Preface

C.M. Zapata
Universidad Nacional de Colombia

L. Castro
Universidad Del Quindío

S. Huang
Florida Atlantic University

P.-W. Ng
Ivar Jacobson International

SEMAT (Software Engineering Method and Theory) has reached a new stage. By the date of this book, SEMAT officially became a new OMG standard. These are good news for the software engineering as a discipline. In this book, we are joining the SEMAT celebration by doing what we do best: promoting the usage of the SEMAT kernel in as many ways as we can. Since we Latin American researchers are also happy with our new OMG standard, we devote our effort to promote the work on the SEMAT kernel. This book is the direct result of this effort. In twelve Chapters, we work on the foundations of the software engineering theory, the creation of SEMAT-kernel-based models, and the promotion of teaching by using the ideas behind the SEMAT initiative.

This third volume of *Software Engineering—Methods, Modeling, and Teaching* was written by authors coming from five countries: USA, China, Mexico, Colombia, and Brazil. Several points of view related to the SEMAT initiative are promoted and a lot of different theories and examples are prepared for the reader. Whether you belong to the Academic world or the Industry Practitioners, this book is very helpful for you as the seed of the usage of this new OMG standard.

This book is divided into three main parts: (i) theoretical development, (ii) method and practice representation, and (iii) teaching. If you belong to the Academy, probably you will be interested in Parts (i) and (iii) more than Part (ii). If you are a practitioner, maybe the Part (ii) will be more helpful for you. Regardless of your point of view, we strongly suggest the reading of the fundamental topics about the SEMAT kernel included in this Preface, since these ideas are cross-cutting to the entire book. Hopefully, we think you will find what you looking for inside this book: different ways to understand, practice, and use the main SEMAT ideas.

OVERVIEW AND KEY CONCEPTS OF ESSENCE

In this Section we give brief overview of Essence with focuses on describing its key concepts (Jacobson *et al*., 2012; 2013), namely alphas and alpha states and their applications.

The Essence constitutes the kernel along with the language supporting the kernel. The Essence kernel includes a stripped-down, light-weight set of elements that are universal to all software engineering endeavors. Through states defined for its elements, the Essence kernel provides a novel and effective instrument for reasoning about the progress and health of the software development endeavors in a method independent way. It helps practitioners to understand where they are, point out what they should do next and, suggest what they should improve and where. The kernel provides a common ground for understanding and describing the commonalities and diversities of software engineering practices, and this common ground is realized as a universal set of elements called alphas. Presenting the essence of software engineering in this way enables us to build our knowledge on top of what we have known and learnt, and to apply and reuse gained knowledge across different application domains and software systems of differing complexity.

The benefits of using these kernel elements are that they are harvested from existing common practices in industry, so they already exist and prevalent in software endeavor. They are what we always have (*e.g*., teams and work), what we always do (*e.g*., specify and implement), and what we always produce (*e.g*., software systems) when we develop software. Even without a well-defined method, the Essence kernel can be used to monitor the progress and health of specific software endeavor, and to analyze the strengths and weaknesses of a team way of working.

Alphas

Alphas (Abstract-Level Progress Health Attribute) are one of the core concepts of Essence. Essence uses an object-oriented approach to identify typical dimensions of software engineering challenges. These objects are called alphas. Each alpha represents a key dimension of endeavors. Alphas are separated into three different areas of concerns, which are Customer, Solution, and Endeavor (see Figure P1).

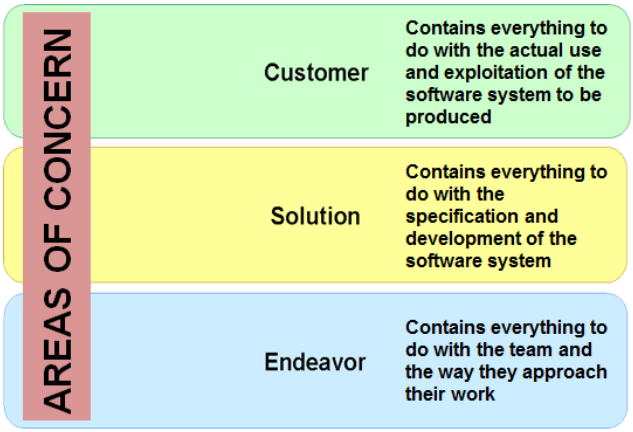


Figure P1. Three Areas of Concerns

The Essence kernel has identified seven method-independent alphas that are common to software development, namely Opportunity, Stakeholders, Requirements, Software System, Work, Team, and Way-of-Working. From Essence OMG specification (OMG, 2014), they are defined as follows:

- Opportunity is 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
- Software 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

Alphas are agnostic to practitioner's chosen practices and methods. For example, the team performs and plans work does not imply any specific order in that they perform and plan the work. These seven alphas and their relationships are shown in Figure P2.

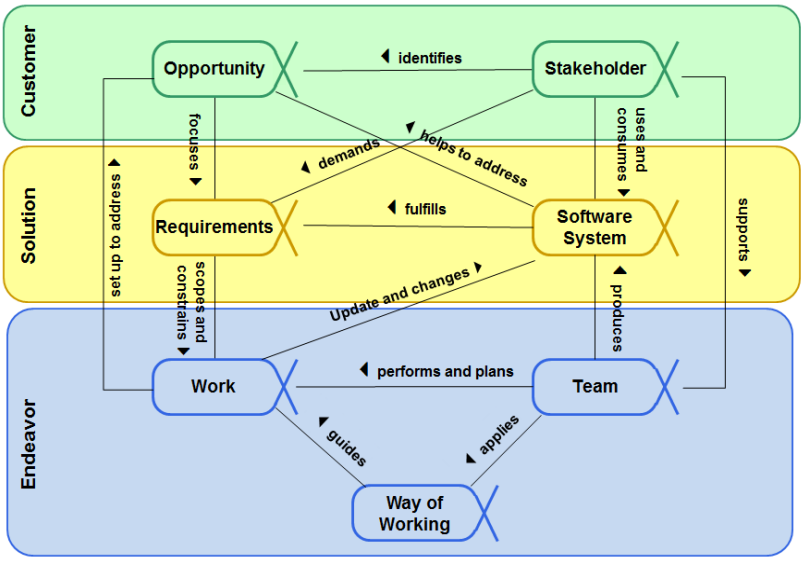


Figure P2. Alphas and their relationships

Alpha States

Each alpha has states that provide guidance for development teams to achieve progress along these dimensions and to detect risks and problems early.

The alpha states are the following:

- Opportunity: *Identified, Solution Needed, Value Established, Viable, Addressed*, and *Benefit Accrued*.
- Stakeholders: *Recognized, Represented, Involved, In Agreement, Satisfied for Deployment and Satisfied in Use*.
- Requirements: *Conceived, Bounded, Coherent, Acceptable, Addressed and Fulfilled*.
- Software System: *Architecture Selected, Demonstrable, Usable, Ready, Operational and Retired*.
- Team: *Seeded, Formed, Collaborating, Performing, and Adjourned*.
- Work: *Initiated, Prepared, Started, Under Control, Concluded and Closed*.

- Way of Working: *Principles Established, Foundation Established, In Use, In Place, Working Well and Retired*.

Essence kernel provides a detailed checklist for each alpha and their states (see Figure P3). An example of the alpha *opportunity* is shown in Figure P4.

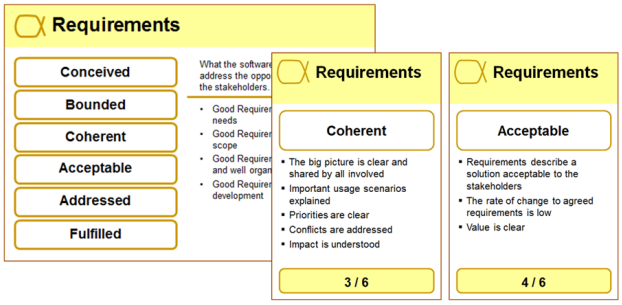


Figure P3: Requirement Alpha and Its Selected States

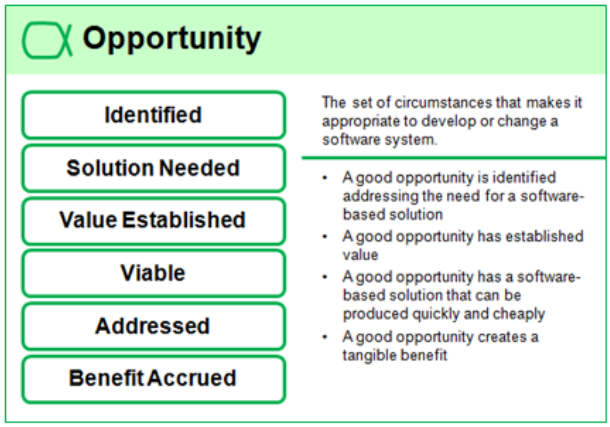


Figure P4. The alpha *opportunity* (Jacobson *et al*., 2013)

As illustrated in Figure P5, both the kernel alphas and their states can be represented as a deck of cards. These cards may be used for monitoring the progress and health of software development endeavors. The monitoring may also be supported by spider graphs (SEMAT accelerator, 2014) in Figure 5 that illustrates the common progress of all alphas.

Figure P5. Cards and progress of the alphas

Alpha states can be also arranged in a roadmap combining the alphas and the project milestones, as shown in Figure P6.

Besides the alphas ("things we always work with"), the Essence kernel defines activity spaces ("things we always do"), also classified into the three areas of concern, as shown in Figure P7.

The other element Essence kernel defines with a close set of values is *competency*, encompassing abilities, capabilities, attainments, knowledge, and skills necessary to do a certain kind of work. The set of competencies defined by the Essence kernel is depicted in Figure P8.

The Essence kernel does not include any instances of other elements—like the ones defined in Table P1—since such instances are acquired by the elements in the context of a specific practice (Jacobson *et al*., 2013). The most important associations among elements are depicted in Figure P9.

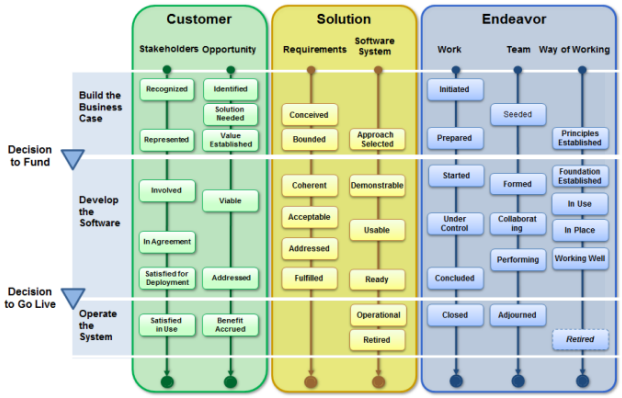


Figure P6. Alpha states arranged in a roadmap.

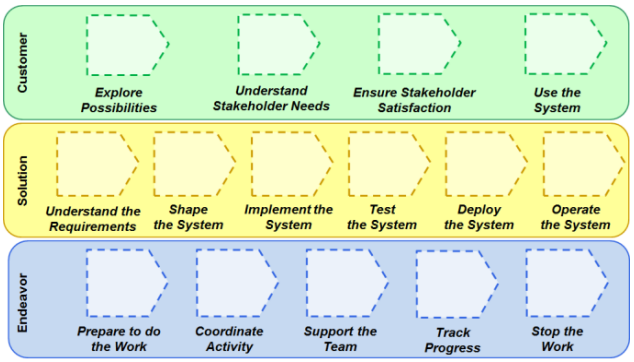


Figure P7. Activity spaces.

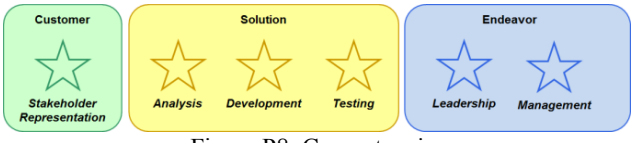
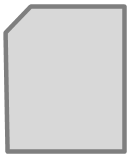
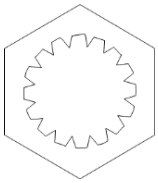
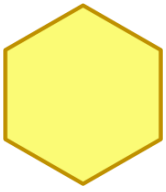


Figure P8. Competencies

Table P1. Essence kernel elements

Element	Symbol	Additional Information
Alpha		Provide descriptions of the kind of things that a team will manage, produce, and use in the process of developing, maintaining, and supporting good software. SEMAT has identified seven alphas: Opportunity, Stakeholders, Requirements, Software System, Work, and Way of working and Team.
Alpha state		Is the progress and health of an alpha. Represented in a checklist. For example the Alpha *Opportunity* has the following states: Identified, Solution needed, Value established, Viable, Addressed and Benefit accrued.
Activity space		Represent the essential things which have to be done to develop good software. For example, the Activity space called System use, which allows use of system in a live environment to benefit the stakeholders.
Activity		Define one or more kinds of works items and gives guidance how to perform these.
Competency		Encapsulate the ability to perform an activity involving the performance of work within the software engineering process. For example the competency *testing* encapsulates the ability to test a system verifying it is usable and it meets the requirements.
Work Product		Is an artifact of value and relevance for a software engineering endeavor. A work product may be a document, a piece of software, a creation of a test environment, or the delivery of a training course.
Kernel		A set of elements used to form a common ground for describing a software engineering endeavor.
Practice		Is considered as an element group which names all Essence elements necessary to express the desired work guidance with a specific objective.

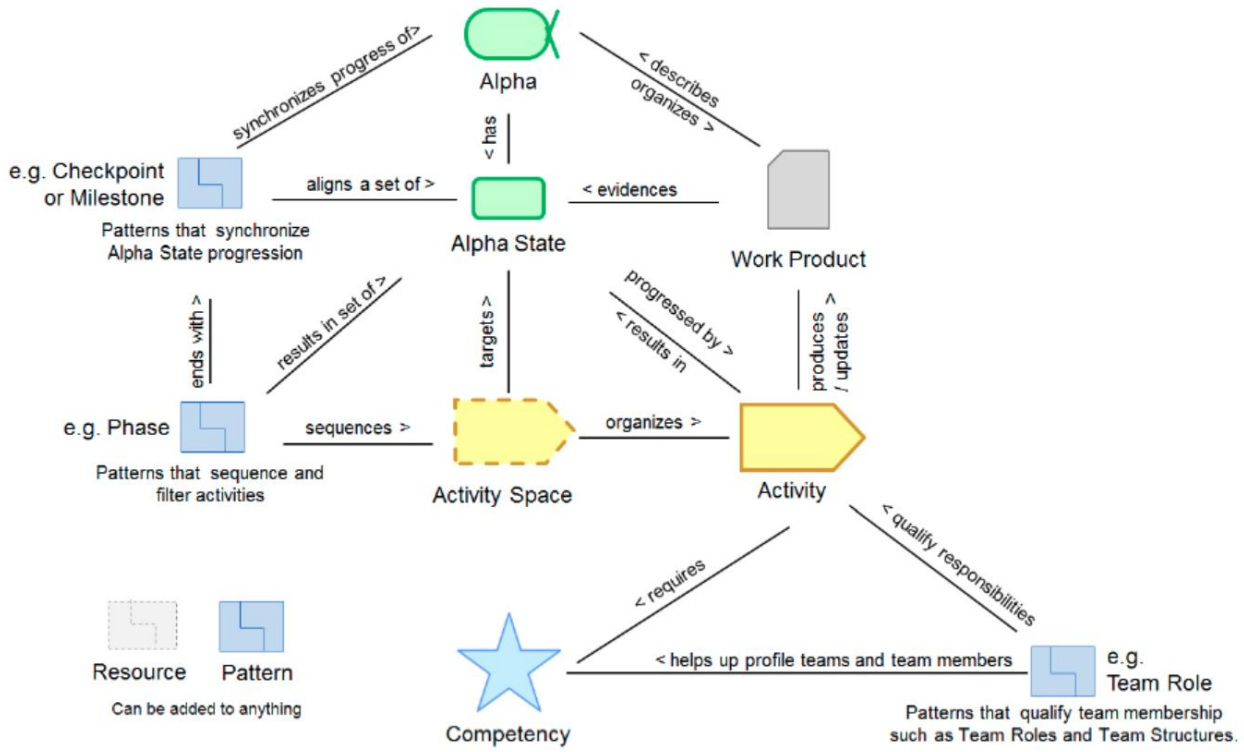


Figure P8. Competencies

REFERENCES

OMG. 2014. *Essence Submission*. Available http://www.omg.org/spec/Essence/

Jacobson, I., Ng, P.-W., McMahon, P., Spence, I. & Lidman, S. 2012b. The Essence of Software Engineering: the SEMAT kernel. *Communications of the ACM* (10): 42-49.

Jacobson, I., Ng, P.-W., McMahon, P., Spence, I., & Lidman, S. 2013. *The Essence of Software Engineering: Applying the SEMAT Kernel*. Addison-Wesley.

SEMAT accelerator. 2014. Available https://sematacc-meteor-com-ddp.meteor.com/