Ending Method Wars:
The Successful Utilization of Essence at Munich Re

Authors: Preben Folkjaer, Burkhard Perkens-Golomb, Frank Rauch, Ian Spence

This article has been submitted for publication to ACM 2015
1 Abstract

“Could you please describe how an endeavor in application development progresses over time in a healthy way?” In 2010 Munich Re’s application development department was asked this question. But as simple as the question may sound, it is as hard to answer. Even more so in a group of people with different views and experiences. One of the big challenges is the lack of a common language to allow the group to discuss, share and compare their ways of working. ESSENCE, an OMG standard finalized in June 2014, provides definitions and tools to overcome this challenge: it enables efficient discussions about a way of working, the identification of the essential elements that teams should share, the sharing of practices and processes, and the unambiguous documentation of defined ways of working. In this manner ESSENCE was successfully utilized at Munich Re to define a common way of working for all endeavors. This articles describes the issues Munich Re faced in their initial failed attempts to discuss a way of working by mainly referring to work products, the breakthroughs made by utilizing ESSENCE, and the benefits brought to Munich Re by the resulting ESSENCE-based way of working.

2 Introduction

“If you ask me, what is art, I do not know. If you do not ask me, I know.” – El Lissitzky

This article is about an art, a particular and very challenging art in application development: the art of communicating ways of working for endeavors and comparing different approaches from different sources.

The above quote appositely describes the predominant feelings when members of Munich Re’s global Application Development department were given the task to discuss, agree and describe how to run their endeavors in a healthy way.

3 Background

In the last decade, it has become apparent that, in the German job market, there is a shortage of developers and testers for application development. Not only is there a current skills shortage but all the forecasts show that the situation will become more and more critical in the future. This is why the senior management of Munich Re’s Application Development function for reinsurance decided to prepare the organization for a high level of outsourcing and offshoring in these areas. In 2008, reorganization around functional disciplines took place to enable the organization to collaborate with its outsourcing partners.

The initial expectation was that everyone – the internal IT staff and the suppliers’ staff – were experts, so did not need to be told how to collaborate and to run projects. After two years of working this way, it was realized that the initial expectations were misjudged – the outsourcing approach, which used a predominantly waterfall process, lacked efficiency and transparency. Even within Munich Re itself the different development teams, from the different countries and subsidiaries, found it difficult to communicate and collaborate. A new standardized process was needed to establish a common way of working for all involved parties: Munich Re’s Application Development functions (about 500 FTEs of internal staff members spread across multiple sites in the United States, Canada, Germany, Australia and other countries) and the suppliers (about 500 FTEs of employees spread across many countries as well). And the process had to be suitable for Munich Re’s many products and projects consuming a budget of about 240 million Euros per year.
To this end a working group was set up consisting of some of the most experienced members of Application Development from around the world. The group started to discuss their different conceptions of a “healthy course” for endeavors. Most of the contributions to the discussion referred to things like use cases, user stories, test strategies, project plans, schedules and a vast array of other work products. But, as we discovered, referencing work products to describe a process has major drawbacks such as:

- **Drawback 1:** People tacitly expected an agreement on the definition of the work products. But this wasn’t the case, not even for widely known work products. In fact people’s notion of use case, test strategy, project plan and any other work product varied considerably. But this didn’t become apparent until very late in the discussions, leading to many misunderstandings and large amounts of wasted time.

- **Drawback 2:** The same work product can occur at many different levels of detail. This led to even more misunderstandings during the discussions, because most people tacitly assume a specific level of detail, and there was no common vocabulary to talk about it.

- **Drawback 3:** You can run the same endeavor successfully with many different sets of work products (user stories instead of use cases, a master test plan instead of a test concept etc.). So there’s no set of work products that uniquely characterizes a good process. This means you can describe the same process with different terms – which again raises the risks of misunderstandings. Equally because of the different interpretations of the same work products different processes can be described in a way that makes them look the same.

For the above reasons, the discussion about a process definition based on work products didn’t go well. We were facing a big challenge: every group member, equipped with a vast amount of experience in application development, had a clear view of what the process should be – and struggled a lot to communicate the ideas in an easily understood easily digestible way. It became very evident that we were lacking a common language to express the concept of a “good” process.

When repeatedly failing to communicate their own ideas this speechlessness caused emotions that reminded people of the story of the confusion of tongues during the construction of the Tower of Babel – something the painter Gustave Doré brilliantly captured in his engraving “The Confusion of Tongues” shown in Figure 1: The feelings of high motivation shot down by the inability to communicate, turning into frustration and desperation.

There were key lessons to be learnt here, especially from drawback 3: even if people use different work products to describe a healthy process, there’s something in common between these different descriptions. The common denominator is the purpose of the work products and related activities. For example if you look at different requirement techniques used at the beginning of projects, like vision statements, use cases, feature lists, user stories and others: they share the same objective to document the common understanding of all stakeholders about the scope of the system, regardless of their different format and structures.

In other words, different work products may have different advantages and disadvantages in different contexts, but they can serve the same purpose. It’s the purpose of the work products or activities that matter, and that makes different approaches comparable.
4 Revealing the common ground: The introduction of ESSENCE

After a series of frustrating workshops and discussions – frustrating because people were more than willing to, but sadly were not able to, communicate their ideas – the working group was introduced to the concepts and terminology of ESSENCE. This was a breakthrough.

The purpose of ESSENCE is to define a common ground for software engineering. This common ground is manifest as a kernel of essential elements that are universal to all software-development efforts, and a simple language for describing methods and practices. The kernel and the language allow people to describe the essentials of their existing, and future, methods and practices so that they can be compared, evaluated, tailored, used, adapted, simulated and measured by practitioners as well as taught and researched by academics and researchers. They also allow teams to continually assess the progress and health of their software development efforts. A comprehensive introduction to ESSENCE is far beyond this article, for an introductive text to the concepts of ESSENCE, please refer to [1]. The OMG specification of ESSENCE [2] is authoritative source for all aspects of ESSENCE.

ESSENCE can be used as a tool to abstract away from the work products and describe ways of working on a objective-oriented level. In ESSENCE progress in an endeavor is described by a kind of “health indicators” known in ESSENCE terminology as “alphas”. These health indicators have state values, and an endeavor crosses these state values when progressing over time.

There’s a fundamental statement about these indicators by ESSENCE: There are seven health indicators that are universal to all kinds of endeavors in application development. They can be universally used for endeavors like in-house development, like the introduction and configuration of 3rd party software, like endeavors dealing with business intelligence, data warehouses, maintenance, and any other types. There might be additional indicators meaningful in particular types of endeavors, but the seven basic indicators are always present.

Table 1 lists the seven universal alphas with their descriptions and their state values. ESSENCE provides detailed checklists for the description of the state values, these checklists are not shown here for lack of space, but can be found at [1].
<table>
<thead>
<tr>
<th>Alpha</th>
<th>Description</th>
<th>State Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholders</td>
<td>The people, groups, or organizations who affect or are affected by a software system.</td>
<td>Recognized &gt; Represented &gt; Involved &gt; In Agreement &gt; Satisfied for Deployment &gt; Satisfied in Use</td>
</tr>
<tr>
<td>Opportunity</td>
<td>The set of circumstances that makes it appropriate to develop or change a software system.</td>
<td>Identified &gt; Solution Needed &gt; Value Established &gt; Viable, Addressed &gt; Benefit Accrued</td>
</tr>
<tr>
<td>Requirements</td>
<td>What the software system must do to address the opportunity and satisfy the stakeholders.</td>
<td>Conceived &gt; Bounded &gt; Coherent &gt; Acceptable &gt; Addressed &gt; Fulfilled</td>
</tr>
<tr>
<td>Software System</td>
<td>A system made up of software hardware, and data that provides its primary value by the execution of the software.</td>
<td>Architecture Selected &gt; Demonstrable &gt; Usable &gt; Ready &gt; Operational &gt; Retired</td>
</tr>
<tr>
<td>Team</td>
<td>The group of people actively engaged in the development, maintenance, delivery and support of a specific software system.</td>
<td>Seeded &gt; Formed &gt; Collaborating &gt; Performing &gt; Adjourned</td>
</tr>
<tr>
<td>Work</td>
<td>Activity involving mental or physical effort done in order to achieve a result.</td>
<td>Initiated &gt; Prepared &gt; Started &gt; Under Control &gt; Concluded &gt; Closed</td>
</tr>
<tr>
<td>Way-of-Working</td>
<td>The tailored set of practices and tools used by a</td>
<td>Principles Established &gt; Foundation Established &gt; In Use &gt; In Place &gt; Working Well &gt; Retired</td>
</tr>
</tbody>
</table>

Our empirical evidence at Munich Re shows: This basic assumption of seven alphas that comprehensively characterize progress in endeavors is valid. In that manner, ESSENCE serves as a “lingua franca”, a universal language, and can be used to describe processes for all kinds of endeavors in application development.
4.1 ESSENCE’s Benefit 1: Definition of an all-purpose lifecycle and variants for Munich Re

Equipped with the language, pre-defined terms and checklists of ESSENCE the working group at Munich Re was able to quickly abstract from work products and agree on a generic lifecycle for every endeavor in Munich Re’s application development (shown in Figure 2).

Figure 2 shows how combinations of the state values were used to define important milestones for the endeavors. These combinations of state values form the borders of our lifecycle phases Preparation, Inception, Elaboration, Construction, and Transition. The advantage of this approach: Even if endeavors use different work products, roles, activities, agile or non-agile approaches to progress – this lifecycle gives general guidance for a healthy course for all endeavors. And this picture, paired with the definitions of the alphas and checklists for the state values, can be used as a tool for checking the health of endeavors, as a self-assessment or for governance purposes.
After a short while the working group noted that the funding process, which is special to the company, should be represented in the lifecycle more explicitly. ESSENCE is extendable, so the working group defined an additional alpha “Funding” and aligned the new alpha with the already existing alphas, shown in Figure 3. This depicts the integration of the funding process in the overall lifecycle.

This advantage of ESSENCE, its extensibility by new alphas, is also used to give guidance to endeavors in special circumstances. E.g. if an endeavor has the objective to select and acquire 3rd party software, an additional alpha is integrated in the lifecycle to describe the progress in the company-specific acquisition process and the acquisition’s relation to the overall lifecycle.

Figure 3: Extending the Software Development Lifecycle to cover Munich Re’s funding process (the state values for the alpha “Funding” are highlighted for clarity)
To help people more effectively apply the new standard, two variants were developed to cope with different project circumstances (shown in figure 5 and 6): One for endeavors with a high risk profile ("Exploratory Lifecycle") and another for endeavors dealing with only small independent changes to an existing application ("Small Enhancements Lifecycle").
The aim of this phase is to identify the business need for the project, and secure funding to start it.

A short phase of one iteration to get the team up and running, prime the backlog and get an outline plan in place.

During this phase the system produced is handed over and accepted by the customer. The system may not be made operational particularly if the project is a feasibility study or a proof of concept. Bringing these risks under control is a major achievement for the project.

The longest phase of any Exploratory Project where the team explore the true nature of the customer’s requirements and address the technical risks threatening the project. Bringing these risks under control is a major achievement for the project.

A short phase where the system is polished to produce a production quality baseline and address any remaining requirements in whatever order the customer desires.

Please note: Often in exploratory projects there’s no clear boundary between Elaboration and Construction because the project’s risk exposure may stay high and/or no formal release is expected as outcome of the project. The intention of this phase is then working on remaining non-high-risk backlog items desired for the project’s goals.

Figure 5: The Software Development Lifecycle for endeavours with a high degree of risk and uncertainty (“Exploratory”)
Transition

Inception

Construction

Preparation

Maintenance projects update existing operational systems by fixing or enhancing to a production quality code base. This is the first phase of the MR Essentials. Projects follow priorities defined in product backlog and the schedule defined by the release plan. Funding is typically setup on a yearly basis for the product.

A short phase (usually only a few days and often less than one iteration) is required to get the delivery team up and running, and initial plans in place. Ideally coding and testing start immediately. In this phase the Maintenance Release Team decides about approach and defines release objectives and plans.

Because maintenance projects are only undertaking bug fixing and small enhancements on top of an already production quality code base the software should always be production quality, and always be complete enough to be useful. This allows the software to be released – if necessary – short term at any point in time.

During this phase the system is accepted by the customer, and goes into a production environment. The planning and maintenance of the system is handled over. Historical data and lessons learned are recorded to improve future projects.

Figure 6: The Software Development Lifecycle for endeavours only caring about small enhancements to existing applications
4.2 ESSENCE’s Benefit 2: Giving more specific guidance with practices

The lifecycles shown above are precise in the definitions of objectives for progressing steps in endeavors. They are defined using the state values of the alphas, each of which is supported by an extensive checklist. But these lifecycles don’t talk about specifics how to progress from state value to state value: what work products to use, which activities to conduct, which competencies are required, what level of detail the work products could have, etc. This reticence can be seen as an advantage, or it can be seen as a lack of guidance: this depends on the necessities and culture of the organization.

ESSENCE provides the concept of practices to describe more specifically how to progress in endeavors. The purpose of executing a practice is achieving progress, manifesting as a progression of the alphas. In that sense the alphas serve as a “backbone” for the practices: For example you might have different approaches in the area of requirement engineering (use-case-driven, or user-story-driven, or feature-driven, or test-driven etc.), but all these practices must explicitly and clearly explain how to progress the alpha “Requirements” (or even other alphas as well) with their specific work products and activities.

Munich Re decided to give, in certain areas, more specific guidance to endeavors: E.g. we want our projects to work with an iterative-incremental approach, driven by Use Case 2.0. We were especially interested in practices joining up the organizational stove-pipes by fostering collaboration between the parties affected by an endeavor (team members, stakeholders, governance units etc.). Therefore we licensed from Ivar Jacobson International the respective practices, providing definitions for work products, activities, competencies and more. We aligned the practices with our company-specific lifecycle, resulting in a precise relation between the alphas’ state values, the work products and the activities.

At Munich Re, for the most important work products their relationships to the alphas’ state values are also pictured in a diagram, part of it shown in figure 6. Shown are the alphas “Requirements” and “Software System”, plus additional practice specific alphas such as “Use Case” (introduced by utilizing the Practice for Use Case 2.0), Architecture and Component. It also shows some of the most important work products introduced by the practices. The diagram depicts the level of comprehensiveness a work product has to reach to provide evidence that a particular state value for an alpha is reached.

Use Case 2.0 is an evolution of the use-case technique. It’s a scalable, agile practice that uses use cases to capture a set of requirements and drive the incremental development of a system to fulfil them. For more information see http://www.ivarjacobson.com/Use_Case2.0_ebook/.
Figure 7: Part of a diagram, for the standard lifecycle defining the alignment between alphas and work products.
4.3 ESSENCE’s Benefit 3: Defining company-specific practices

The practices provided by external parties like Ivar Jacobson International don’t completely cover the needs of Munich Re. There are company-specific regulations, internal processes etc. influencing the way-of-working in endeavors. Before releasing the practices from Ivar Jacobson International to the staff, the most important internal guidelines were turned into ESSENCE practices as well, further practices were added over time: for example for the integration of PMBOK® Guide by PMI in our iterative-incremental way of working, for the definition of Munich Re’s funding process, for the definition of Munich Re’s role model, for the definition of company-specific processes for Test Management, for the definition of the acquisition process for 3rd party products, for the integration of activities in the development lifecycle to ensure installability, operability and supportability of the application in production, and for others aspects, too.

Using the ESSENCE approach for the documentation of our internal guidelines has several benefits:

- ESSENCE provides a “grammar” the description of practices, so that practice descriptions always share a common structure. This makes it easier for readers to grasp the content of the practice because the structure has to be understood only once.
- The pre-defined structure of a practice description forces practice authors to always answer essential questions, e.g. how the practice’s techniques progress the endeavor, depicted as progress of the alphas.

4.4 ESSENCE’s Benefit 4: Combining practices to build a Way-of-Working

In endeavors practices that define a way of working in a certain area are rarely applied in isolation. In most cases several practices (e.g. a practice describing an approach driven by Use Case 2.0, a practice describing an iterative-incremental approach, a practice describing an architecture-centric approach and others) are applied in combination. Together the practices form a process, a way of working for an endeavor.

In other words, i.e. in the terms of ESSENCE, an endeavor’s way of working, often referred to as its process or method, is an assembly of practices covering all necessary areas of work.

There’s an additional aspect when assembling practices: often practices are not independent from each other. Consider a practice describing an iterative-incremental approach introducing a backlog consisting of requirements (or “cutouts” of requirements) as backlog items. Then the practice describing the approach for requirements engineering should be able to answer the question how to slice the requirements so that these slices become well-formed backlog items. As another example, if a practice describing an architecture-focused approach is introduced it should refer to the practice describing the overall lifecycle for the endeavor, because the question when a certain level of architectural progress is reached should be aligned with the lifecycle.

Therefore, to more precisely define the creation of a process as the assembling of practices: an endeavor’s process is an assembly of practices accompanied with a resolution of the dependencies between the practices.

ESSENCE provides mechanisms to define dependencies between practices and resolve them when practices are assembled to a process. Figure 8 shows the set of key practices for Munich Re. As an assembly these practices define “MR Essentials”, the way of working in Munich Re’s endeavors in application development.

Figure 8: The key practices of MR Essentials. As an assembly they define the basic way of working for endeavours in the application development at Munich Re.
4.5   ESSENCE’s Benefit 5: Linking the way of working to the organization

Every endeavor at Munich Re is embedded in an organizational ecosystem: There are departments, sections, organizational processes, governance processes and so on, interacting with the endeavor. To work efficiently the relationships between the way of working in endeavors and the surrounding organization must be clearly defined.

For this it’s very beneficial to have the way of working described as ESSENCE practices. Because all the practices adhere to the same structure, it’s very easy to identify in every practice the elements that have to be linked to the organizational environment. E.g. for work products and activities in the practices of MR Essentials we defined the link to responsibilities in the organization, depicted in a RACI matrix. And because the practices are comprehensive regarding the way of working in Munich Re’s application development, so are the responsibility links to the organization.

5   Summary

The usage of ESSENCE was beneficial for Munich Re in several regards.

For endeavors in application development, ESSENCE provides a universal, high-level, objective-based language to describe a way-of-working. Based on this, Munich Re described several lifecycles for endeavors giving guidance for teams, stakeholders and governance units. On this level there’s no need to reference work products, roles, activities which is beneficial as they are too specialized to be valid for all the different kinds of endeavors at Munich Re.

Furthermore, ESSENCE provides the concept of practices as a tool to describe a way of working with more precision. Munich Re introduced practices from 3rd party vendors and documented the way of working in company-specific areas with their own practices. The practices then get assembled to construct the ways of working for the endeavors.

We also faced some inconveniences when introducing ESSENCE: Learning ESSENCE was somewhat challenging when we started to use it. The supporting material for ESSENCE is partially in an early state, the tool support can be improved for the usage in large and complex organizations.

Overall, ESSENCE provided great benefit for the discussion and definition of the way of working the application development department of Munich Re: It provides a solid foundation, clear structures and a precise language, both for discussion and definitions of a way of working on a high level and on a detailed level.

6   Acknowledgements

Preben, Olaf, Ivar, SEMAT initiative
7 References

- SEMAT website: http://www.semat.org
- Article “The Essence of Software Engineering: The SEMAT Kernel”, acmqueue
- Articles about practices
- The Essence of Software Engineering: The SEMAT Kernel http://queue.acm.org/detail.cfm?id=2389616
About Ivar Jacobson International
IJI is a global services company providing high quality consulting, coaching and training solutions for customers implementing enterprise-scale agile software development.

IJI improves the performance of software development teams by introducing new practices, and removing barriers to their wider adoption.

Through the provision of high calibre people, innovative practices, and proven solutions, we ensure that our customers achieve strong business/IT alignment, high performing teams, and projects that deliver.

www.ivarjacobson.com