# REPRESENTING ESSENCE OF SOFTWARE ENGINEERING IN ENTERPRISE ARCHITECTURE KNOWLEDGE DOMAIN

**[1]MURAT PASA UYSAL, [2]ALİ HALICI**

[1,2]Department of Management Information Systems, Baskent University, 06790, Ankara, Turkey
E-mail: [1]mpuysal@baskent.edu.tr, [2]ahalici@baskent.edu.tr

**Abstract -** Essence Framework (EF) is developed for addressing the core problems of Software Engineering (SE) domain. It provides a common ground for the definition and implementation of SE practices and methods. On the other hand, a software system with its own SE domain-specific views, processes and tools, can be regarded as an important part of an Enterprise Architecture (EA). Therefore, representing EF in the knowledge domain of EA may not only extend its usage, but also provide the understanding of EF's components and their relationships to its organizational environment that guides its design and development. In this paper, we propose a concept mapping and representation method for EF by using EA concepts. Although primarily and potential contributions are presented, there are also limitations that should be addressed by future research studies.

**Keywords -** Software Engineering, Essence Framework, Enterprise Architecture

## I. INTRODUCTION

Technical innovations, which are changing from time to time, have been driving factors for the trends and practices of Software Engineering (SE) Domain. Although various tools, techniques and methods are proposed, the fundamental problems in SE plus the gap between academic research and software industry still exists. Recently, the Essence Framework (EF) is proposed by the community of SEMAT organization for addressing these problems [1]. Its specification mainly defines a language and a kernel for the development, improvement and use of software engineering methods. The Essence Kernel and the Essence Language, together called as "Essence", allow "people to describe the essentials of their existing and future methods and practices so that they can be compared, evaluated, tailored and used by practitioners as well as academics and researchers [2]".

EF mainly provides a common base, separates the "what" of software engineering from the "how, enables method building by the composition of various practices, applies the principle of separation of concerns etc. It uses a layered architecture, where a software development method is assumed as a simple composition of software practices, which are also described by using the Essence Kernel and Essence Language. The Kernel has three discrete areas of concern, each of which focuses on a specific and core aspect of SE, namely, the customer, solution, and endeavor areas as shown by Fig. 1.

The Alphas of EF, as depicted by Fig. 1, capture key SE concepts, allow progress, and they provide a common ground for the definition of SE practices and methods. The much of EF's focus is on the software development processes, the use of practices for the composition of methods, and the agile approach that allows monitoring the health of SE endeavors and artifacts [2]. On the other hand, we believe that it is not possible to think of a single software product, let alone a SE project, isolated from its surrounding environment comprised of various enterprise processes, IT components and stakeholders.

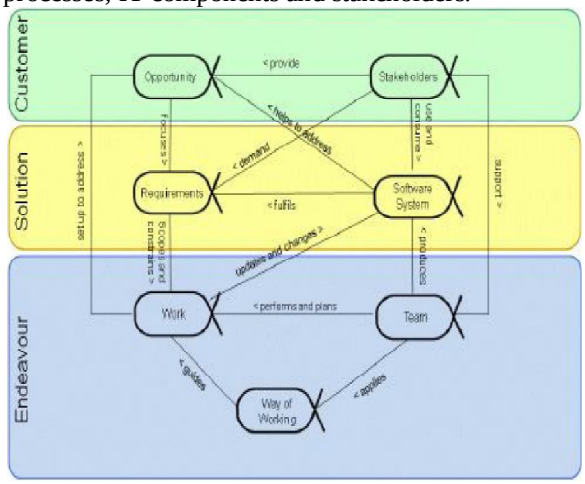



**Fig 1.Aplhas in Essence[1]**

In the same context, Enterprise Architecture (EA) is also defined as "a coherent whole of principles, methods, and models that are used in the design and realization of an enterprise's organizational structure, business processes, information systems, and infrastructure [3]". There are various frameworks used for EA development, i.e. TOGAF, Zachman, DoDAF, etc. [4]. They provide methods and principles for the EAs while presenting core architectural concepts, definitions and prescriptions on a common basis. Therefore, software, as being either a process or a product, should be regarded as an integral part of an EA. Thus, representing EF in the knowledge domain of EA may not only extend its usage, but also provide the understanding of software components and their relationships to its organizational environment guiding its design and development. However, the review of literature on EF

and EA cannot provide the sufficient work and examples.

In this paper, we representthe core EF components by using the EA concepts and Concept Algebra principles. This EA-oriented approach to EF can be regarded as the main contribution brought by this study. The next parts of this paper include the method, proposed EA and conclusion sections respectively.

## II. METHOD

The study is conducted according to the guidelines of Design Science Research (DSR) [5], which focuses on the creation of scientific knowledge when solving a real-world problem in IT domain. Therefore, the representation model proposed for EF is created by following the cyclic steps (design-build-evaluate) of DSR method [6]. The detailed steps are given in the following sections.

### 2.1.Mapping Essence Framework Concepts to Enterprise Architecture Concepts

Mapping EF concepts to EA concepts cannot be straightforward. Since modeling is a kind of knowledge representation, conceptual mapping and semantic evaluations usually require formal methods. Therefore, we initially extract core concepts from the Essence Specification, and then, the mapping procedures are conducted by using the formal theories and definitions of Concept Algebra (CA) [7]. This type of algebra is "an abstract mathematical structure for the formal treatment of concepts and their algebraic relations, operations, and associative rules for composing complex concepts [7]". CA provides denotational mathematical means for algebraic manipulations of concepts. It is used to specify, manipulate and model problems related to various knowledge domains, such as, system architectures, software and system engineering.

The compositional and relational operations are the two main operations of CA. However, when considering our research purpose, we use only the relational operations to compare and map the corresponding abstract concepts in EF to the those in the EA semantic context (Θ).A concept is defined as "a cognitive unit to identify and/or model a real-world concrete entity or a perceived-world abstract subject [7]". Accordingly, a concept connotes attributes or properties, and it denotes members or instances.Given that Θ is a semantic context, the conceptual mapping definitions are as follows:

$$\Theta = (O, A, R) \quad (1)$$

Where, the symbol *O* denotes a finite/infinite nonempty set of objects, *A* is a finite/infinite nonempty set of attributes, and *R* is a set of relations between *O* and *A*. On the basis of above elements, an abstract Essence concept can be regarded as the composition of different elements. The general structured model of an abstract concept is illustrated in Fig. 2.

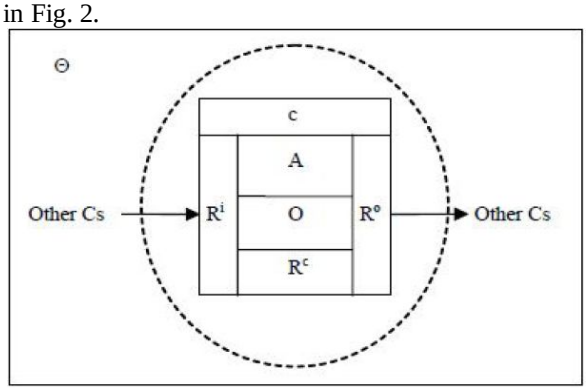



**Fig 2. The structured model of an abstract concept[7].**

Thus, an EF concept, with its attributes and objects, internal and external relations, can be defined as follows:

$$C_{EF} = (O_{EF}, A_{EF}, R_{EF}^{c}, R_{EF}^{i}, R_{EF}^{o}) \quad (2)$$

Where,

- $C_{EF}$ is a concept in Essence,
- $O_{EF}$is a non-empty set of objects extended fromthisEssenceconcept, $O_{EF}= \{o_1, o_2, \ldots, o_m\}$,
- $A_{EF}$is a non-empty set of attributes, $A_{EF}= \{a_1, a_2, \ldots, a_n\}$,
- $R_{EF}^{c}= O_{EF}\times A_{EF}$is a set of internal relations of the Essence concept,
- $R_{EF}^{i}\subseteq C'\times C_{EF}$is a set of input relations of the Essence concept and where $C'$is a set of external concepts,
- $R_{EF}^{o}\subseteq C_{EF}\times C'$is a set of output relations.

The corresponding abstract EA concept, $C_{EA}$, can be defined by adopting the same approach:

$$C_{EA} = (O_{EA}, A_{EA}, R_{EA}^{c}, R_{EA}^{i}, R_{EA}^{o}) \quad (3)$$

The relationships between the two concepts in EF and EA knowledge domains are determined by the relations of their set of attributes *A*, and the set of objects *O*. As being a dynamic mathematical structure, it is important to note that an abstract concept can adapt and interrelate itself to other concepts via input relations $R^i$ and output relations $R^o$. In our case, these are $R_{EA}^{i}$-$R_{EA}^{o}$ and $R_{EF}^{i}$-$R_{EF}^{o}$ respectively. The relational operations are defined as "related", "independent", "sub-concept", "super-concept", "equivalent", "consistent", "comparison", and "definition"; and they are represented by the $\{\leftrightarrow, \nleftrightarrow, \prec, \succ, =, \cong, \sim, \triangleq\}$ symbols respectively.

### 2.2. Concept Definitions

Let's take the concept $c_1$, "*requirements*", from EF Θ and the concept $c_2$, "*requirement*", from the EA Θ. Suppose that they have the sets of attributes ($A_1A_2$) and the sets of objects ($O_1$, $O_2$). The following definitions can be used when finding the similarity of the concepts in EA and EF:

***Definition 1*:** See whether the related concepts $c_1$ and $c_2$ share some common attributes in $A_1$ and $A_2$, which is denoted by:

$$c_1 \leftrightarrow c_2 \Rightarrow A_{1,} \cap A_2 \neq \varnothing (4)$$

***Definition 2***: Compare $c_1$ and $c_2$ and determine their equivalency or similarity levels as below:

$$c_1 \sim c_2 \Rightarrow \frac{\#(A1 \cap A2)}{\#(A1 \cup A2)} * 100\% (5)$$

Where # means the cardinal operator giving the number of elements in a given set, and thus,0% means no similarity whereas 100% means a full similarity.

***Definition 3*:** Assume the concepts "equivalent concept" as denoted by:

$$c_1 = c_2 \Rightarrow (A_1 = A_2) \wedge (O_1 = O_2) (6)$$

Which means that the two concepts having the attributes ($A_1 = A_2$) and the objects ($O_1 = O_2$) are identical.

### 2.3. Representation of EF in EA

Initially, tables or lists, which include both EA and EF concepts, are initially created by using expert judgment method.

Input : A Concept List of EF
Output: A Corresponding Concept List of EA for EF

1) While (Concepts Exist in the List EF) Do
2) Begin
3) Form a non-empty set of attributes $A_E$ for a concept $C_E$
4)While (Concepts Exist in the List EA) Do
5)Begin
6)Take a concept $C_A$ from the list EA
7)Form a non-empty set of attributes $A_A$ for the concept $C_A$
8)Find the equivalency/similarity level of $A_E$ and $A_A$
9)If (the level is satisfied) Then
10)Add $C_A$ to the Corresponding Concept List of EA
11)End
12) End

As can be seen from Fig 3, the algorithm is applied, (a) to find whether the concepts share common attributes, (b) to calculate concept similarity levels and (c) to form a corresponding concept list of EA for EF.

### 2.4. Sample Case: Implementation of EF Using EA Description Language

It is seen from Table 2 that the concepts in Essence, such as, software system, requirements, work, and way of working can be mapped to the corresponding concepts in EA straightforwardly, such as, application component, requirement, work package, and process/function respectively. Although a comprehensive comparison of theconcepts existed in the EF and EA knowledge domains is beyond the scope of this paper, however, a sample case is given as follows: Suppose that the sets of properties for the concept "Concern" in EF and the concept "View" in EA (AEF, AEA) are given in Table 1:

**Table 1: Two sets attributes of EF And EA**

| **Set of properties for the concept "Concern" in EF ($A_{EF}$= {$a_1$, $a_2$, …, $a_n$})** | **Set of properties for the concept "View" in EA ($A_{EA}$ = {$b_1$, $b_2$, …, $b_n$})** |
|---|---|
| $a_1$ = software practices divided into a collection of main areas (customer, solution, endeavor) | $b_1$ = representation of a system from the perspective of a related set of concerns |
| $a_2$ = separating the software essence from the details | $b_2$ = leaving out unnecessary information and details |
| $a_3$ = understanding stakeholders' concerns | $b_3$ = stakeholders' views of an architecture that focus on their concerns |
| $a_4$ = identifying key aspects of Software Engineering | $b_4$ = relating two or more aspects of EA elements |
| $a_5$ = containing a small number of alphas, activity spaces, competencies | |
| $a_6$ = applying the principle of separation of concerns | |

By using the definition (4), we find that
$A_{EF,} \cap A_{EA} \neq \varnothing$, and then

By using the definition (5);

$$\frac{\#(\{a1, a2, a3, a4, a5, a6, \} \cap \{b1, b2, b3, b4\})}{\#(\{a1, a2, a3, a4, a5, a6\} \cup \{b1, b2, b3, b4\})} * 100$$

The similarity level is found as

$$= \frac{4}{6} * 100 = 66.7\%$$

Consequently, it is possible to state that the two concepts are approximately similar at the level of 66.7%, and thus, the concept "Concern" can be represented in the EA's knowledge domain as the concept "View". As the result of application of this process to other EF concepts, the concept lists are formed as given inTable 2. Note that it presents only the Alpha concepts of EF that are mapped to their corresponding concepts of EA:

**Table 2. A Corresponding EA Concept List for EF**

| ***$C_E$ : Concept in Essence*** | ***$C_A$ : Corresponding Concept in Archimate*** | ***Category in Archimate*** |
|---|---|---|
| Stakeholders | Actor, Role | Behavioral Element; Motivation Element |
| Opportunit | Goal, Driver | Motivation |

| | | |
|---|---|---|
| y | | Element |
| Requirements | Requirement | Motivation Element |
| Software System | Application Component, Interface, Deliverable | Application Element |
| Work | Work package; Deliverable | Business Element; Imp.& Migr. Element |
| Team | Actor, Role | Behavioral Element Motivation Element |
| Way of Working | Process; Function | Business Element |
| Concern | View | Layer |

At the final step, the corresponding EA concepts related to EF concepts are represented using an EA development method and description language. In this study, ArchiMate [8] is used as both for the EA modeling language and IDE for the implementation processes. Moreover, The Open Group Architecture Framework (TOGAF) [9] provided the required guidance and principles for EA development processes (Fig. 4).

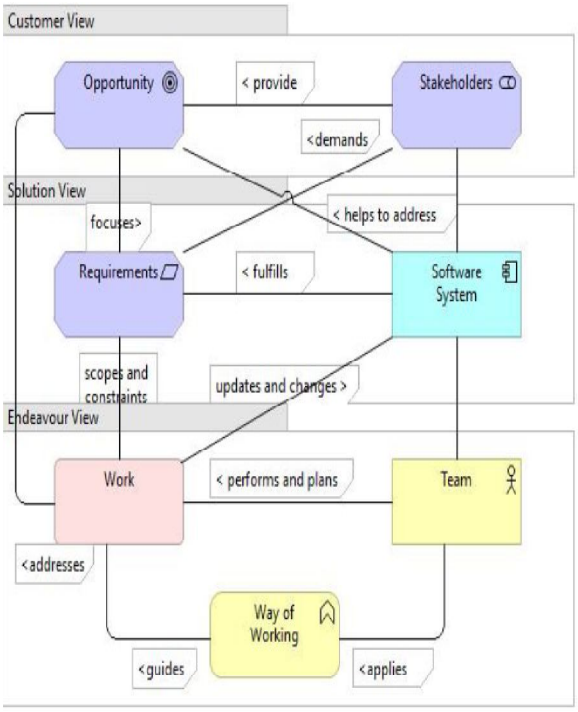



**Fig 4. Essence represented in Enterprise Architecture**

As can be seen from Figure 4, the Alpha concepts of EF Kernel are developed by the Archimate EA Development Language and IDE.

## CONCLUSIONS

As aforementioned, EF is proposed both for the problems of software industry and SE research domain. While it provides a common base and separates software essence from the details, it brings a formalism to the efforts of creating solutions to fundamental software problems. However, the main argument of this paper is that SE methods, practices and tools need to be handled in a holistic manner in an enterprise's IT context. This approach can not only extend the EF's usage, but also it can help its high level representation of relationships to its environment from the viewpoint of EA domain principles.

In this paper, therefore, the result of a this type of an applicationis presented. The study is conducted according to the principles of DSR. However, the space limitations led us to give only the conceptual background and its limited application in a small sample case. First, a mapping mechanism is formed by using CA principles and definitions. Second, a concept mapping algorithm is developed. Third, the mapping is accomplished through the use of this algorithm and CA. Finally, EF is represented using an EA description language and a development method. Consequently, this study can be regarded as an initial attempt to extending EF in a different practice domain. Therefore, future research efforts should focus on addressing the limitations and also on gathering more empirical evidences.

## REFERENCES

[1] S. Park, I. Jacobson, B. Myburgh, P. Johnson and P.E. McMahon, "*SEMAT yesterday, today and tomorrow*," SEMAT, 2014. (http://semat.org/).

[2] OMG, "SMSC/15-12-02Essence–Kernel and Language for Software Engineering Methods", Specification v. 1.1, 2015.

[3] M. Lankhorst, *Enterprise Architecture at Work: Modelling, Communication, and Analysis*, Springer-Verlag Berlin Heidelberg,USA, 2009.

[4] EABOK, Guide to the (Evolving) Enterprise Architecture Body of Knowledge, the MITRE Corporation, 2004

[5] V.K. Vaishnavi, W.J. Kuechler, "Design Science Research methods and patterns: Innovating information and communication technology", USA, Auerbach Publications, Taylor & Francis Group, 2008.

[6] Hevner, S. Chatterjee, "Design Research in information systems", Integrated Series in Information Systems, vol. 22, DOI 10.1007/978-1, 2010

[7] Y. Wang,"On Concept Algebra: A denotational mathematical structure for knowledge and software modeling", Int. Journal of Cognitive Informatics and Natural Intelligence,2(2), 1-19, 2008.

[8] Archimate, "Enterprise Architecture Description Language and Integrated Development Environment", (https://www. archimatetool. com), 2017.

[9] Open-Group, *TOGAF Version 9.1 Evaluation Form*, published in the U.S. by The Open Group, 2011.

★★★