Munich Re and ESSENCE – Kernel and Language for Software Engineering Methods: A Case Study

By Ann McDonough, Marketing Communications Specialist, Object Management Group®

The Situation

Munich Re is an insurance company combining primary insurance and reinsurance. It operates in all lines of insurance, with headquarters based in Munich, Germany and with almost 45,000 employees throughout the world. With premium income of around €28bn from reinsurance alone and with approximately 11,000 employees worldwide in reinsurance, it is one of the world's leading reinsurers.

In the last decade, Germany has experienced a shortage of testers and developers in application development. The skill shortage is expected to become more critical in the future. That is why senior management at the company's application development function for reinsurance decided to prepare the organization for a high level of outsourcing and offshoring in this area. In 2008, the function reorganized for better collaboration with its outsourcing partners.

After two years of working this way, the internal staff and the suppliers' staff realized that they misjudged how well they could work together. That year, the IT function for reinsurance at Munich Re came to the conclusion that it required a new way of developing applications.

A main objective was to define a standard lifecycle as a blueprint for a "healthy" project. The challenge, however, was that there was no common language that allowed the internal staff and its outsourcing partners to characterize the "healthiness" of a project. Participants used different approaches and different terms or even the same terms but with different meanings. As Burkhard Perkens-Golomb, IT Architect, Munich Re, stated, "My colleagues and I could sense the healthiness of a project, but we couldn't describe it."

Initially, Munich Re tried to describe a project's lifecycle based on artifacts from previous projects. However, there were a few pitfalls from this approach. For one thing, it was possible to run projects successfully with different sets of artifacts (i.e. a use case during one project, feature lists during another, etc.). Another pitfall was that documents can have varying degrees of details making uniformity difficult. Finally, there was no common understanding of the artifacts themselves. For example, team members asked what a use case was and attempted to define the difference among test strategy, test concept, test plans and master test plans. At the same time, the outsourcing agile approach lacked efficiency and transparency.

A new standardized process was needed to establish a common way of working for the 500s FTEs from the company's different countries and subsidiaries as well as the 500 FTEs working at the outsourcing firms. Although every member on both sides had a clear understanding of what the process should be, they struggled to communicate their ideas in an easily understood way. It became evident that what was lacking was a common language to express the concept of a "healthy" process.

And the new process had to be sustainable for the company's many products and projects consuming a budget of about 240 million euros per year.

At this point, Munich Re reached out for external help to overcome this situation and engaged Ivar Jacobson International (IJI), a global services company that provides consulting, coaching and training solutions for customers implementing enterprise-scale agile software development. Munich Re determined that the company needed a common view or a kernel to describe the specifics of each individual project. Therefore IJI introduced "ESSENCE" at Munich Re.

The Solution

After a series of workshops and discussions, the two groups had a breakthrough when IJI introduced the concepts and terminology of "ESSENCE – Kernel and Language for Software Engineering Methods" -- or "ESSENCE" for short. This standard was officially adopted by the OMG in June 2014. As a Platform-level member of the Object Management Group® (OMG®), IJI helped lead the effort to adopt the OMG ESSENCE standard.

The ESSENCE kernel provides common ground for defining software development practices. This common ground includes essential elements that are universal in every software development effort (i.e. Requirements, Software System, Team, and Work) and includes a simple language for describing methods and practices. These elements have states representing progress and health, so as the project moves forward, the states associated with these elements progress. The kernel also helps practitioners compare methods since instead of comparing entire methods, they can now compare practices and make better decisions about their practices since decisions can be done for one practice at a time instead of for all practices within a method.

According to the ESSENCE standard, software development is a multidimensional effort and work in all endeavors needs to progress in some proper way concurrently. In ESSENCE, progress in an endeavor is described by a kind of "health indictor" known in ESSENCE terminology as an "alpha." There are seven such alphas (see Table 1). Each alpha can take a number of states and the state of an alpha tells how far the alpha has reached in the software development endeavor. These states describe collectively the status of the project and therefore characterize the overall progress and health of the project itself.

Alphas were the key to Munich Re's success in defining healthy projects. According to Perkens-Golomb, Munich Re needed a "lingua franca" or universal language that everyone could agree on to characterize the "healthy course" of a project. With the introduction of alphas, teams were able to verbally describe the lifecycle of a healthy project.

Based on combinations of alpha values, Munich Re used the ESSENCE kernel to define three standard lifecycles for projects:

1. Exploratory: for high-risk profile projects, new development where requirements and architecture are unstable or unknown quantities
2. Standard: for medium-risk profile projects characteristic of projects making bigger changes to applications
3. Small Enhancements: for low-risk profile projects, e.g. maintenance

One of these lifecycles is assigned to a particular project based on its individual risk exposure.

In addition to the lifecycles, the ESSENCE kernel was also used as a platform to describe practices. Munich Re introduced Use Case 2.0 and Iterative and Incremental Development practices from IJI. These

practices were then aligned with the three standard lifecycles and allowed Munich Re to define responsibilities throughout the organization.

Since then, the company has defined its own practices – all practices must be reusable in the three standard lifecycles.

Moving Forward

The OMG ESSENCE standard continues to play an important role at Munich Re for several reasons: it guarantees consistency across the board for all software development projects; it helps create a learning organization by providing a solid basis and framework for the discussion and integration of learning; it helps the teams in their daily work to measure progress and health; and it helps Munich Re in scaling agile.

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.

Overall, ESSENCE provided great benefits for discussing and defining the way of working on a high level and on a detailed level within the application development department of Munich Re.

As an OMG standard, ESSENCE is freely available to the public and can be downloaded at: http://www.omg.org/spec/Essence/. A more detailed write-up of the case study on Munich Re and ESSENCE can be found here at the Ivar Jacobson website.

Alpha	Description	State Values
Stakeholders	The people, groups, or organizations who affect or are affected by a software system.	Recognized → Represented → Involved → In Agreement → Satisfied for Deployment → Satisfied in Use
Opportunity	The set of circumstances that makes it appropriate to develop or change a software system.	Identified → Solution Needed → Value Established → Viable, Addressed → Benefit Accrued
Requirements	What the software system must do to address the opportunity and satisfy the stakeholders.	Conceived → Bounded → Coherent → Acceptable → Addressed → Fulfilled
Software System	A system made up of software hardware, and data that provides its primary value by the execution of the software.	Architecture Selected → Demonstrable → Usable → Ready → Operational → Retired
Team	The group of people actively engaged in the development, maintenance, delivery and support of a specific software system.	Seeded → Formed → Collaborating → Performing → Adjourned
Work	Activity involving mental or physical effort done in order to achieve a result.	Initiated → Prepared → Started → Under Control → Concluded → Closed

Way-of-Working	The tailored set of practices and tools used by a team to guide and support their work.	Principles Established → Foundation Established → In Use → In Place → Working Well → Retired

Table 1 shows Seven Universal Alphas with Their Descriptions and State Values. ESSENCE provides detailed checklists for the description of the state values.