Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 9, No. 1, pp. 20-27, Junio 2025
Selección de los Mejores Artículos de CoNCISa 2024
https://doi.org/10.63870/revecom.2025.09.01.003

Customizing Software Development Methods: A Process Model Approach

Judith Barrios, Jonás Montilva
ijudith@ula.ve, jmontilva@gmail.com

Department of Computer Science, University of Los Andes, Mérida, Venezuela

Abstract: This article presents an instantiation process model for facilitating the understanding and customization activities of software development methods The process accepts as input a method represented by at least one process model that prescribes the set of software development activities. A method is also defined through a product model that prescribes the set of product parts that can be built following the process model and, a team model that prescribes the set of roles needed to execute the activities proposed in the process model. The proposal integrates the teaching, practical and consulting experience of the authors, which is essential to understand and handle usual difficulties found during the process of adapting software development methods. The main contribution of our proposal is to provide students of systems and software engineering with a global vision of a method and the know-how implicit in its process model description. Therefore, the proposal simplifies users understanding of method background concepts, and guides them whereas customizing it according to a particular software project context. The proposal is illustrated by an example of the White_Watch method customization to cope with a hypothetical software project situation.

Keywords: Software Methods; Instantiation Process; Method Customization; Teaching Practice.

I. Introduction

Software development methods (SDM) intend to assist project leaders and the whole development team along a development project. A SDM is typically used as a guide to establish main activities as well as technical products that the project team needs to complete to produce a software application. Nevertheless, while following the prescribed guidelines of a method, the project leader -software engineer- has to take instant decisions about the project plan workflow by considering, among other variables, the software product complexity and characteristics, the specific product dynamic requirements, the set of restrictions and needs coming from the current working environment including team size and experience, the developing tools capacities and the essentials of the programming languages, project schedule, and user participation.

In general, a SDM needs some adjustments before being used as a development guide. These adjustments serve to define the project's preliminary schedule and to organize the work that the team must do. Nevertheless, there are still some eventual context factors that may disturb project workflow like technological infrastructure, financial, and human resources complications. A project leader, therefore, needs to frequently adapt the method development process workflow. This problem of tuning a method gets bigger if the project leader/software engineer´s background in method understanding and practice is not deep enough. This is the main problem encountered during teaching activities, the students just begin to understand a method when they must already adapt it to solve a concrete problem. The main motivation of this proposal is, therefore, to assist students to customize a method that they do not completely understand. For that reason, this method instantiation process can be assumed as a teaching practice because it is effective for introducing systems and software engineering students into the context of understanding methods and then adapting them properly. Nevertheless, the proposition may also help software and systems professionals to understand and customize methods and other methodological guidelines.

The process of fine-tuning or customizing a method is called an Instantiation Process (IP). Generally, an instantiation process is done over the set of general concepts prescribed by a method model or just by a broad methodological textual description – tables or a set of steps or phases. A general method model is prescribed, at a high abstraction level, by a comprehensive set of concepts and their relationships which are involved in the process of developing a particular product or service [1].

In this article, we propose an instantiation process (IP) model for properly guiding the customization of a SDM. It means that the IP accepts a SDM represented by at least a process model, which is one of the three formal method description models. The other two are the product model which prescribes the set of product parts that can be built by following method process guidelines, and the team model which prescribes the set of roles and responsibilities that developers need to take for executing the activities prescribed as process guidelines [1][2]. This is a generic proposition; thus, it can also be instantiated to couple with methods that only have one or two of the method models mentioned. The instantiation process model is illustrated with the customization of a SDM example that describes how to adapt the method for fitting a particular project situation of a hypothetical example.

The proposal integrates the teaching, practical and consulting experience of the authors as the basis of a good understanding of the problems faced by students and professionals during the adaptation of methods. The feedback gained from teaching system engineering students to customize methods to a specific software project and product factors has been essential. The IP model proposal is completed by ways of working derived from specific methods adaptations and characterizations completed in many Venezuelan software organizations.

The main contribution of our proposal is to help systems and software engineering students with method understanding and its customization; especially, the concerns related to a software development method adaptation for satisfying specific project needs. In addition, and considering that the SDM already customized may be applied over and over in similar project contexts, students can enhance their ways of working which are perceived as the enhancement of the quality of their development processes as well as that of the software products or services elaborated.

The article is structured as follows: the next section presents the problem of customizing SDM along with the review of some related works. Section III presents a summary of method engineering and business process model background concepts. Section IV describes, at two levels of detail, the instantiation process model proposed; first, a general contextual level and then, the corresponding detailed workflow level for describing the main activities included in the general process model. Section V shows how to apply the proposed process by instantiating the White_Watch method in a hypothetical software project situation. Section VI concludes the paper and gives some practical recommendations to take advantage of the use of the instantiation process model proposed.

II. Problem and Related Work

Use this template to prepare your camera ready paper. In this template, all margins, column widths, line spaces, and text fonts are prescribed; please do not alter them.

A. Interpretations of the Problem

The problem of adapting methods is not new, and it is, generally, neglected by software organizations and project leaders since they prefer to solve method adapting problems incidentally [3]. Nevertheless, in preventing to take rush decisions related to tailoring the SDM throughout the execution of a development project, many software organizations define and institutionalize their own development methods to swift project performance and to adequately assure that a high-quality development process shall produce a high-quality software product. But the problem of adapting methods still causes difficulties that disturb project plan work breakdown structure, timelines, costs, and final product or service quality.

The software engineering community early began to work on method selection and its adapting problem. For instance, at the beginning it was identified as a problem of programmers and the knowledge they had of the development cycle along with their capacities to define what to do, when to do it, and how much effort they needed to complete a prescribed task; so, they could organize their work and, consequently, help project leaders to estimate the required work of the whole team. These were the personal software process model (PSP) and TSP [4][5]. Afterward, the SEI proposal of the CMMI model expected to help organizations and software engineers to understand the whole software development process and, organize team development work by approving a predefined set of business development processes. Accordingly, an organization defines the complete set of software processes that ought to be installed, executed, measured, and monitored to improve organizational performance as well as the corresponding quality of the production process and its products and services [6].

Nevertheless, another customizing problem emerges when a software organization defines or selects, as mandatory, a particular SDM or decides to define and install a set of organizational software processes for elaborating and managing demanded software products. That is, the project leader and the work team must plan and execute each one of the prescribed activities to produce each prescribed technical and management product or document for realizing later, at the end of a project, that some of these activities were not useful because they do not positively contribute to obtaining the final product or service and its documentation, as required by the client. In consequence, there are a lot of time and effort lost, many expenses, and any added value either for the client or for the software organization itself. Many related proposals for software processes assessment and improvement searched to enhance and elevate installed software process performance and quality like SCAMPI, SPICE, ISO 9000 [7][13].

Some other propositions tackled the problem of adapting and defining the required product development activities by assembling a set of generic ones with or without modifications. This kind of solution was triggered by, among main issues, the software development cycle, the product type, the project management schedules according to pertinent project situational or practical variants, etc. The WATCH suite of methods is a good example of this kind of method proposal [8][9] as well as the Crystal method family [10]. These propositions preconize a set of guidelines arranged to manage some of the method adapting problems; as a result, a set of method variants is obtained where they can be selected according to the predefined set of project and product features. Other approaches extend generic activities with practical strategies like RUP (Rational Unified Process) and its agile version RUP agile [11], the ASD (Adaptive Software Development) that repeat iteratively an adapted development cycle as a practical strategy to accomplish software product dynamic or uncompleted requirements [12], [13]; and, the SCRUM project management practical approach that has been used as an agile way for organizing the development work for speedy build functional software products [14].

The SEMAT approach extends the range of the approaches reviewed. The OMG Essence standard has been published as the kernel for software engineering methods. It recommends and organizes, in a generic model, a reduced set of essential elements or concepts which are associated with any software production process. This model may well be applied by the development team to define the practical work to be done along with the things to be produced and manipulated in a particular development project [15]. There are many related works where practitioners apply and extend the essentials to cope with development process issues [16-18].

Lastly, we add to this variety of methodological research works and practical approaches, the huge number of specialized SDM that have been proposed to help, assist, and guide the development process according to, among other settings:

- the type of product and its complexity,
- the team size, and its experience,
- the tools available and their capacity,
- the time to have a functional version,
- the dynamic change of product requirements, and
- the development technology available.

Some of these propositions recommend their method by including a list of product characteristics, time to have a functional product, team size, and project typical situations where their methods have proved to be effective [19-22]. The work presented in [23] addresses the methods configuration problem but only for the agile development approach.

None of these methodological propositions and strategies explicitly include a user guide, method use guidelines, or a set of tips to better adjust or customize their proposals. Knowledge level and understanding of a SDM or approach as well as sufficient software development experience seem to play a relevant role in the selection and, subsequently, customization of a SDM to fit the project and contextual factors. The literature review presented in [24] discusses the types of reasons behind software engineers' decisions to select and use software development methods – adoption of methods. It includes discussion about method deviations and method operationalization problems where a solid understanding of SDM background concepts along with the awareness of why and how to implement some of the method activities, seems to be a significant factor to avoid method misconception, and consequently, its misalignment to software project intentions.

We also found that many of these methodological and strategical offers are supported by social network broadcasts and an experienced set of users, programmers, and developers who communicate their practices and recommend effective ways of working like [25] and [26].

B. The Problem from Teaching Experience Perspective

To complete the description of the problem, we take some examples from our teaching experience in systems and software engineering. For instance, while working on class projects with our students, we notice that central difficulties of method usage were related to the know-how to apply a method. That is, first, students exposed problems to select, and then, how to execute a prescribed guideline/activity or to choose one of them instead of another among the set of the proposed ones. It is not easy for the students to discern if a particular action is necessary (ought to do it), if it is optional or not required considering the type of product/service or the development tool available, or a particular management exigence of the client/teacher; similar difficulty with an earlier configuration of technical products parts that are needed to complete the final software product or service. A concern case is when a student or a project course team decides to follow each one of the prescribed guidelines to produce each one of the prescribed products parts or documents ("because the method is explicit so we have to do it") for realizing later, at the end of the project, that a lot of the diagrams or executed actions were not necessary because they were redundant or do not contribute at all to the final product/service required.

As observed, the problem of customizing a SDM disrupts software engineering work from many perspectives. It is an individual problem, a team problem, a project leader problem, and a business process problem. We may conclude that it is not a method adapting problem but a process domain knowledge problem or a deficient practical experience. It may be true but not completely because the process of learning requires a well-founded background to have an understanding and a comprehensive picture of a method before applying it.

That is one of the reasons why this method instantiation process model was shaped: to assist method understanding and comprehension before instantiating it. From the business process perspective, this problem needs to be solved at the institutional level by defining and installing the required and flexible set of software processes by using a specific process improvement approach or standard as mentioned earlier in this section, but this discussion is out of the scope of this proposal.

We are convinced[1] that a generic instantiation process model, like our proposal, may contribute to facilitate the software engineering task of adjusting a SDM to project context situations, product/service characteristics, and team size and experience issues. It means that before starting a development project, the student, the project leader, or the system/software engineer in charge, may analyze and take decisions about the final product or service, its product parts, and documents that must be produced; besides, what would be the workflow that best fit project context situation and team members' knowledge, skills, experiences, and competencies. Method instantiation process model guidelines assist not only to adjust a method to a particular scenario but to understand what is prescribed by the method before adapting it. This premise persists and is independent of the SDM approach, i.e., if it is disciplined, balanced, or agile.

III. Background Concepts

Software methods are formally defined as a set of cohesive and complementary models that represent the software final product and its partial components, the software process/activities which need to be executed to produce each part or component of the product/service, and the competencies and understanding that each member of the software team must attest to, adequately, execute each prescribed activity, and assure that the product part elaborated has the expected quality. These models are the product model, the process model, and the team model, respectively [1][2].

The process of fine-tuning or customizing a method is called an instantiation process (IP). Generally, this kind of process is done over a set of generic concepts prescribed by a method model. This method model is general and is placed at a high abstraction level. It means that a software development model prescribes the complete set of concepts and their relationships which are

[1] After applying several times some others associated learning strategies

involved in the method development process of a software product/service. Some of them may as well include the roles that the members of a software team have to play throughout a development project.

The process of instantiating a method implies the selection, extension, reduction, or modification of any of the concepts included in the general model. In the case of a SDM represented by a process model, a product model, and a team model, the instantiation process ought to be done coherently and consistently over each one of the generic models so the relationships and dependencies between concepts and models may be correctly kept. After an instantiation process, the customized method model has a step lower abstraction level than the general one.

Figure 1 shows the links between the generic method model and the customized model obtained after an instantiation process.

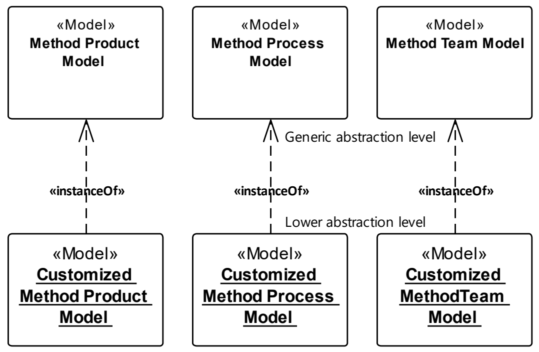
«Model» Method Product Model
«Model» Method Process Model
«Model» Method Team Model
Generic abstraction level
«instanceOf»
«instanceOf»
«instanceOf»
Lower abstraction level
«Model» Customized Method Product Model
«Model» Customized Method Process Model
«Model» Customized MethodTeam Model

Figure 1: Links between Generic and Instantiated Method Models

A method product model represents the set of product parts and the relationships among them which can be elaborated by applying a particular method process guideline. In the case of a SDM, a method model must include the set of partial technical and management product parts as well as the deliverable ones. Figures 2, 3, and 4 present, at a very high abstraction level, a set of generic method concepts that may be instantiated to build a particular SDM product model. These concepts can be instantiated to fit specific method requirements [8][9].

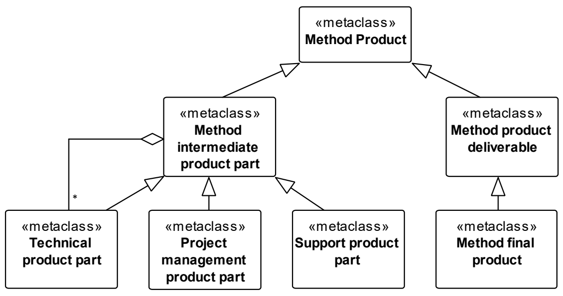
«metaclass» Method Product
«metaclass» Method intermediate product part
«metaclass» Method product deliverable
*
«metaclass» Technical product part
«metaclass» Project management product part
«metaclass» Support product part
«metaclass» Method final product

Figure 2: Generic Product Model Concepts. Adapted from [8]

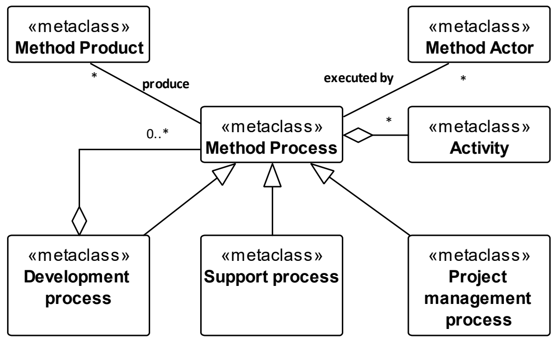
«metaclass» Method Product
«metaclass» Method Actor
*
produce
executed by
*
0..*
«metaclass» Method Process
*
«metaclass» Activity
«metaclass» Development process
«metaclass» Support process
«metaclass» Project management process

Figure 3: Generic Process Model Concepts. Adapted from [8]

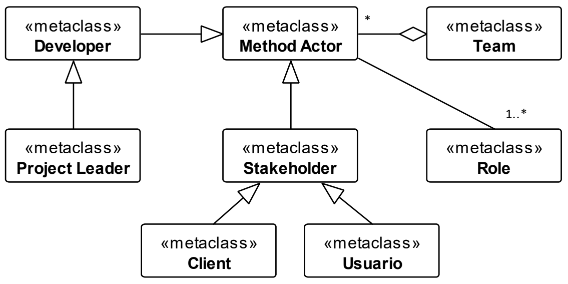

Figure 4: Generic Team Model Concepts. Adapted from [8]

As model concepts are represented at a high abstraction level, they could be chosen and extended for being part of a particular SDM. These meta-models were used to elaborate the Blue, Yellow, and White variants of the WATCH method suite [9].

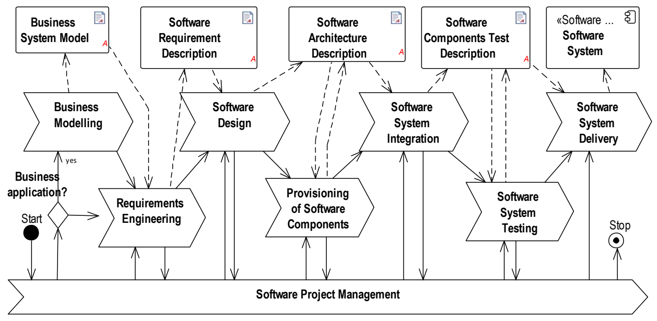
Business System Model
Software Requirement Description
Software Architecture Description
Software Components Test Description
«Software ... Software System
Business Modelling
Software Design
Software System Integration
Software System Delivery
yes
Business application?
Start
Requirements Engineering
Provisioning of Software Components
Software System Testing
Stop
Software Project Management

Figure 5: Workflow of the White_Watch Method [27]

Figure 5 presents an example of the results of the instantiation process of the workflow of the White_Watch method [27]. This version of the WATCH suite was defined for assisting software engineering students in their course projects. Notice that product and process generic concepts (Figures 2 and 3) were extended to have a more specific description of the technical products and their corresponding building processes. In this case, a software product is built as an assembled set of reused software components. The intermediate technical products of Figure 2 were extended with a business system model, documents for expressing software requirements, architecture, and testing descriptions. These technical products are represented by using explicit software engineering techniques and UML diagrams where applicable.

A. Process Model

The IP model proposed is graphically represented by a business process model using the UML Business notation [28]. Accordingly, a business process explicitly has a process goal to aim, is executed and supervised by actors (and their roles), has a set of inputs that may be transformed into outputs, generates other outputs, and is regulated by some precise procedures, standards, and rules, and is supported by some business resources (technology, money, documents, etc.). A business process may be complex or simple so it can be decomposed and detailed (into activities and tasks) according to modeling description requirements. A graphical representation like business process models is easy to understand and follow and provides students and engineers with a complete perspective of what they must do for customizing a software development method [29] and [30].

For that reason, the IP model is represented by a general process description diagram and a process decomposition diagram that shows the set of four sub-processes of the general process. Each one of the sub-processes is detailed by using a UML activity diagram.

IV. The Instantiation Process Model

As we explained earlier, an instantiation process implies the selection, extension, reduction, or modification of any method element included in the general method model. This instantiation aims to generate a particular version of the method. Besides, in the case of a SDM represented by a process model, a product model, and a team model, the instantiation process ought to be done coherently and consistently over each one of the general models, so the relationships and dependencies between concepts and models can be respected.

According to the research works [8] and [9], method guidelines suggest starting the instantiation process by first customizing the method product model determining what is going to be produced by the adapted method; then, the selection of the process model elements that describe what must be done to produce the product model elements already selected. The instantiation process ends by defining the actors and their roles that are required for executing those processes/activities expressed by the process model already instantiated. Afterward, a validation process is necessary to assure that the resulting method models are coherent among them. According to these process model guidelines, the student/project leader has by now shaped a set of method models to a particular software project scenario.

Figure 6 shows the Instantiation Process description by using a general-level UML Business diagram [28]. As described in the diagram, the IP is modeled as a business process whose main goal is “to adapt method models according to project and product situational factors”. The IP process accepts general Method Models (Product, Process, and Team) and after processes guidelines, it produces the set of corresponding customized method models. The process is under the responsibility of the project leader (actor) and it may be supervised by a software engineer. There are some rules, standards, and restrictions related to method application, organizational or business domain, and other method features that must be considered while customizing method models. Project documents and other items like initial product requirements, organizational context, and some relevant technological parameters support decisions related to the selection of model elements. Similarly, if there are any required addition or modification to complete general method model customization.

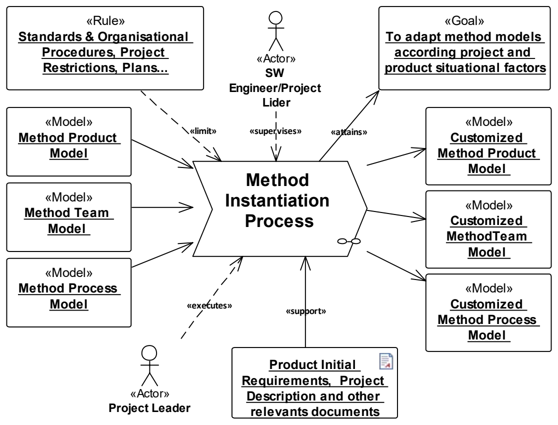
«Rule»
Standards & Organisational Procedures, Project Restrictions, Plans...
«Actor»
SW Engineer/Project Lider
«Goal»
To adapt method models according project and product situational factors
«Model»
Method Product Model
«Model»
Method Team Model
«Model»
Method Process Model
«limit»
«supervises»
«attains»
Method Instantiation Process
«Model»
Customized Method Product Model
«Model»
Customized MethodTeam Model
«Model»
Customized Method Process Model
«executes»
«support»
«Actor»
Project Leader
Product Initial Requirements, Project Description and other relevants documents

Figure 6: General Description Diagram for the Instantiation Process (IP)

Considering that the IP is a complex process it has been decomposed into four sub-processes as represented in Figure 7.

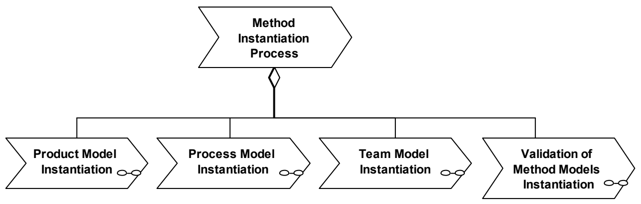
Method Instantiation Process
Product Model Instantiation
Process Model Instantiation
Team Model Instantiation
Validation of Method Models Instantiation

Figure 7: Sub-processes that Compose the IP

To precisely prescribe IP model guidelines, the detailed workflow for each one of the four sub-processes depicted in Figure 7 is presented in Figures 8, 9, 10, and 11, respectively.

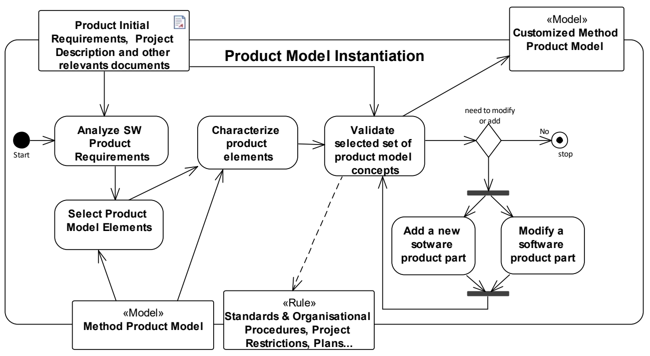
Product Initial Requirements, Project Description and other relevants documents
Product Model Instantiation
«Model»
Customized Method Product Model
Start
Analyze SW Product Requirements
Characterize product elements
Validate selected set of product model concepts
need to modify or add
No
stop
Select Product Model Elements
Add a new sotware product part
Modify a software product part
«Model»
Method Product Model
«Rule»
Standards & Organisational Procedures, Project Restrictions, Plans...

Figure 8: Product Model Instantiation

According to Figure 8, for customizing a method product model, it is necessary to analyze the initial software product requirements document as well as consider project contextual and technological aspects that may influence the selection and characterization of a set of product parts to be produced as represented in the workflow of the figure. It may be necessary to add or modify by extension or reduction one or more product parts of the customized product model. The IP of the method product model ends after a well-structured validation of the whole set of method product elements.

Once the method product model is customized, the IP continues with the method process model to select the required set of processes/activities that are required for building each one of the product model elements included in the customized product model. If there are some new product parts or some of them have been modified, the corresponding set of method process model elements should be defined or redefined as appropriate. It is possible that the general method process model does not have all the prescribed processes, i.e., maybe it just prescribes the development processes but no the support or the project management processes. Therefore, the IP is done only on the available process model elements as represented in the activity diagram of Figure 9. It is important to have in mind that any modification to current method models must be properly described by using the same formality and/or notation that in the SDM general model.

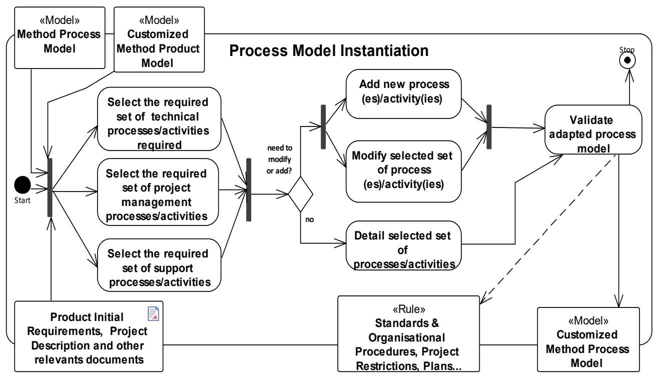
«Model» Method Process Model
«Model» Customized Method Product Model
Process Model Instantiation
Stop
Start
Select the required set of technical processes/activities required
Select the required set of project management processes/activities
Select the required set of support processes/activities
need to modify or add?
no
Add new process (es)/activity(ies)
Modify selected set of process (es)/activity(ies)
Detail selected set of processes/activities
Validate adapted process model
Product Initial Requirements, Project Description and other relevants documents
«Rule» Standards & Organisational Procedures, Project Restrictions, Plans...
«Model» Customized Method Process Model

Figure 9: Process Model Instantiation

If a method team model exists in the general SDM models, it is instantiated by using as inputs the already customized method process model. This process consists of identifying and characterizing actors, roles, and responsibilities required for the execution of processes/activities included in the method process customized model as it is represented in Figure 10.

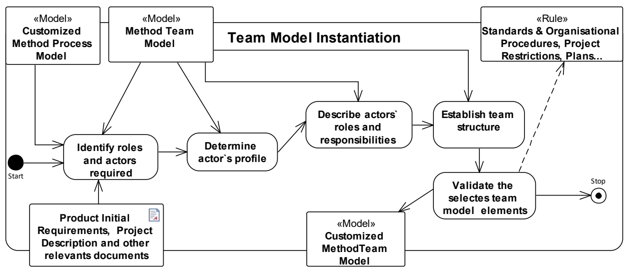
«Model» Customized Method Process Model
«Model» Method Team Model
Team Model Instantiation
«Rule» Standards & Organisational Procedures, Project Restrictions, Plans...
Start
Identify roles and actors required
Determine actor´s profile
Describe actors´ roles and responsibilities
Establish team structure
Validate the selectes team model elements
Stop
Product Initial Requirements, Project Description and other relevants documents
«Model» Customized MethodTeam Model

Figure 10: Team Model Instantiation

Finally, a global validation process is done to assure that the set of customized models is coherent and consistent with what is expressed in the SDM general models, and with the project and technological factors delimited by the official project documents, procedures, rules, and restrictions (see Figure 11). For example, in a SDM expressed by the three method models, the process model must use, produce, or complete a product part included in the product model. In the same way, each element included in the process model must be described or assigned as a responsibility of an actor´s role from the team model.

As we mentioned, the proposal is a generic process model, thus it is adjusted to different kinds of method descriptions and formalities. As mentioned before, the proposed IP requires, as input, at least a process model. Most methodological descriptions are represented textually through tables or as a list of steps or phases. This type of representation express activities that are performed to produce a software product (partial or complete). Therefore, the product model is implicitly included in such activities; it is the task of the project leader to extract the products involved, and to determine what part of the product they may represent and whether or not they form part of the deliverable product. The general IP workflow is showed in Figure 12.

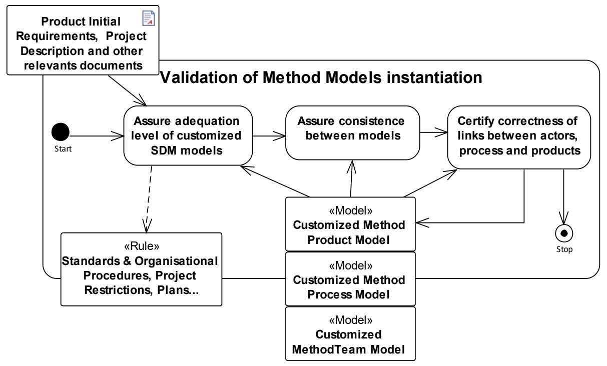
Product Initial Requirements, Project Description and other relevants documents
Validation of Method Models instantiation
Start
Assure adequation level of customized SDM models
Assure consistence between models
Certify correctness of links between actors, process and products
«Model» Customized Method Product Model
«Model» Customized Method Process Model
«Model» Customized MethodTeam Model
«Rule» Standards & Organisational Procedures, Project Restrictions, Plans...
Stop

Figure 11: Validation of the Customized Set of Method Models

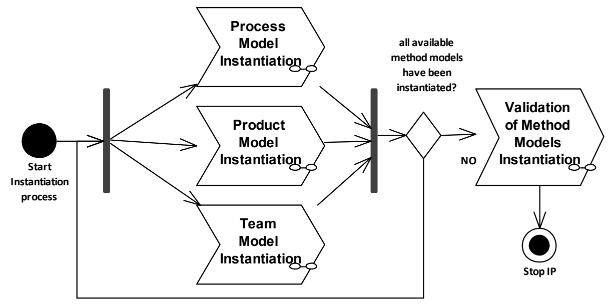
Start Instantiation process
Process Model Instantiation
Product Model Instantiation
Team Model Instantiation
all available method models have been instantiated?
NO
Validation of Method Models Instantiation
Stop IP

Figure 12: Instantiation Process General Workflow

V. Applying the Instantiation Process Model

To illustrate the benefits of our proposal, we present the customization of the White_Watch method [27] for a hypothetical course project where the students work in a team of two software engineers.

A. Hypothetical Project Description

The first part of a software engineering course project consists in detailing the set of business processes involved in an online booking system for a small theater. These processes are part of the organizational or business system model of the small theater; thus, it is necessary to define what are the activities, actors, events, resources and objects involved in the process of booking seats in the plays offered.

B. IP Workflow for Adapting the White_Watch Method

The White_Watch method workflow showed in Figure 5 is detailed through a descriptive table that organize those processes into steps. Each step has a set of prescribed activities along with the set of techniques or notations suggested to elaborate the involved products. In view of that, students should adapt the method from the table description which has four columns: method steps, activities, notations/techniques and products. Table I shows the Business System modeling step of the White_Watch method.

Initially, it is important to state that, according to Figure 6, both students, alternatively, ought to play the role of project leader during the personalization of the process prescribed in the IP model.

Table I: Excerpt of White_Watch Method Model [27]

Steps	Activities	Notations/Techniques	Products
Business System (BS) Modeling	-Modeling BS value chain (if needed) -Modeling fundamental processes -Modeling Support processes -Modeling process´ activities	-Interview with BS and domain experts -Direct observation of BS context -Review of technical documentation -UML Business value chain diagram -UML Business process description diagram -UML activity diagram	-Value Chain -Hierarchy of processes -Descriptions of Processes -Diagrams of activities

The relationship between product and process models is explicit and direct. Students have to select and instantiate those products, steps, and activities considering the above mentioned course project requirements; and then they can select the suggested technique or notation to produce them. IP guidelines suggest to start the instantiation process by the product model, then the process model, and finally, if required, the team model.

The outcome of the IP execution should be:

- Product model required (from Table I column "products"): one or more business processes descriptions and the corresponding set of activity diagrams.
- Process Model instantiation (Table I column "activities"): modeling fundamental processes (for the booking process) and modeling process activities (for detailing the booking process). These activities should be performed by applying techniques and notations selected from the "Notations/ Techniques" column. In that case, after direct observation of the business context, it is necessary to represent perceived processes and activities by using UML business process description diagrams and UML activity diagrams.
- The adaptation of the Team Model seeks to define the roles that each of the two students has to play to execute the process model already instantiated. The actor roles prescribed by the method are "project leader", "analyst", "designer", "programmer" and "tester". In this case and as part of the teaching strategy, both students must play the role of analyst at some point in the modeling process.

Table II presents the customization of the White_Watch method after applying the IP to the example introduced above.

Table II: White_Watch Method Model Customized for the Example

Steps	Activities	Notations/techniques	Products
Business System (BS) Modeling	-Modeling fundamental processes -Modeling process´ activities	-Direct observation of BS context -UML Business process description diagram -UML activity diagram	-Descriptions of Processes -Diagrams of activities

VI. CONCLUSIONS

Throughout this article we presented an IP model as a teaching strategy to help students while personalizing a method they do not fully understand. The proposed model provides students with a more complete view of the components of the method and their dependency relationships, which are essential to adapt a method to the situation of a particular project. In fact, it allows to better apprehend what is the purpose of a method and how it is structured, the kind of product that can be built through the execution of a pertinent set of processes and activities, and to detect what are the competencies that developers need to have to properly implement prescribed activities.

As explained, the problem of adapting methods has many relative solutions; some of them are oriented to the implementation of method variants that partially solve the problem of choosing a suitable SDM according to some predefined factors. Others, characterize the methods according to their development approach to then suggest how to select a method from the collection. Our proposal is generic and is based on method engineering concepts so it can be adjusted to different types of method descriptions and formalities, independently of any approach, paradigm and type of method. For instance, it equally works for disciplined, balanced or agile development methods.

This proposal has proven to be an effective teaching practice for introducing system and software engineering students into the conceptual context of the development methods and their adaptation. The IP model may also assist systems professionals to understand and customize other methods and methodological guidelines than SDM.

REFERENCES

[1] S. Brinkkemper, *Method Engineering: Engineering of Information Systems Development Methods and Tools*. Information and Software Technology, vol. 38, pp. 275-280, 1996.

[2] J.J. Odell, *A Primer to Method Engineering*. INFOSYS: The electronic newsletter for information systems, vol. 3, no. 19, 1996.

[3] D. Rivero, J. Montilva, J. Barrios, and M. Murúa, *Un Análisis del Desarrollo de Software en Empresas Venezolanas*. Seventh LACCEI Latin American and Caribbean Conference for Engineering and Technology- LACCEI 2009. San Cristobal, Venezuela, pp. WE1: 1-10, 2009.

[4] W. Humphrey, *Introduction to the Personal Software Process*, Addison-Wesley, 1997.

[5] W. Humphrey, *Introduction to the Team Software Process*, Addison-Wesley, 2000.

[6] SEI, *CMMI for Development, Version 1.3*. Software Engineering Institute. Carnegie Mellon University, USA, Technical Report No.CMU/SEI-2010-TR-033, 2010.

[7] M. de la Villa, M. Ruiz, and I. Ramos, *Un Estudio Crítico Comparativo de ISO 9001, CMMI e ISO 15504,* CISTI 2006, ISBN: 978-989-20-0271-2, vol. II, pp. 551-573, 2006, https://www.researchgate.net/publication/235661307_Un_estudio_critico_comparativo_de_ISO_9001_CMMI_e_ISO_15504.

[8] J. Barrios, J. Montilva, and D. Rivero, *The WATCH Method Suite in Practice: Two Complementary Perspectives of Use*, Conference CLEI 2011, Quito, Ecuador, Octubre, 2011.

[9] J. Barrios and J. Montilva, *Watch: A Suite of Methods for Facilitating Software Development Process Adaptability* in Software Engineering: Methods, Modeling, and Teaching, Sello Editorial Universidad de Medellin, Colombia, 2011.

[10] A. Cockburn, *Crystal Clear*, Addison-Wesley, 2005.

[11] S. Ambler, *The Agile Unified Process (AUP)*, 2006, http://www.ambysoft.com/unifiedprocess/agileUP.html.

[12] J. Highsmith, *Adaptive Software Development: An Evolutionary Approach to Managing Complex Systems*, Dorset House Publishing, 2000.

[13] R. Pressman, *Software Engineering a Practitioner's Approach,* Seventh Edition, McGraw-Hill, 2010.

[14] K. Schwaber and J. Sutherland, *The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game*, Scrum.org, October, 2017, https://www.scrumalliance.org/learn-about-scrum/the-scrum-guide.

[15] I. Jacobson, Ng Pan-Wei, P. McMahon, I. Spence, and S. Lidman, *The Essence of Software Engineering: The SEMAT Kernel*, Communications of the ACM, vol. 55, no. 12, pp. 42-49, December 2012.

[16] OMG, *Essence Kernel and Language for Software Engineering Methods Version 1.0*, November 2014, http://www.omg.org/spec/Essence/1.0/.

[17] V. Savić and E. Varga, *Extending the SEMAT Kernel with the TDD practice - IET Software,* Wiley Online Library, 2018, https://doi.org/10.1049/iet-sen.2016.0305.

[18] J. S. Park, *Essence-Based, Goal-Driven Adaptive Software Engineering*, IEEE/ACM 4th SEMAT Workshop on a General Theory of Software Engineering, 2015, pp. 33-38, DOI: 10.1109/GTSE.2015.12.

[19] J. Montilva and J. Barrios, *Ingeniería del Software: Un enfoque Basado en Procesos*, Sello Editorial del Vicerrectorado Académico, Universidad de los Andes, Venezuela, August, 2021, Published by Amazon Books, https://www.amazon.com/-/es/Jon%C3%A1s-Montilva-C/dp/9801120290.

[20] J. Barrios and J. Montilva, *A Balanced and Adaptable Method for Software Development in very Small Enterprises: The Blue Watch Variant*, in Carlos M. Zapata et al. (eds), Software Engineering: Methods, Modeling and Teaching: pp. 39-54, Medellin, Sello Editorial, 2011.

[21] J. Barrios and J. Montilva, *Gestión Ágil de Proyectos a Través de la Integración de Blue-WATCH y SCRUM*, X Ibero-American Conference on Software Engineering and Knowledge Engineering – JIISIC 2013, University of Medellín, Medellín, Colombia, November, 2013.

[22] M. A. Khan, A. Parveen, and M. Sadiq, *A Method for the Selection of Software Development Life Cycle Models Using Analytic Hierarchy Process*, International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014, pp. 534-540, DOI: 10.1109/ICICICT.2014.6781338.

[23] D. Gupta and R. Dwivedi, *Configurable Method Model of Agile Methods for Creating Project-Specific Methods*, International Conference on Software Eng. Research and Practice, SERP'16, ISBN: 1-60132-446-4, CSREA Press, 2016.

[24] T. Havstorm and F. Karlsson, *Software Developers Reasoning Behind Adoption and Use of Software Development Methods – A Sytematic Literature Review*, in International Journal of Information Systems and Project Management, vol. 11, no. 2, article 4, 2023, https://aisel.aisnet.org/ijispm/vol11/iss2/4.

[25] Toptal Engineering Blog, https://www.toptal.com/developers/blog.

[26] SoVisionIT, *Software and APP Development*, https://www.sovisionit.com/software-app-development/interesting-stuff/blog/.

[27] J. Barrios and J. Montilva, *Método W_Watch*, Revised version 2020. Internal Report, Group GIDyC, School of Systems Engineering, University of Los Andes, Mérida, Venezuela, 2020, partially published in J. Montilva & J. Barrios [21].

[28] H. Eriksson and M. Penker, *Business Modeling with UML: Business Patterns at Work*, John Wiley & Sons, 2000.

[29] K. Figl, *Comprehension of Procedural Visual Business Process Models: A Literature Review*, Business and Information Systems Engineering, vol. 59, no. 1, February 2017, pp. 41–67.

[30] E. Kornyshova and J. Barrios, *Visual Representation of the TOGAF Requirements Management Process*, ER Workshops 2225, October 2018, LNCS, vol. 11158, pp. 239-248.