as an antidote against planning chaos

As structured as the TOGAF-driven approach is, there is a major practical issue of how to coordinate all the nested and parallel architecture activities. Tracking them by the usual ad-hoc means like an Excel sheet distributed on regular base via email does not provide a really good overview where the teams stands. On the other hand, heavy-weight project planning tools, with their focus on the long-term time horizon, offer little help in the volatile world of architecture work.

Adapting the Kanban approach from lean management provides a flexible, efficient, and TOGAF-compliant way of organizing EA work. The principle of Kanban is that task cards flow through a series of process steps. In software development, these steps typically form a sequence like *Plan – Implementation Ongoing – Testing Ongoing – Waiting for Deployment – Live*. Whenever there is a status change for a task – for instance, the implementation has been finished – the corresponding card is moved one cell to the right on the so-called *Kanban board*, as depicted in Figure 7.

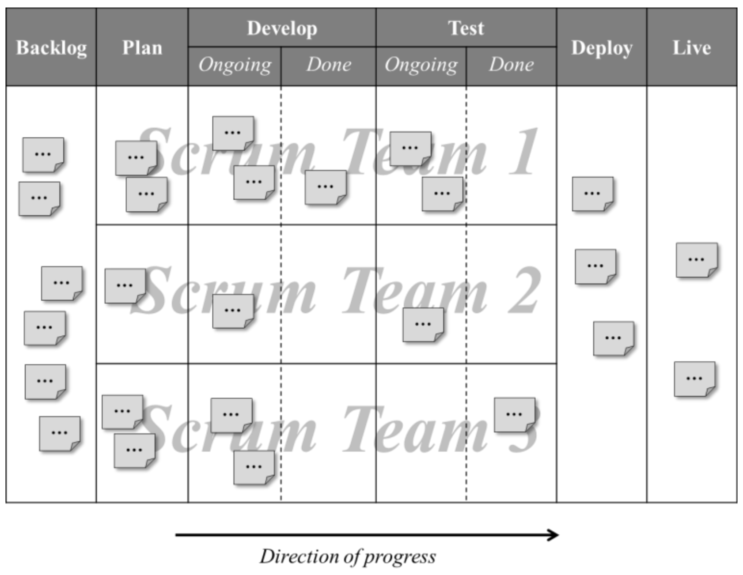

Figure 7. Kanban board as used in software development

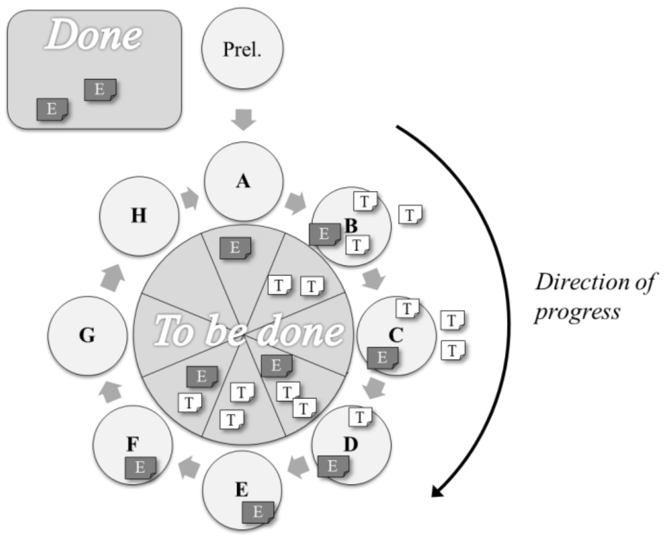

Figure 8. EA Kanban board

For EA tasks, there is a perfect match in TOGAF ADM, which already provides a detailed process model for EA activities with standardized phases. Instead of the above steps for software development, the TOGAF phases A – H (plus Preliminary phase) are used. The rich set of predefined activities within each phase provides a blueprint for possible EA tasks. An *EA Kanban board* looks like shown in Figure 8.

Architecture work items like *3. Explore option to slightly extend application X, in order to make application Y obsolete* flow clockwise through the ADM phases (instead from left to right, as in Software Kanban), triggering phase-specific tasks (like for example *Re-model business architecture with hindsight to possible X extension* in phase B) along the way. The Requirements Management phase in the middle of the TOGAF figure is replaced by a joint backlog for all phases, marked "To be done".

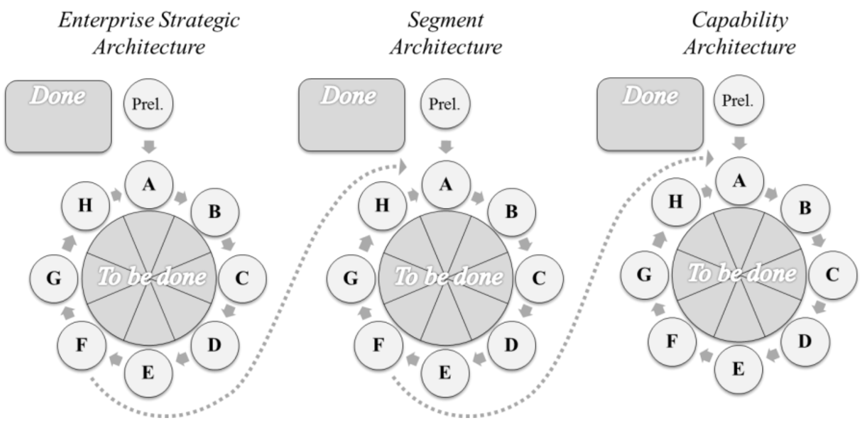

Figure 9. Nested EA Kanban boards for Strategic, Segment and Capability architecture

The real strength of the approach is achieved when EA Kanban boards for all three levels of detail are combined, as depicted in Figure 9. Detailing the rules for task flow through the EA Kanban board would go beyond the scope of this paper. The interested reader is referred to [1] for an elaboration.

Summary of tool #3

The EA Kanban approach is not a silver bullet against work overload for the EA team. It is a tool for prioritization and focusing on nearby deliveries. That way, it specifically helps maintaining a clear overview at all times. The EA Kanban Board visualizes a clogging of the task flow at process bottlenecks, making it easier to resolve them, or even avoid them altogether.

Tool #3 altogether shows that a well-established EA framework can easily be adapted to an iterative work model. Combined with three week iterations and the scrum setup described in tool #2, work results can be forwarded to the next level much faster. This way, all actors in the process have a clear synchronization point for handovers between the different roles (enterprise and project architects, business, project managers, etc.) and the parallel TOGAF cycles on Enterprise Strategy, Segment, and Capability level.

In a way, this is the successful application of the central agile notion of *continuous integration* to EA. It allows the enterprise architects to deliver intermediate results – strategies, visions, guidelines, and plans – in a flexible and light-weight manner to their recipients. EA work can be tuned to prefer *communication over perfection* when it comes to modeling.

Takeaways from the lean and agile toolbox

In this paper, we have outlined three tools for a lean and agile EA and enterprise transformation – some "waste removal" techniques adapted from lean management in tool #1, a number of specific scrum team setup patterns in tool #2, and the EA Kanban approach to handle architecture work items in a transparent fashion in tool #3.

All these tools help to structure EA activities on a day to day basis. They introduce a demand-driven workflow at all levels of operation, and make it easier to achieve a holistic result. They also elicit the participation of all, in particular the ground level stakeholders, in order to balance the helicopter view and the ground level perspective.

In that sense, the tools in this paper can be freely combined. The reader is invited to select and pick from it what looks useful in his or her daily work.

References

1. Bente, S., Bombosch, U., Langade, S.: Collaborative Enterprise Architecture: Enriching EA with lean, agile, and enterprise 2.0 practices. Morgan Kaufmann, Burlington, Sept. 2012
2. Krafcik, J.: Triumph of the Lean Production System. Sloan Management Review, 30(1), p. 41-52, 1988.
3. Leffingwell, D.: Agile software requirements: lean requirements practices for teams, programs, and the enterprise. Addison-Wesley, Upper Saddle River, NJ, 2011
4. McManus, H.: Product Development Value Stream Mapping (PDVSM) Manual 1.0. The Lean Aerospace Initiative, Boston, 2005.
5. Ohno, T.: Toyota production system: beyond large-scale production. Productivity Press, Cambridge, Mass, 1988.
6. Poppendieck, M., Poppendieck, T. D.: Lean software development: an agile toolkit. Addison-Wesley, Boston, 2003.
7. Poppendieck, M., Poppendieck, T. D.: Implementing lean software development: from concept to cash. Addison-Wesley, Upper Saddle River, NJ, 2007.
8. The Open Group: *TOGAF^{TM} Version 9.1*. Document Number G116, retrieved December 16th, 2011 from http://pubs.opengroup.org/architecture/togaf9-doc/arch/
9. U.S. Government Accountability Office (U.S. GAO): Information technology: the federal enterprise architecture and agencies' enterprise architectures are still maturing (2004). Retrieved July 27, 2011, from http://www.gao.gov/products/GAO-04-798T
10. Womack, J. P., Jones, D. T., Roos, D.: The machine that changed the world: based on the Massachusetts Institute of Technology 5-million dollar 5-year study on the future of the automobile. Rawson Associates, New York, 1990.