International Review on Computers and Software (I.RE.CO.S.), Vol. 10, N. 7
ISSN 1828-6003 July 2015

SP2MN: a Software Process Meta-Modeling Language

H. Khdair, Z. A. Othman

Abstract – *In the last two decades, software process modeling has been an area of interest within both academia and industry. Software process modeling aims at defining and representing software processes in the form of models. A software process model represents the medium that allows better understanding, management and control of the software process. Software process metamodeling rather, provides standard metamodels which enable the defining of customized software process models for a specific project in hand by instantiation. Several software process modeling/meta-modeling languages have been introduced to formalize software process models. Nonetheless, none of them has managed to introduce a compatible yet precise language to include all necessary concepts and information for software process modeling. This paper presents Software Process Meta-Modeling and Notation (SP2MN); a meta-modeling language that provides simple and expressive graphical notations for the aim of software process modeling. SP2MN has been evaluated based upon the well-known ISPW-6 process example, a standard benchmark problem for software process modeling. SP2MN has proved that it presents a valid and expressive software process modeling language.*

Keywords: *Software Process Model, Software Process Metamodel, Software Process Modeling/Meta-Modeling, Software Process Modeling Language*

I. Introduction

The software process is a critical factor for delivering quality software systems, as it aims to manage and transform the user need into a software product that meets this need. The software process defines the way in which software system development is conducted and supported. A software process is a partially ordered set of activities undertaken to manage, develop and maintain software systems. In order to manage the software process, a wide range of software process engineering solutions have been proposed for that aim [1]-[50].

Software process modeling presents the most notable software process engineering paradigm that aims at enhancing the usefulness of the software process. As the name refers, it is the act of defining explicit software process models. A single software process model provides an abstraction of a specific view of the software process. However, a software process is not abstracted by only one single view. In the literature, there are several proposed software process models that intend to convey different views of the software process [1]-[3].

A software process model can be a kind of an activity process model, which focuses on the types, structure and properties of the activities in the software process and their interrelations. While, a product process model describes the types, structure and properties of the software artifacts of the software process.

A resource process model, on the other hand, describes the resources which are either needed by or provided to the software process.

Whereas, a role process model describes a particular set of resources, known as the performing role, which concerns the performing agents, the skills they provide and the responsibilities they accept.

Since there is no single software process model that is capable to represent all views of the software process, software process meta-modeling paradigm presents a shift in software process abstraction level, from software process model to software process metamodel. Software process meta-modeling, by analogy to software process modeling, is defined as the act of creating and defining software process abstract and generic software process metamodels instead of software process models.

Thus, a specific software process model can be created by instantiating a certain pre-defined software process metamodel [4].

A software process modeling/meta-modeling formalism denotes the modeling language or notation that is used for modeling and formalizing the software process. There are many different software process formalisms have been provided, with different forms (e.g. graphical, textual and etc.) and different levels of formalism (formal, semi-formal, and informal). When a software process formalism is formally formalized, it can be described as a language, known as Software Process Modeling Language (SPML).

Any designed SPML is based upon fulfilling an objective of the essential software process modeling objectives (see [42], [43]). For instance, programming language-based SPMLs (such as [5]-[7]) play a vital role

in automating the software process itself, and its execution. While, graphical-based SPMLs (such as [8]-[10]) are very essential in facilitating the human understandability of process software process models and communication among large numbers of software process model users (e.g. software process owners, software process engineers, project managers, software engineers and executives, and etc.).

A software process metamodel within a software process meta-modeling approach presents a sort of formal software process modeling language, where its abstract syntax is denoted by the set of meta-elements (each meta-element abstracts a single software process concept) and the relationships among them. Yet, there are many different and distinct software process meta-modeling approaches have been introduced [5], [9], [11]-[18]. Each of them still provides distinct concepts and views for expressing the software process.

Since that some software process meta-modeling approaches provide distinctive software process concepts which are missed in others, this paper presents an expressive language named as, SP2MN.

The expressiveness (also expressivity or expressive power) of a language is the breadth of ideas and concepts that can be represented and communicated within that language [19], [20]. SP2MN is designed in order to be as expressive as to include the most common software process modeling concepts, in terms of the most common software process engineering modeling concepts, as well as situational method engineering concepts (as discussed in Section 2). Furthermore, it's essential for this language to enhance human understandability and communication as well, therefore, SP2MN presents a graphic-based language that provides simple and expressive graphical notation. As a final point, SP2MN is evaluated with the well-known ISPW-6 Software Process Example [21], a standard benchmark software process problem developed by experts in the field of software process modeling.

This paper is organized as follows; Section 2 introduces a discussion on the software process, its associated modeling and engineering concepts, with respect to the most prominent software process engineering/meta-modeling languages. Section 3 presents the specification of the proposed SP2MN formalism, while Section 4 provides the validation and evaluation of such proposal. Finally, Section 5 concludes the work presented in this paper.

II. Related Works

Modeling and/or Meta-modeling is a widely embraced approach in software engineering field. For instance, in the context of Model Driven Architecture (MDA), models play a very essential role, not only in the description and representation of the concepts within the domain but also in the production and the automation process [44]-[46]. The success of MDA has attracted several researchers to apply its principles on software process meta-modelling.

Software process meta-modeling provides a mean for software organizations to create their specific software process model by instantiating a certain pre-defined software process metamodel [4]. Hence, a software process metamodel is a description at the type level of a software process model, and at the meta-type level with respect to a process [22]-[24], as depicted in Fig. 1, that follows.

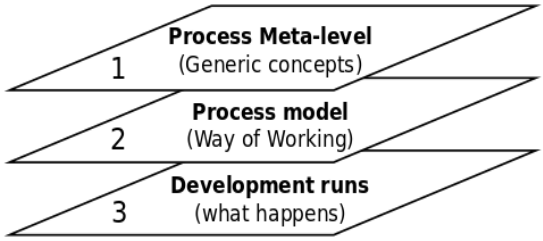


Fig. 1. Software process abstraction levels

Generally, modeling/meta-modeling is not only captured in one single perspective. It considers recording the static concepts and data of the domain, in addition to the activities and other dynamic concepts of the methodical development [25], [26]. Therefore, software process meta-modeling has to be considered in those various perspectives using suitable modeling formalism (language and/or notation) for conceptual (data) modeling plus activity modeling.

This section presents a critical analysis of the most widely used and prominent software process meta-modeling approaches. The approaches are discussed based on their metamodels with respect to the constituent static software process concepts and the underlying semantics of such concepts, along with their associated modeling formalisms. Additionally, the dynamic activity modeling concepts and formalisms are further illustrated and studied as well. The aim of this section is to exhibit the concepts as well as the requirements that SP2MN should encompass and achieve.

II.1. Software Process Engineering

Software Process Engineering Metamodel (SPEM), is a wide known standard software process metamodel currently in version 2.0 [18]. SPEM presents an activity-oriented (also known as, process-focused) metamodel, which focuses on describing the concepts that allow building software process models concentrating on the activities and tasks performed in producing software artifacts together with their ordering.

SPEM 2.0 separates the concept of the reusable method from its application in a software process.

According to SPEM, a method provides step-by-step explanations, describing how specific development goals are achieved independent of the placement of these steps within a development lifecycle. A Process, on the other hand, takes these method elements and relates them into semi-ordered sequences that are customized to specific types of projects.

A method is represented by Method Content meta-package, while software process structure is represented by Software Process Structure meta-package, see SPEM 2.0 [18]. Main meta-classes (concepts) that present reusable methods are; Tasks, Work Products and Roles.

Tasks are the work steps to be performed. Tasks have Work Products as input and output and which are being performed by Roles. Roles define important responsibility relationships to work products.

A software process would apply these elements in Activities in a different parts of a lifecycle differently.

The Activity meta-element presents the structured work definitions which describes the work to be performed along a timeline or lifecycle and organize it in so called breakdown structures. Fig. 2 below shows how such main concepts are related together to represent a real time software process. SPEM 2.0 reuses elements from the Unified Modeling Language (UML) (version 2.0 [47]) metamodel, as UML profile model.

This meta-package presents the set of UML 2.0 stereotypes which provides the necessary meta-elements (concepts) as well as the notation for modeling and formalizing the software process. A UML profile presents a good chance to use existing generic UML 2.0 modeling tools for software process modeling instead of inventing new inventing tools. Fig. 3 shows a UML Class Diagram that represents several SPEM 2.0 concepts as UML classes with their respective stereotypes applied. Note, that the Task "Use Case Analysis" is actually represented by a UML 2.0 Activity which is a specialization of UML 2.0 Class and can therefore have associations and be used in a UML Class Diagram.

II.2. Situational Method Engineering

SME is a special software process engineering meta-modeling paradigm [27]. SME concerns about improving the decomposability, hierarchy and understandability of the methods/software process by introducing modularization concepts and principles.

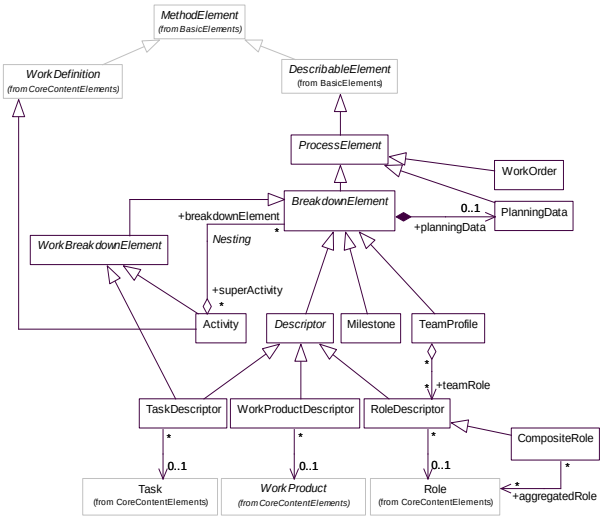


Fig. 2. SPEM 2.0 Software Process Structure meta-package

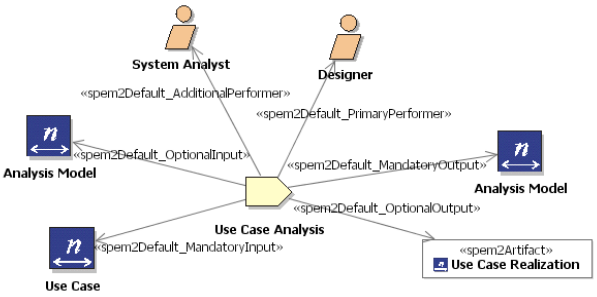


Fig. 3. Sample software process structure represented by SPEM 2.0

Moreover, it ensures flexibility and adaptability of the methods/software process models where their modular components/constructs can be reused and adapted into different applications with similar situational applicability, based on reuse strategy [28].

Therefore, SME meta-modeling approaches present specific concepts concerning the situational discoverability, retrieval and selections of the modular method/software process components.

Yet, there are many different and distinct software process meta-modeling approaches have been proposed (for illustration, [10], [13]-[15], [17], [27], and etc.).

Each one presents its own modularization concepts, notions, semantics and consequently distinct situation, retrieval and selection concepts.

SME metamodels present sort of context-oriented software process metamodels which allow building software process models representing the situation and the intention of an actor at a given moment of the project [23]. The key concepts of this kind of metamodel are the Context that is composed of a Situation and an Intention.

The situation is a part of a product under design that is the object of a decision. The intention represents the objective, i.e. the goal that an actor wants to achieve according to the situation [24], [29].

Although all approaches adopt component-based modularization of software process. Method chunk [10] presents the most complete component-based software process/method modular construct. Contrary to method fragment concept [27] and [14], in which the modular construct represents a process part (activities, tasks, etc.) or product part (work products) of the software process, method chunk ensures a tight coupling of some process part and its related product part of the software process.

Thus, any software process is viewed as a set of loosely coupled method chunks expressed at different levels of granularity [9]. Furthermore, a method chunk is based on the decomposition of the software process model into reusable represented modularization, known as guidelines. Thus, the core of a method chunk is its guideline to which are attached the associated product parts needed to perform the process encapsulated in this guideline. A guideline is built upon intention and situation concepts (context), where the body of the guideline represents the body of the method chunk (process part associated to product part of the software process) that achieves the intention.

Fig. 4 shows the structure, interrelation and semantics of such concepts. Guidelines have three distinct kinds of modeling formalisms, as; simple, tactical and strategic.

A strategic guideline is also known as a MAP [9], (for more information about these modeling formalisms, see [10]) which uses a graph structure to relate its sub-guidelines. A MAP is a labelled directed graph in which the nodes are the intentions and the edges between intentions are strategies to achieve such intentions. For illustration, Fig. 5 shows an example of MAP for representing a part of a method chunk.

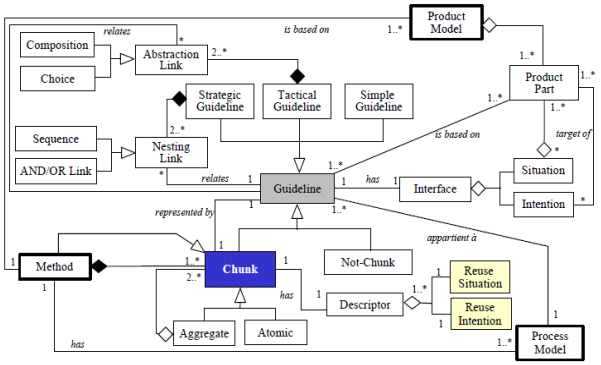


Fig. 4. Method chunk metamodel

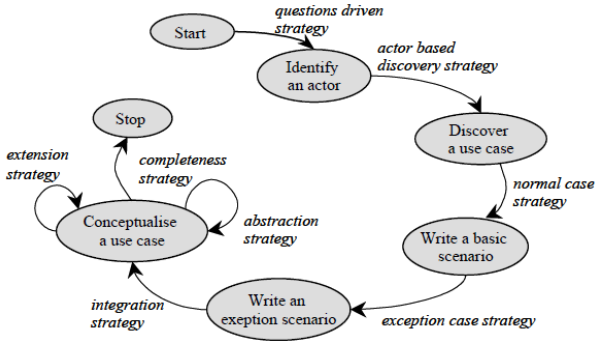


Fig. 5. Sample process part of a method chunk represented by MAP

Another interesting idea introduced by SME is method service [15], [17]. A method service proposes to capture the software process/ method modular construct in a service-based specification. This in turn would pave the way for new promises in software process knowledge sharing and reuse in the service-oriented architecture [48]. However, the modularization of software process models and its presentation in such approaches is not handled according to concepts, principles and modeling formalism that conform to service-oriented architecture and service-orientation design principles.

II.3. Activity Modeling Concepts and Formalisms

The preceding has mostly discussed static software process modeling concepts and formalisms. However, this is insufficient to model and formalize the dynamic concepts of the software process, such as the ability to express events, decisions, exceptions, interactions among software process participants and so on.

In this context, as this work focuses on enhancing human understandability and communication of software process models, the concern is on graphical-based formalisms. As discussed before, many SPMLs present graphical-based formalisms. The most prominent technique presented and adapted for such aim is UML, the success of UML for activity modeling [49] has lured many SPMLs to reuse and adapt this widely used modeling standard instead of inventing from scratch. For illustration; SPEM, PROMENADE language, Di Nitto et al. approach, UML4SPM, and etc.

Alike, there are some promising activity modeling formalisms have been introduced mainly for business process modeling. Business Process Model and Notation (BPMN) is a one contender process modeling technique that has attracted many modelers in the industry as well as researchers [50].

BPMN (currently in version 2.0 [30]) brings a series of enhancements to process modeling, compared to UML for instance, in regards to exclusive/parallel event-based gateway, fine granular tasks and sub-processes for activities, sequential multi-instance activity, flow connections and interdependencies, data objects, a wide set of events for a process, and so on [31]-[40].

Therefore, the work in this paper proposes to reuse BPMN for software process modeling, and to extend its concepts as well as notation to suit the software process modeling specialty.

In summary, given these points, it is essential when designing SP2MN in this paper to be as expressive as possible to express the previously discussed software process modeling concepts with their associated details and semantics while preserving the meta-type level of abstraction of software process models. To sum up, such concepts comprise; activity-oriented software process concepts (such as, activity, task, phase, lifecycle, and etc.), product concepts (including work products details, categories and states), resources and responsibilities concepts (role, human actors/agents, and tools), in addition to modularization and context-oriented concepts (situation and intention).

The essence of this effort is to provide a SPML that acts as a common medium for software process modeling with an understandable, clear, simple and widely used set of graphical notation and diagrams. We believe that this, in turn, would enhance the adoptability and the use of SPMLs from software process users. The following section provides the detailed specification of SP2MN.

III. The SP2MN Formalism

SP2MN presents a new SPML. Generally, a language is composed of syntax and semantics [41]. The syntax of a language means the structure of that language and it is further divided into two complementary types, namely, abstract and concrete syntax. Abstract syntax presents the rules that specify well-formed expressions of symbols (presented here as a conceptual metamodel). While, concrete syntax is the set of graphical and textual

symbols used to render the software process model and other related representing diagrams (in this context, reuses and extends BPMN 2.0). On the other hand, the semantics of a language refers to the meaning and interpretations of the meta-model constructs, attributes, properties as well as the relationships with other constructs and the things being modeled.

The abstract syntax of SP2MN is represented by UML compliant metamodel. It is divided into Software Process Structure metamodel and the Foundation metamodel, as mentioned in Sub-section 1 and Sub-section 2, respectively.

III.1. SP2MN Software Process Structure

Software process structure metamodel, as shown in Fig. 6, provides the overall syntax and semantics needed for software process modeling. At the very highest level, a software process is decomposed into a number of Work Units which can be a coarse-grained Activities or finer-grained Tasks. Work Units are specified by the Context and situation of applicability.

An Activity in turn is composed a number of Tasks. A Task therefore is the adopted unit of work in this specification. A Task handles a number of Work Products as input and/or outputs.

A single Work Unit can be under the responsibility of a number of Roles that are performed by a number of assigned individuals or Human Actors. Automated software or Tools can help such individuals in performing their work as well. A Task is identified by an intention. A Technique models the way of performing or implementing a specific task.

A Work Product is an abstraction of the descriptions of content elements that are used to define anything used, produced, or modified by a Task. In a software process model, a Work Product is either an Artifact or a Deliverable work product or an Outcome.

A Stage is an abstract meta-class that models the intended timing of the performance of a temporally cohesive set of activities during the enactment of a software process. A Life Cycle consists of all phases during which a single system or application is produced, used, and retired. The role of the life cycle is to provide overall organization to the associated activities and milestones. And to support top-level scheduling of activities, personnel, and resource acquisition.

The Context is composed of a Situation and an Intention. The Situation is a part of a product under design that is the object of a decision. The Intention represents the objective, i.e. the goal that an actor wants to achieve according to the situation.

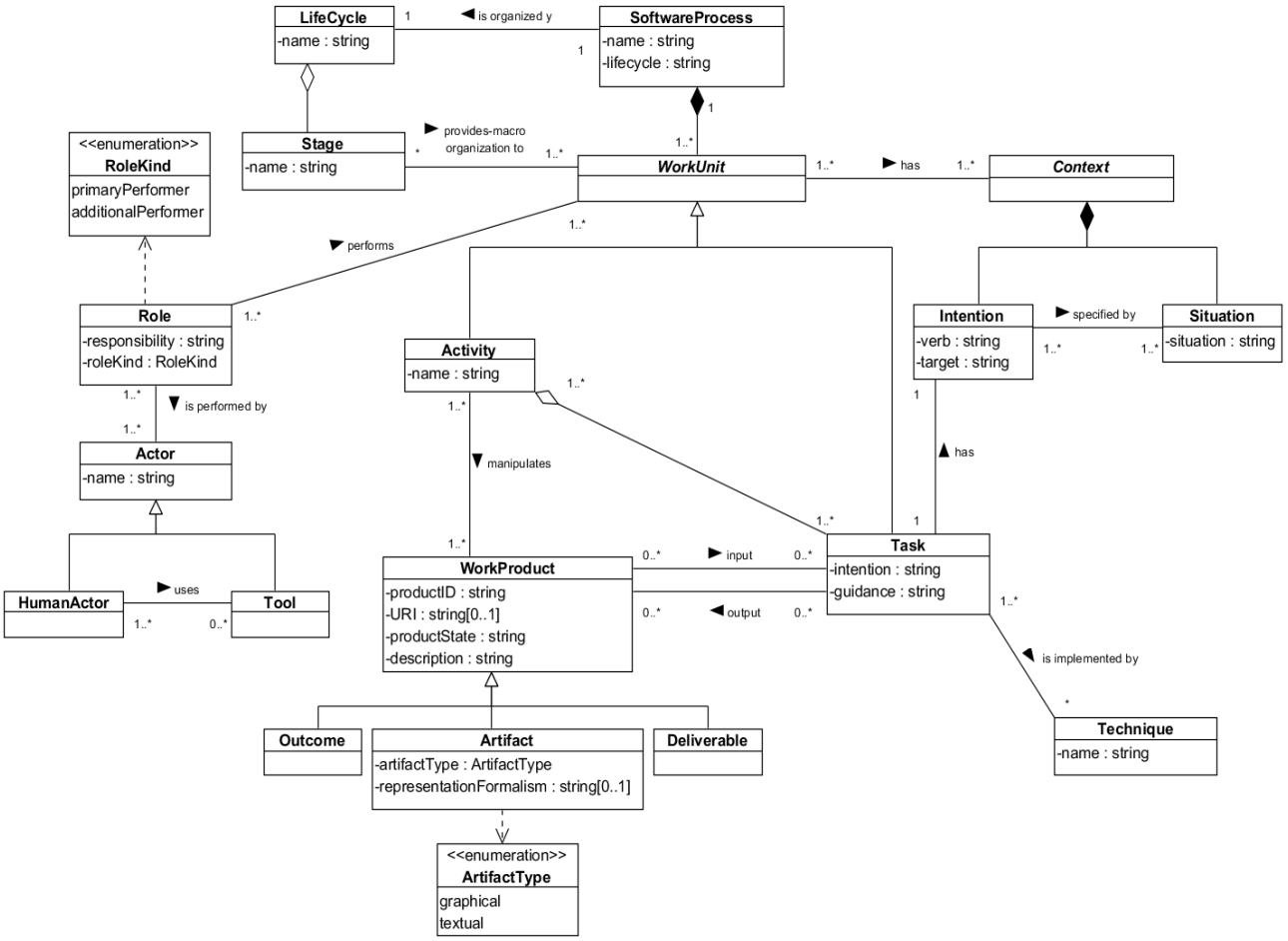


Fig. 6. SP2MN software process structure metamodel

III.2. SP2MN Foundation

Foundation package represents the subset of BPMN 2.0 [30] elements that are reused in SP2MN metamodel and other elements that are extended and adapted to suit the software process modeling.

As shown in Fig. 7, the dynamic concepts for software process modeling are shown in white background classes, while other related static concepts that to be reused are shown in light gray background classes, whereas new extension concepts are shown in dark gray back ground classes. As shown in the metamodel, BPMN basic elements are a special classes of the generalized BaseElement meta-class.

BaseElement is the abstract super class for most BPMN basic elements. It provides the attributes, id and documentation, which other elements will inherit.

The basic elements are particularly; a set of Flow Elements (Activities, Events, Gateways, Sequence Flows, Message flows), In addition to Data Objects, Pools, and Lanes, as well as other BPMN Artifacts, such as Groups, Text Annotations, and Associations. A FlowElement is a special class from BaseElement and it the abstract super class for Activities, Events, Gateways, Sequence Flows, and Message flows. Where such meta-elements constitute the key elements that affect the flow of process and the interaction between its participants.

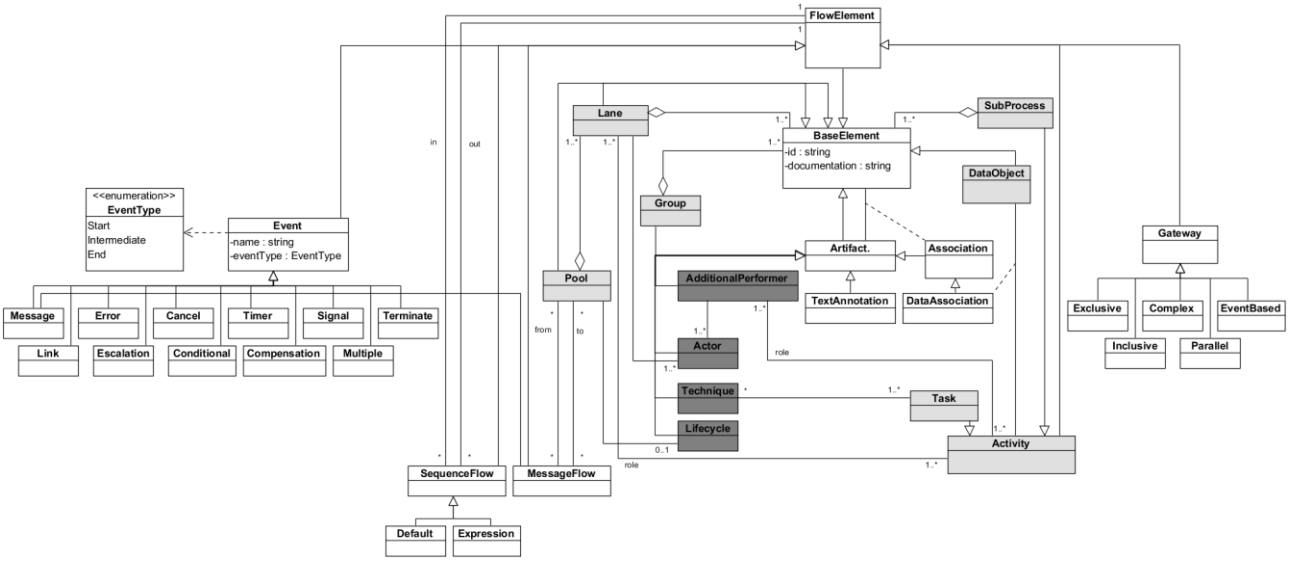


Fig. 7. SP2MN foundation metamodel

III.3. SP2MN Notation

The concrete syntax of SP2MN is a graphical-based notation adopted, adapted and extended from BPMN 2.0 constructs. Each of the aforementioned software process elements and constructs in the abstract syntax is mapped to a visual graphical symbol.

Due to limited space, Table I below shows a subset of the most important SP2MN constructs.

IV. Evaluation of SP2MN with ISPW-6 Software Process Example

The 6th International Software Process Workshop has produced a standard benchmark software process modelling example problem [21]. A problem that comprehensively exercises the various modeling approaches being developed, throughout coverage of several important components of real-world software processes. The primary purpose behind that was to facilitate understanding, assessing and comparing the various approaches that are being pursued for software process modelling. The core problem is scoped as a relatively confined portion of the software change process. It focuses on the designing, coding, unit testing, and management of a localized change to a software system. The change is prompted by a change in requirement that happens during the development life-cycle.

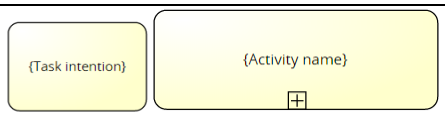
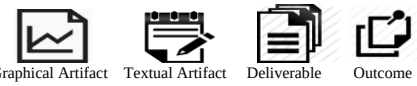
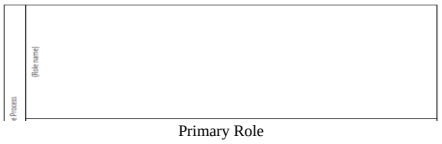
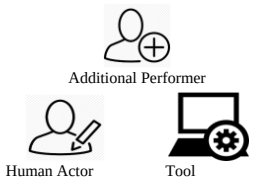
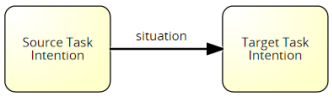
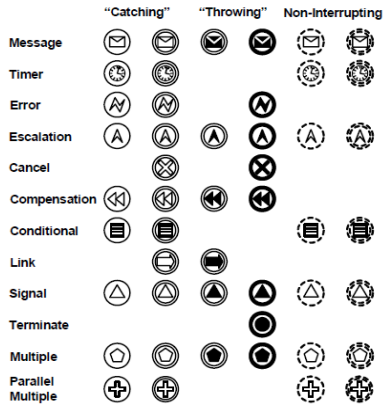
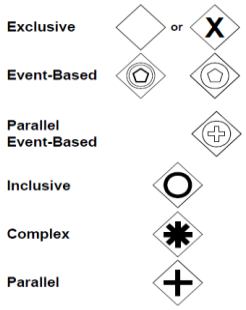
The entire example process is entitled Develop Change and Test Unit, and it is decomposed of a set of major tasks as: Schedule and Assign Tasks, Modify Design, Review Design, Modify Code, Modify Test Plans, Modify Unit Test Packages, Test Unit, and finally Monitor Progress. The process is described by a narrative description.

The software process problem as described in [21] is considered to demonstrate the applicability and validity of the SP2MN. Due to limited space, Fig. 7 shows an instance of the ISPW-6 software process problem as represented by SP2MN notation, whereas Fig. 8 respectively shows the overall represented software process model.

V. Results

The following shows how SP2MN has fullfiled the stated requirements and design goals.

TABLE I
SP2MN NOTATION

Software Process Concept	Notation (Graphical Symbol)
Work Units (Tasks or Activities)	
Techniques	
Work Products (Artifacts, Deliverables, , Outcomes)	
Role	
Actor	
Life Cycle	
Stage	
Context (Intention plus situation)	
Events	
Gateways	

The expressiveness of SP2MN was defined as the breadth of ideas and concepts that can be represented and communicated within it. These concepts include, static software process concepts, such as, work unit and stage, work product, role, actor, and context, in addition to dynamic software process concepts, such as, communication within the software process and between participants, software activities coordination, and software process events, as well as other concepts, such as, representing a modularized software process entities.

By applying SP2MN on the standard benchmark ISPW-6 software process example [21] it has proved its applicability and validity. It also has shown how each of the aforementioned concepts have been demonstrated on its software process elements, as mentioned in the previous section and shown in Fig. 8. For illustration, Work units are represented as tasks, sub-processes.

Moreover, the technique, as well as the stage and lifecycle concepts representations are also represented.

Moreover, different kinds of work products (such as deliverable, or artifact) and the delivered product types (graphical, or textual), as well as the products formalism, in addition to product state are represented.

ISPW-6's pre-condition process elements are represented by intentions which are associated to tasks labels. While post-condition elements are represented by situations which are associated to flow connectors.

Where, the communication within the process is by signal events and data association flow. While the communication between software process participants is by messages. Above all, modularization is supported by tasks that are represented as autonomous sections (associated with Contexts), which are eligible to be defined and represented as method services.

Finally, SP2MN was designed with the aim to achieve the human understandability. This achieved by reusing sets of BPMN 2.0 elements and notations. This is considered as a good significance, since that BPMN 2.0 has attractive features. It is standard, graphical, intuitive, and easy to understand. A wide community of software process modeling is already familiar with BPMN2.0 and variety of tools and training supports are proposed.

Therefore, SP2MN has an important competitive advantage compared to any existing SPML.

VI. Conclusion

While there are multiple SPMLs have been proposed, nonetheless, they have failed to gain the attention of the industry. They are abandoned from many software process users. The mere existence of existing software process metamodels and consequently the distinct software process concepts, notions and structures are seen as the main causes of such a problem. Additionally, it might be due to the complexity and ambiguity of their formalisms. SP2MN presents a SPML that concerns the reusing and adapting BPMN, as a clear, simple, understandable, as well as being standard, widely acceptable and used formalism for process modeling.

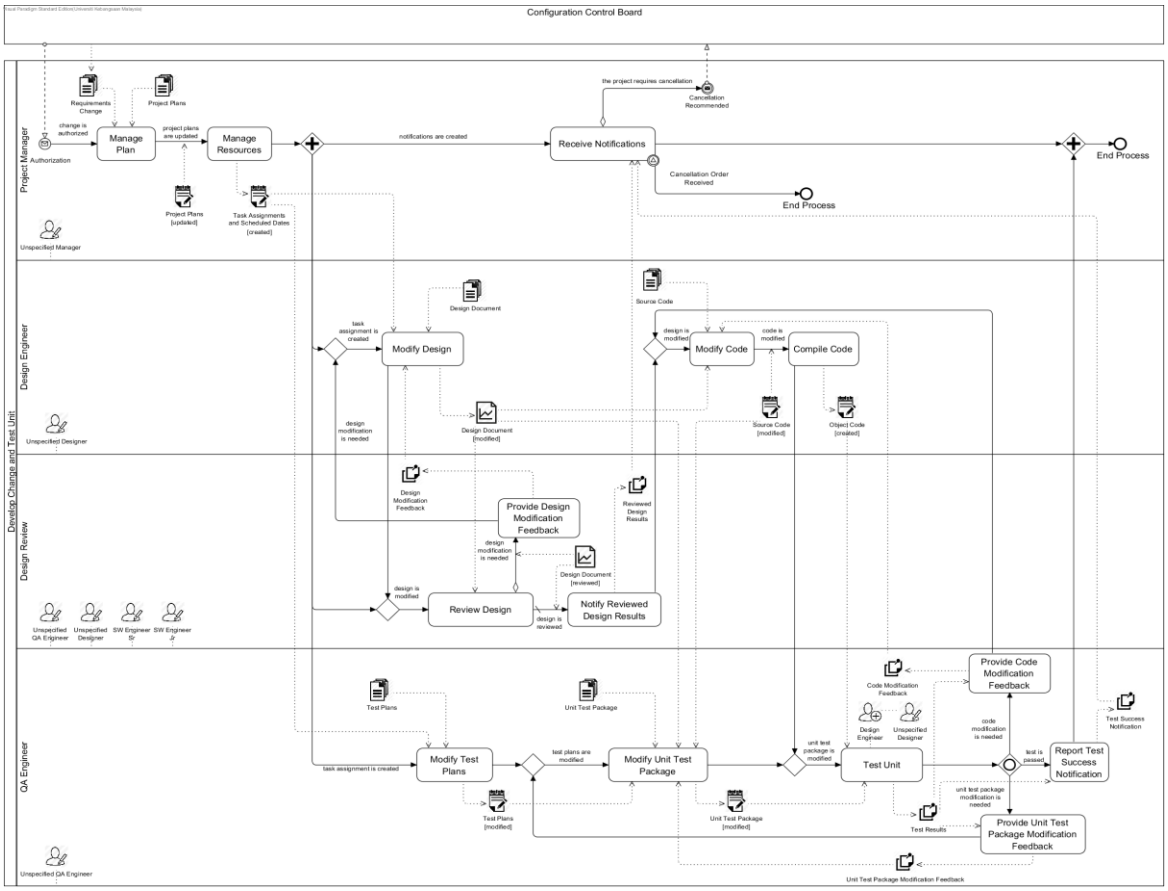


Fig. 8. Overall ISPW-6 software process represented by SP2MN

More importantly, SP2MN is designed with an aim to be expressive to the most common software process modeling concepts that are being recorded within real time software process modeling.

SP2MN metamodel is a UML-compliant metamodel, which simply can be instantiated in order to produce specific software process models. In conclusion, validating and evaluating SP2MN on the standard ISPW-6 benchmark for software process modeling has demonstrated its workability, validity and significance.

The constructs of the language and its graphical notation has proved its power to express all software process elements and concepts of the standard benchmark ISPW-6 software process example [21].

However, SP2MN notation has been introduced as conceptual graphical symbols. Yet, implementing such symbols within a specific and compliant to SP2MN software process modelling tool is planned to be a future work.

References

[1] V. Ambriola, R. Conradi, and A. Fuggetta, "Assessing process-centered software engineering environments," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 6, pp. 283-328, 1997.

[2] J.-C. Derniame, B. A. Kaba, and D. Wastell, *Software process: principles, methodology, and technology*, vol. 1500: Springer, 1999.

[3] A. Finkelsteiin, J. Kramer, and B. Nuseibeh, *Software process modelling and technology:* John Wiley & Sons, Inc., 1994.

[4] B. Henderson-Sellers and C. Gonzalez-Perez, "On the ease of extending a powertype-based methodology metamodel," *Meta-Modelling and Ontologies. WoMM* 2006, pp. 11-25, 2006.

[5] S. Brinkkemper, K. Lyytinen, and R. Welke, *Principles of Method Construction and Tool Support*: Springer, 1996.

[6] N. Goldman and K. Narayanaswamy, "Solution to ISPW-7 Process Example", Manuscript USC Information Sciences Institute, Marina Del Rey CA October 1991.

[7] M. P. Christopher M. Lott , H. Dieter Rombach, "A MVP-L Solution for the Software-Process Modeling Problem," 1991.

[8] F. Karlsson and K. Wistrand, "Combining method engineering with activity theory: theoretical grounding of the method component concept," *European Journal of Information Systems*, vol. 15, pp. 82-90, 2006.

[9] J. Ralyte and C. Rolland, "An approach for method reengineering," in Conceptual Modeling—ER 2001: Springer, 2001, pp. 471-484.

[10] J. Ralyte, "Ingénierie des méthodes à base de composants," 2001.

[11] J. Ralyte and I. Mirbel, "DeneckèreR (eds)(2011) Engineering methods in the service-oriented context," *Proceedings of the 4th IFIP WG8*, vol. 1, 2011.

[12] C. Cauvet, "Method engineering: a service-oriented approach," in *Intentional Perspectives on Information Systems Engineering*: Springer, 2010, pp. 335-354.

[13] M. Cossentino, S. Gaglio, A. Garro, and V. Seidita, "Method fragments for agent design methodologies: from standardisation to research," *International Journal of Agent-Oriented Software Engineering*, vol. 1, pp. 91-121, 2007.

[14] D. G. Firesmith and B. Henderson-Sellers, The OPEN process framework: An introduction: Pearson Education, 2002.

[15] G. Guzelian and C. Cauvet, "SO2M: Towards a service-oriented approach for method engineering," presented at the *2007 World Congress in Computer Science, Computer Engineering and*

Applied Computing, in the proceedings of the international conference IKE, 2007.
[16] A. F. Harmsen, J. N. Brinkkemper, and J. L. H. Oei, *Situational method engineering for information system project approaches*: University of Twente, Department of Computer Science, 1994.
[17] A. Iacovelli, C. Souveyet, and C. Rolland, "Method as a service (MaaS)," presented at *Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on*, 2008.
[18] OMG, "*Software & Systems Process Engineering Metamodel Specification (SPEM)* - Version 2.0," 2008.
[19] M. Felleisen, "On the expressive power of programming languages," *Science of computer programming*, vol. 17, pp. 35-75, 1991.
[20] M. Felleisen, "On the expressive power of programming languages," in *ESOP'90*: Springer, 1990, pp. 134-151.
[21] M. Kellner, P. Feiler, A. Finkelstein, T. Katayama, L. Osterweil, M. Penedo, and D. Rombach, "*ISPW-6 software process example*," 1991.
[22] C. Rolland, "Modeling the requirements engineering process," presented at *Information Modelling and Knowledge Bases V: Principles and Formal Techniques: Results of the 3rd European-Japanese Seminar*, Budapest, Hungary, May, 1993.
[23] C. Rolland, "A comprehensive view of process engineering," presented at *Advanced Information Systems Engineering*, 1998.
[24] C. Rolland, C. Souveyet, and M. Moreno, "An approach for defining ways-of-working," *Information Systems*, vol. 20, pp. 337-359, 1995.
[25] J. Souer, I. Van De Weerd, J. Versendaal, and S. Brinkkemper, "Situational requirements engineering for the development of content management system-based web applications," *International Journal of Web Engineering and Technology*, vol. 3, pp. 420-440, 2007.
[26] I. van de Weerd, J. Souer, J. Versendaal, and S. Brinkkemper, "Situational requirements engineering of web content management implementation," presented at Proceedings of the *First International Workshop on Situational Requirements Engineering Processes: Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes (SREP'05)*, Paris France, 2005.
[27] S. Brinkkemper, "Method engineering: engineering of information systems development methods and tools," *Information and software technology*, vol. 38, pp. 275-280, 1996.
[28] B. Henderson-Sellers and J. Ralyte, "Situational Method Engineering: State-of-the-Art Review," *J. UCS*, vol. 16, pp. 424-478, 2010.
[29] N. Prat, "Goal formalization and classification for requirements engineering, fifteen years later," presented at *Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on*, 2013.
[30] OMG, "*Business Process Model and Notation (BPMN)*," 2011.
[31] W. M. P. van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distributed and parallel databases*, vol. 14, pp. 5-51, 2003.
[32] J. C. Recker, M. zur Muehlen, K. Siau, J. Erickson, and M. Indulska, "*Measuring method complexity: UML versus BPMN*," 2009.
[33] M. R. Khabbazi, M. K. Hasan, R. Sulaiman, A. Shapi'i, and A. Taei-Zadeh, "Business Process Modelling in Production Logistics: Complementary Use of BPMN and UML," *Middle-East Journal of Scientific Research*, vol. 15, pp. 516-529, 2013.
[34] G. Aagesen and J. Krogstie, "BPMN 2.0 for modeling business processes," in *Handbook on Business Process Management* 1: Springer, 2014, pp. 219-250.
[35] L. Eloranta, E. Kallio, and I. Terho, "A Notation Evaluation of BPMN and UML Activity Diagramsâ€–," Special course in information systems, 2006.
[36] C. V. GEAMBAŞU, "*BPMN vs. UML Activity Diagram for Business Process Modeling*," 2012.
[37] S. Meyer, K. Sperner, C. Magerkurth, and J. Pasquier, "Towards modeling real-world aware business processes," presented at *Proceedings of the Second International Workshop on Web of Things*, 2011.
[38] D. Peixoto, V. Batista, A. Atayde, E. Borges, R. Resende, and C. PÃ¡dua, "A comparison of BPMN and UML 2.0 activity diagrams," presented at *VII Simposio Brasileiro de Qualidade de Software*, 2008.
[39] W. M. P. Van der Aalst and K. M. van Hee, "Framework for business process redesign," presented at 2012 *IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2012.
[40] P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell, *On the suitability of BPMN for business process modelling*: Springer, 2006.
[41] G. Génova, "*Modeling and metamodeling in Model Driven Development - What is a model: syntax and semantics*," 2009.
[42] S. T. Acuna and X. Ferre, "Software Process Modelling," presented at *ISAS-SCI* (1), 2001.
[43] S. T. Acuna and N. Juristo, *Software process modeling*, vol. 10: Springer, 2006.
[44] Sabraoui, A., El Koutbi, M., Khriss, I., A MDA-based model-driven approach to generate GUI for mobile applications, (2013) *International Review on Computers and Software (IRECOS)*, 8 (3), pp. 844-852.
[45] Rahmouni, M., Mbarki, S., Combining UML class and activity diagrams for MDA generation of MVC 2 web applications, (2013) *International Review on Computers and Software (IRECOS)*, 8 (4), pp. 949-957.
[46] Rahmouni, M., Mbarki, S., An end-to-end code generation from UML diagrams to MVC2 web applications, (2013) *International Review on Computers and Software (IRECOS)*, 8 (9), pp. 2123-2135.
[47] OMG, "*UML Version 2.2*," 2009.
[48] Hanafiah, M., Abdullah, R., Murad, M.A.A., Din, J., Towards developing collaborative experience based factory model for software development process in cloud computing environment, (2015) *International Review on Computers and Software (IRECOS)*, 10 (3), pp. 340-350.
[49] Aabidi, M.H., Jakimi, A., El Kinani, E.H., Elkoutbi, M., A new approach for code generation from UML state machine, (2013) *International Review on Computers and Software (IRECOS)*, 8 (2), pp. 500-506.
[50] Saib, S., Benmoussa, R., Bengoud, K., Collaborative business process specification and a mapping from BPMN model to service model, (2015) *International Review on Computers and Software (IRECOS)*, 10 (3), pp. 351-361.

Authors' information

School of Computer Science, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia (UKM), Malaysia.

Hisham Khdair received a BE degree in Computer Systems Engineering in 2007 from Al-Azhar University of Gaza – Palestine and a MSc degree in Information Technology in 2010 from UUM university - Malayasia. He is currently a PhD candidate in Computer Science at UKM. His research interests include software engineering and technology, software processes, SDLC, software methods, and software process modeling.
E-mail: h.s.khdair@hotmail.com

Zulaiha Ali Othman was born in Malaysia. She was awarded a PhD in Software Engineering by Sheffield Hallam University, England, in 2004. Since then she has been a lecturer and researcher at Faculty of Information Science and Technology, UKM. Associate Prof. Dr.Zulaiha is a member of several computer science committees. She has written more than 100 conference and journal papers that have been published around the world and has several research awards.
E-mail: zao@ftsm.ukm.my