# A Process Framework for Applying Situational Method Engineering (SME) on OMG's Essence

Yani Widyani[1], Muhammad Zuhri Catur Candra[2], Eko Kuswardono Budiardjo[3], Benhard Sitohang[4]

[1.2.4]School of Electrical Engineering and Informatics, Institut Teknologi Bandung
Bandung, Indonesia
[3]Faculty of Computer Science, Universitas Indonesia
Jakarta, Indonesia
yani@itb.ac.id, m.candra@itb.ac.id, benhard@stei.itb.ac.id, eko@ui.ac.id

*Abstract:* Situational method engineering (SME) is an engineering process used to construct context-specific software development methods. The advantage of SMEs is to allow software development teams to work using a context-specific or situational method, that is, a method that suits their project characteristics. A situational method comprises method parts; each part has a context description that details the appropriate situation for applying that particular method. There are several types of method parts, such as method fragment, method chunk, method component, and method service. In this research, we adopt the concept of method chunk. We also use the modified metamodel from our previous study. Although there are advantages to applying SMEs, it does require extra effort. Method chunks are not easy to find, and a different notation decreases the method chunk's interoperability. This research proposes a process framework for applying SMEs. The framework's benefits are to guide method engineers in applying SMEs and provide a reference for software engineers to develop the supporting system. This framework use Essence language as a standard for method modeling to improve the interoperability of method chunks. We also apply the concept of service-oriented in the SME process to enhance the accessibility of method chunks by providing method chunk description as a service. Following the proposed framework, method engineers can extract method chunks from existing methods, publish them at a centralized publishing system to make them available as a service, and construct situational methods from selected method chunks. Software engineers can use the proposed framework to develop the supporting system. Our framework defines the complete processes for applying SMEs in a software project. The proposed framework has been validated by using the framework in a case study and building a prototype of the supporting system. Our objective is to validate the applicability of the proposed framework as a guideline. We conclude that the proposed framework is applicable, and in the end, it can support method engineers in applying SMEs in their software projects with less effort.

*Keywords*: situational method; method chunk; service-oriented; Essence; framework

## 1. Introduction

Selecting the most appropriate method for a software development project can be challenging. This paper uses the term software development method, or "method" for short, to denote the complete element needed to describe a software development endeavor in all relevant aspects [1]. Although some authors use the term methodology instead of method, we will use these terms as synonyms. In many cases, the method selection process often leads the project team to use the most popular method, which is not necessarily the method that best fits their needs. Furthermore, surveys show that many project teams mix their methods [2]. For example, Scrum combined with XP [3], or Scrum mixed with Kanban [4]. This survey shows that the existing method is insufficient for a particular situation. Therefore, a specialized situational method is needed that is appropriate for the context; it becomes necessary to combine method parts from different existing methods.

Situational method engineering (SME) focuses on constructing an organization-specific or project-specific method by composing several method parts [5]. Each method part has an

Received: August 3rd, 2021. Accepted: December 14th, 2021
DOI: 10.15676/ijeei.2021.13.4.1

attribute that describes its use context, called a situational factor. Method engineer selects appropriate method parts by inspecting the situational factor of method parts.

By applying SME, an organization or developer team can use a situational method with the help of the method engineer. To compose a new situational method, the method engineer first defines the project characteristics, such as organization characteristics, developer team characteristics, domain problem characteristics, the technology used, and others. Then, they will map these characteristics into situational factors and use them to select the most appropriate method parts.

Although there are some advantages in using a situational method, SMEs have not been widely applied in software projects. One reason is that method engineers need an extra effort to find existing and appropriate method parts. Furthermore, existing method parts may have a different notation and terminology, making it harder to compose all method parts to form a new situational method. Therefore, method engineers need a new approach to apply SMEs easier.

An example of implementing SME can be found in [6], where there was a need to develop a method for a small company in France. The authors identified two main objectives that needed to be satisfied by the new method: (1) document the enterprise business situation and (2) discover potential business evolution options based on the analysis of the produced models. They suggested using three modeling perspectives: business, business processes, and information (data). First, they searched several suitable method chunk candidates from the literature based on the author's knowledge. Then, they selected two method chunks: e3value and Business Model Canvas (BMC), as business modeling techniques. In addition, BPMN as a process modeling technique was also suggested, along with a class diagram as a data modeling technique. Finally, they merged the two method chunks into a complete method. This method was then used to assist the company in clarifying and understanding the business by producing semi-formal documentation of its business structure, processes, and domain concepts. The authors demonstrated that this new situational method was well adapted to the project and satisfied the requirements.

The above scenario shows that extra effort is needed to apply SME. Therefore, our motivation for this research is to find a way to minimize the effort involved in implementing SMEs in software projects. We propose a process framework to guide SMEs' application and development of the SME support system, which will reduce the effort involved. In this research, our process framework essentially lists the key processes of SMEs that should be performed, grouped hierarchically to show how they relate to each other. Method chunk is used as a method part definition. Using the framework and the supporting system, method engineers can extract method chunks from any existing software development method, define a method chunk in Essence language and publish it to be accessed as a service by other method engineers. Later, method engineers can build any situational method by selecting appropriate method chunks and accessing a complete description of each method chunk from the providers. Since each method chunk is described in the same notation, i.e., Essence language, it will be easier to merge them into a new situational method. Furthermore, since method chunks are described in digital format, the merging process can be automated using the computer-aided method engineering (CAME) tool.

The remainder of this paper is organized as follows. Section 2 describes some fundamental topics, including SME, the OMG Essence Framework, the concept of service-oriented architecture (SOA), and the method chunk metamodel from our previous research. In Section 3, we explain our research objectives and methodology. Section 4 describes the proposed framework. The validation of the proposed framework is conducted by applying the framework (Section 5) and building the prototype of the supporting system (Section 6). In Section 7, we describe some related works. Finally, the conclusion and our future work are described in Section 8.

## 2. Fundamentals

*A. Situational Method Engineering (SME)*

Method engineering (ME) aims to design, construct, and adapt methods. Situational method engineering (SME) applies ME in specific situations or characteristics of the software development process [5]. It consists of several activities that method engineers should perform to provide a situational method for a particular software project. This discipline has become more popular since the need for a particular software development method that suits one specific situation became apparent. For example, some developers use Scrumban (Scrum and Kanban) to suit their needs instead of only using Scrum. Furthermore, several studies have indicated that developer teams often have to modify an existing method to suit their needs. In [2], a survey of 100 organizations showed that more than 2/3 had developed or adapted the software development method they used. Furthermore, 89% of respondents stated that the method needed to be adapted for each software project.

The SME process is generally carried out by selecting method parts from a repository known as a method base. The chosen method parts are then composed to become a new method. To meet the need for a specific method, the selection of method parts must consider the characteristics of the project.

Various metamodels have been proposed for method parts: these include method fragments [5], method chunk [7], Open Process Framework (OPF) method fragment [8], and method component [9]. The method fragment defines the elementary part of a method while the method chunk combines process and product method fragments. The OPF method fragment defines its method fragments extracted from the definition of OPF. Last, the method component adopts the concept of a software component; it establishes a method part as an individual component with a defined interface.

This research chooses the method chunk metamodel since the assembly process model is used to construct a new method [10], the most similar process to the merging mechanism in the Essence Framework. The method chunk is part of a method; it consists of a process and product part. The method chunk interface defines the context use of the method chunk.

Various studies on the method construction process model have also been proposed. One of these is the assembly-based process model. A new method is constructed in the assembly-based process model by assembling selected method chunks.

Most of the proposed process models are described using map diagrams. A map is a labeled graph consisting of nodes and arcs. Each node represents an intention, while arc represents a strategy to achieve the corresponding intention. Therefore, a map diagram can be defined as a set of sections. Each section is represented by a tuple <source intention, target intention, strategy>, which describes a strategy for achieving the target intention from the source intention [11]. The proposed processes will also be described using map diagrams in this research.

*B. OMG Essence Framework*

In 2014, the Object Management Group (OMG) published a standard for modeling software development methods, known as the Essence Framework [12], and consists of Essence Kernel and Essence Language. The framework is the main product of SEMAT (Software Engineering Method and Theory), a community composed of several universities and industries. This community saw the need for a fundamental change in working with software development methods. It would be better if there were standard terminologies in the software development process.

The Essence Framework is scalable, extensible, and easy to use. It allows people to express the essentials of their existing and future methods and practices. They can essentialize the various software development methods. When described in Essence, method engineers can compare, evaluate, and tailor methods and practices. As a result, choosing a method for a particular software project can be more manageable. They can also continually assess the progress and health of the software development efforts through the kernel elements.

The kernel uses language to provide essential elements of the software development method described. It defines three areas of concern: the Customer, the Solution, and the Endeavor. The kernel defines the following three types of elements in each concern: (1) **Alpha** represents essential aspects in software development. Alpha describes things to manage, produce, and use by the development team to build, maintain, and support the software. A project manager will assess the project's progress through alpha's states. (2) **Activity space** represents every essential task. Activity spaces describe challenges the development team will face in building, maintaining, and supporting a software system and other things needed to address. (3) **Competency** represents the main capabilities required to carry out activities in a software development process.

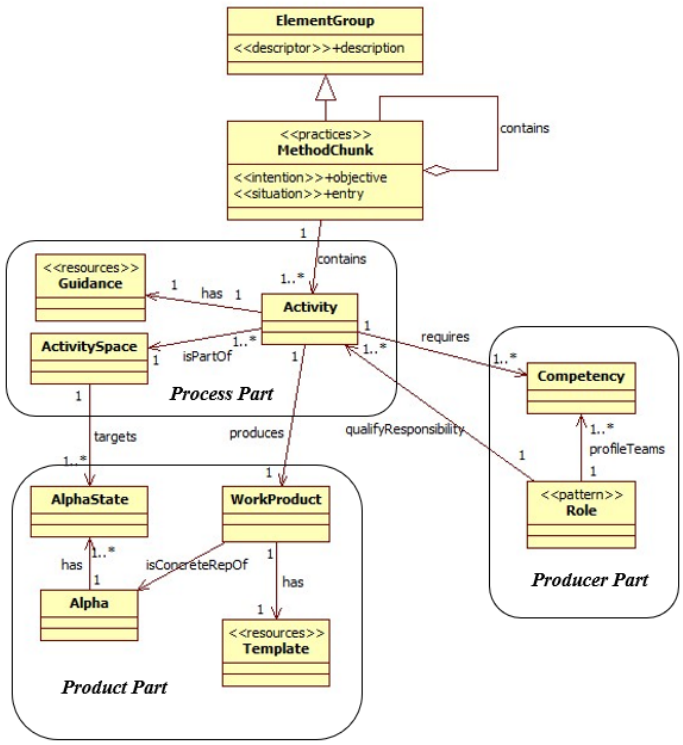


Figure 1. Revised method chunk metamodel

Our previous research proposed the method chunk metamodel in Essence [13]. Using this metamodel will improve the interoperability of the method chunks, therefore easing the construction of a method from several method chunks. The metamodel -is shown in Figure 1- describes the product part, the process part, and the producer part of a method chunk expressed in Essence Kernel elements. Alpha defines the product part of a method chunk with its states and work products in the first part. Next, activity space and activities with related guidelines represent the process part of a method chunk. Finally, Competency with its corresponding role pattern expresses a producer part of a method chunk.

*C. Service-oriented Concept for SME*

Service-oriented architecture (SOA) is part of a concept that forms a service-oriented computing (SOC) platform. SOC is a computing paradigm that utilizes service as a fundamental element of applications or solutions [14]. Service is a self-describing and platform-agnostic computational element that quickly supports distributed applications' composition at lower costs. Service can perform a specific function that can vary from a simple function to a complex one

representing one particular business process. A service-based application is built from interacting services with well-defined interfaces to meet the needs of its users. Thus, SOA is a type of architecture that defines an approach to designing a software system that provides services for other applications or other services. Usually, the services are distributed in the network and have published and discoverable interfaces.

SOA defines interactions between software agents by exchanging messages between service requesters (clients) and service providers. However, SOA is not just a service-focused architecture. SOA also defines the relationship between its three participants: service provider, service discovery agency, and service requestor (client).

In SME, we adopted the service-oriented concept of SOA to enable method engineers to provide their method chunk descriptions for access by other method engineers as a service. Method engineers from different providers can also publish their method chunks in a centralized service discovery agency to make it easier to find method chunks. Once method engineers construct a situational method, they can discover method chunk candidates from the service publisher and then access the method chunk's complete description from the provider.

## 3. Research Objective and Methodology

Our research objective is to provide a guide in applying situational method engineering (SME) by proposing a process framework for method engineers and software engineers. Following the proposed framework, method engineers can extract method chunks from existing methods, publish them to make them available as a service, and construct situational methods from selected method chunks. Software engineers can use the proposed framework to develop the supporting system.

The proposed framework uses OMG Essence Kernel as a standard method modeling notation to improve the interoperability of method chunks. The framework also applies the concept of service-oriented by providing a complete cycle of SME application in a software development project that involves three participants of SOA: the provider, the publisher, and the client. This approach allows method engineers to provide their method chunk descriptions for access by other method engineers as a service.

The cycle starts with the method chunk extraction process, stores method chunks at the repository (as a provider), publish method chunk at the publisher system. Finally, method engineers (as a client) can construct a complete method from several method chunks. Each method chunk has an attribute that describes the situational factor, a suitable situation for its application. As a result of the method construction process, the method description will also include situational factor attributes. Later, project managers can enact the resulting situational method in a software development project.

The contribution of this research is providing a framework to guide method engineers to apply SMEs in their software projects. Furthermore, software engineers can use the definition of the proposed process framework to develop the supporting system. In the end, method engineers can apply SMEs with less effort.

Several intermediate goals are defined:

1. Propose the process flow involving the three participants: the method chunks provider, the publisher, and the client. This process flow includes the method chunk repository building, the method chunk publication, and the situational method construction.
2. Propose the method chunk repository building by extracting method chunks from an existing software development method, each assigned its situational factor. The process itself will use Essence Kernel elements as a reference.
3. Propose the method chunk construction process. It begins with defining method requirements by identifying the project characteristics, mapping them to situational factors, selecting and retrieving the appropriate method chunks, and composing the situational method.

The detail of the method chunk publication process is not derived since it depends on the technology used.

Our research methodologies are as follows:

1. The process flow is designed to accommodate a complete cycle of the SME process, following the assembly process model [10]. Detailed steps in the SME process are then identified by elaborating the method engineer's tasks. Next, the concept of SOA is adopted to improve the accessibility of method chunks. Finally, the processes are grouped into the three participants of SOA: provider, publisher, and client.
2. The method chunk repository building is designed by referring to previous SME research. This research also uses the Essence Kernel as a reference to identify elements of the method chunk, using existing practices as method chunk candidates.
3. The method chunk construction process is also designed by referring to previous SME research, combined with the mechanisms proposed in OMG Essence Framework. It includes defining the method requirement by mapping the project characteristics into situational factors, searching method chunk candidates based on the situational factors, selecting the appropriate method chunks, and retrieving a complete description of all method chunks from their providers. Finally, a new method will be constructed from selected method chunks.

The proposed process framework is validated by applying the process framework to a case study and building a prototype of the supporting system. The objective is to show that the proposed framework can be applied and used to develop the supporting system.

## 4. Proposed Framework

The proposed framework consists of the process flow that includes a complete cycle of constructing a situational method and the details of subprocesses. The process flow includes the extraction of method chunks from existing methods, selecting the appropriate method chunks according to the project characteristics, and the composition of a situational method from selected method chunks. The process flow is designed by following the assembly process model; modified to accommodate the application of Essence Framework in method modeling and construction process. The framework also applies the service-oriented concept by promoting the three participants of SOA and providing method chunk description as a service.

As mentioned in Section 2.A, the details of subprocesses are described using a map diagram with their corresponding guidelines. Three guidelines for each process -adopted from [11]- are developed: (1) Intention Achievement Guideline (IAG) for each section, (2) Intention Selection Guideline (ISG) for each source intention, and (3) Strategy Selection Guideline (SSG) for each couple of source and target intention. IAG describes how to achieve an intention by providing a detailed algorithm. ISG describes the condition that matches to progress to other intentions. SSG describes which strategy should be chosen from one intention, which leads to the selection of corresponding IAG. Examples of the guidelines can be found in Appendix A. Overall, there are seven map diagrams with 21 guidelines. Method engineers may follow the guidelines to perform the process manually, while software engineers can use the guidelines as a reference when developing the supporting system.

### *A. Process flow*

The process flow is designed to accommodate a complete cycle of the SME process. The steps in the SME process are defined by elaborating the method engineer's tasks when performing the SME. A new method is constructed in the assembly-based process model by assembling selected method chunks from the repository. Method engineers must first build a repository of method chunks by extracting method chunks from existing methods. To create a new method, method engineers must define the method requirement, select appropriate method chunks, and construct a new method from them.

SOA is adopted to improve the accessibility of method chunks by promoting the three participants of SOA: provider, publisher, and client. As shown in Figure 2, the process flow involves all the processes of the three participants: (1) The method chunk provider, (2) The method chunk publisher, and (3) The method chunk client. The method chunk provider is a subsystem that maintains a method chunk repository, or a method-base, by providing complete

descriptions of the method chunks and making these available to be accessed as a service. Many providers can be in the system, allowing method engineers to share their method chunks. The method chunk publisher is a subsystem that publishes method chunks from various providers. This subsystem maintains a centralized method chunk publisher to facilitate the method chunk selection process based on defined criteria. Finally, the method chunk client subsystem provides a CAME tool to construct a situational method. Method engineers can use this tool to build a situational method.

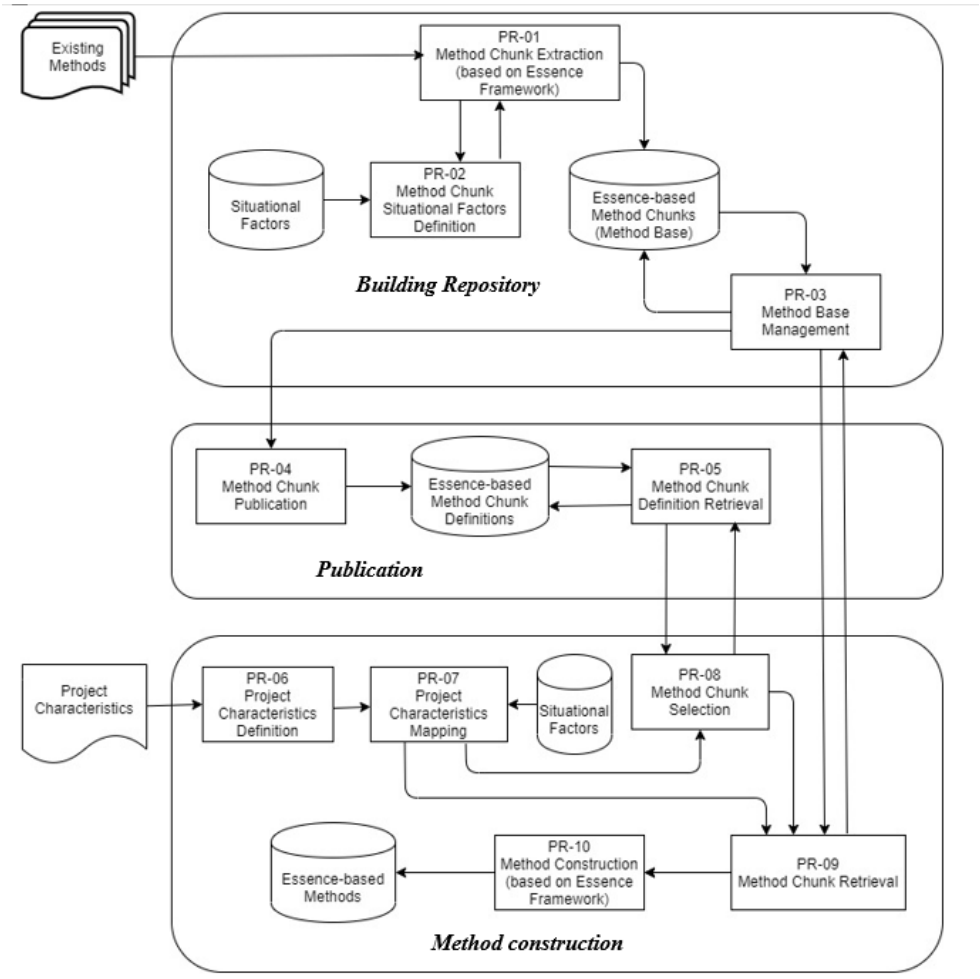


Figure 2. Process flows in applying SME

As seen in Figure 2, there are three groups of processes: (1) building the repository at the method chunk provider, (2) method chunk publication in the method chunk publisher, and (3) method construction on the client-side. The groups of the process are defined based on the role of each SOA participant. The process details are designed according to the method engineer's task description when performing SME. The method chunk repository is built by extracting method chunks from existing software development methods. The situational factor attributes for each method chunk are defined along the process. The method chunks are then stored in the method base. The method base management process provides the capability to publish the method chunk definitions at the publisher. It also provides the ability to retrieve the method chunk detail description when a client needs it.

Method chunks from providers can be published in the method chunk publisher. In addition, the publisher provides the capability to find method chunks that match specific criteria. The selection criteria come from the client due to project characteristics mapping into situational factors. Next, the client will use the selected method chunk definition to retrieve the complete description of the method chunk from the corresponding provider. Finally, the client can perform the construction process to build the situational method from retrieved method chunks.

The following sections describe the detailed design of several processes from the process flow. Section B describes the building method chunk repository, includes the method chunk extraction process from existing methods (PR-01), and defines situational factor attributes for each method chunk (PR-02). Next, Section C describes method requirement definition consisting of two processes: project characteristics description (PR-06) and project characteristics mapping into corresponding situational factors (PR-07). Finally, Section D describes the method construction process (PR-10). As mentioned earlier, this research will not derive other processes because of their simplicity or dependency on the technology used.

*B. Building Method Chunk Repository*

Before applying SME, the method chunk repository should be built first. Method engineers can define method chunks or extract them from any existing method. Next, digital representation of method chunk description must be stored at the repository and published at the publisher.

Building method chunk repository consists of two processes: (1) method chunk extraction (PR-01) and (2) method chunk situational factor definition (PR-02). The extraction process is designed based on the method reengineering approach [7] and the method chunk construction process [15]. The objective of the process is to provide method chunks by extracting them from an existing method. Later, method chunks will be stored in the repository and published as services. In addition, OMG Essence Framework will be consulted as a reference to produce Essence-based method chunks.

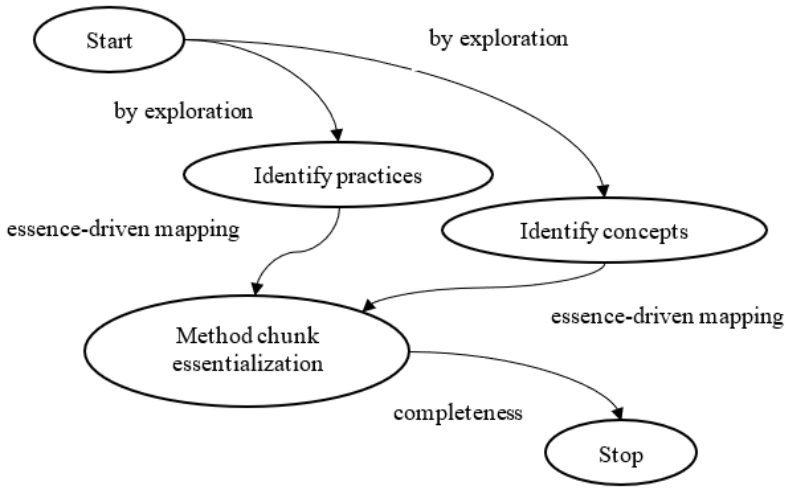


Figure 3. Method chunk extraction process

As shown in Figure 3, the extraction process identifies practices and concepts from the existing method. First, each practice will be promoted as a method chunk candidate. Next, all method chunk candidates and concepts not included in all candidates are essentialized to produce Essence-based method chunks. After that, situational factor attributes are added to each method chunk. This process is conducted using the essence-driven mapping strategy. Using practices as method chunk candidates complies with the Essence Framework method architecture that defines a method as a composition of practices.

Another map is derived to describe the method chunk essentialization intention. As shown in Figure 4, two different strategies to identify a method chunk can be selected: (1) 'process (activity-space) driven decomposition' and (2) 'product (Alpha) driven decomposition'. The process can begin with identifying the process part or the product part of the method. The first strategy must be followed by the 'by completing product part' strategy to define a complete method chunk. Likewise, the second strategy must be followed by the 'by completing process part' strategy. After that, the producer part and situational factor attributes must be added to method chunk candidates. Finally, the process ends at the stop intention by performing 'the check for completeness' strategy. Three guidelines for each process are developed: the IAGs for each intention, the ISGs for each path from one intention to the next intention (called a section), and the SSGs for each strategy. Overall, there are six guidelines for both processes.

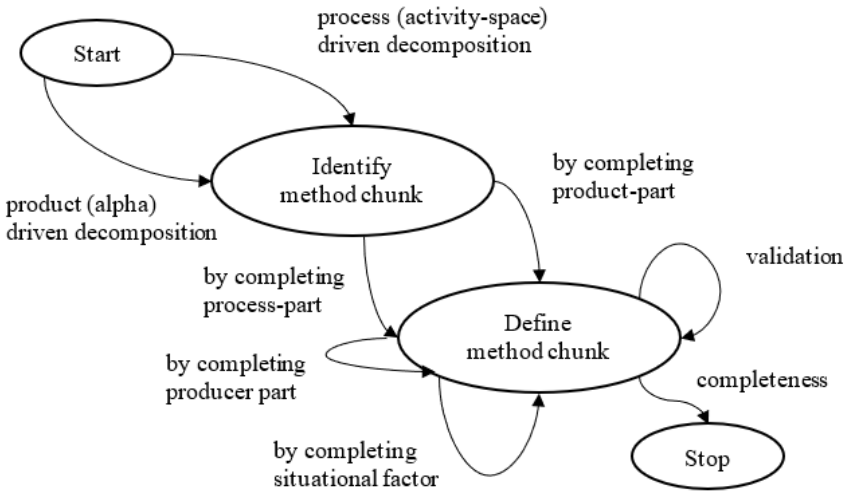


Figure 4. Method chunk essentialization process

*C. Method Requirement Definition*

The first step in applying SME to a particular software project is to define the method requirement. Method engineers can determine the requirement based on the project characteristics. Method requirement definition consists of two processes: (1) project characteristics definition (PR-06) and (2) project characteristics mapping into situational factors (PR-07). When applying SME, the project manager must first define the project characteristics to determine a method requirement. The method requirement is defined as a set of situational factors. Therefore, the project characteristics must be mapped into their corresponding situational factors.

Mapping the project characteristics into situational factors is conducted by converting project characteristics into method chunk 'reuse situation' attributes. These attributes define criteria that describe an appropriate situation for using the method chunk. As seen in Figure 5, the process consists of two intentions: (1) Define the characteristics of the project by selecting the classifications, the characteristics, the sub characteristics (optional), and their values, and (2) Map each characteristic or sub-characteristic into reuse situation attributes. For this map diagram, three guidelines have been developed: the IAGs for each intention, the ISGs for each called a section and the SSGs for each strategy.

A systematic literature review on project characteristics that affect the choice of the software development method has been performed. This study aims to develop a guide for project characteristics definition. Following a systematic selection, we have developed a reference table for defining the project characteristics, shown in Appendix B, collected from [16], [17], [18], [19], [20], [21], [22], [23], and [24].

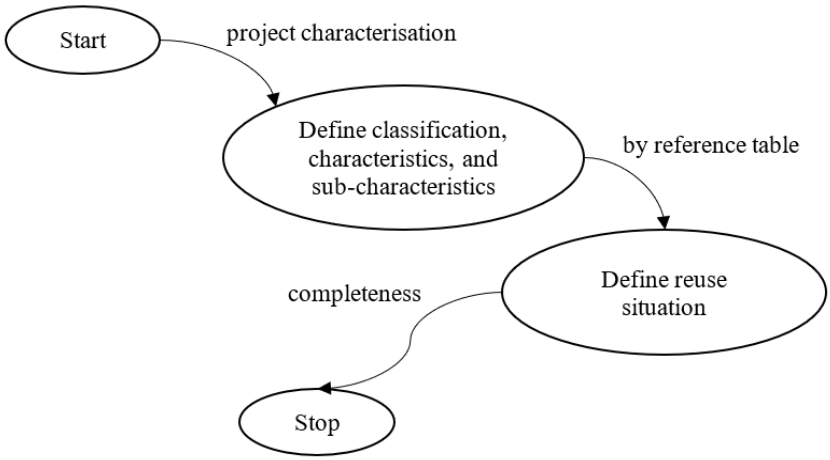


Figure 5. Mapping process of project characteristics into situational factors

*D. Method construction process*

The method construction process (PR-10) is adopted from the assembly-based process model [10]. This process model is used as it is most similar to the merging mechanism in the Essence Framework. This process aims to compose several method chunks into a situational method. There are two strategies involved in assembling method chunks in the original process model. The first one is the association strategy, which merges two method chunks with different elements. The result is a new method chunk with more elements since the process combines all the aspects of the two method chunks. The second strategy is the integration strategy, which merges two method chunks with common elements but different attributes. This strategy merges different elements of the two method chunks as the association strategy and then combines all attributes of the common element, thus enriching the common elements.

Table 1. Three types of method chunk merging

| # | Cases | Type of merging |
|---|---|---|
| 1 | Method chunks A and B have different kernel elements. | Association by merging |
| 2 | Method chunks A and B have common kernel elements but different attributes or common attributes with different values. Method engineers must choose one of the conflict attribute values. | Integration by merging |
| 3 | Method chunks A and B have common attributes with different meanings and values; one of the conflict attributes must first be renamed (extend). | Integration by merging with extension |

This research applies the merging mechanism of the Essence Framework [12] to refine these two original strategies allowing to resolve conflict in common attributes. In Essence Framework, two different method elements can be merged to form a more significant method element. When a conflict in method element attributes arises, an extension mechanism is used to modify the method element to match the new context. Therefore, instead of two strategies, this research proposes three strategies to assemble method chunks: (1)' association by merging', (2) 'integration by merging', and (3) 'integration by merging with extension'. The original integration strategy is split into two strategies that can be selected to handle conflict attributes of the combined method elements. In the first case, the method engineer must choose one of two

conflict attribute values. In the other case, one of the two conflict attributes can be renamed using the extension mechanism of Essence. These three strategies can be seen in Table 1.

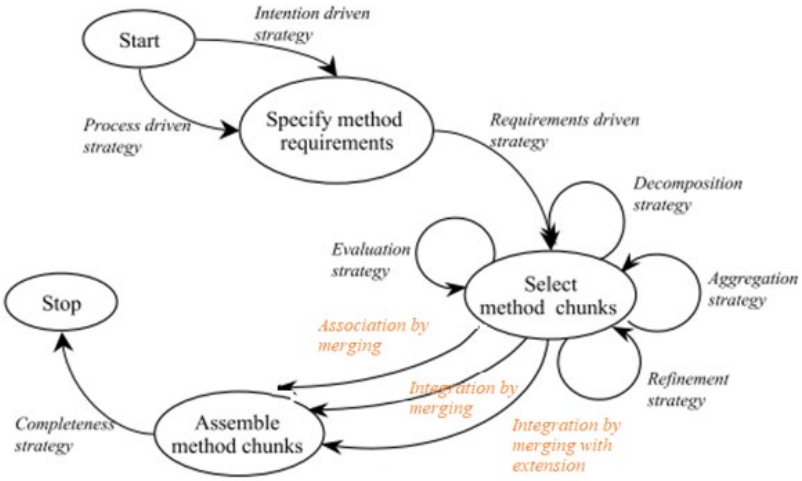


Figure 6. Assembly-based process in Essence (adopted from [10])

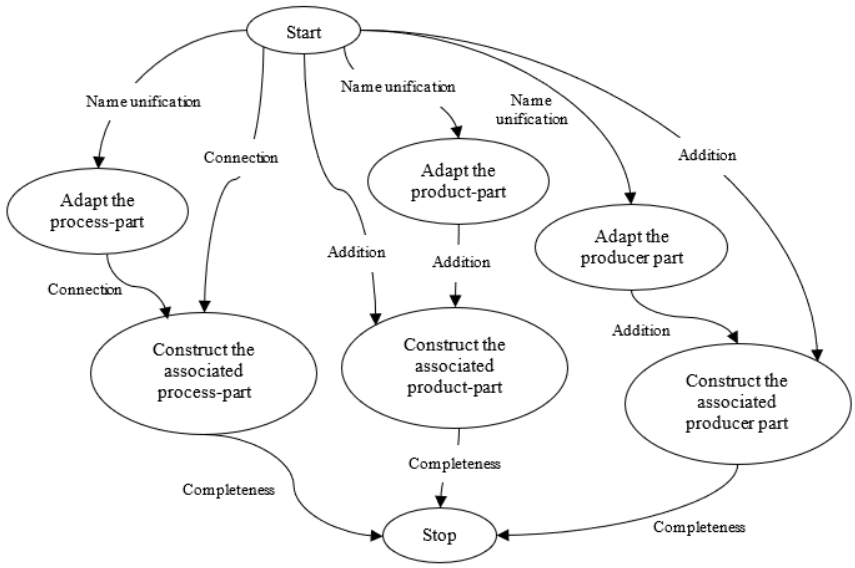


Figure 7. Association by merging (adopted from [25])

The proposed method construction process model can be seen in Figure 6. This process model is a revised version of the assembly-based process model. The process begins with specifying the method requirement, followed by selecting appropriate method chunks. The last intention is to assemble method chunks by applying three strategies as described in Table 1. The detailed process model for each strategy can be seen in Figures 7, 8, and 9, adopted from [25]. For each map diagram, three types of guidelines have also been developed: the IAGs for each intention, the ISGs for each section, and the SSGs for each strategy. Overall, there are 12 guidelines for four map diagrams.

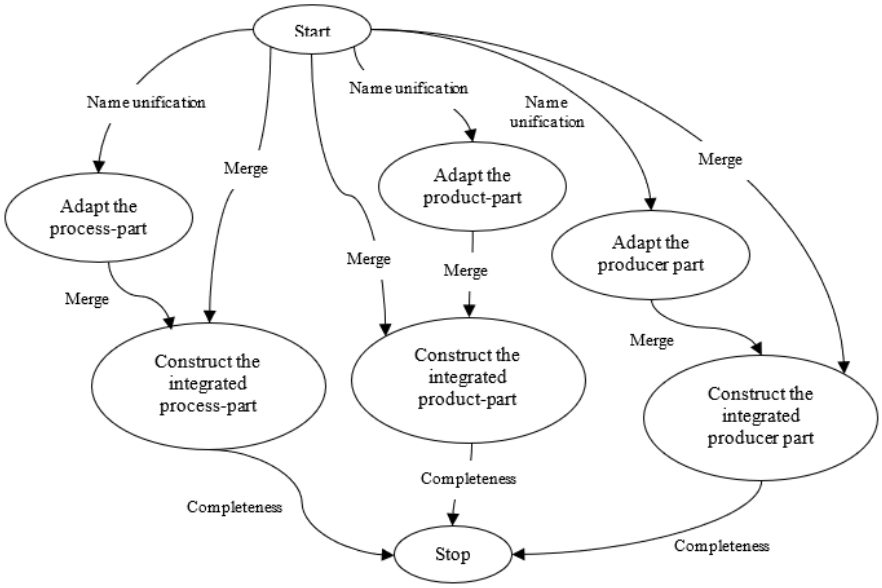


Figure 8. Integration by merging (adopted from [25])

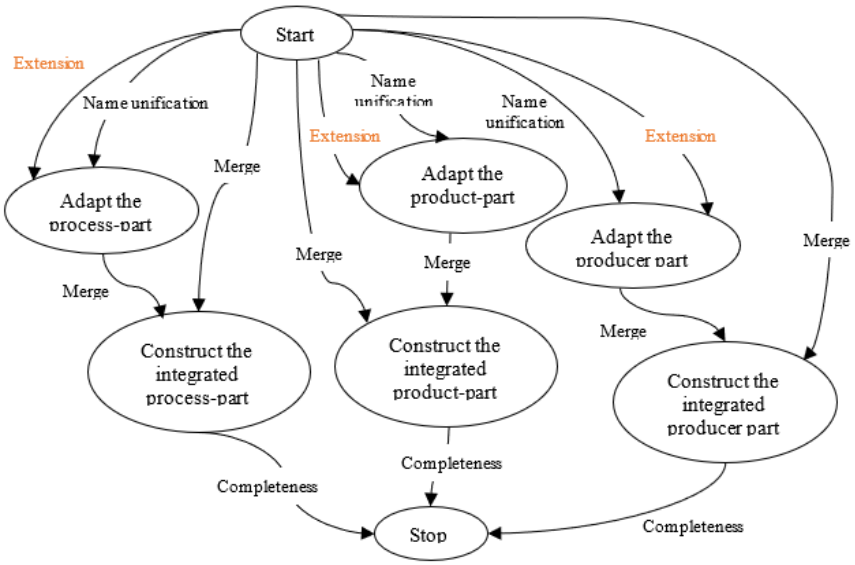


Figure 9. Integration by merging with an extension (adopted from [25])

Using the 'association by merging' strategy (Figure 7), the process elements of the two method chunks will be connected as a sequence of activities. The product and producer elements of the two method chunks will be combined to form a more comprehensive product part and producer part of the method chunk. The 'integration by merging' strategy (Figure 8) will merge the process, product, and producer elements of the two method chunks. The merging process begins with a similar process as the first strategy. But when a conflict occurs in the merging process, the method engineer must decide which element will be chosen. The 'integration by merging with extension' strategy (Figure 9) will merge the two method chunks' process, product, and producer elements. It is a similar process as the second strategy. But when a conflict occurs,

the method engineer will perform the extension mechanism by renaming one of the conflict attributes.

## 5. Application of Framework

This section describes the first part of the validation process. The main processes of the proposed framework have been manually executed to show the applicability of the proposed framework. It includes extracting method chunks from an existing method, mapping the project characteristics into situational factors, and composing a new method from several method chunks.

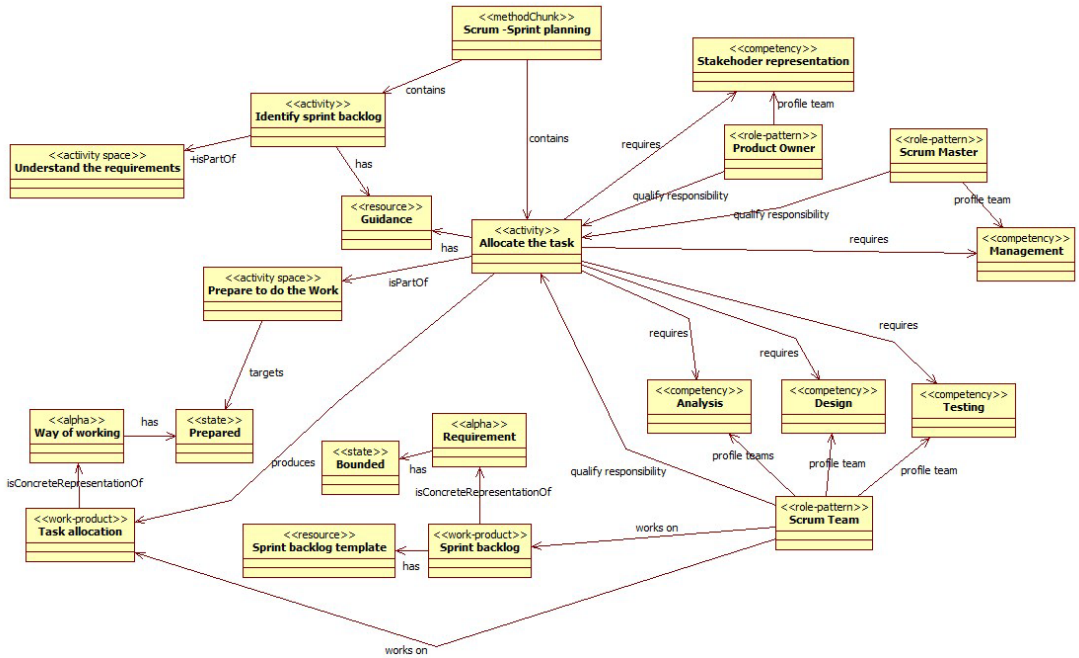


Figure 10. Example of method chunk in Essence

Taken from Scrum [26], five method chunks have been successfully extracted by using Scrum ceremonies as method chunk candidates. These are Sprint Planning, Sprint, Daily Scrum, Sprint Review, and Sprint Retrospective. Following the proposed method chunk extraction process, the method chunk elements are identified. For example, the Sprint Planning method chunk consists of the product, process, and producer parts. As seen in Figure 10, the product part consists of two alphas: (1) Way of Working with the corresponding alpha state (Prepared) and work product (Task Allocation), and (2) Requirement with the corresponding alpha state (Bounded) and work product (Sprint Backlog). Next, the process part consists of two activity spaces: (1) Prepare to Do the Work with the corresponding activity (Allocate the Task), and (2) Understand the Requirement with the corresponding activity (Identify Sprint Backlog). Finally, the producer part consists of three role patterns: (1) Product Owner, (2) Scrum Master, and (3) Scrum Team.

Extracted method chunks have aligned with our metamodel (Fig. 1 at Section 2.B). Each has the process part, product part, and producer part. The process part consists of activity space and activity, while the product part consists of alpha, sub-alpha, and work product. Last, the producer-part consists of competencies and roles. As for their granularity, they can still be split into several atomic method chunks. The coarser granularity of the extracted method chunks came from our approach, using practices of the existing method as method chunk candidates.

Table 2. Example of project characteristics

| Category - Characteristics | Sub-characteristics | Value |
|---|---|---|
| People - Personel | Experience | High |
| People - Team | Size | Small |
| Product - Requirement | Changeability | High |

Table 3. Corresponding reuse situation attribute value

| # | Reuse Situation |
|---|---|
| 1 | People-Personel->Experience->High |
| 2 | People-Team->Size->Small |
| 3 | Product-Requirement->Changeability->High |

Table 4. New Scrum method in Essence standard

| | |
|---|---|
| **Method chunk name:** | Scrum |
| **Origin:** | Scrum |
| **Interface** | <Problem statement, Develop the software system following Scrum Method> |
| **Reuse Situation** | Product – Strategy -> Development -> Iterative<br>Product – Strategy -> Deployment -> Incremental<br>People – Personnel -> Expertise -> High<br>People – Personnel -> Commitment -> High<br>People – Stakeholder -> Involvement -> Real<br>People – Stakeholder -> Participation -> High<br>People – Management -> Commitment -> High |
| **Reuse Intention** | Develop a software system following Scrum Method |
| **Process part** | {Activity space - Activity}<br>1. Explore possibilities – Identify the problem<br>2. Ensure stakeholder satisfaction – Inspect the increment<br>3. Support the team – Improve the process<br>Parallel<br>1. Understand the requirements – Identify sprint backlog<br>2. Prepare to do the work – Allocate the task<br>3. Shape the system – Design the increment<br>4. Implement the system – Build the increment<br>5. Test the system – Test the increment<br>6. Deploy the system – Deploy the increment<br>Parallel<br>1. Coordinate activity – Coordinate the team<br>2. Track progress – Inspect progress toward the sprint goal |
| **Product part** | {Alpha/Sub Alpha – Work product}<br>Requirement - Sprint backlog<br>Way of working - Task allocation<br>Way of working - Kanban board<br>Way of working - Burndown chart<br>Way of working - Process definition<br>Software system - Increment<br>Software system - Software release |
| **Producer part** | {Competency – Role}<br>Stakeholder representation – Product owner<br>Analysis – Scrum Team<br>Design – Scrum Team<br>Testing – Scrum Team<br>Management – Scrum Master |

The mapping process of project characteristics into situational factors has also been conducted. For example, project characteristics in Table 2 are mapped into method chunk' reuse situation' attributes in Table 3. The project characteristics are identified from a reference table. The mapping process is simple; it is performed to generate values of reuse situation attribute of a method chunk.

Finally, the third process has been executed to construct a new Scrum method from several method chunks. The new Scrum method is built from five method chunks extracted earlier. Therefore, the five method chunks from the Scrum method are reconstructed into a new Scrum method. Method chunks are merged one by one using three strategies: (1) association by merging, (2) integration by merging, and (3) integration by merging with extension. The new Scrum method, in Essence, can be seen in Table 4; it consists of 11 activity spaces with 11 related activities, three alpha with seven corresponding work products, and five competencies with three corresponding roles.

The quality of the resulting method can be measured by inspecting the relationships between method elements and duplicate elements. The three strategies of the method construction process -association by merging, integration by merging, integration by merging with extension- assure the existence of the relationship between activities from different method chunks. The method engineers must determine the sequence of activities. Duplicate method elements can be avoided since the above strategies will detect and resolve a conflict.

For other cases, method engineers may consult the proposed framework and the corresponding guidelines to apply SMEs in their software projects. But, it does require a comprehensive knowledge of existing methods they will extract.

## 6. Prototype of Supporting System

The second validation process of the proposed framework is the development of the SME supporting system prototype, called ESME Environment. The process has been conducted in several final projects of undergraduate students under author supervision [27][28][29][30], which will be summarized below. This work shows that the supporting system for the proposed framework can be built using current technologies.

The work begins with designing a support system architecture that applies a service-oriented concept [27]. The architecture is designed by following the process flow of the proposed framework, which consists of three components: (1) the provider component, which provides method chunks complete description, (2) the publisher component, which publishes method chunks from providers, and (3) the client component which provides the capability to construct a new method.

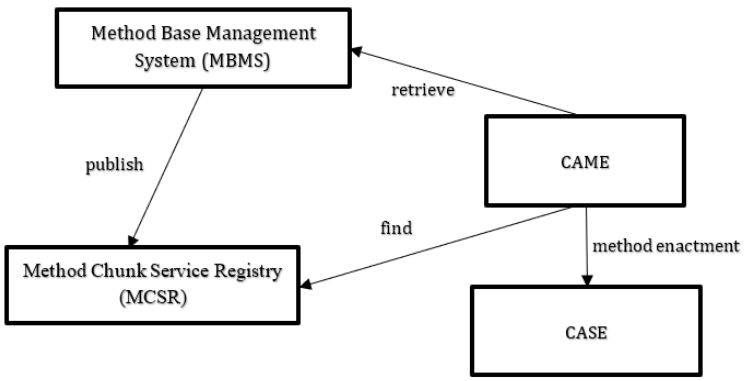


Figure 11. The proposed support system architecture [19]

The proposed architecture can be seen in Figure 11. The provider component is the MBMS (Method Base Management System). The publisher component is the MCRS (Method Chunk Registry System), and the client component is CAME (Computer-Aided Method Engineering)

and CASE (Computer-Aided Software Engineering) tools. Method engineers use the CAME tool to define the project characteristics, find a list of method chunk candidates that match the project characteristics from the MCRS, and retrieve the selected method chunks from the corresponding MBMS. Finally, the CAME tool will help them to compose a new method from selected method chunks. Later, the project manager can enact the method in a particular software development project.

A prototype for each component has been developed based on the proposed architecture. First, the prototype of MBMS that provides services to access the method chunks in its method base [28]. The method chunks can be retrieved through defined Application Programming Interfaces (APIs). The method chunk complete descriptions are stored in MongoDB in the form of JSON documents. Currently, the method base contains 19 method chunk descriptions and can be accessed at:

https://github.com/gejimayu/mbms/tree/master/method%20chunks

The API consists of a list of endpoints, headers in the HTTP protocol, query parameters, delivery payloads, and response payloads. Since the method chunk description is stored in JSON format, the delivery payload and response payload are also JSON documents.

The prototype of the second component, the MCRS, has also been developed [29]. This component provides services for publishing method chunks from various providers (MBMS) and performs the method chunks searching process through the defined APIs. The MCRS provides 27 API endpoints. The searching process for method chunks will generate a list of method chunk candidates that match specific criteria. This research uses the multicriteria decision-making (MCDM) algorithm [31] to find the matched method chunk candidates.

The last component is the prototype of a CAME tool [30]. This tool incorporates a method composition module that composes several method chunks into a complete situational method semi-automatically [32]. Method engineers can use the CAME tool to construct methods from several method chunks. First, this component will interact with the MCRS to prepare a list of method chunk candidates, which are selected based on the project characteristics defined by the method engineer. Once the method engineer has chosen the method chunks, the tool will retrieve the complete method chunks description from the MBMS. This approach allows the method engineer to find the method chunk without knowing the method chunk provider. Finally, the tool will compose the selected method chunks to form a complete situational method.

Table 5. Comparison of CAME Tools

| No | Environment | Method Representation | Architecture |
|---|---|---|---|
| 1 | MENTOR | NATURE approach | Monolithic |
| 2 | MetaEdit+ | Graph, Object, Port, Property, Relationship, and Role (GOPPRR) | Service-oriented |
| 3 | Decamerone | Method Engineering Language (MEL) | Monolithic |
| 4 | MERU | Method Requirement Specification Language (MRSL) | |
| 5 | Method Editor | Method Engineering Language (MEL) | Monolithic |
| 6 | **ESME Environment** | Essence Language | Service-oriented |

The ESME Environment has been compared to other CAME tools that apply the assembly process model. Those are Mentor, MetaEdit+, Decamerone, MERU, dan Method Editor [33]. The indicators are interoperability and method chunk ease of access. Interoperability is defined as a quality that allows method chunks from different providers can be composed into a complete method. It required standard notation of method chunk description. Ease of access can be

achieved when method chunk descriptions are available as services. Method engineers can share their method chunks and publish them through a centralized publisher.

Table 5 shows the comparison between existing CAME tools and the ESME Environment. It shows that ESME Environment has the following advantages:

1. Using Essence Language as a notation of method chunk and method description will improve the interoperability since OMG has published it as a standard notation for method modeling. Other CAME tools use various notations, a generic notation such as the NATURE approach and GOPPRR, or their languages such as MEL and MRSL.
2. Appling SOA will enhance access to available method chunks published by many providers. Many existing tools apply monolithic architecture that will decrease the capability to share method chunks.

## 7. Similar Work

Some related works have been conducted to promote the application of SMEs in a software development project. The idea of providing method parts as a service was proposed in [34][35][36] and is called method service. Furthermore, a framework called Method Engineering with Method Services and Method Pattern (MESP) was proposed in [37]. This framework is a complete solution for managing and using project-specific software development methods in a software development project. It defines three groups of tasks: (1) method content definition, (2) method tailoring, and (3) method enactment. This work transforms a method service to a BPEL process model and deploys it into a BPEL engine.

An effort to build a method part repository has also been conducted. In [38], a repository of agile method fragments has been introduced to organize the evidential knowledge of agile method fragments according to their objectives and requisites. The authors gathered the knowledge through a systematic review of empirical studies investigating the enactment of agile methods in various project situations. In [8], a repository of method components has been developed to support the development of situation-specific methods.

Ivar Jacobson Institute has developed a library of practices described in Essence Kernel. Currently, this library contains 28 practices described in Essence Kernel elements using text and pictures. Anyone can browse the practices through the library interface at https://practicelibrary.ivarjacobson.com/.

Compared to similar work such as MESP, our proposed framework promote several advantages. First, the proposed framework applies the service-oriented concept, which facilitates the method engineers in sharing their method chunks. Each provider can publish their method chunk through a centralized publisher to minimize the effort of finding method chunks. In our proposed solution, the service provided by the method chunk providers is to retrieve method chunk descriptions. It does not include an automatic execution of the method chunk as proposed in MESP. This approach is chosen to cover all types of method chunks, which mostly cannot be automatically executed. Thus, the software development team can manually perform most of the method chunks.

Next, all method chunk descriptions are defined using Essence Kernel elements instead of another standard such as Software & Systems Process Engineering Meta-Model (SPEM) [39]. Essence is chosen since it promises better support for method enactment than SPEM [40]. Finally, the method chunk description is stored in digital format as JSON documents in the method base. This approach has not been applied in similar works described above; it allows us to do a semi-automatic composition process as suggested in [30] and [32]. As a result of the construction process, the new method is also in a JSON document. Therefore, it allows us to use the method descriptions as an input for a CASE tool. See our future work for further explanation.

## 8. Conclusions and Future Work

This research proposes a process framework that combines the advantages offered by SME and the Essence Framework as a standard for method definition. The proposed process can be used as a reference for building a service-oriented SME support system. When applying SMEs,

method engineers have an essential role in software development projects. They have to define the method requirements that suit the characteristics of the project, find appropriate method chunks, and construct a new method. The supporting system can facilitate and minimize the effort of method engineers by providing various method chunks that are easy to access.

In this research, method chunk is defined using uniform notation and terminology, making it easy to understand and construct a new situational method. Furthermore, the Essence Kernel element is an interesting choice to represent the method chunk since it provides predefined elements of a comprehensive method.

Several validations for the proposed framework have been carried out to show that the combination of SME with the OMG Essence Framework is applicable and provides advantages. We have executed the processes of the proposed framework. Some are still conducted manually, especially the building process of the method chunk repository. Others can be performed using the supporting system prototype. However, further research can be conducted to transform manual processes into automatic or semi-automatic operations. In the end, method engineers can apply the SME to any software development project with less effort since they can easily find the method chunks they need and use the supporting system instead of performing the process manually.

As for future work, complete supporting systems development is still in progress. We still develop a graphic method editor called EssenceBoard, allowing method engineers to essentialize method chunks. This editor is part of the MBMS. We also built a CASE tool called EssenceProject to support method enactment in a particular software development project. This tool will use a JSON document that contains a method description in Essence. EssenceProject is a generic software management project tool that will behave according to the input method.

## 9. Acknowledgment

This research was funded by the BUDI-DN Program and provided by the Ministry of Finance, the Republic of Indonesia.

## 10. Appendix

*A. Examples of map diagram guidelines*

Examples of guidelines for the map diagram in Figure 4 (Method chunk essentialization process):

1. Intention Achievement Guideline (IAG) for each section; using an algorithm notation to describe the process in achieving an intention.

Table 6. Example of Intention Achievement Guideline (IAG)

| # | Description | |
|---|---|---|
| 1 | ID | IAG2-1 |
| | Section | <Start, Identify method chunk, process (activity-space) driven decomposition> |
| | Guideline | For each activity space in the Essence Kernel, identify relevant activities from the source practices description to form the process-part of the target method chunk<br>Algoritm:<br>repeat<br>for each activity_space in essence_kernel<br>create (chunk)<br>if is_relevant (activity_space, practice_desc)<br>then<br>insert_kernel (chunk, activity_space)<br>activities ← identify_activities<br>(activity_space,practice_desc)<br>for each activity in activities<br>insert (chunk, activity, activity_space)<br>insert (chunks, chunk)<br>until there is no more practice desc |

2. Intention Selection Guideline (ISG) for each source intention; describing when a target intention can be chosen.

Table 7. Example of Intention Selection Guideline (ISG)

| # | Description | |
|---|---|---|
| 1 | ID | ISG2-1 |
| | Intention | $I_0$: Start |
| | Guideline | Progress to $I_1$: Identify method chunk; when at least one activity space or one Alpha of the Essence Kernel has corresponding concepts in the source practice description<br>{I.S. Several practice descriptions (practice_desc) have been identified}<br>{F.S. Process part or product part of each method chunk candidate has been determined} |
| 2 | ID | ISG2-2 |
| | Intention | $I_1$: Identify method chunk |
| | Guideline | Progress to $I_2$: Define method chunk; when the process part (consists of activity space and activities) has been added to the product part (consists of alpha and work products) of the method chunk candidates or vice versa<br>{I.S. The process part or product part of each method chunk candidate has been identified}<br>{F.S. The product part or process part has been added to the process part or product part of method chunk candidate} |

3. Strategy Selection Guideline (SSG) for each couple of source and target intention; describing when a strategy can be chosen.

Table 8. Example of Strategy Selection Guideline (SSG)

| # | Description | |
|---|---|---|
| 1 | ID | SSG2-1 |
| | <Source Intention, Target Intention> | <Start, Identify method chunk> |
| | Guideline | Select IAG2-1 when at least one activity space of the Essence Kernel has a corresponding concept in the source practice description<br><br>Select IAG2-2 when at least one Alpha of the Essence Kernel has a correspondings concept in the source practice description |
| 2 | ID | SSG2-2 |
| | <Source Intention, Target Intention> | <Identify method chunk, Define method chunk> |
| | Guideline | Select IAG2-3 when we need to add the product part to the method chunk candidate<br><br>Select IAG2-4 when we need to add the process part to the method chunk candidate |

*B. Reference Table for Defining the Project Characteritics*

Table 9. Reference table to defining project characteristics

| # | Category | Characteristics | Sub-characteristics |
|---|---|---|---|
| 1 | People | Personel/developer | Experience |
| 2 | | | Competence/expertise |
| 3 | | Team | Size |
| 4 | | | Teamwork |
| 5 | | | Conflict |
| 6 | | Stakeholder | Commitment |
| 7 | | | Participation |
| 8 | | Management | Commitment |
| 9 | Product | Requirements | Understanding, clearness |
| 10 | | | Problem domain |
| 11 | | | Complexity |
| 12 | | Application | Size |
| 13 | | | Risk |
| 14 | | Technology | Knowledge |
| 15 | | | Tool supports |
| 16 | Project | Type | |
| 17 | | Duration | |
| 18 | | Budget, Cost | |
| 19 | | Criticality | |
| 20 | Other | Company | Size |
| 21 | | | Number of employees |
| 22 | | | Stability |
| 23 | | Communication | User-Developer understanding |

## 11. References


[1]. G. Engels and S. Sauer, "A meta-method for defining software engineering methods," in Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5765 LNCS, pp. 411–440, 2010

[2]. N.L. Russo, J.L. Wynekoop, and D.B. Walz, "The use and adaptation of system development methodologies" in *Proceedings of the International Resources Management. Association Conference*, Atlanta, 1995

[3]. H. Kniberg, "Scrum and XP from the Trenches - How We Do Scrum", C4Media the Publisher of InfoQ, 2015

[4]. N. Nikitina and M. Kajko-mattsson, "From Scrum to Scrumban : a case study of a process transition From Scrum to Scrumban : A Case Study of a Process Transition", 2015

[5]. F. Harmsen, S. Brinkkemper, and H. Oei, "Situational Method Engineering for Information System Project Approaches", *Information Systems* (Vol. 55), 1994

[6]. J. Ralyté, "Situational method engineering in practice: A case study in a small enterprise," CEUR Workshop Proc., vol. 998, pp. 17–24, 2013

[7]. J. Ralyté, J. Rolland, C., "An Approach for Method Reengineering" in 20th *International Conference on Conceptual Modeling ER*, Yokohama Japan, November 2001

[8]. D.G. Firesmith, B. Henderson-Sellers, "The OPEN Process Framework. An Introduction", Addison-Wesley, London, UK, 330pp, 2002
[9]. K. Wistrand and F. Karlsson, "Method Components – Rationale Revealed", *Advanced Information Systems Engineering 16th International Conference, CAiSE*, 3084, 189–201, 2004
[10]. J. Ralyté, R. Deneckère, and C. Rolland, "Towards a Generic Model for Situational Method Engineering Generic Process Model for Situational Method Engineering", *Advanced Information Systems Engineering*, 95–110, 2003
[11]. C. Rolland, N. Prakash, and A. Benjamen, "A Multi-Model View of Process Modelling", *Requirements Engineering Journal, Springer*, 1999
[12]. I. Jacobson, P. Ng, P. McMahon, I. Spence, and S. Lidman, "Kernel and Language for Software Engineering Methods (Essence)", OMG, (December), 2014
[13]. Y. Widyani, B. Hendradjaya, E.K. Budiardjo, B. Sitohang, "Essence-based Method Chunk Metamodel", Proceedings of the *International Conference on Electrical Engineering and Informatics*, 2019, 2019-July, pp. 152–157, 8988825, 2019
[14]. M.P. Papazoglou, "Service-Oriented Computing: Concepts, Characteristics, and Directions", in the *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE 2003)*, 2003
[15]. J. Ralyte', "Towards situational methods for information systems development: engineering reusable method chunks", In: Vasilecas O, Caplinskas A, Wojtkowski W, Wojtkowski WG, References 297 Zupancic J, Wrycza S (eds) Proceedings of the 13th international conference on information systems development. Advances in theory, practice, and education. Vilnius Gediminas Technical University, Vilnius, Lithuania, pp 271–282, 2004
[16]. I. Sharon, J. Barjis, and J. Vrancken, "A Decision Framework for Selecting A Suitable Software Development Process", In *Proceedings of the 12th International Conference on Enterprise Information Systems* 3 ISAS, 2010
[17]. E. Kornyshova, R. Deneckere, and B. Claudepierre, "Contextualization of method components", 2010 *Fourth International Conference on Research Challenges in Information Science (RCIS)*, 2010
[18]. P. Clarke and R.V. O'Connor, "The situational factors that affect the software development process: Towards a comprehensive reference framework", *Information and Software Technology*, 54(5), 2012
[19]. J.A. Hurtado, M.C. Bastarrica, S.F. Ochoa, and J. Simmonds, "MDE software process lines in small companies", *Journal of Systems and Software*, 86(5), 2013
[20]. K.M.B. Silva, and S. Santos, "Critical Factors in Agile Software Projects According to People, Process, and Technology Perspective", *In Proceedings - 6th Brazilian Workshop on Agile Methods*, WBMA 2015, 2015
[21]. L.R. Vijayasarathy, and C.W. Butler, "Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?", *IEEE Software*, 33(5), 2016
[22]. D. Carrizo, O. Dieste, and N. Juristo, "Contextual attributes impacting the effectiveness of requirements elicitation Techniques: Mapping theoretical and empirical research". *Information and Software Technology*, 92, 2017
[23]. M. Vhutshilo, "Factors Influencing the Design of Unbounded Rule-Based Expert Architecture for Selection of Software Development Methodologies", 2018 Open Innovations Conference (OI), 2019
[24]. B. Guérineau, L. Rivest, M. Bricogne, A. Durupt, and B. Eynard, "Towards A Design-Method Selection Framework for Multidisciplinary Product Development", 2018
[25]. J. Ralyte' and C. Rolland, "An assembly process model for method engineering", In: Dittrich KR, Geppert A, Norrie MC (eds) Advanced information systems engineering. Lecture notes in computer science, vol 2068. Springer, Berlin, pp 267–283, 2001
[26]. K. Schwaber, "The Scrum Guide TM", (November), 2017

[27]. Y. Widyani, B. Hendradjaya, E.K. Budiardjo, B. Sitohang, "Service-oriented Situational Method Engineering (SOSME) Model and Architecture", *Proceedings of the International Conference on Electrical Engineering and Informatics*, 2019, 2019-July, pp. 256–260, 8988863
[28]. G. Hwandiano and Y. Widyani, "Method Base Management System Berbasis Service dalam Standar Essence", Tugas Akhir Program Studi Sarjana Teknik Informatika ITB, 2019
[29]. A. Nyonata and Y. Widyani, "Pembangunan Method Chunk Registry System untuk Sistem Pendukung Service-oriented Situational Method Engineering", Tugas Akhir Program Studi Sarjana Teknik Informatika ITB, 2019
[30]. R.N. Hafizha and Y. Widyani, "Pembangunan Kakas CAME as a Service sebagai Sistem Pendukung Service-oriented Situational Method Engineering", Tugas Akhir Program Studi Sarjana Teknik Informatika ITB, 2020
[31]. E. Kornyshova, R. Deneckère, and C. Salinesi, "Method chunks selection by multicriteria techniques: An extension of the assembly-based approach: *IFIP International Federation for Information Processing*, 244, 64–78. https://doi.org/10.1007/978-0-387-73947-2_7, 2007
[32]. C.C. Andreas and Y. Widyani, "Essence-based Method Chunk Composition", Proceedings of 2019 International Conference on Data and Software Engineering, ICoDSE 2019, 2019
[33]. A. Niknafs and R. Ramsin, "Computer-aided method engineering: an analysis of existing environments," *Proc. Conf. Adv. Inf. Syst. Eng.*, pp. 525–540, 2008.
[34]. G. Guze'lian and C. Cauvet, "SO2M: towards a service-oriented approach for method engineering", In: Proceedings of IKE '07, Las Vegas, NV, USA, 2007
[35]. R. Deneckère, A. Iacovelli, E. Kornyshova and C. Souveyet, "From method fragments to method services", CEUR Workshop Proceedings, 337, 80–96., 2008
[36]. A. Iacovelli, C. Souveyet, and C. Rolland, "Method as a Service (MaaS)," Proc. 2nd Int. Conf. Res. Challenges Inf. Sci. RCIS 2008, pp. 371–379, 2008
[37]. M. Fazal-baqaie, "Project-Specific Software Engineering Methods: Composition, Enactment, and Quality Assurance", Doctoral Dissertation at Faculty of Computer Science, Electrical Engineering, and Mathematics Paderborn University, 2016
[38]. H.C. Esfahani and E. Yu, "A repository of agile method fragments," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6195 LNCS, pp. 163–174, 2010
[39]. OMG, "Software & Systems Process Engineering Meta-Model Specification V2.0," no. April, p. 236, 2008.
[40]. B. Elvesæter, G. Benguria, and S. Ilieva, "A Comparison of the Essence 1.0 and SPEM 2.0 Specifications for Software Engineering Methods," *Pmde 2013*, 2013.

**Yani Widyani** was born in Bandung, on January 1970. She received a Master's degree in Software Engineering from Institut Teknologi Bandung (ITB) (Indonesia) in 2001. Currently, she is a Doctoral student in Electrical Engineering and Informatics at ITB. Her research interest concerns Software Development Methods.
Email: yani@itb.ac.id.

**Muhammad Zuhri Catur Candra** is a lecturer and researcher at the School of Electrical Engineering and Informatics of Institut Teknologi Bandung (ITB). He received his Doctoral degree from Technische Universitat Wien, Vienna, Austria, in 2016. His expertise areas are Service Computing and Cyber-Physical-Social Systems.
Email: m.candra@itb.ac.id.

**Eko Kuswardono Budiardjo** is a professor and researcher in Sofware Engineering at the Faculty of Computer Science, Universitas Indonesia (UI). He graduated from Institut Teknologi Bandung (ITB) before receiving his Master's degree from the University of Brunswick and Doctoral degree from Universitas Indonesia. He is also actively involved in several professional organizations, such as Ikatan Profesi Komputer dan Informatika Indonesia (IPKIN), as a general chair. His research interest areas are Software Engineering, Requirement Engineering, and Information Systems.
Email: eko@ui.ac.id.

**Benhard Sitohang** is a professor and researcher in Database at the School of Electrical Engineering and Informatics of Institut Teknologi Bandung (ITB). He received his Master's and Doctoral degree from USTL-Montpellier II, Montpellier in France. His current research areas are Data and Knowledge Engineering. Email: benhard@stei.itb.ac.id.