Journal of Computer Science

Research Paper

A Design Method of SPEM-based Feature Methodology Framework

Chee-Yang Song[1], Eun-Sook Cho[2]

[1]*Department of Software, Kyungpook National University, Daegu, Korea*
[2]*Department of Software Engineering, Seoil University, Seoul, Korea*

Article history
Received: 3 May 2025
Revised: 25 June 2025
Accepted: 11 July 2025

Corresponding Author:
Chee-Yang Song
Department of Software, Kyungpook National University, Daegu, Korea
Email: getachewmekuria@dbu.edu.et

Abstract: Existing feature methodologies have been presented primarily at the research level, lacking generalized frameworks suitable for commercial application. Critically, the absence of SPEM (Software & Systems Process Engineering Metamodel) standard compliance limits their standardization and systematization, hindering practical adoption. Furthermore, work-product-based feature methodology frameworks that facilitate methodological structuring are needed to address the evolving complexity of software development processes. This paper presents a design method for an SPEM-based feature methodology framework using metamodel principles. The proposed framework adheres to SPEM's method content and method process classification systems, incorporating the fundamental development process elements of role, task, and work product. Specifically, this study introduces a work-product-centric methodology framework aligned with SPEM elements, optimizing methodological structuring capabilities. Additionally, standardized specifications are provided to define activities within feature methodology phases. The feature methodology metamodel is formally specified using Z notation, with metamodel accuracy verified through Z/Eves toolset validation. Standardized SPEM adoption systematizes and enhances the feature methodology, improving usability and accessibility. By structuring existing methodologies around work products to accommodate diverse application and methodology evolution, the framework enables flexible feature development process construction suitable for varied project requirements and organizational contexts.

Keywords: Software Process Engineering, SPEM, Feature-Oriented Development, Methodology Framework, Work Product-Centric Design, Metamodeling, Formal Specification

Introduction

The continuous evolution of software computing environments has driven the emergence of novel development methods and process improvements aligned with shifting software development paradigms. Software lifecycle perspectives have encompassed business methodologies, management methodologies, development methodologies, and quality assurance methodologies. Concurrently, software development paradigms have progressed through object-oriented methodology, component-based methodology, service-oriented methodology, agile methodology, mobile methodology, ubiquitous methodology, cloud methodology, and big data/AI methodologies.

Research efforts have focused on metamodel-based methodology frameworks utilizing SPEM (Software & Systems Process Engineering Metamodel) (OMG, 2008) to systematically integrate, define, and manage heterogeneous enterprise-wide methodologies. These frameworks aim to support customized and optimized development processes matching specific project characteristics (scale and nature), while improving reusability, minimizing redundancy, and providing scalability for previously developed methodology components.

However, formal frameworks employing SPEM standards for feature methodologies remain underdeveloped. Existing feature methodology components—including phases, activities (tasks, steps), roles, work products, guidelines, and tailoring mechanisms—lack systematic establishment, diminishing methodology usability. This deficiency creates challenges in maintaining consistency and traceability when integrating feature methodologies with UML (Unified Modeling Language)-based methodologies. Conversely, UML-based software development methodologies have been successfully constructed using SPEM foundations (Cho, 2015). Consequently, an SPEM-based feature development process framework is essential to provide standardized and systematic feature methodologies.

Furthermore, flexible frameworks are necessary to accommodate evolving development methodologies. Existing methodologies employ structures separating work activities (or tasks) from work products, complicating continuous methodology improvement and specialized project-customized process support. Additionally, since multiple work products are generated per task, identical work products may be duplicated across tasks, hindering methodology structuring. This separation weakens activity modularity and reduces work product reusability. Given that development processes fundamentally aim to generate work products according to temporal milestones, a work-product-oriented feature methodology framework is required.

Cho (2015) presented a metamodel for SPEM-based development methodology frameworks without addressing feature methodology specifics. This metamodel outlined Method Class, Method Component, Process Component, and Delivery Process structures. However, specific methods for constructing work-product-based frameworks were not addressed. Specifically, the study did not describe: (1) how to create method classes by combining work product creation items with task-specific operations, (2) how to construct Method Components by combining Method Classes, or (3) how to provide structural and behavioral models as application examples. Therefore, structured methods for work-product-based feature methodology design are required.

This paper presents a design method for metamodel-based feature methodology frameworks according to SPEM standards. Specifically, a metamodel for SPEM and work-product-based feature methodology frameworks is established. This paper extends Cho (2015) by incorporating feature methodology and work-product-based methodology framework structuring methods.

The SPEM-based methodology metamodel is defined according to SPEM's classification system encompassing Method Content, Process Component, and Delivery Process, utilizing Method Content elements of Role, Task, and Work Product. A mapping profile between SPEM and feature methodology is established to facilitate this integration. The work-product-centered methodology framework creates method classes for each work product, combines them into method components, and subsequently assembles them according to methodology workflow to create Process Components and Delivery Processes. To ensure systematic feature methodology construction, formal specifications for phase-internal activities are defined.

The proposed feature metamodel demonstrates reliability through application to existing feature methodologies. Furthermore, to verify the presented feature metamodel's accuracy, formal specification using Z notation and verification using Z/Eves are provided.

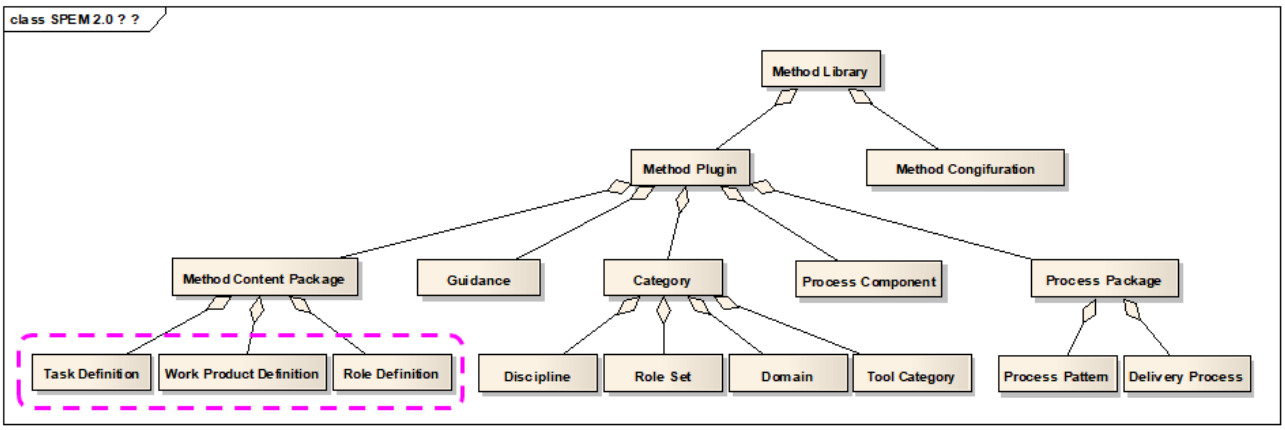


Fig. 1. Metamodel structure of SPEM V2.0 (OMG, 2008)

This feature methodology framework enables simplified creation of optimized feature development methodologies by assembling necessary methodology elements according to project characteristics or organizational (Information Technology company) requirements.

The remainder of this paper is organized as follows: Section 2 analyzes the current status of SPEM and feature methodologies utilizing SPEM principles. Section 3 defines the metamodel for SPEM-based feature methodology and the work-product-based feature methodology framework. Section 4 presents a case study demonstrating the practical application of the feature methodology based on the proposed metamodel. Section 5 addresses Z-formal specification and verification results for the feature methodology metamodel. Finally, Section 6 provides comparative evaluation with existing methods and discusses the characteristics and limitations of the proposed approach.

Literature Review

SPEM

OMG (Object Management Group)'s SPEM (OMG, 2008), an international standard that standardizes the process architecture of SW methodology, is a metamodel for the software development process. The structure of SPEM, defined as part of the UML metamodel (OMG,2008 and Sparx, 2025), is shown in Figure 1. SPEM is largely structured into Method Content, Method Process, and Process Package. Method Content consists of Work product definition, Role definition, Task definition, and Category. On the other hand, Method Process is composed of Task use, Role use, Work product use, Activity, and Process. Process Package is published as a combination of method processes.

SPEM 2.0 defines three independent methodology authoring units, including method content, process pattern, and process, and defines three components of behavior, work product, and role for each authoring unit. These concerns and components increase the complexity of actual development process authoring. SPEM has method content interrelated with the elements of Work-Product, Task, and Role, but is composed of separate classes. So, SPEM defines the process focusing on work behavior in process authoring. Ultimately, this activity (work)-centric perspective causes various problems, such as increased complexity and the occurrence of side effects.

To solve this problem, this paper presents a work product -based framework that integrates these three elements. A method class is constructed based on work product using SPEM's Task definition, Work product definition, and Role definition.

SPEM-based Methodology Metamodel

Cho (2015) has written a metamodel of a SPEM-based development methodology framework that addressed that can easily develop a methodology optimized for each project characteristic. In other words, it defined a metamodel that combines S/W development methodology and SPEM elements. This metamodel was used to cover a case where it was applied to a specific S company's development methodology. However, this was aimed at object-oriented development methodology, not feature methodology. This constitutes a Method Class for SPEM's Method Content and a Method-Component, which is a combination of these. Next, we presented a framework of S/W methodology that creates Process-Component and Delivery-Process by combining Method-Component by work order. However, it did not suggest how to create Method Class and Method-Component based on work product.

In other studies related to SPEM, modeling method using a UML metamodel and rational tools for software process modeling was presented in Bendraou *et al.* (2009), Bendraou *et al.* (2005). In addition, in Combemale and Cregut (2006), there is a study on the formal specification of the constraints of SPEM structures using OCL (Object Constraint Language). In Debnath *et al. (*2006), there is a UML-EWM (Extended Workflow Metamodel) based on SPEM for business process modeling. Kołcz (2006) has a modeling study on the management process through SPEM.

In Park *et al.* (2009), a teaching-learning process model was presented based on SPEM. By defining SPEM's “Method Content” and reusing it to define “Process Content,” it was defined by mapping it to the teaching-learning model. “Method Content” has elements of role, work, task, step, guidance, and discipline, and “Process Content” has elements of activity, repetition, phase, and delivery process, establishing a teaching-learning model. This demonstrates how SPEM can be applied within educational curriculum modeling. Park *et al.* (2008) defined a service-oriented development methodology based on SPEM. By SPEM's phases, goals, activities, unit activities, and work products, the service-oriented development method was defined in four phases: definition, design, development, and deployment. In other words, each phase was schematized and presented as work activities and work products. However, it is not considered for feature methodology. Pillat *et al.* (2015) proposes a method of customizing the S/W development methodology by tailoring SPEM

and then expanding and converting it into a BPMN model in order to provide process behavior expression and simulation in SPEM. In addition, Park *et al.* (2007) presented a simulation model of the development process through SPEM and DEVSHybrid mapping. Morale (2022) introduced SPEM-based development process construction and authoring tools to develop new smart software. In Simeckova *et al.* (2020), a method for modeling process anti-patterns using SPEM and an Eclipse-based implementation were presented. Baumgarten *et al.* (2015) used SPEM in the CRYSTAL platform development process for optimization of complex embedded systems. Meanwhile, Bezerra *et al.* (2023) evaluated the usability of SPEM and BPMN (Business Process Modeling Notation) by applying them to modeling in application fields such as cloud and machine learning. This provides a cutting-edge example of how process modeling using SPEM is applied in practical and educational settings. Agh *et al.*, (2023) applied the metamodel to machine learning systems proposes an integrated approach to handle consistencies of different views of ML (Machine Learning) system analysis named Multi-View Modeling Framework for ML Systems.

Work Product-based Methodology Metamodel

Legacy research on the framework of work product-based methodology is weak. In Cho (2015), a methodological framework centered on work product was mentioned. In this framework, the structures of Method Class, Method Component, Process Component, and Delivery Process were presented. SPEM's Method Content consists of Work-Product, which is an output of activity, Method Class, which is a collection of activities, and Method Component, which is a collection of Method Classes. Here, Method Class is defined as a reusable unit class centered on the work product. Method Component is a structure of Method Classes and is defined as a unit model. However, it did not suggest how to create a method class based on the work product by combining the work product component of the methodology with an operation.

In this article, we will cover how to define a framework for feature methodology using this SPEM-based and work product-oriented structure.

Feature Methodology

Lee and Kang (2002) and Kim *et al.* (2005) show the process of the feature-based FORM (Feature-Oriented Reuse Method) method. In Figure 2, the FORM methodology consists of domain engineering and application engineering. Domain engineering consists of domain analysis, domain architecture design, and domain component development. Through feature modeling, FORM analyzes and models the commonalities and variability between products focusing on features. This is a method of developing applications by reusing feature assets and adapting them to changes. In the feature model, features are expressed as required or optional due to commonality and variability. Additionally, features are expressed by layering techniques of abstract development into capabilities, operating environment, domain technology, and implementation technology.

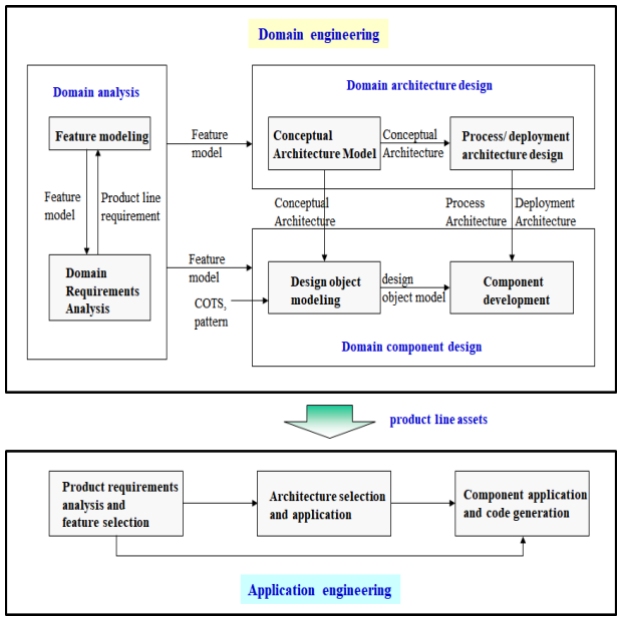


Fig. 2. Feature reuse methodology (Lee et al., 2002; Kim et al., 2005)

In Bae and Kang (2013), a method and process for configuring features and feature attributes that meet the business and quality goals of the product were presented. Lee (2012) dealt with the conversion of the FORM architecture model to the UML architecture model. In addition, as a study on various application fields applying feature methodology, Kim and Kang (2006) presented a feature development method for embedded system development. Furthermore, Lee et al. (2016) presented a feature-based IoT (Internet of Things) development method, and Song and Cho (2022) presented a feature-based cloud development method. In Halimeh et al. (2023), A Model-Driven Approach for Software Process Line Engineering was provided.

In Song et al. (2022), a feature modeling method for developing cloud applications using feature models is presented. Figure 3 shows the framework of the feature-oriented cloud modeling process presented in Song et al. (2022).

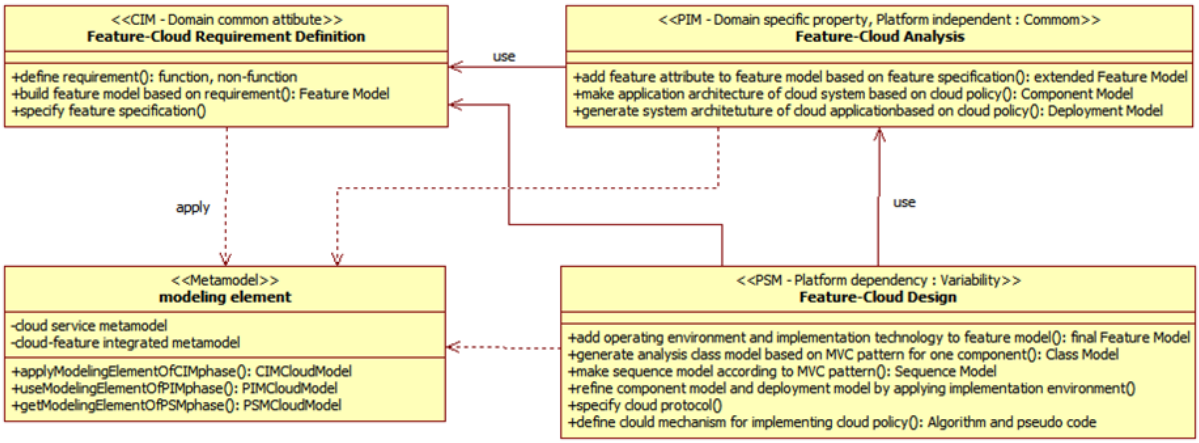


Fig. 3. Structural model of feature-oriented cloud modeling process: SM-CMP (Song et al., 2022)

In this paper, the activities of each phase in this framework will be further specified and formalized and defined based on SPEM. In particular, this paper drives to structure the feature methodology shown in Figure 3 into a standardized development process based on SPEM. Halimeh *et al.* (2023) propose a novel MDD (Model Driven Development) approach specialized for Software Process Line Engineering (SPrLE).

Metamodel of SPEM-based Feature Methodology Framework

The purpose of this paper is to provide a method for designing an efficient feature methodology structure using an object method. To do this, the first step is to define a metamodel of the SPEM-based and work product-centered feature methodology framework ("Metamodel of SPEM-based feature methodology" section and "Conversion profile between SPEM and Feature methodologies" section). Second, when constructing the Feature methodology, a specification is established to define the activity tasks within the development phase ("Activity specification of feature methodology" section).

This section defines the feature methodology framework (architecture) as a metamodel based on SPEM for the construction of a systematic methodology and on the basis of a work product for the authoring of a flexible methodology.

SPEM proposed defining the methodology according to the classification system of Method Content, Method Process, and Process Package. At first, the metamodel of the SPEM-based feature methodology in Figure 4 is defined using these SPEM package elements and elements of the feature methodology. Next, the work product-based feature methodology metamodel in Figure 5 is defined according to the structure of Method Class, Method Component, Process Component, and Delivery Process in Cho (2015). At this time, the Method Class is made based on the work product of the methodology. After that, a conversion profile between detailed SPEM and feature methodologies is specified. Lastly, SPEM-based Feature methodology construction defines the phases of the Feature methodology based on SPEM and defines the activity specification within the phase in Table 5.

The design principle of the proposed method is as follows and is reflected in Figure 4 and Figure 5.

- Based on SPEM, the framework of the feature methodology is defined and provided to ensure standardity and systematicity.
- By constructing a methodology framework based on work product, a SPEM element, rather than based on the activity of the methodology, reuse is increased and duplication of work products between activities is eliminated.
- The framework of the feature methodology is defined as a metamodel to support the construction of an optimized and flexible methodology according to the scale and characteristics of the project.
- Establish the existing feature methodology as a framework to facilitate understanding and publishing of the methodology.

Metamodel of SPEM-based Feature Methodology

The feature methodology framework metamodel (Methodology Modeling Metamodel) is defined by adding feature methodology components to SPEM elements. This metamodel is defined as a subclass of SPEM.

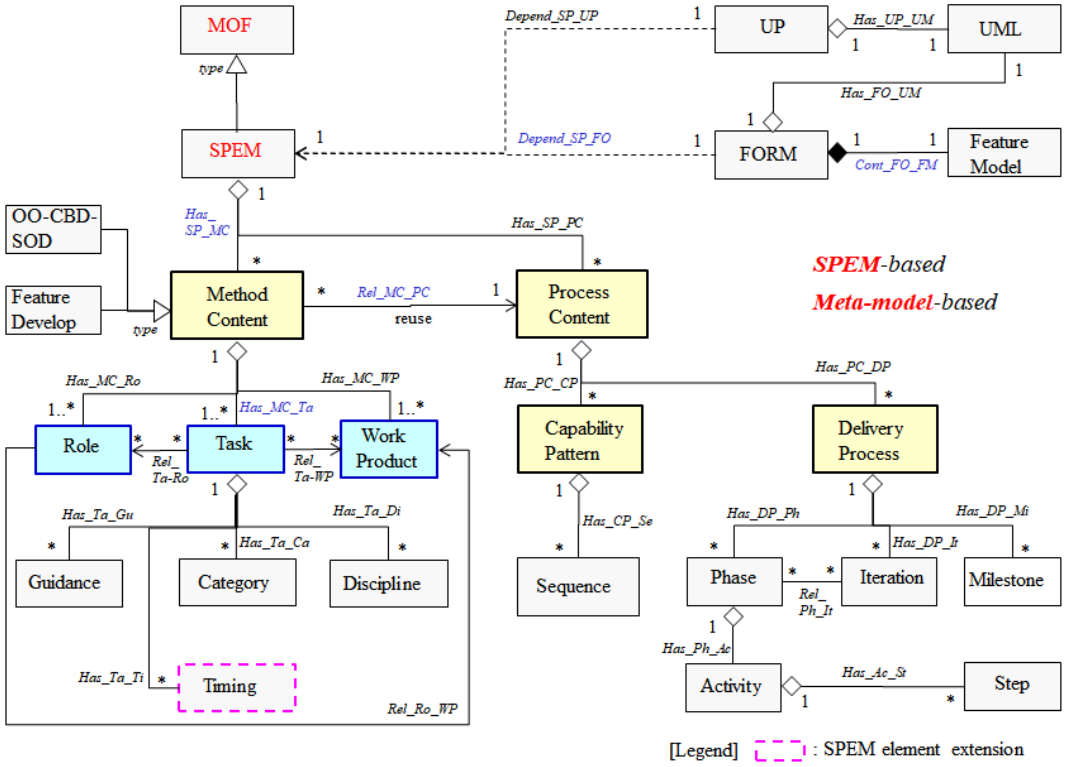


Fig. 4: Meta-Model of Feature Methodology Architecture based on SPEM: MM-FMA

The meta-model of the **SPEM-based Feature Methodology Framework** (MM-FMA) is shown in Figure 4 (extension and modification of Cho, 2015). This metamodel is based on SPEM and consists of Method content and Process Contents. Method Content consists of SPEM's work-product, task, and role elements. Process Content is a managerial unit with a temporal concept and a work order to achieve milestones. It consists of Capability Pattern and Delivery Process. Capability Pattern defines the order of tasks to be performed. Delivery Process defines the process of the entire development phase.

In Figure 4, the "Timing" element, which represents the duration of the task, was added to SPEM in this paper. Timing means the timing and duration of work. OO-CBD-SOD stands for Object Oriented - Component Based Development - Service Oriented Development, and FeatureD stands for Feature Development. Method Content describes who, what, why, and how in the S/W development process. Role means ability, skill, and responsibility. Task is a unit of work performance. Guidance is detailed instructions on how to perform a task. Category refers to a grouped field of work. Discipline represents the detailed work principles of a task.

The **Work product-centered feature methodology metamodel** is shown in Figure 5.

Figure 4 shows the relationship with the legacy S/W development methodology based on the components of the SPEM standard in an overall aspect. Figure 5 is a hierarchical representation of how to structure the methodology based on work product using the SPEM components (Role, task, Method content, Process content, etc.) of Figure 4.

First, Method-class, which creates reuse assets for each work-product, is created with Work products for each activity of the feature methodology. That is, it defines a method class of an independent technology asset. Method-classes are combined to create a larger independent reusable unit, method-component. Create a method component as an application asset of the process unit. Next, from the view of method using independent assets, process-component combines method-component to establish a general-purpose feature methodology (FORM, UP: Unified Process). Finally, a delivery-process (specialized S company-FORM, S company-UP) is established by customizing and publishing a general-purpose methodology to suit the characteristics of the project or IT company.

The creation rules for work product-based method classes and method components are set as follows. Process components and delivery processes, which have a time concept of development, are created similarly, so their creation rules are omitted.

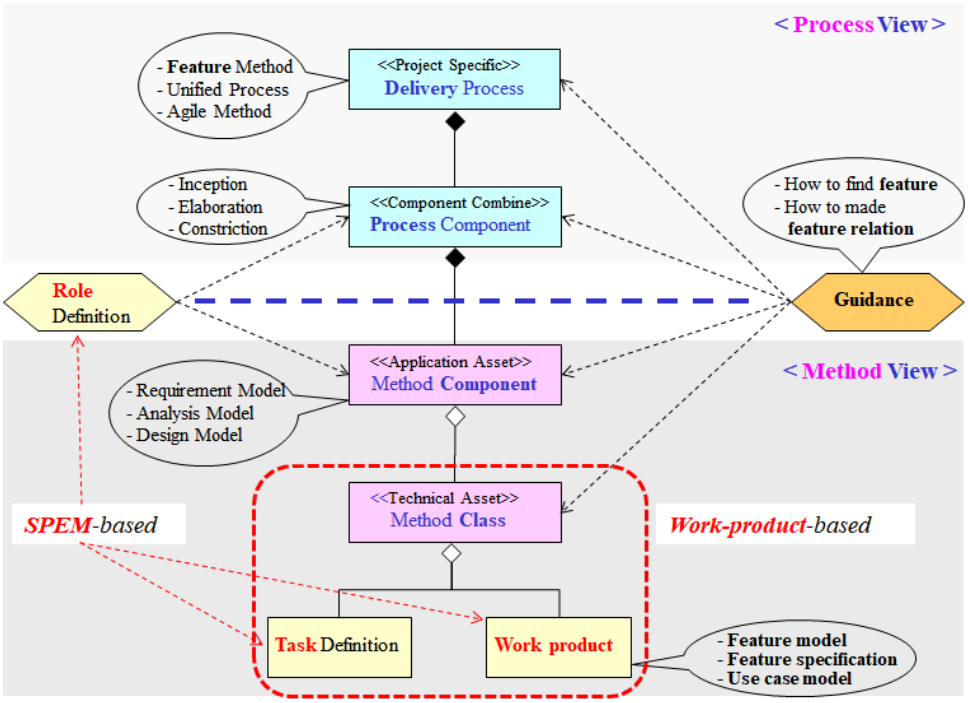


Fig. 5: Metamodel for Methodology Construction Process based on Work-Product

Rule 1: (Method class) The creation of a method class, which is a unit technical asset for reuse in method components, is as follows:

- Method classes are defined using SPEM's Work-product, Task, and Role elements.
- Define one method class for each work product.

  The method class name becomes the work product name, and the properties are composed of the work product's written items and role player. The operations required to create this work product are defined by consisting of activities (or sub-tasks), which are the lower units of activity.
  - Method Class = {name(Work-product) + attribute(Write item, Role player) + operation(Activity, work)}

· The *method class creation process* is as follows.

① Identify the target work product.

② Classify work product types (technical, common).

③ Identify attributes, operations and roles for each work product.

④ Define method classes for each work product in Figure 6.

⑤ Deploy work products by type.

Rule 2: (Method component) The creation of a method component, which is an application asset reused in process components, is as follows.

- Define a reusable method component by combining previously defined method classes.

$$\text{- Method Component} = \sum_{i=1}^{n} \text{Method Class}$$

· Method co33mponents create models with a hierarchical structure of abstraction according to the scale of the system.

- Depending on the stakeholder's perspective, define models for each perspective as requirements definition model, analysis model, architecture model, design model, implementation model, and test model.
- Create method components of the structural model and behavioral model by combining view-related method classes.

- The structural model is constructed by identifying method classes related to the model for each perspective and identifying input/output relationships between them.
- The behavior model is defined based on the structural model to represent the temporal execution procedures between models.

· The *creation procedure of the method component* is as follows.

① Identify perspectives and define models for each perspective.

② Identify method classes related to the creation model.

③ Identify inputs for each model.

④ Identify the relationships between them and create a structural model for each level of abstraction in Figure 8.

⑤ Create a behavior model by creating a structural model-based object and identifying messages in Figure 9.

⑥ Arrange structural models and behavioral models by type.

Conversion Profile between SPEM and Feature Methodologies

In the previous section, the feature methodology framework Figure 5 was defined as a metamodel based on the SPEM standard Figure 4. Next, a conversion profile between the standard modeling elements of SPEM and the components of the feature methodology is needed. In other words, it is shown how SPEM elements are accepted and satisfied as elements of the feature methodology.

Regarding the mapping between SPEM and legal methodologies, the relationship between "Method-Content" between SPEM and features and "Process-Content" between SPEM and features is analyzed. After that, the structural transformation profile of the SPEM-based feature methodology framework that integrates these is defined.

"Method-Content" between SPEM and Feature

Table 1 shows the "Method-Content" between SPEM and Feature. This shows how SPEM's Method-Content Package is expressed in the general-purpose methodology and feature methodology. For example, SPEM's work product is expressed as an artifact (or work product) in the methodology, and is expressed as a feature attribute in the feature methodology. In Table 1, Step refers to the detailed steps of the work, Guidance represents reference materials, and Discipline represents a set of tasks.

Table 1: "Method-Content" profile between SPEM and Feature methodology

SPEM	Methodology Elements	Feature Methodology
Method Content Package	Method-Content	
Role	Performer	
Work product	Work product (artifact)	Feature (Attribute)
Task	Activity	Feature (Operation)
	Step	
	Guidance	
	Discipline	
	Process	Interaction

"Process-Content" between SPEM and Feature

Table 2 shows the "Process-Content" between SPEM and Feature. It shows how SPEM's Process Component Package is expressed in the methodology and feature methodology. SPEM does not define in detail how to define the process by combining the contents of the "Method Content Package" in Table 1. For this purpose, Table 2 shows the interrelationship between general-purpose methodology and feature methodology in terms of process. SPEM's Process Package was added to Table 2 to show the conversion profile with the methodology. Activity refers to the connection of tasks, Iteration refers to the connection of activities, Phase refers to the connection of repetitions, and Delivery process refers to the connection of phases and the entire development phase.

SPEM-based Feature Methodology Conversion Structure

Table 3 compares the interrelationships according to methodological components between SPEM and the proposed feature framework structure in Figure 5.

Table 2: "Method-Process" profile between SPEM and methodology

SPEM	Methodology Elements	Feature Methodology
Process Component	Process-Content	
Package	Activity	Activity
	Iteration	Iteration
	Phase	Phase
Process Package	Delivery process	Interaction

Table 3: Conversion structure to SPEM-based feature methodology

SPEM structure		Feature Methodology Framework Structure	Meaning
Method Content Package - Task definition - Work product definition - Role definition		**Method class** - By methodology - By technology **Work product** - Method component - Independent asset unit of process organization	Independent Work product - Technology assets - Application assets
Process Component		- Process component	Independent process unit
Process Package	Process Pattern	- General development process - General process patterns by methodology	Pattern organized by process
	Delivery Process	- Specialized development process	Specialized process on a project basis

Table 4: Conversion profile to method class of feature methodology

Construction Target Methodology	Class model	Note
Classification system of Feature methodology	Package declaration	Apply classification system
Work product of feature methodology	Class declaration	Applies below phase
Feature methodology phase	Super class name / operation name	- Create a superclass model - Method component
Feature Methodology Activities	Composed of subclasses	- Create subclass model - Method component
Work product name of feature methodology / work product write element/process activity	Subclass name/Attribute name / Operation name	- Method class

Table 4, which shows how to create unit method classes and method components for each work product, which is the result of work within the feature methodology. This defines the conversion of the feature methodology structure into a class model. As an application example, Figure 8 is created by applying Table 4.

Activity Specification of Feature Methodology

Generally, the development process structure of commercial object-oriented and CBD methodologies consists of hierarchical elements of phases, activities, tasks, and steps. The form that defines the methodology's tasks consists of task name, Goal, Metric, Role, Entry criteria, Exit criteria, Input, Work product (Output), Dependency, Related technique, Work step, Guideline, etc. At this time, work products are created for each task. However, in the case of the feature methodology, the process structure defines the level of activities within the phase, and the activities of the methodology are not defined using a standardized format as above.

In order to build a SPEM-based feature methodology, the development phase and the work tasks of the activities within the phase must be established. So, to speak, the activities of the feature methodology must be defined using the core elements of SPEM: Role, Task, and Work-product. Through this, developers can easily perform S/W development work using well-defined feature methodology procedures, methods, and guidelines.

The activities within the phases of the feature methodology are defined as follows.

Definition 1: (Feature Methodology Activity: FMA).

Activities within the phases of the feature methodology are composed of FMA= (n, Goal, Role, Input, Activity, →, Work-product, Criteria). (1) name of activity n; (2) Goal that this activity seeks to accomplish; (3) Activity responsibility

and role; (4) Input required for the activity; (5) A finite set of detailed activities to perform a task Activity, a, b, c ∈ FMA; (6) Transitive relationship between Activities → ⊆ (Activity x Activity); (7) Work-product, the result of the activity; (8) Criteria for starting this activity and ending criteria for ending the activity.

Each phase in the feature methodology in Figure 3 consists of a number of activities being performed by the phase. This is a work group with subdivided phase and is a work unit where actual work is performed. The activity specification format for defining this based on SPEM is shown in Table 5. That is, SPEM's Role, Work-product, and Task (blue font) are used to define the activity specification at the feature modeling phase. SPEM's tasks are expressed as activity names in Table 5. In particular, the specification is defined by adding security, time taken, and quality review checklists.

The specifications for the work of unit activities are broadly classified into outline, performance procedure, and performance guide, as shown in Table 5. The overview section consists of subdivided elements such as goals and measurement indicators. Here, the Role element of SPEM is defined as "Responsibility and Role", the Task element as "Activity Name", and the Work-product element as "Work product" as the main items of the process activity specification. This means that the feature methodology is defined based on the SPEM standard. As a result, the characteristic of the specification of the feature methodology is that it satisfies the three components of the SPEM standard. In particular, by reflecting security work and quality review, work for functional requirements as well as work to satisfy the quality of non-functional requirements have been improved and strengthened.

Case Study

This paper presents a method for creating a structure of a development methodology that is easy to change according to the development technology advancement. Usually, a project applies a development methodology to create an application model at the model level. The proposed method is a method for structuring a development methodology at the metamodel level. Therefore, this method targets a methodology, not a project, as case study. The case study is to transform the structure of a legacy feature methodology into a work product structure using the elements of the proposed methodology framework and create a restructured methodology. That is, by combining the components of the existing feature methodology, a Method class, Method component, Process component, and Delivery process based on the four-layer structure of Figure 5 were made. This framework has the generality to be used not only in the feature methodology as a case study, but also in the methodologies of object-oriented, CBD (Component-Based Development), and SOA (Service Oriented Architecture).

In this section, in order to show the effectiveness of the method presented in the previous section, by applying the SPEM - based and work product - centered feature Methodology metamodels in Figure 4 and Figure 5 , and the conversion profiles (Table 3 and Table 4), the structure of feature methodology is built. The existing feature methodology to be applied is Figure 3. Elements are defined in a hierarchical structure based on SPEM and Work product. In this way, it is shown as an example that the structure of the existing feature methodology is designed in an object manner using the proposed metamodel.

The SPEM-based hierarchical structure according to Method Class, Method Component, Capability Pattern, and Delivery Process in Figure 4 is shown in Figure 6 and Figure 7. Figure 8 represents that the method class and method component were created by based on the work product in Figure 5. This is expressed in class notation. Figure 6 (a) shows the entire framework structure of the S/W development methodology including object-oriented and feature methodology. Figure 6 (b) and Figure 7 represent the detailed structure of each of the four layers. For each layer that composes Figure 6 (a), Figure 6 (b) shows the Method Class, Figure 7 (a) represents the Method Component, and Figure 7 (b) shows the detailed structure of the Delivery Process. Next, Figure 8 and Figure 9 are instances created from the overall framework of the methodology and applied to a specific feature methodology. These are Method Class and Method Component written for the requirement definition phase of the feature methodology. Figure 8 shows the structural model of the "Requirement Definition" phase based on work products, and Figure 9 shows the execution structure. This shows an example of expressing the feature methodology as an object-oriented class centered on work products. In Figure 6, the UML and OO-CBD sections refer to Cho (2015). In (Cho, 2015), Method class is a unit class that reuses the work product. Method component is a set of method class and is defined as a unit reuse model. Method Process is an actual execution process with time and work order and is composed by assembling method components. Based on this, work product-based methodology elements were created targeting the FORM methodology.

Table 5: Activity specification within each phase of the feature methodology

Activity name	Work activity name	**Phase** Phase name	**Revision date** Date of revision
1. Overview	Goal	What this activity aims to achieve	
	Metric	Measuring criteria for the work product performed by the activity	
	Responsibilities and Roles	A performer who performs a task and is responsible for the work-product	
	Entry Criteria	Conditions that must be satisfied before performing an activity	
	Exit Criteria	Conditions that must be satisfied after performing an activity	
	Input	Inputs required to perform this activity	
	Work product	Work product created after performing this activity	
	Process relevance (Dependency)	Whether it is related to other activities	
	Security	Security-related tasks in this activity	
	Tool	Tools that support performing this activity	
	Technique	Techniques applied in performing this activity	
	Standard	Standards involved in performing this activity	
	Related Activity	Other activities related to the input/output transition of this activity	
	Work Time	When should this work be done and how long does it take?	
2. Perform Procedure (Step)	- Detailed and procedural steps from starting this activity until the work is completed		
3. Performance guide	- Specific guidelines for practical application of this activity That is, detailed guidance on the methods and techniques used.		
4. Quality Review Checklist	- Detailed quality assessment of results based on measurement indicators - Check and evaluate whether the work product of this activity meets the goals and requirements, is consistent, and is free from complexity.		

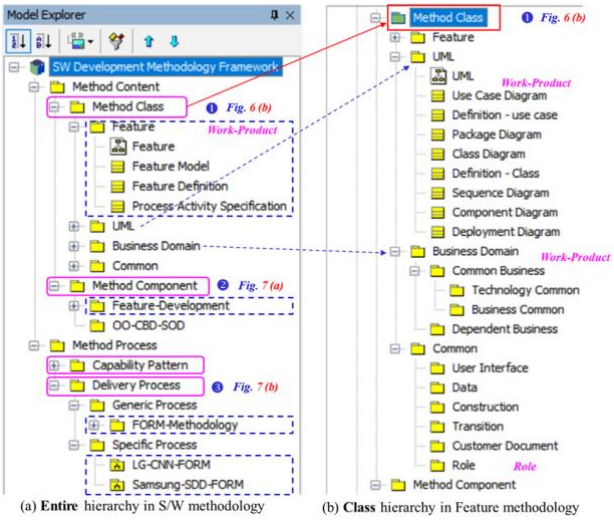
Layers of the Feature Methodology Framework

The overall hierarchical structure of the S/W methodology is shown in (a) of Figure 6.

The blue dotted line indicates that it is related to the feature methodology. It was constructed based on SPEM's Work-Product, Role, and Task elements. (b) in Figure 6 shows the hierarchical structure of Method

DOI: 10.3844/jcssp.2025.2663.2684

Class. Method Class is created for each Work-Product created at each phase.

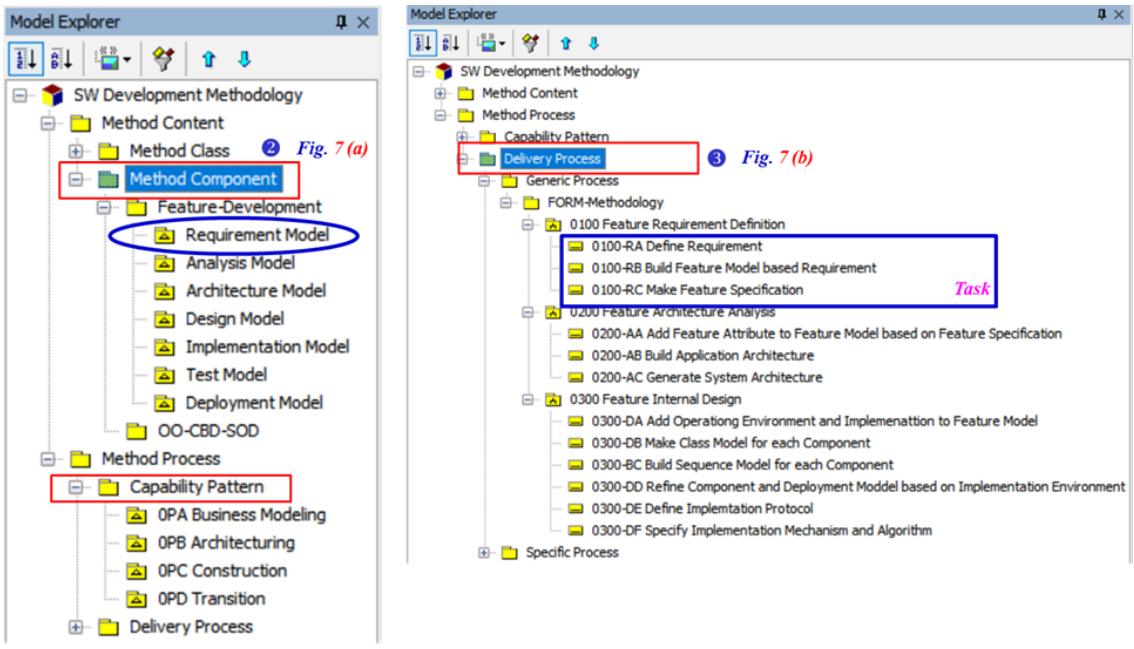


(a) **Entire** hierarchy in S/W methodology (b) **Class** hierarchy in Feature methodology

Fig. 6: Hierarchy of Feature Methodology Framework

This proves that the methodology structure is defined based on work product. Method classes were classified into Feature, UML, Business Domain, and Common. The feature package consists of work products such as feature model and feature definition. The common package consists of work products that are used independently for a specific technology, business, or paradigm (CBD, Feature method, etc.). Role means project manager (PM), software architect, data architect, etc.

Figure 7 shows the hierarchical structure of Method Component, Capability Pattern, and Delivery Process. Method component was classified into Feature-Development and OO-CBD-SOD. As shown in (a) of Figure 7, the Feature-Development package is a complex class expressed by Requirement Model, Analysis Model, and Architecture Model, which are the work products of each phase. Capability pattern is defined as a phase of a large task with a time concept, such as business modeling and architecture modeling, for the development process. Finally, the delivery process defines the feature execution process by reusing the already defined method component. It was separated into a generic process with general purpose and a specific process specialized for a specific project (or IT development company). As shown in (b) of Figure 7, the FORM-Methodology of the generic process package was applied by deleting the cloud term from the feature methodology in Figure 3. The specific process package shown in (a) of Figure 6 establishes a general-purpose feature methodology by customizing it to suit the project (or company). Since there is no feature methodology currently in use by IT companies, it was named hypothetically, for example, Samsung-SOD-FORM.



(a) Hierarchy of Method **Component** and **Capability** Pattern (b) Hierarchy of **Delivery** Process

Fig. 7: Hierarchy of method components and delivery process

DOI: 10.3844/jcssp.2025.2663.2684

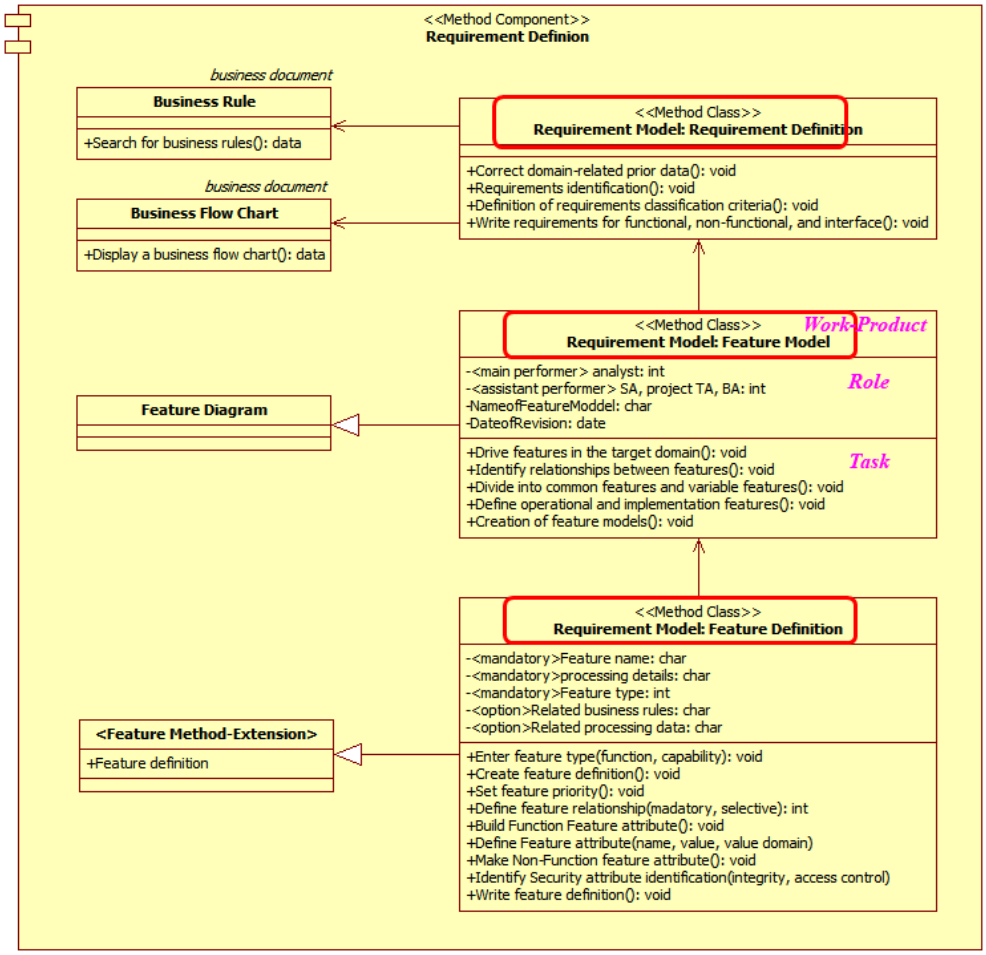


Fig. 8: Structural model of "Requirements Definition" work product: Method-component

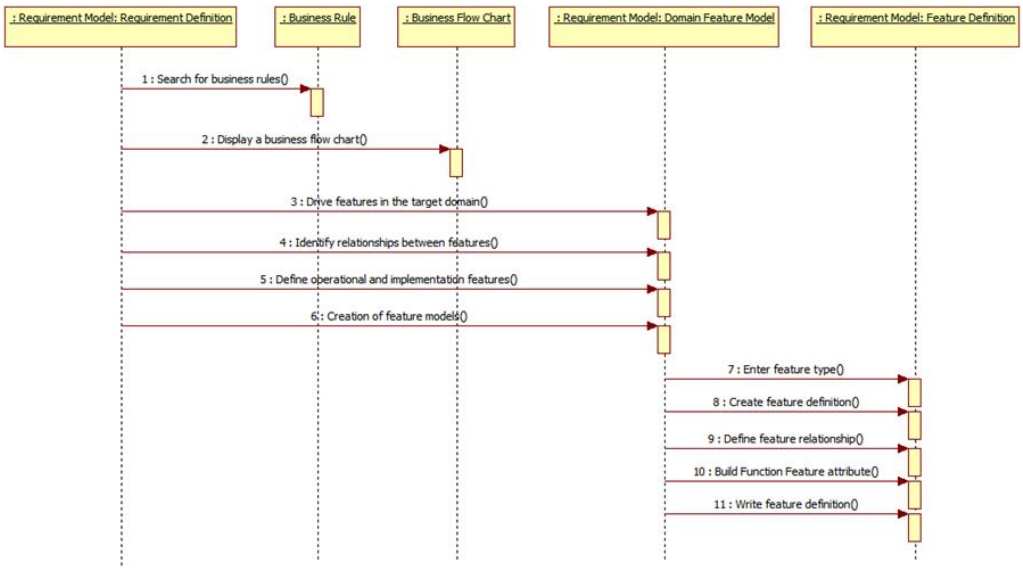


Fig. 9: Behavior model of "Requirements Definition" Work product: Method-Component

Architecture of Work Product-based Feature Methodology

According to the metamodel of the work product-based methodology framework in Figure 5, the Method class, Method component, and Process component in Figure 6 and Figure 7 must be defined for the existing feature methodology. To this end, we show how to create a method class and method component by applying Rule 1 and Rule 2. Figure 8 shows the creation of method-class and method-component for the "requirement definition phase" of the feature methodology.

Method Class

Method class is defined according to Rule 1 and Table 4. For an instance, the "Feature model" work product in (a) of Figure 6 should be defined as a class method (red square) in Figure 8. Based on rule 1, the work product name is defined as the class name in the "Feature Model" method class in Figure 8. The properties of the method class consist of the "main performer" who creates this feature model as the role and the "NameofFeatureModel" as the creation item of the work product. Next, the task for creating the feature model is defined as operation. This shows that method class is defined with a focus on Work-product according to SPEM's Role, Task, and Work-product. Afterwards, we can generate method component and process component by combining the defined method classes.

Method Component

According to Rule 2, the method component is defined as a combination of method classes. As an example, the structural model at the activity level within the "requirements definition" phase is shown in Figure 8 and the behavioral model is shown in Figure 9. Figure 8 is a construction for the "Requirement Model" method component (blue square) in Figure 7. That is, the method component was defined by combining previously defined method classes such as "Requirement Definition" for the "Requirement Model" work product that must be created in the requirements definition phase of the feature methodology. That is, the "Requirements Definition" method component is composed of three method classes: Requirement Definition, Feature Model, and Feature Definition. The behavior model of the method component in Figure 9 is made by reversing the dependency relationship between classes and determining the order of messages between objects based on the structural model in Figure 8. In the process of processing the behavior model for creating a work product, the request definition object searches the business rule object for rules and requests flow chart data from the business flow chart object. Receive this and define the requirements. Next, create a Feature model from the domain feature model object based on these requirements.

Lastly, the work proceeds in the order of creating a feature definition object and specifying each feature.

As a result, examples reflecting SPEM's structure-related method content and process content were shown in Figure 6 and Figure 7 based on the work product. In addition, method class and method component were created in Figure 8 and Figure 9 based on the Role, Task, and Work-product of SPEM method content. Using the proposed framework, we can see that the feature methodology is structured based on SPEM and work product.

Create Activity Specifications within Each Phase of the Feature Methodology

Work activities within the phases of the feature methodology in Figure 3 must be defined according to the form of SPEM-based activity specification in Table 5. This means that when all activities at each development phase are defined by this specification, a SPEM-based feature methodology is constructed. Unlike the previous sections, which deal with the structure of the actual feature methodology, this unit deals with cases of writing an activity specification for the feature methodology.

Table 6 presents the "Requirements-based Feature Model Creation" activity within the "Feature Requirements Definition" phase (see Figure 7b), following the format of Table 5. This activity results in the "Feature Model" shown in Figure 8

Z-Formal Specification and Verification of Feature Metamodel

The metamodel of the proposed feature methodology framework (Figure 4 MM-FMA) is informally specified for structures expressed as class models. Therefore, the syntax and meaning of the feature metamodel must be formally specified, and the accuracy of the model must be verified. To do this, in this section, the metamodel is specified in Z using the conversion method of the class model to the Z formal specification language (Shroff and France, 1997; Song, 2003) and application examples (Song *et al.*, 2011; Cho and Song, 2022). After that, syntactic defects in the structure of the metamodel. That is, the relationships between components, are verified. This can be done by showing that the feature methodology metamodel is proven through the written Z specification through Z/Eves (Saaltink, 1999). The metamodel subject to Z verification is formally specified for the metamodel in Figure 4, and the verification results are proven.

Table 6: "Requirements-based Feature Model Creation" activity specification

Activity name	Creation of feature model based on requirements	**Phase**	Feature requirement definition	**Revision date**	2025. 06. 30
1. Overview	Goal	Create a feature model for one use case.			
	Metric	Feature model consistency and complexity			
	Responsibilities and Roles	Domain analyst who performs domain analysis to create a feature model			
	Entry Criteria	Requirements Specification			
	Exit Criteria	Completed creation of feature model			
	Input	Requirements Definition			
	Work product	Feature model			
	Process relevance (Dependency)	It depends on the work product of the previous "Requirements Definition" activity.			
	Security	Security-related tasks in this activity			
	Tool	ASADAL			
	Technique	Feature modeling technique			
	Standard	Feature model notation			
	Related Activity	- "Requirements Definition" activity - "Feature Statement" activity			
	Work Time	Computation based on the domain of the target system			
2. Perform Procedure (Step)	- Identify function-oriented features based on the requirements specification. - Analyze relationships between features. - Create a feature model.				
3. Performance guide	- Identify features centered on characteristics (functions) based on the requirements specification. - Analyze hierarchical relationships between features. - Identify the relationship between two features. . Resolve required/optional/optional relationships. . Analyze the "composed-of," "generalization / specialization," and "implemented by" relationships between features. . Examine constraints such as "require" and "mutually-exclude" relationships between features. - Divide the identified features into common and variable features. - Create an initial feature model. - Investigate features related to the operating environment, domain, and implementation technology. - Create the final feature model.				
4. Quality Review Checklist	- Check consistency between requirements and features. - Investigate and confirm missing and duplicate features. - Review the complexity of the feature model.				

Formal Specification of Feature Methodology Metamodel

The proposed metamodel in Figure 4 of the feature methodology framework is formally specified using the Z language. The Z specification of the metamodel (class model) defines a basic type, specifies an element-based Z schema for each class using this type, and then follows the relationships between classes (association, inheritance, inclusion, dependency). Create a relationship-based Z schema. For Z specification, the feature metamodel must be given both a relationship name and multiplicity.

As a result of the Z specification, 22 basic types including Timing_ID were defined as the first basic

DOI: 10.3844/jcssp.2025.2663.2684

type, as Z schema, 25 including MOF were specified as the element-based Z schema, and 25 including Has_Ta_Ti were specified as the third relationship-based schema specification.

Z Basic Type Definition

In Z, basic type definitions are written as basic types and free types. The basic type defines the basic type for each element for all classes in the feature metamodel. 23 basic types were defined, including Timing_ID and Guidance _ID.

Z Schema Specification

The Z schema specification is written as a Z schema using the previously declared Z basic type based on relationships and all elements (classes) included in the feature methodology metamodel MM-FMA in Figure 4.

The element-based Z schema specification is 1:1, with one Z schema for each element. Since this element-based schema specification is a specification based on relationships between elements, the order in which the schema is defined is important. For example, the inclusion relationship between two elements (classes) creates a Z schema from the lower element to the upper element. Meanwhile, in the inheritance relationship, the parent element is first defined as a Z schema, and then the lower elements are specified by inheriting it. Figure 10 shows the element-based Z schema specification for MM-FMA in Figure 4.

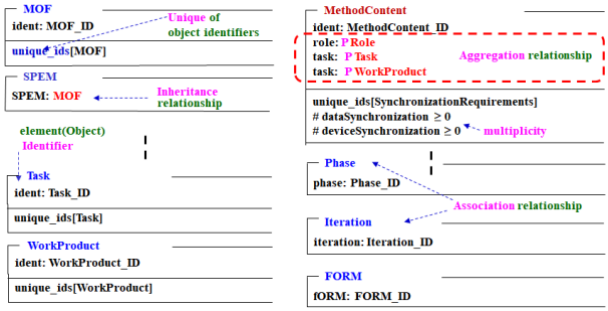


Fig. 10: Element-based Z schema specification of the MM-FMA metamodel

In Figure 10, the Z specification with an inheritance relationship declares the parent element "MOF" schema first, and specifies the "MOF" schema as a variable in the child element "SPEM" schema.

Next, the relationship-based Z schema specification creates a Z schema for each type of relationship between two elements. Through this, it is possible to express multiplicity clearly and in accordance with the type of relationship between elements. Figure 11 shows part of the relationship-based schema for the MM-FMA metamodel. In Shroff and France (1997), Song (2003), Song *et al.* (2011), and Cho *et al.* (2022), the relationship-based schema is specified as a declaration part and a predicate part. The declaration part (signature part) declares the elements connected for each relationship and the functional relationships between them. The predicate part describes multiplicity and constraints.

As an example, in Figure 11, for the Z specification of the Association relationship, the elements related to "Rel_MC_PC" are "MethodContent" and "ProcessContent". The relationship between these two elements is declared by reflecting the multiplicity in the declaration part, and the multiplicity between the two elements is expressed as *:1 by giving constraints on the domain and range in the predicate part.

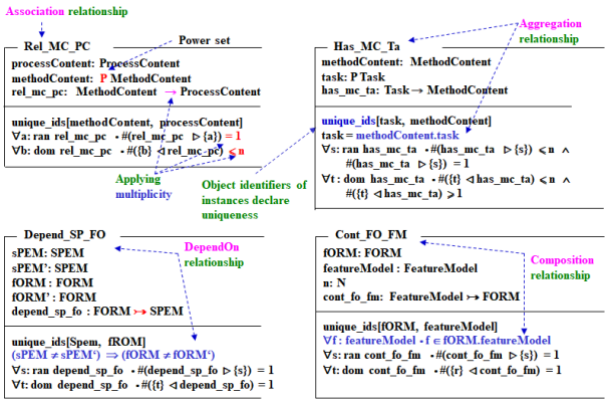


Fig. 11: Relationship-based Z schema specification of MM-FMA metamodel

Model Verification of Z Specification

The Z specification created in the previous section needs to be checked for accuracy. This is because the syntactic and semantic properties of the Z specification must be shown to be correct to prove that the model is clear. We can use the Z/Eves Tool to check the accuracy of the syntactic structure and properties of the Z schema specification. In other words, syntax checking, variable range checking, and consistency type checking can be performed on the Z schema specification.

Through the Z/Eves tool (Saaltink, 1999), to check type consistency, 23 basic type declarations and 2 free types from "Z basic type definition" section were registered and syntactic and semantic checks were performed. As a result, the execution result of " syntax" and "proof" attribute was out as "Y". This confirmed that there were no errors or conflicts in the basic types and free types of the Z specification.

DOI: 10.3844/jcssp.2025.2663.2684

Next, the schema of 27 was checked against the element-based Z specification, and the execution results of both "syntax" and "proof" were "Y", respectively. As an instance, Figure 12 is the result screen of a check for the "SPEM" schema. 'Syntax' at the top left shows syntax and type checking, and if it appears as 'Y', it means there is consistency between types. 'proof' shows the results of domain checking, and if it appears as 'Y', it means that the Z specification is accurate.

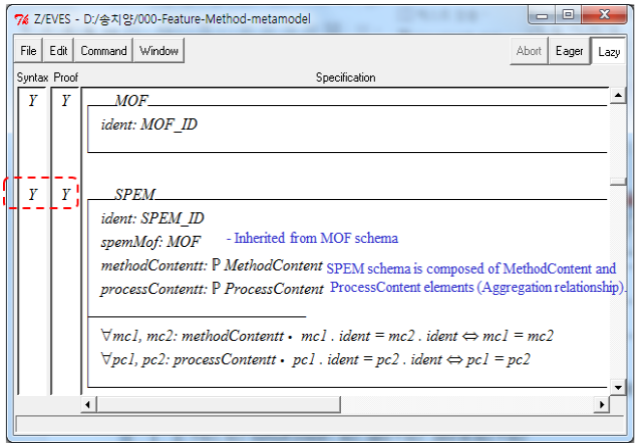


Fig. 12. Syntax check results of element-based Z schema of MM-FMA metamodel

Figure 13 shows some of the results of model checking for 26 schemas (20 composition relationship, 4 association relationship, 2 dependency relationship) by examining the relationship-based Z specification. In Figure 13, the test result screen for "Has_SP_MC" (SPEM and MethodContent element) with a composition relationship is shown. "Rel_UCD_UC" (UseCaseDescription and UseCase element) below indicates the relationship. Here, the syntax and proof sections show a result of "Y", indicating that the Z specification is correct. Proof is proved by reduce.

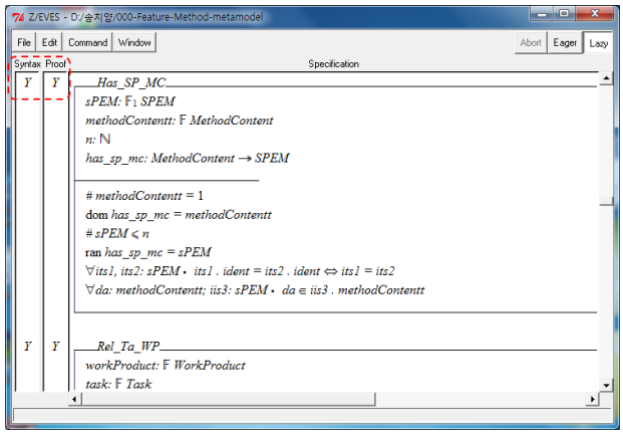


Fig. 13. Syntax check results of relationship-based Z schema of MM-FMA metamodel

Table 7: Comparison of DMF-MM and the proposed method

Evaluation Item	**DMF-MM** (Cho, 2015)	**Proposed Method**
SPEM-based methodological framework	- Support	- Support
Target methodology	- Object-oriented, CBD methodology	- Object-oriented, CBD methodology - Feature Methodology
Metamodel of work product-based methodology framework	- Support	- Support
How to create method classes and method components	- Not supported	- Support . Method Class: [Rule 1] . Method Component: [Rule 2]
Application examples of Method Class and Method Component	- Partial support - Hierarchy of Method - Class and Method Component	- Support . Hierarchy of Method Class and Method Component in Figure (6-7) . Structure and behavior model of Method Component in Figure (8-9)
Translation profiles between SPEM and methodologies	- Not supported	- Support (Table 2-4)

Table 8: Comparison of correlation between SPEM and Feature metamodel

Comparison metrics	SPEM structure	Proposed Feature metamodel structure
Paradigm	Task (behavior) oriented structure	Work product-oriented structure
Architecture	Method content package - Task definition - Work product definition - Role definition	- Method class . Work product by methodology technology . Independent Work-product - Method component . Independent unit of process organization
Modeling elements	- Task	- Feature, Class
Atomic reuse unit	- Task, Work-product, Role	- Method Feature, Class
Composite elements	- Method content	- Method component
Combination reuse unit	- Category	- Method component

Consequently, through this model checking, it was proven that there were no errors in the syntactic and semantic properties of the proposed feature methodology metamodel (MM-FMA). By specifying and inspecting the relationships and number of correspondences between classes, no mismatch errors between elements or conflicts between types occurred in the syntactic structure of the metamodel.

Evaluation

Previously, in order to systematically and flexibly structure various applications and evolving feature methodologies, the article addressed a metamodel of SPEM and work product-based feature methodology framework. As an application example, we designed the structure of the existing feature methodology by constructing method classes and method components using this metamodel.

Meanwhile, in order to create a standardized feature methodology, an activity specification within each phase was defined based on SPEM as shown in Table 5, and an application example was shown. Especially, existing studies have rarely studied the structuring of feature methodology architecture based on work products.

In this section, we compare and evaluate the structure of the presented feature methodology framework and the SPEM structure, and the structure of the existing methodology and the proposed feature methodology. Moreover, the characteristics and limitations of the presentation method are highlighted.

Table 9: Comparison of correlation between SPEM and Feature metamodel

Comparison items	Legacy Existing S/W development methodology	Proposed Framework (Modularity based)
Applicable target	- Development process and procedure focused	- A modular framework applied to development
Modularity	- Relatively low	- Very high
Reusability	- Limited to process and document reuse	- **Modular reusability** of methodology components
Scalability	- Somewhat limited in new requirements	- Easy extensibility of methodology component modules
Flexibility	- Many limitations in process changes	- Ability to flexibly adapt to changing technological environments
Quality improvement	Quality management through standardized procedures	Improvement of quality and maintainability of methodology through modular reuse
Ease of development methodology	- A lot of time and effort is required to develop a new methodology	- Ability to quickly develop a new methodology based on a framework

In Cho (2015), which is highly relevant to this paper, a metamodel of a SPEM-based development methodology framework was presented. A comparison of the presentation method with Cho (2015) is shown in **Table 7.** It did not consider feature methodology and did not support detailed methods for building a work product-based methodology framework. In other words, it did not cover how to create method classes and method components based on work products. This paper, which solves this problem, extends Cho (2015) to feature methodology. In addition, Cho (2015) did not present a conversion profile such as the "SPEM-based feature methodology conversion structure" section. Therefore, it is unclear how the elements of SPEM are mapped to Method Class, Method Component, etc. This may lead to inconsistencies when converting to a work product-centered structure according to the SPEM standard.

Structural Correlation between Proposed Method and SPEM

The proposed feature metamodel was defined based on the configuration criteria of SPEM 2.0, an international standard. As shown in Figure 6, they constructed "method classes and method components" (independent work products) according to the metamodel process based on the work product. By combining these, it is composed of "process components" (independent process units), "area processes" (patterns combined into process units), and "project execution processes" (project-specific specialized processes).

Meanwhile, **Table 8** shows a comparison between SPEM and feature metamodel in terms of structure and reuse. SPEM is a behavior-oriented structure and is divided into tasks (actions, activities), work products, and roles.

On the other hand, the distinguishing feature of this study is that it has a single method class structure centered on the integrated work product. The metamodel of the feature methodology accepts and satisfies all three components of SPEM. In Table 8, the sector related to this construction is a component of SPEM's "Method Content Package". These were expressed by mapping into "method class" and "method component" of the presentation method." That is, SPEM's tasks, work products, and roles are defined as work product-centered method classes in the proposed method, and from the perspective of the configuration model of the development process, method classes are recombined in component units. Figure 5 of the proposed metamodel shows how the three elements of SPEM are mapped to the work product-centric structure and the transformation profiles between them are shown in Tables 1 to 3. Examples of this are shown in Figure 8 and Figure 9.

Structural Evaluation of Existing and Proposed Methodologies

Table 9 is a comparative analysis of the existing methodology and the proposed framework, showing the advantages of structuring the methodology into a framework. Compared to the existing methodology, the proposed framework has good modularity, reusability, expandability, flexibility, and S/W methodology development. In particular, the proposed framework is useful when newly constructing or improving a methodology.

Table 10: Comparative evaluation of architecture between proposed methodologies compared to existing methodologies

Evaluation Metrics	Existing Feature Methodology Structure	Proposed Feature metamodel structure
Modularity	Methodology components and processes are not independently defined - Difficult to do process tailoring - The order of work products/ techniques / activities of the methodology is not defined independently from each other.	Methodology components and processes are independent - Easy tailoring of the process - Work product/technique/activity is defined as a single object and independence as work product - Clarification of structure through independent definitions between work products
Flexibility	It is difficult to structure the methodology as the activity, work-product, and role are separated.	Activity/Work product/Role is classed as a method based on Work product, making structuring easy.
Reusability	The structure of the feature methodology is not a reuse-based structure. - The structure is not based on combination elements between activity levels and work product levels.	The structure of the methodology is based on reuse. - Method class with minimal independent technical asset reuse - Composed of method components of application assets that reuse method classes

Table 10 shows a comparative evaluation of the problems of the existing behavior-centered methodology architecture and the advantages of the proposed work product-based methodology architecture. A metric for comparative evaluation of methodology architectures is evaluated in terms of modularity, flexibility, and reusability. Through the construction of a work product and object-based architecture model, the modularity of each work product in the methodology process can be strengthened and reusability can be improved. For example, regarding the modularity of existing feature methodologies, when creating multiple work products within a task, the steps related to the work products are intertwined. That is why this can cause side effect problems. Regarding reusability, the Method class or Method Component of the restructured feature methodology created by applying the presented framework can be reused to build or improve another IT company's feature methodology.

Characteristics and Limitations

The advantages of this paper are as follows:

- Structure of SPEM-based feature methodology framework
  The architecture of the feature methodology was structured by reflecting the SPEM schemes in Figure 1:
    Method Content, Capability Pattern, and Delivery process. Therefore, the feature methodology was improved to a standard and practical level.
- SPEM's method content elements, Work-product, Task, and Role, were defined in the method class to establish a feature methodology.
- The reflection of SPEM in the framework of the feature methodology is shown in Figure (4-7) and Table 5.
- Establishment of work product-based feature methodology framework
- Work product independence and reusability have been improved.
- Independent method classes were built around the work product, and these were reused to define method components and process components in Figure 5.
- Redundancy in multiple activities of the same work product within the methodology was eliminated.
- The modularity of the methodology makes it easier to convert structures with high scalability and reusability.
- Work product-centered project execution and quality control management becomes possible.
- Metamodel-based feature methodology framework design
- By applying the modeling elements and relationships of a standardized methodology metamodel, the structure of the methodology becomes clear and the structuring of methodology changes becomes easier in Figure 4 and Figure 5.
- An optimized methodology can be established according to the characteristics of the project.
- Establishment of architecture system of existing feature methodology
- Provided a blueprint of the feature methodology by Phase, Activity, Step, etc.
- Feature methodology becomes easier to understand, expand, and publish.

The **limitations** (disadvantages) of the presentation technique are as follows:

- Because the existing feature methodology was not specific, lower-level tasks and steps within the activities at the development phase could not be reflected in the metamodel of the feature methodology framework.
- Methodological web service through tailoring and publishing of the authored feature architecture model was not supported.
- It does not provide automated tools to expand and develop elements of emerging methodologies and support them.

Conclusion

The existing feature methodology is at a research level that does not support a practical and standardized methodology based on SPEM. A flexible work product-based feature methodology structuring method is needed to adapt to the evolution of various applications due to inter-industry convergence.

In this paper, a metamodel-based feature methodology framework was presented as a way to design the structure of the feature methodology framework. For this purpose, the architecture and conversion profile between SPEM and feature methodology were also defined. In particular, we presented a structuring method for the work product-based feature methodology framework, which is a SPEM element. In other words, method classes of independent technical assets were defined for each work product, and method components at the unit model level were created by combining them. By combining these, the general-purpose process component of the methodology was defined, and the Delivery Process creation was established by customizing the process component for each project (by IT company). As an application example, the creation of method class and method component was shown for the requirements definition phase of the feature methodology. In addition, the activity specification of the feature methodology was defined based on SPEM

elements. We showed how to define the "Requirements-based Feature Model Creation" activity in the requirements definition phase according to the activity specification of the feature methodology. The metamodel of the presented feature methodology was formally specified in Z and model checked to show that it was accurate.

As an expected effect, by using the metamodel process framework of the SPEM-based feature methodology, the structuring work of the feature methodology will become convenient. Constructing a work product-based methodology framework can solve the problem of work product duplication between activities. As a result, modular construction of the feature methodology becomes easier and the reusability of the created feature application model can be improved.

Future research requires a metamodel framework that accommodates the addition of Tasks and Steps within the activities of the phases to specify the feature methodology. In addition, it is necessary to provide web services using feature methodology through tailoring and publishing of authored architecture models.

Author's Contributions

Chee-Yang Song: Worked on most of the parts, introduction, design, and evaluation of the research method and to the writing of the manuscript.

Eun-Sook Cho: Contributed to the design of the research plan and content, and engaged in the literature view.

Acknowledgment

We thank the anonymous reviewers for their useful and valuable comments to make the paper more organized.

Funding Information

This research was supported by the Basic Research Program of the Education and Research Fund at Kyungpook National University."

Ethics

This manuscript is original and has not been published elsewhere. The corresponding author confirms that coauthors have review and approved the article and there are no ethical issues in the future.

References

Agh, H., & Ramsin, S. (2023). A model-driven approach for software process line engineering. *Software*, *2*(1), 21–70. https://doi.org/10.3390/software2010003

Bae, S. J., & Kang, K. C. (2013). A feature-based product configuration method for product line engineering. *Journal of Software Engineering Society*, *26*(2), 31–44.

Baumgarten, G., Rosinger, M., Todino, A., & Marín, R. J. (2015). SPEM 2.0 as process baseline meta-model for the development and optimization of complex embedded systems. In *Proceedings of the IEEE International Symposium on Systems Engineering (ISSE)* (pp. 1–6). IEEE. https://doi.org/10.1109/SysEng.2015.7302749

Bendraou, R., Gervais, M.-P., & Blanc, X. (2005). UML4SPM: A UML 2.0–based metamodel for software process modelling. In *Model Driven Engineering Languages and Systems (MODELS 2005)* (LNCS Vol. 3713, pp. 17–38). Springer. https://doi.org/10.1007/11557432_3

Bendraou, R., Jezéquel, J.-M., & Fleurey, F. (2009). Combining aspect and model-driven engineering approaches for software process modeling and execution. In *Proceedings of the International Conference on Software Process (ICSP 2009)* (LNCS Vol. 5543, pp. 148–160). Springer. https://doi.org/10.1007/978-3-642-01680-6_15

Bezerra, C., & Coutinho, E. (2023). Modeling software processes from different domains using SPEM and BPMN notations: An experience report of teaching software processes. *Journal of Software Engineering Research and Development*, *11*(1). https://doi.org/10.5753/jserd.2023.3186

Cho, E. S. (2015). Design of methodology framework based on meta-model. *Journal of the Korea Academia-Industrial Cooperation Society*, *16*(10), 6969–6976. https://doi.org/10.5762/KAIS.2015.16.10.6969

Cho, E. S., & Song, C. Y. (2022). Meta-model and formal specification for design of cloud-based IoT software. *The Society of Convergence Knowledge Transactions*, *10*(4), 121–132. https://doi.org/10.22716/sckt.2022.10.4.042

Combemale, B., Cregut, X., Caplain, A., & Coulette, B. (2006). Towards a rigorous process modeling with SPEM. In *Proceedings of the Eighth International Conference on Enterprise Information Systems (ICEIS)* (pp. 530–533). https://doi.org/10.5220/0002455205300533

Debnath, N., Riesco, D., Cota, J. B., Perez-Schofield, G., & Uva, D. R. M. (2006). Supporting the SPEM with a UML extended workflow metamodel. In *Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)* (pp. 1151–1154). https://doi.org/10.1109/AICCSA.2006.205234

Husen, J., Washizaki, H., Yoshioka, N., Tun, H., Fukazawa, Y., & Takeuchi, H. (2023). Metamodel-based multi-view modeling framework for machine learning systems. *Science and Technology Publications*. https://doi.org/10.5220/0011699600003402

Kim, G. B., Kang, K. C., Kim, K. S., Lee, J. J., Ko, E. M., & Kang, K. S. (2005). Process control system development using software product line engineering: A case study applied to a steel system. In *Proceedings of the Korea Conference on Software Engineering* (pp. 295–306).

Kim, K. S., & Kang, K. C. (2006). A feature-oriented approach to developing dynamically reconfigurable embedded software in product line engineering. *Journal of Computing Science and Engineering*, *24*(11), 59–71.

Kołcz, K. (2006). *Using SPEM/UML profile for specification of IS development processes* (Master's thesis). Sweden.

Lee, H. S., Lee, K. B., & Bang, H. C. (2016). Systematic development of mobile IoT device power management: Feature-based variability modeling and asset development. *Journal of the Korean Institute of Information Scientists and Engineers*, *43*(4), 460–469. https://doi.org/10.5626/JOK.2016.43.4.460

Lee, J. J., & Kang, K. C. (2002). Product line software development process. *Communications of the Korean Institute of Information Scientists and Engineers*, *20*(3), 23–30.

Lee, K. W. (2012). Managing and modeling variability of UML-based FORM architectures through feature-architecture mapping. *Journal of Korea Information Processing Society*, *19*(1), 81–94. https://doi.org/10.3745/KIPSTD.2012.19D.1.081

Morale, S. (2022). Software process modeling with SPEM. https://modeling-languages.com/process-software-modeling-spem

Object Management Group. (2008). *Software & systems process engineering meta-model (SPEM) specification* (Version 2.0). https://www.omg.org/spec/SPEM/2.0

Park, C. S., Lee, S. N., Kim, J. H., & Bak, Y. W. (2009). Software development methodology-based instruction process modeling. In *Proceedings of the Korean Institute of Information and Communication Sciences Conference* (pp. 822–826).

Park, J. S., Moon, M. K., Nam, T. W., & Yeom, K. H. (2008). A 4D process for service-oriented software development. *Journal of the Korean Institute of Information Scientists and Engineers*, *35*(11), 653–660.

Park, S. H., Choi, K. S., Yoon, K. A., & Bae, D. H. (2007). Deriving software process simulation model from SPEM-based software process model. In *Proceedings of the Asia-Pacific Software Engineering Conference (APSEC)* (pp. 382–389). https://doi.org/10.1109/APSEC.2007.28

Pillat, R. M., Oliveira, T. C., Alencar, P. S., & Cowan, D. D. (2015). BPMNt: A BPMN extension for specifying software process tailoring. *Information and Software Technology*, *57*, 95–115. https://doi.org/10.1016/j.infsof.2014.09.004

Saaltink, M. (1999). *The Z/EVES 2.0 user's guide* (Technical Report TR-99-5493-06A). ORA Canada.

Shroff, M., & France, R. B. (1997). Towards formalization of UML class structures in Z. In *Proceedings of COMPSAC '97* (pp. 11–15). IEEE. https://doi.org/10.1109/CMPSAC.1997.625087

Simeckova, L., Brada, P., & Picha, P. (2020). SPEM-based process anti-pattern models for detection in project data. In *Proceedings of the 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE. https://doi.org/10.1109/SEAA51224.2020.00024

Song, C. Y. (2003). *A metamodel-based modeling mechanism for hierarchical design in UML* (Doctoral dissertation). Korea University.

Song, C. Y., & Cho, E. S. (2022). A feature-based cloud modeling method. *Journal of Korean Knowledge Information Technology Society*, *17*(1), 101–120. https://doi.org/10.34163/jkits.2022.17.1.011

Song, C. Y., Cho, E. S., & Kim, C. J. (2011). An integrated GUI-business component modeling method for MDD- and MVC-based hierarchical designs. *International Journal of Software Engineering and Knowledge Engineering*, *21*(4), 1–44. https://doi.org/10.1142/S021819401100529

Sparx Systems. (2025). *Software & systems process engineering meta-model (SPEM)*. https://sparxsystems.com/resources/user-guides/17.1/large-print/model-domains/languages/spem.pdf