Essence: Reference Architecture for Software Engineering
Representing Essence in Archimate Notation

Nestori Syynimaa
Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland
Gerenios Ltd, Tampere, Finland
Sovelto Plc, Helsinki, Finland

Keywords: SEMAT, Essence, Kernel, Software Development Method.

Abstract: Essence is a standard for working with methods in software engineering. As such, it can be seen as the reference architecture for software engineering. The Essence consists of the Kernel, and a notation called the Language. This representation is not widely known and likely hinders the adoption of the Essence. This paper represents the work-in-progress of representing the Essence using ArchiMate, the de facto notation for enterprise architecture. Our purpose is to help organisations to adopt Essence by representing it in the language already understood by different stakeholders.

1 INTRODUCTION

The Essence is a standard for working with methods in software engineering. It consists of a kernel and a language for the software engineering methods. It is built by the Software Engineering Methodology and Theory (SEMAT) community. Essence allows people to describe the essentials of their software engineering methods and practices enabling further analysis and comparison (OMG, 2015). The Essence Kernel captures the essential elements of the software engineering, but it is described using the Essence Language, which is not widely known.

This paper attempts to analyse and describe the Essence using widely known ArchiMate 3 language.

2 ESSENCE

In this section, we will introduce the Essence and its constituent elements as described in Essence specification by the Object Management Group (OMG, 2015).

The Essence uses a layered higher-level method architecture as illustrated in Figure 1. The key concepts of the higher-level method are *method*, *practice*, *The Kernel* and *The Language*. As it can be seen in the architecture, a method is a composition of practices which are described using the Kernel elements.

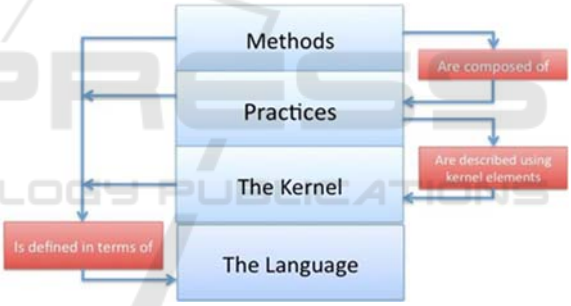

Figure 1: Method architecture (OMG, 2015, p. 10).

The Kernel is organised into three areas, namely *Customer*, *Solution*, and *Endeavor*, as illustrated in Figure 3. Each area contains *Alphas*, *Activity Spaces*, and *Competencies*.

2.1 Alphas

Alphas are representations of the essential things to work with. They capture the key concepts involved in software engineering, allow tracking and assessing of any software engineering endeavour, and provide a common ground for the definition of software engineering methods. Alphas are defined in Table 1 and illustrated in Figure 1, and illustrated in Figure 4.

Syynimaa, N.
Essence: Reference Architecture for Software Engineering.
DOI: 10.5220/0006793603450350
In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018)*, pages 345-350
ISBN: 978-989-758-298-1

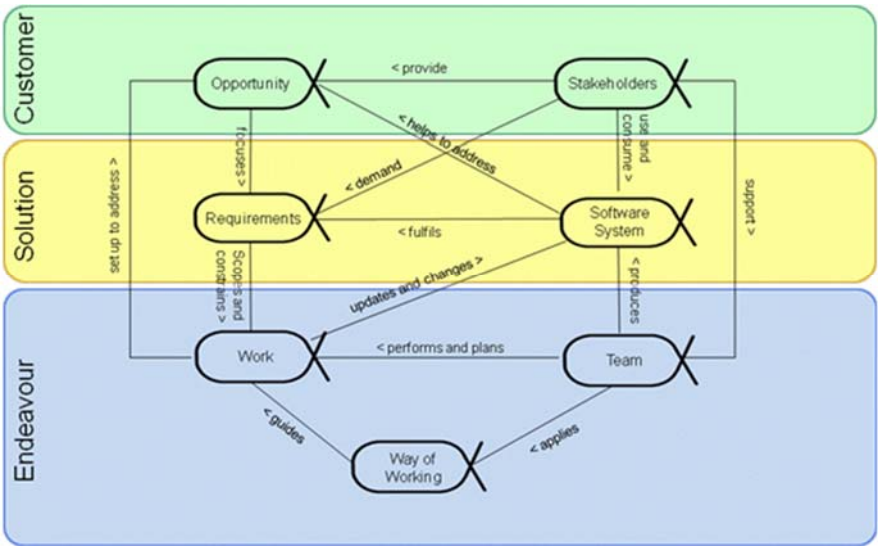

Figure 2: The Kernel Alphas (OMG, 2015, p. 17).

Figure 3: The Three Areas of Concern (OMG, 2015, p. 16).

Table 1: The Kernel Alphas.

Alpha	Description
Opportunity	The set of circumstances that makes it appropriate to develop or change a software system.
Stakeholders	The people, groups, or organizations who affect or are affected by a software system.
Requirements	What the software system must do to address the opportunity and satisfy the stakeholders.
Sofware System	A system made up of software, hardware, and data that provides its primary value by the execution of the software.
Work	Activity involving mental or physical effort done in order to achieve a result
Team	A group of people actively engaged in the development, maintenance, delivery, or support of a specific software system.
Way-of-Working	The tailored set of practices and tools used by a team to guide and support their work

2.2 Activity Spaces

Activity Spaces are representations of the essentials things to do, and they complement the Alphas by providing an activity-based view to software engineering. Activity Spaces are described in Table 2

Table 2: The Activity Spaces.

Activity Space	Description
Explore Possibilities	Explore the possibilities presented by the creation of a new or improved software system. This includes the analysis of the opportunity to be addressed and the identification of the stakeholders.

Table 2: The Activity Spaces (cont.).

Activity Space	Description
Understand Stakeholder Needs	Engage with the stakeholders to understand their needs and ensure that the right results are produced. This includes identifying and working with the stakeholder representatives to progress the opportunity.
Ensure Stakeholder Satisfaction	Share the results of the development work with the stakeholders to gain their acceptance of the system produced and verify that the opportunity has been successfully addressed.
Use the System	Observe the use of the system in a live environment and how it benefits the stakeholders.
Understand the Requirements	Establish a shared understanding of what the system to be produced must do.
Shape the system	Shape the system so that it is easy to develop, change and maintain, and can cope with current and expected future demands. This includes the overall design and architecting of the system to be produced.
Implement the System	Build a system by implementing, testing, and integrating one or more system elements. This includes bug fixing and unit testing
Test the System	Verify that the system produced meets the stakeholders' requirements.
Deploy the System	Take the tested system and make it available for use outside the development team.
Operate the System	Support the use of the software system in the live environment.
Prepare to do the Work	Set up the team and its working environment. Understand and commit to completing the work.
Coordinate Activity	Co-ordinate and direct the team's work. This includes all on-going planning and replanning of the work, and re-shaping of the team.
Support the Team	Help the team members to help themselves, collaborate and improve their way of working.
Track Progress	Measure and assess the progress made by the team.
Stop the Work	Shut-down the software engineering endeavour and handover of the team's responsibilities.

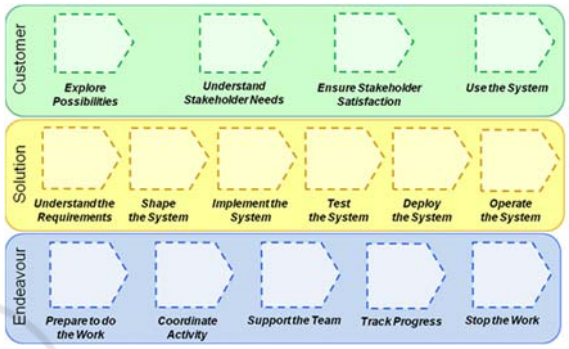

Figure 4: The Kernel Activity Spaces (OMG, 2015, p. 19).

2.3 Competencies

Competencies are representations of the key capabilities required in software engineering, which complement the Alphas and Activity Spaces to provide the key capabilities required in software engineering. The competencies are described in Table 3 and illustrated in Figure 5.

Table 3: The Competencies.

Competency	Description
Stakeholder Representation	This competency encapsulates the ability to gather, communicate, and balance the needs of other stakeholders, and accurately represent their views.
Analysis	This competency encapsulates the ability to understand opportunities and their related stakeholder needs, and transform them into an agreed and consistent set of requirements.

Table 3: The Competencies (cont.).

Competency	Description
Development	This competency encapsulates the ability to design and program effective software systems following the standards and norms agreed by the team.
Testing	This competency encapsulates the ability to test a system, verifying that it is usable and that it meets the requirements.
Leadership	This competency enables a person to inspire and motivate a group of people to achieve a successful conclusion to their work and to meet their objectives.
Management	This competency encapsulates the ability to coordinate, plan and track the work done by a team.

Figure 5: The Kernel Competencies (OMG, 2015, p. 20).

Each competency has five levels of achievements. The levels are 1) assists, 2) applies, 3) masters, 4) adapts, and 5) innovates.

3 ENTERPRISE ARCHITECTURE AND ARCHIMATE

Enterprise architecture can be defined as a formal description of the current and future state of the enterprise (Syynimaa, 2015). Typically these descriptions are provided for four layers; business, information, information systems, and technology (Pulkkinen, 2006). Some industries have industry-specific architectures which can be adopted and adapted by anyone. For software engineering, there are frameworks such as Scrum, and Kanban, which can be categorised as industry-specific architectures. These kinds of architectures are generally called reference architectures. Therefore, the Essence can also be categorised as a reference architecture for software engineering.

Boundary objects (Star and Griesemer, 1989) are artefacts that support knowledge sharing between different communities of practice (Abraham, 2013). For this purpose, enterprise architecture descriptions are often produced using ArchiMate (The Open Group, 2015) language, a de facto notation of enterprise architecture.

ArchiMate framework is organised into layers and aspects as illustrated in Figure 6.

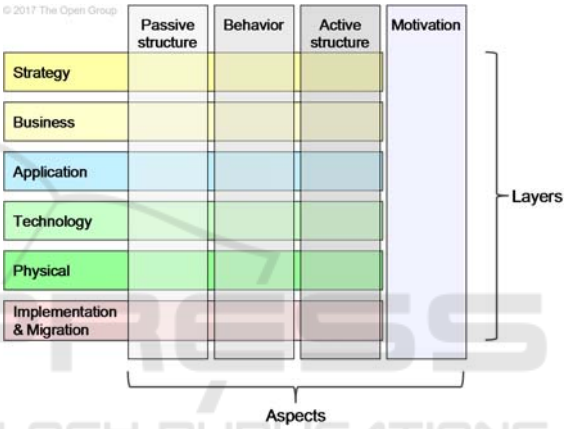

Figure 6: Full ArchiMate Framework (The Open Group, 2017).

3.1 Alphas

ArchiMate presentation of Alphas can be seen in Figure 7. First, we mapped the *Areas of concern* to ArchiMate *Group*. A *Stakeholder* is mapped to ArchiMate *Stakeholder* element. We mapped *Opportunity* to ArchiMate *Assessment* element and added an *Access* association between it and *Stakeholder*. In other words, *Stakeholder* provides the *Opportunity*. The *Opportunity* is *Realised* by *Requirements* which maps directly to the *Requirement* element. The *Requirements* are *Realised* by the *Software System*, which we mapped as a *Product*. The *Software System* is *Accessed* by the *Stakeholders*. The *Software System* is *Accessed* by the *Team*, which is mapped to a *Business Role* element. The *Team Uses Way of Working*, which is mapped to a *Business Process* element. Finally, we mapped *Work* to a *Business Collaboration* element, which *Aggregates* the *Team*. Mapping of Alphas to ArchiMate elements is summarised in Table.

3.2 Activity Spaces

All *Activity Spaces* are mapped to *Business Process* ArchiMate element and are illustrated in Figure 8.

Table 4: Mapping of Alphas to ArchiMate elements.

Alpha	ArchiMate element
Stakeholders	Stakeholder
Opportunity	Assessment
Requirements	Requirements
Software System	Product
Team	Business Role
Way of Working	Business Process
Work	Business Collaboration

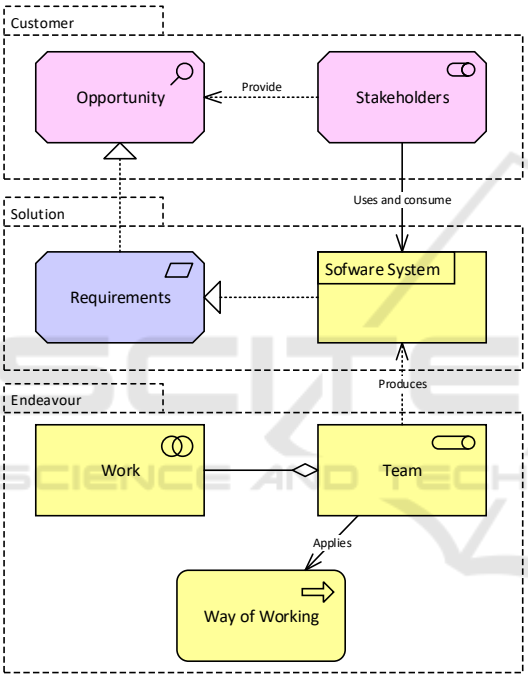

Figure 7: The Kernel Alphas in ArchiMate notation.

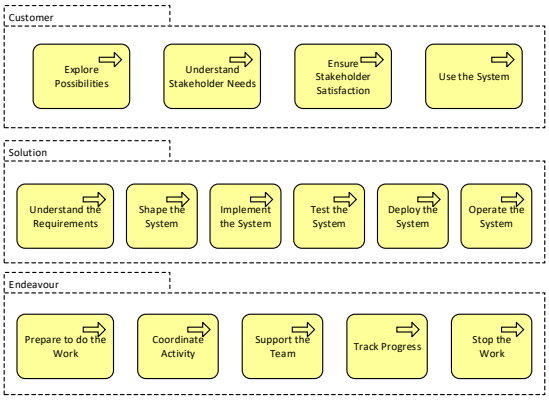

Figure 8: The Kernel Activity Spaces in ArchiMate notation.

3.3 Competencies

All *Competencies* are mapped to *Capability* ArchiMate element are illustrated in Figure 9.

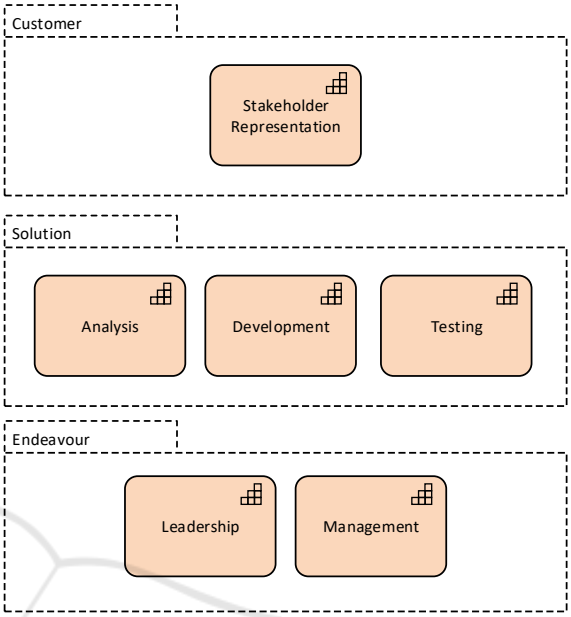

Figure 9: The Kernel Competencies in ArchiMate notation.

4 DISCUSSION

4.1 Conclusions

This study is the first attempt to represent Essence using ArchiMate, the de facto enterprise architecture notation. Our purpose is to help organisations to adopt the Essence and use it as a reference architecture for software engineering.

4.2 Limitations

In this paper, we have presented only a subset of Essence, namely the Kernel Alphas, Activity Spaces, and Competencies. The study does not cover for instance the different states of Alphas, or various Ways of Working, such as scrum.

4.3 Directions for Future Research

A more detailed representation of Alphas, including their states, needs to be created. This would further help organisations in adopting Essence. Moreover, different Ways of Working should be represented using ArchiMate notation. For instance, representing Scrum and Scaled Agile Framework (SAFe) using

ArchiMate notation would help many organisations struggling in their adoption.

REFERENCES

Abraham, R. (2013). Enterprise Architecture Artifacts As Boundary Objects - A Framework Of Properties. *ECIS 2013 Completed Research.*, (Paper 120). URL: http://aisel.aisnet.org/ecis2013_cr/120.

OMG (2015). Essence - Kernel and Language for Software Engineering Methods. Version 1.1.

Pulkkinen, M. (2006). Systemic Management of Architectural Decisions in Enterprise Architecture Planning. Four Dimensions and Three Abstraction Levels. *Proceedings of the 39th Annaul Hawaii International Conference on System Sciences, HICSS'06.*

Star, S. L. & Griesemer, J. R. (1989). Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science, 19*(3), 387-420.

Syynimaa, N. (2015). Modeling the Dynamics of Enterprise Architecture Adoption Process. *In:* Hammoudi, S., Maciaszek, L., Teniente, E., Camp, O. & Cordeiro, J., eds. ICEIS 2015, LNBIP 241, 2015. Springer International Publishing Switzerland, 577-594.

The Open Group. (2015). *ArchiMate®*. URL: http://www.opengroup.org/subjectareas/enterprise/archimate [Apr 21st 2015].

The Open Group. (2017). *ArchiMate® 3.0.1 Specification*. URL: http://pubs.opengroup.org/architecture/archimate3-doc/ [Dec 4th 2017].