A Formal Semantic for Scenario-Based Model Using Algebraic Semantics Framework for MOF

M. A. Isa, Dayang N. A. Jawawi and M. Zulkifli M. Zaki

Department of Software Engineering
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia (UTM),Skudai 81300, Johor Bahru, Malaysia
{mohdadham, dayang, zulkiflizaki}@utm.my

Abstract

Model-driven development uses models to represent system artifacts to improve the visibility of the system towards the real world. The development of models is underlying in the meta-object facility (MOF) standard in order to define the building concepts of metamodel and models. However, these concepts, especially within MOF standard, are not yet formally define which will be difficult to check the consistency between metamodel and models. Therefore, defining a formal semantic for MOF-based metamodel is essential for discovering the meaning of the model and to ensure a structural and behavioral conformance between metamodel and its model. In this paper, we define a formal semantic for a scenario-based model called Performability Failure Behavior Awareness Metamodel (PerFAM) by applying algebraic semantics for MOF framework which provides a formal stage: metamodel, model and model conformance. For this purpose, a formal consistency checking can be applied as to ensure the accuracy of the produced model towards its metamodel.

Keywords: *MOF, Metamodelling Semantics, formal analysis, metamodel*

1. Introduction

Modeling a system is important to be able to represent the abstract level of the system from a real world aspect. The process of modeling the software system can be realized through model-driven development in which system artifacts are represented as models in order to enhance visibility, understanding and quality. Presently, models are first class artifacts in a modeling domain where models are capable of providing a rigorous understanding of the system on a more abstract level [1]. The meta-object facility (MOF) [2] provides a systematic framework to define a new modeling language in which the abstract syntax can be declared in a specific domain concept. Hence, the MOF also provides a basis for better model-driven development in terms of model and metamodel definition, as well as defining the conformance between models and metamodels. However, the MOF framework does not provide a standard notion for formally defining the metamodel semantics in a systematic way [3]. This shortcoming then produces a few problems in a model-driven development process.

The metamodel is a collection of data based on MOF framework. This data builds up the modeling languages that will be used to produce a set model that conforms to that metamodel. However, when the metamodel is not formally defined in the syntax and semantic definition, problems such as misinterpretation of models among developers cannot be avoided [4]. This problem could cause some metamodel derivation problems when interpreting from a metamodel to its model. Therefore, it is important to have a formal semantic for metamodel and models so as to ensure the consistency for models production from its metamodel and the correctness of system design.

In this paper, we defined a formal semantic for our previous metamodel [5] by properly defining its syntax and semantic in a more comprehensive representation based on algebraic semantics for the MOF framework [3]. This framework is advantageous in two aspects: (1) a mathematical representation of metamodel and models will provide a more comprehensive definition; and (2) the conformance between metamodel and models can be traced. We clearly defined the metamodel, model and model conformance semantics within this framework theoretical background. This formal semantic is capable of providing a more useful representation of metamodel for checking of consistency between metamodel and models.

The rest of the paper is organized as follows: Section 2 reviews some related works. The overview of the PerFAM model and its features is highlighted in Section 3. Section 4 shows the semantic framework that we applied and Sections 5 and 6 discuss the formal semantic declaration and motivation example respectively. Section 7 describes a finding that can be gained through motivation example and Section 8 concludes this paper with the expected future works.

2. Related Works

Seminal works have appeared for defining a formal semantic for MOF/UML modeling languages, in particular for behavior model. The behavioral model in UML builds up the scenario of the system, which triggers how the system behaves with a series of action or tasks.

In [6], the formal definition for sequence diagram was defined by using QVT relations' language to ensure the correctness of the model. Similar work in [7] defines the formal semantics of sequence diagram using Petri Nets to express the sequential and parallel behavior of systems. In a more general form, work in [8] proposed the formal semantics for a state chart diagram for UML by using graph grammar and a graph transformation approach. This approach is beneficial for design verification purposes, which investigate the consistency during system requirements to system design transformation. In addition, the formal semantics for a sequence diagram is defined for automated analysis [9] for testing, tool development or simulation. The rest of the overview of formal semantics for a behavior model in UML can be found in [10].

The method for defining a formal semantic within MOF/UML domain is different based on what is required for details metamodel/model. In [3], the framework for defining semantics for MOF is presented. This framework introduced two new notations, namely, model realization and model types that are formalized by using Membership Equational Logic (MEL) theory supported by MAUDE language and MOMENT2 tool. Another work in [11] defines the metamodel semantics by applying attribute grammars in order to integrate with a supporting tool for automation interpretation. The systematic framework for defining semantics for class, object and state diagrams is presented [12] by using graph transformation approach. This framework provides the advantages for syntax checking, validation checking and verification. A similar work also can be found in [13-15].

3. PerFAM Model Overview

The PerFAM model [5] is an intermediate model, between the UML sequence diagram and analysis model (DTMC, Petri Nets, etc.). This model provides a modeling view for performance and reliability aspects, particularly for outlining failure behavior of timing requirements. The idea is underlying in model-driven development where the UML sequence diagram with performance annotation from MARTE profile is then transformed into a PerFAM model for performance and reliability viewing models. The PerFAM model is then

transformed into analysis model for in depth analysis. Figure 1 shows the PerFAM metamodel and its model in Figure 2.

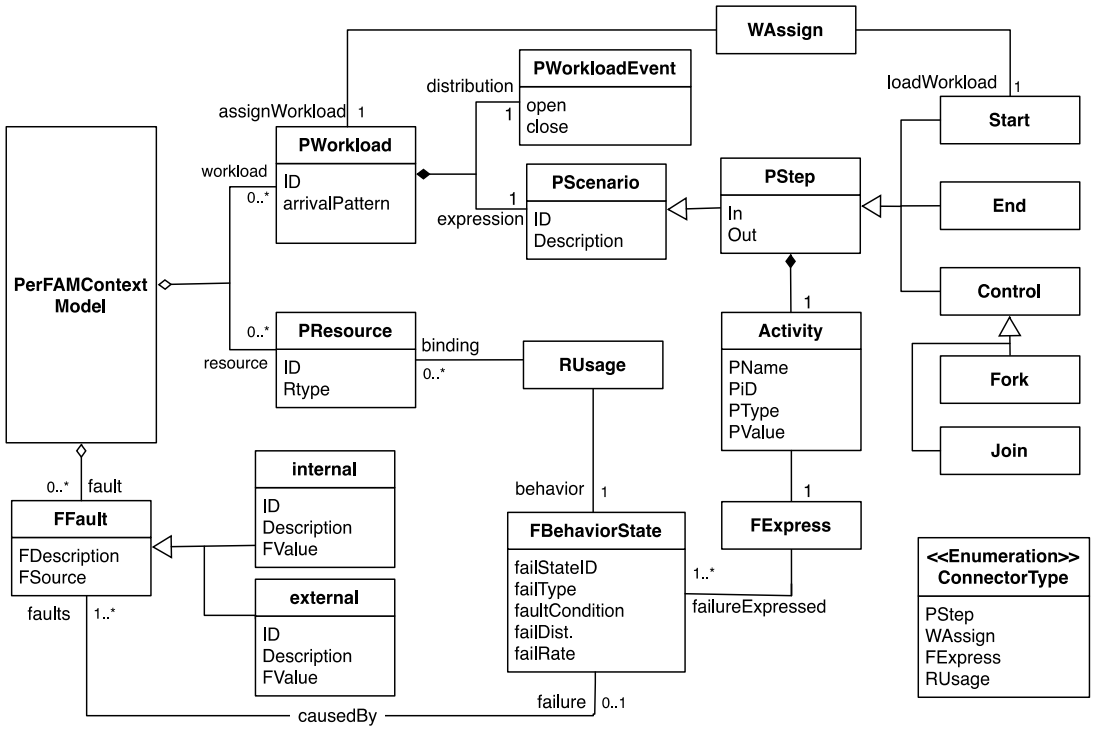

Figure 1. PerFAM Metamodel [5]

The PerFAM metamodel as shown in Figure 1 builds up the modeling languages, which underlie the domain concepts for performance, reliability and a scenario-based model. Each concept poses its own meta-class representing its own definition and constraints as depicted in Table 1.

Table 1. Domain Concepts in PerFAM Metamodel

Domain Concepts	Meta-class
Performance	*PWorkload, PWorkloadEvent, PResource*
Reliability	*FFault, FBehaviorState, internal, external*
Scenario-based model	*PScenario, Activity, Start, End, Control, Fork, Join, WAssign, PStep, FExpress, RUsage*

Performance domain concepts, for instance, pose meta-classes known as *PWorkload, PWorkloadEvent, PResource.* These meta-classes interconnect with the meta-classes for reliability. *FFault, FBehaviorState, internal, external* exist via scenario-based model meta-classes, and *PScenario, Activity, Start, End, Control, Fork, Join, WAssign, PStep, FExpress, RUsage* show the relationship that builds up the PerFAM model.

In Figure 2, the PerFAM model is produced based on the PerFAM metamodel. This model is responsible for expressing a failure behavior of the system task regarding timing failure, as well as the capacity to identify which resource is being used by the system task during a

particular execution time. In addition, this model also poses several benefits in terms of system modeling and analysis. The benefits of PerFAM model are:

(i) The model is easy to implement, as this model is well suited to the UML MARTE profile for performance annotation and smooth integration into a design environment.

(ii) The model poses a linear sequence process, originally from a UML sequence diagram that makes the integration between this model and the UML sequence diagram semantically easy to implement.

(iii) This model is derived solely from MOF generic framework and can integrate smoothly with model-driven development.

(iv) This model provides a clear view of failure behavior of the system that enables this model to produce early analysis on performance and reliability aspects.

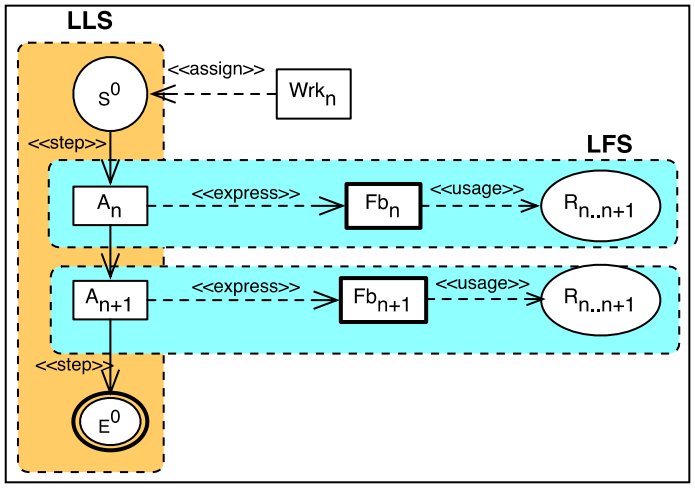

Figure 2. PerFAM Model

A formal semantic for both PerFAM metamodel and model is needed to ensure that the consistency of model inheritance conforms to its metamodel. The formal semantic can also provide the accuracy of the model and provide for the development of a tool support. The details of formal semantics will be explained in detail in Section 5.

4. The Framework for Semantic Declaration

The metamodel semantic is defined within the context of *metamodel*, *model* and *model conformance*. The metamodel is defined based on its static and dynamic features. These features will build up the model realization and are described in a mathematical representation to precisely describe the meaning of metamodel elements. Once the metamodel and model realizations have been declared, a set model can then be developed based on metamodel and metamodel realization context and its model conformance can be defined as precisely tracing the conformance between metamodel and models.

In order to realize this process, we used the algebraic semantic for MOF [3] framework to formally define a notion for *metamodel*, *model* and *model conformance* as shown in Figure 3.

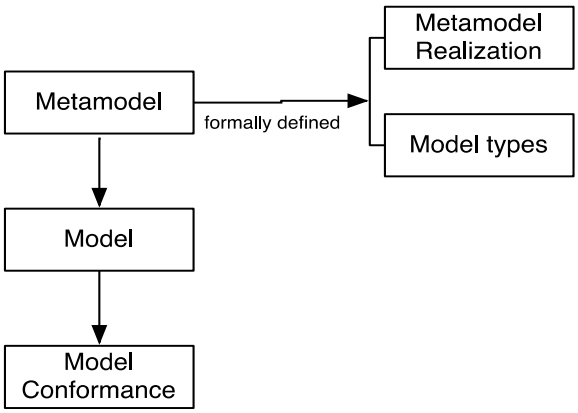

Figure 3. Semantics for MOF Framework

The framework as depicted in Figure 3 establishes a formal representation of MOF, as the MOF is a semiformal approach to define a modeling language. Generally, this framework consists of three levels of notions, namely, *metamodel*, *model* and *model conformance*. The metamodel is usually described as a data or theory. However, at present, the MOF based metamodel lack of mathematical representations as to provide the accuracy of the metamodel towards its model. Therefore, this framework provides two additional notions of metamodels called *metamodel realization* and *model type*. A *metamodel realization* enables a precise mathematical representation for metamodel using MEL theory to describe its *model types.*

As shown in Figure 3, the first artifact is called MOF metamodel, $\mathbb{M}$. The MOF metamodel $\mathbb{M}$ realizes a model type through *metamodel realization* within the context of MEL theory. The metamodel realization is denoted as $\mathbb{A}(\mathbb{M})$ and is formally associated with MEL theory. The simplified MEL theory, $T_{(\Sigma,E)}$ [3] defines the initial algebra for each associated MOF metamodel elements in $\mathbb{M}$. The second artifact is called model, *model.* The model that is defined from a metamodel conforms to the MOF metamodel $\mathbb{M}$. Once the MOF metamodel $\mathbb{M}$ and model, *model* is defined, a model, *model* is said to conform to a MOF metamodel $\mathbb{M}$ when the elements in model are properly inherited from a MOF metamodel $\mathbb{M}$ and this relationship is then called *model conformance*, $model : \mathbb{M}$. In conclusion, the framework for metamodel semantic declaration gives advantages in terms of:

- *Metamodel*: The metamodel is formally detailed into a precise mathematical representation for *metamodel realizations* and *model types*. The metamodel realization establishes the meaning of metamodel in depth that defines a set of model types.
- *Model*: The declaration of model semantics based on a concrete syntax that precisely defines a static and dynamic of model behavior.
- *Model conformance*: The conformance relation between metamodel and its model.

5. The Formal Semantics for PerFAM

In this section, we formally defined the formal semantics for the PerFAM metamodel, the PerFAM model and its model conformance. The formal semantics for the PerFAM metamodel are defined within the context of its metamodel elements, action, data types, attributes and associations. Thus, from this metamodel, we can define the formal semantics for its correspondence model contributing to concrete syntax, system and configuration state.

Finally, we defined the model conformance for consistency checking between the PerFAM metamodel and model.

5.1. Metamodel, Metamodel Realization and Model Types

The MOF metamodel is defined as being a metamodel $\mathbb{M}$ and the PerFAM metamodel as a special metamodel, M_{PerFAM} from a core MOF metamodel, $\mathbb{M}$. The PerFAM metamodel M_{PerFAM} defines a metamodel, which conforms to the MOF metamodel $\mathbb{M}$, which contains all the metamodel elements in a PerFAM metamodel. The extensional semantics of MOF metamodel $\mathbb{M}$ can be denoted as

$$[\![\mathbb{M}]\!] = \{\mathrm{M}_{PerFAM} | \mathrm{M}_{PerFAM} : \mathbb{M}\}$$

The extensional semantics for the MOF metamodel $[\![\mathbb{M}]\!]$ defines a set of the PerFAM metamodel M_{PerFAM}. Since the PerFAM metamodel conforms to the MOF metamodel $\mathbb{M}$, the informal MOF semantics will be described in terms of initial algebraic semantics with MEL specification as mentioned in Section 4. Therefore, a *metamodel realization* for a PerFAM metamodel is defined as:

$$[\![M_{PerFAM}]\!] = T_{\mathbb{A}\,(M_{PerFAM}),model}$$

The metamodel realization $T_{\mathbb{A}\,(M_{PerFAM})}$ provides the algebraic representation for PerFAM metamodel M_{PerFAM}. Differentiating from [3], we define our PerFAM metamodel M_{PerFAM} semantic using equational logic so as to provide a general semantic specification for the metamodel. The general semantic of the PerFAM metamodel M_{PerFAM} is defined within the context of an MOF metamodel $\mathbb{M}$ where the formal semantic for PerFAM metamodel M_{PerFAM} is a tuple,

$$\mathbb{M} \rightarrow \mathrm{M}_{PerFAM} \stackrel{\text{def}}{=} (ATT, ACTION, ELEMENT, ASC, datatypes)$$

where

- $ELEMENT$= is a set of static metaclass
- ATT= is set of attributes that define the $ELEMENT$
- $ACTION$= is a set of $ELEMENT$behaviour
- ASC= is set of association between elements
- $datatypes$ is a definition of type of data for element attributes, $dt \in ATT_{ELEMENT}$.

Definition 1. (metamodel elements)

General aspects of the PerFAM metamodel consist of ELEMENT, where ELEMENT is a finite set of metaclassess, $ELEMENT \subseteq MC$. $ELEMENT$ is defined as the joint sets of $PASSIVE$ and $ACTIVE$, $ELEMENT \stackrel{\text{def}}{=} PASSIVE \cap ACTIVE$. Elements define the static metaclass, $el \in ELEMENT$ in PerFAM that specify its dynamic operation, $act \in ACTION$ in the PerFAM model. In the PerFAM metamodel, each element, $el \in ELEMENT$ is associated with a set of attributes, ATT_{el} that determine the identification of elements through its characteristics and data types.

Definition 2. (Action)

The actions in PerFAM metamodel are defined by a set of action and association,

$$ACTION_c \stackrel{\text{def}}{=} \{act: el_1 \times el_2 \times \ldots \times el_n \rightarrow el | act \in ACTION, n \in \mathrm{I}_0\}$$

The action, act describes the action name and the finite number $n \in \mathrm{I}_0$ of the element el which determines the $ELEMENT$ in the PerFAM metamodel M_{PerFAM}, which applies the action, act. The PerFAM metamodel actions are used to describe the behavior of the metamodel from the view of the elements relationship and association, but from high level views. The details will be explained in Section 5.1.2.

Definition 3. (Association)

The association for each element is connected by a special relationship through action, act. The association has a name, $n \in \mathbb{N}$ that specifies the domain of element actions. We define function

$$act: ACTION \rightarrow ASC(\mathbb{N})$$

to describe the set of association of actions that can be referred from a given associated element by navigating along the participant elements.

5.1.1. Abstract Syntax of PerFAM Metamodel: Once the general semantics for a PerFAM metamodel M_{PerFAM}has been defined, the abstract syntax for a PerFAM metamodel M_{PerFAM} can be outlined in the lower views. The abstract syntax for a PerFAM metamodel is obtained from the metamodel as shown in Figure 1, which defines its specific static elements. In our views in Definition 4, we divide our metamodel semantic into 4 views, namely, *Performance*, *Reliability*, *Connector* and *Behavior* that build the relationship between domain concepts and metamodel elements.

Definition 4. (Domain concepts and elements)

Let $el \in ELEMENT$ is a special $ELEMENT$ for the PerFAM metamodel M_{PerFAM} that represents the reliability and performance aspects of instances of $ELEMENT$. The abstract syntax for PerFAM metamodel M_{PerFAM} has 4 tuples,

$$\mathrm{M}_{PerFAM} \stackrel{\text{def}}{=} \langle PERFORMANCE_{el}, RELIABILITY_{el}, CONNECTOR_{el}, BEHAVIOR_{el} \rangle$$

The lists of elements in the PerFAM metamodel are then defined as follows:

1. $PERFORMANCE_{el} \in ELEMENT$ is a set of performance elements that build the metamodel. Let $per \in PERFORMANCE_{el}$, the performance element is defined as

$$[\![PERFORMANCE_{el}]\!] = \{PWorkload_{per}, PWorkloadEvent_{per}, PResource_{per}\}$$

2. The set of reliability elements provides internal information for performance failure in PerFAM. Let $rel \in RELIABILITY_{el}$

$$[\![RELIABILITY_{el}]\!] = \{FFault_{rel}, FBehaviorState_{rel}, internal_{rel}, external_{rel}\}$$

3. The joint sets of performance and reliability elements; $CONNECTOR_{el}$ is a set of connectors specifying the relationship between elements. Let $con \in CONNECTOR_{el}$

$$[\![CONNECTOR_{el}]\!] = \{WAssign_{con}, PStep_{con}, FExpress_{con}, RUsage_{con}\}$$

4. $BEHAVIOR_{el} \in ELEMENT$ is a set of dynamic behaviors. We assume that it specifies operational behaviors such as scenario and activity. Let $bhv \in BEHAVIOR_{el}$

$$[\![BEHAVIOR_{el}]\!] = \{PScenario_{bhv}, Activity_{bhv}, Start_{bhv}, End_{bhv}, Control_{bhv}, Fork_{bhv}, Join_{bhv}\}$$

From Definition 4, the PerFAM metamodel M_{PerFAM} semantic clearly distinguished the domain concepts for performance and reliability, as well as its behavior through a connector. The fragmentation of metamodel elements can be clearly seen to be associated between domain concepts and metamodel elements in general.

5.1.2. Action in PerFAM Metamodel: Action in a PerFAM metamodel M_{PerFAM} depicts its association described in Section 5.1.1. The action $act \in ACTION$ describes the behavior of the model, $BEHAVIOR_{el} \in \mathrm{M}_{PerFAM}$ with the subtle association between elements, $CONNECTOR_{el} \in \mathrm{M}_{PerFAM}$.

$$act: ACTION_c \rightarrow BEHAVIOR_{el}$$
$$\mathrm{M}_{PerFAM}(PB) = \left\{\exists act \rightarrow PERFORMANCE_{el} \xrightarrow{ASC_{con \in CONNECTOR_{el}}} BEHAVIOR_{el}\right\}$$
$$\mathrm{M}_{PerFAM}(BR) = \left\{\exists act \rightarrow BEHAVIOR_{el} \xrightarrow{ASC_{con \in CONNECTOR_{el}}} RELIABILITY_{el}\right\}$$
$$\mathrm{M}_{PerFAM}(RP) = \left\{\exists act \rightarrow RELIABILITY_{el} \xrightarrow{ASC_{con \in CONNECTOR_{el}}} PERFORMANCE_{el}\right\}$$
$$\mathrm{M}_{PerFAM}(BB) = \{\exists act \rightarrow (BEHAVIOR_{el})^n\}$$

The defined actions act in the PerFAM metamodel M_{PerFAM} establish the relationship between performance and reliability elements via an association as annotated with $CONNECTOR_{el}$ elements.

5.2. A Formal Definition for PerFAM Model

Once the PerFAM metamodel semantics are provided, we then define a formal semantic for the PerFAM model $\mathcal{M}$ (concrete syntax), such that the PerFAM model $\mathcal{M}$ is realized from the PerFAM metamodel M_{PerFAM} from its corresponding metamodel realization $\mathbb{A}(\mathrm{M}_{PerFAM})$. The PerFAM model $\mathcal{M}$ for the PerFAM metamodel M_{PerFAM} is defined within the extensional semantics of M_{PerFAM} and denotes as:

$$[\![\mathrm{M}_{PerFAM}]\!] = \{\mathcal{M} | \mathcal{M}: \mathrm{M}_{PerFAM}\}$$

The conformance between metamodel and model is defined as $\mathcal{M}: \mathrm{M}_{PerFAM}$ and each model of the PerFAM metamodel, $\mathcal{M}$ semantically inherits all the elements and actions in metamodel M_{PerFAM}. Based on the theory in [3], the metamodel realization is defined as

$\mathbb{A}$ ($\mathbb{M}$) so as to define the realization of metamodel M_{PerFAM}within the context of MOF. Thus, we define our metamodel realization as:

$$\mathcal{M}\colon \mathrm{M}_{PerFAM} \Leftrightarrow \mathcal{M} \in \mathbb{A}\,(\mathrm{M}_{PerFAM})$$

Based on the framework for semantic declaration as depicted in Figure 3, the PerFAM model $\mathcal{M}$ is realized by applying the metamodel realization $\mathcal{M} \in \mathbb{A}\,(\mathrm{M}_{PerFAM})$ where a set of PerFAM model semantically inherits all elements in PerFAM metamodel M_{PerFAM}. In order to provide an ease of use of the PerFAM model, we provide a formal semantic for the PerFAM model within the context of $\mathcal{M} \in \mathrm{M}_{PerFAM}$.

Definition 5. (Concrete syntax)

The PerFAM model $\mathcal{M}$ is defined within the context of PerFAM metamodel, M_{PerFAM}. Formally defined, the PerFAM model is a 7-tuple

$$\mathcal{M} \stackrel{\text{def}}{=} \langle \boldsymbol{S^0}, \boldsymbol{E^0}, \boldsymbol{Act}, \boldsymbol{Fb}, \boldsymbol{Res}, \boldsymbol{Wrk}, \boldsymbol{Asc} \rangle$$

where

- $\boldsymbol{S^0}$ is an initial state, $\boldsymbol{S^0} = BEHAVIOR_{el} \in \mathrm{M}_{PerFAM}$
- $\boldsymbol{E^0}$ is an final state, $\boldsymbol{S^0} = BEHAVIOR_{el} \in \mathrm{M}_{PerFAM}$
- $\boldsymbol{Act}$ is the Activity state, $\boldsymbol{Act} = BEHAVIOR_{el} \in \mathrm{M}_{PerFAM}$
- $\boldsymbol{Fb}$ is a failure state, $\boldsymbol{Fb} = RELIABILITY_{el} \in \mathrm{M}_{PerFAM}$ represents the failure behavior for $\boldsymbol{Act}$
- $\boldsymbol{Res}$ is a set of resource, $\boldsymbol{Res} = PERFORMANCE_{el} \in \mathrm{M}_{PerFAM}$, presenting the resources used by a system in which either being hardware or software resources.
- $\boldsymbol{Wrk}$ is a workload of the system, $\boldsymbol{Wrk} = PERFORMANCE_{el} \in \mathrm{M}_{PerFAM}$
- $\boldsymbol{Asc}$ is a partial sequence of action, $\boldsymbol{Asc} = \mathrm{M}_{PerFAM}(act)$ consists of two logical sequence called Logical Linear Sequence (LLS) and Logical Failure Sequence (LFS).

For all $el \in ACTIVE$.

5.2.1. Operation State: The PerFAM model $\mathcal{M}$ Operation Guide states that $OS_{\mathcal{M}}$ complies with the predefined behavior in the PerFAM model, $\mathcal{M}$. The behavior in PerFAM model $\mathcal{M}$ can de divided into two main operations, namely, LLS and LFS.

$$OS_{\mathcal{M}} = LLS \bigcap_{Asc} LFS\colon \mathrm{M}_{PerFAM}(act)$$

The LLS is responsible for executing the list of Activities, $\boldsymbol{Act}_n$ that compromise LFS. The term 'compromise' means that the LFS is a composite operation state of LLS, where each Activity, $\boldsymbol{Act}_n$ in LLS consists of LFS elements and can be denoted as

$$PerFAM\ operation\ state, OS_{\mathcal{M}} = \begin{cases} Asc_{LFS} \in LFS \rightarrow Composite_{Asc_{LLS} \in LLS} \\ ASC_{LFS} \rightarrow' ASC_{LFS}, for\ \forall ASC_{LFS} \in Composite_{Asc_{LLS}} \end{cases}$$

The operation state, Asc_{LLS} executes all the sequential paths of system scenarios with applyication of its composite operation state, Asc_{LFS}. With this configuration, all possible

failure behaviors of Activity, $\boldsymbol{Act}_n$ in LLS can be derived through the operation state, Asc_{LFS}. The details of operation will be explained in Sections 5.2.3 and 5.2.4 respectively.

5.2.2. System Configuration State: In PerFAM model $\mathcal{M}$, the system configuration state for a system behavior denotes the current state of the system. The current state is established when the system is executed and how the possible failure reacted within the predefined requirements. The first system configuration state is a *failure state*, $LFS \in OS_{\mathcal{M}}$,where this configuration state outlines the possible failure behavior state in Activity, $Activity_{bhv}$.

$$\boldsymbol{LFS_n} = \{\boldsymbol{Fb}_n \cap \boldsymbol{Res_n} | \boldsymbol{Fb} \in \mathcal{M}, \text{Res} \in \mathcal{M}\} \rightarrow \boldsymbol{OS}_{\mathcal{M}}$$

Each Activity, $Activity_{bhv}$ is defined as $\boldsymbol{LFS_n}$ where $n = 1,2,3 \ldots n$ and consists of failure state, $Fb \in FBehaviorState_{rel}$ and respective resource, $Res \in PResource_{per}$. The failure of Activity and Resource is said to be independent of $\boldsymbol{Fb}_n \cap \boldsymbol{Res_n}$,as each failure event in Activity and Resource do not reflect upon each other.

The second system configuration state is called the *system state*. The system state depicts the overall system environment and subsequently organizes the respective failure state at failure state, $LFS \in OS_{\mathcal{M}}$ to comply with all the active model elements, $SysState_{el} \in ACTIVE$.

5.2.3. Logical Linear Sequence (LLS): A set of primary sequential steps for LLS is a deterministic transition that correlates with the transition between Activities, $Activity_{bhv}$. Each Activity, $Activity_{bhv}$ is connected by the specific connector, $PStep_{con} \in PerFAM_{el}$. The sequence of transition is motivated by the UML sequence diagram for message transition in order to execute a system scenario. Let the formal semantic for LLS is defined as:

$$LLS = LLSasc: Act_1 \times Act_2 \times \ldots \times Act_n | LLSasc \in ASC_{LLS}, PStep_{con}, Act \in Activity_{bhv}, \mathcal{M}$$

where the instance of LLS association, $LLSasc$ defined the sequential transition for an Activity Act_n. The LLS is a sequential process in which the value for the whole $LLSasc$ is represented as one cycle scenario execution of the system.

5.2.4. Logical Failure Sequence (LFS): Passing the LLS to LFS is a straightforward task, as the LFS is a composite sequence for LLS, $LFS \rightarrow Composite_{Asc_{LLS} \in LLS}$ in a PerFAM operation state, $OS_{\mathcal{M}}$. Let the sequence of LFS is defined as:

$$LFS_{Act_i} = LFSasc_i: Fb_n \times Res_n^i | LFSasc \in ASC_{LFS}, FExpress_{con}, RUsage_{con}, Fb, Res \in \mathcal{M}$$

where the instance of LFS association, $LFSasc$ denotes the sequential expression for Activity failure behavior, Fb_n through the related resources, Res_n.

In conclusion, the operation state for a PerFAM model consists of LLS and LFS. The *Step* connector, $PStep_{con}$ describes the sequential process of *Activity* in order to realize the system scenario. Each *Activity,* Act_n poses its own failure behavior as expressed by *Express* connector, $FExpress_{con}$. This connector establishes a relationship between *Activity* and its failure behavior, Fb_n. Thus, each *Activity* can also incur specified resources through *Usage* connector, $RUsage_{con}$. Both *Express* and *Usage* connector are a subset of LFS.

In order to express the operational state of *LFS* in LLS sequence, let all the combinations for possible *Activity* in LLS reflect *n* resources where:

$$os_{\mathrm{n}}: \{(\boldsymbol{Act_1}(\boldsymbol{Fb_1} \times \boldsymbol{Res_1}) \times (\boldsymbol{Act_2}(\boldsymbol{Fb_2} \times \boldsymbol{Res_2}) \times \ldots \times (\boldsymbol{Act_n}(\boldsymbol{Fb_n} \times \boldsymbol{Res_n}) | os_{\mathrm{n}} \in OS_{\mathcal{M}}\}$$

5.3. Model Conformance

Once the formal semantics of the PerFAM metamodel and model are formally defined, the next stage is to define the semantic of model conformance between the former and the latter. The model conformance, $\boldsymbol{\mathcal{M}}: \mathbf{M}_{\boldsymbol{PerFAM}}$ defines all the sets of PerFAM model $\mathcal{M}$ and is developed within the aspects of PerFAM metamodel M_{PerFAM}. The benefit of model conformance is that the traceability between metamodel and its model can be assured correctly by checking with its metamodel realization. Consequently, the consistency mapping between PerFAM model $\mathcal{M}$ and PerFAM metamodel M_{PerFAM} is traced within the context of $\mathrm{M}_{PerFAM}(act)$ and can be semantically denoted as:

$$\forall \mathcal{M} \rightarrow \forall \mathrm{M}_{PerFAM}$$

where all model and metamodel elements' relationship can be traced back based on Definition 4 and Definition 5.

In conclusion, the formal semantic of PerFAM metamodel M_{PerFAM} was defined based on algebraic semantics for the MOF framework. This framework consists of three stages, these being: metamodel, model and model conformance. The metamodel declaration is further introduced with two notions: namely, metamodel realization and model types. The metamodel realization is a systematic definition for a metamodel semantic using mathematical representation within the MEL theory for describing model types. Thus, the model semantic was defined based upon two aspects: domain concepts and action (behavior). Once the metamodel and model semantic are provided, the model conformance semantic is developed to establish a traceability between both metamodel and model elements so as to ensure the traceability is correct.

6. Motivation Example

The example we used in this paper shows how to realize a PerFAM model from its metamodel and how to check the consistency between the correspondence metamodel and a model using model conformance. The example we used is from [5] as shown in Figure 4.

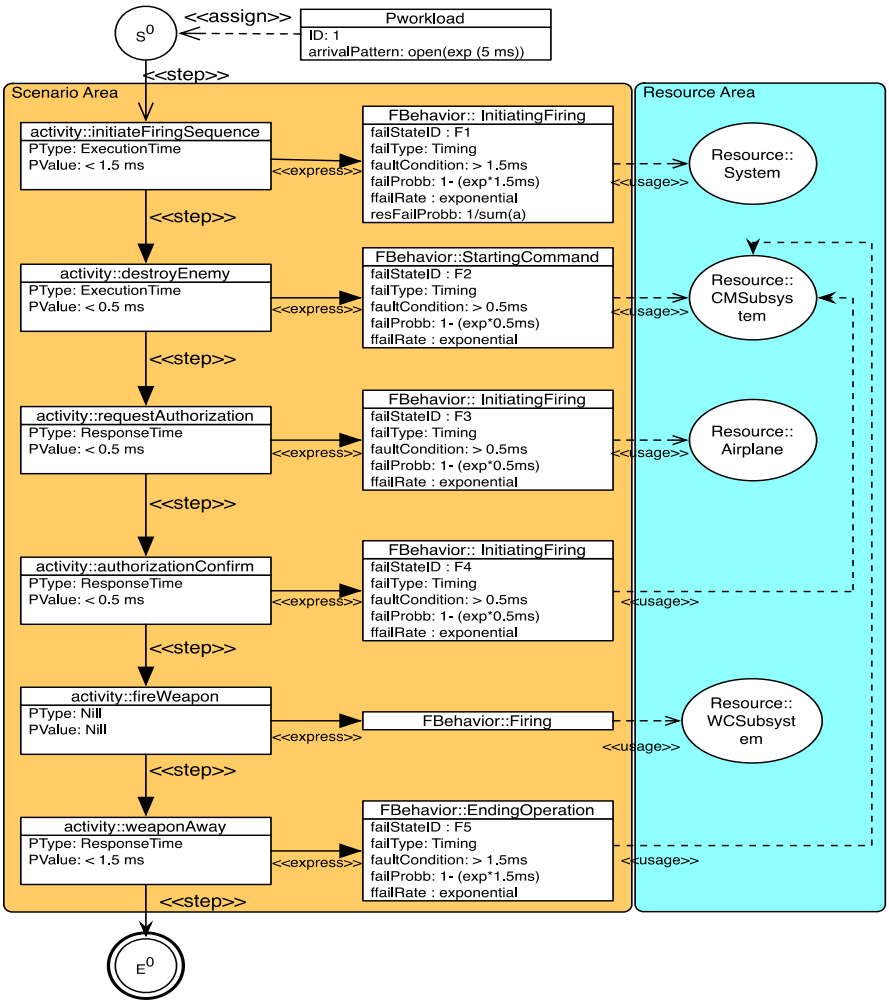

Figure 4. Example of PerFAM Model [5]

Figure 4 denotes an example of a PerFAM model for destroying an enemy scenario from the Aircraft Defense Management System [16]. This system is responsible for managing a defense system and consists of a number of functions including destruction of an enemy function.

The model consists of 6 *Activities* which build up the LLS sequential scenario with an implicit composite failure behavior, LFS. Each Activity is associated with the allocated resources that are responsible to support the Activity operation with a specific timing allocation. From the PerFAM model shown in Figure 5, we can trace the conformity of PerFAM model $\mathcal{M}$ with its metamodel M_{PerFAM} by executing one-to-one mapping with the metamodel realization semantic, as well as with the model and model conformance semantic. Figure 5 to Figure 7 show this conformity checking for a PerFAM model by conducting one-to-one mapping from a PerFAM model with its defined semantic in the shape of abstract and concrete syntaxes. The abstract syntax mapping is based on action semantics (Section 5.1.2) which denotes how the PerFAM metamodel elements relate to each other in building up the system scenario. In addition, the concrete syntax mapping is based on the PerFAM model semantic (Section 5.2, Definition 5).

In Figure 5, an initial scenario is starting up with the workload allocation, *PWorkload* to the initial state of the system, $\boldsymbol{S^0}$**.** This initial scenario in the PerFAM model is then mapped to its semantic in order to check the model conformance for its abstract and concrete syntaxes. As for the abstract syntax, all elements for *PWorkload*, *assign* and $\boldsymbol{S^0}$ conform to the PerFAM metamodel element ($PERFORMANCE_{el}, BEHAVIOR_{el}, CONNECTOR_{el}$) with consideration given to metamodel action $\mathrm{M}_{PerFAM}(PB)$.

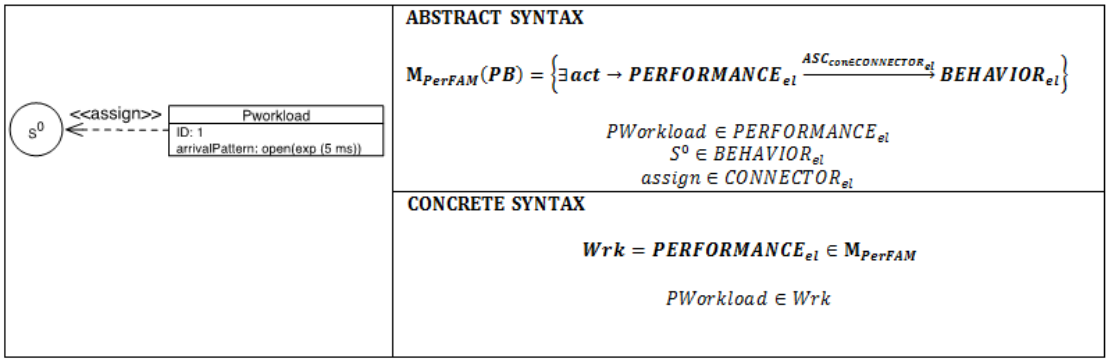

Figure 5. Mapping from Performance to Behavior Element

For the next system scenario, as shown in Figure 6, the sequential elements of *Activity* for the system (initiate Firing Sequence, destroy Enemy, request Authorization, authorization Confirm, fire Weapon, weapon Away) build up the actions for $\mathrm{M}_{PerFAM}(BB)$ where all elements conform to its metamodel elements ($BEHAVIOR_{el}, CONNECTOR_{el}$). As for the concrete syntax, the sequential elements conform to the *LLSasc* association that correlates all *Activity* in a sequential operation of the PerFAM model, LLS.

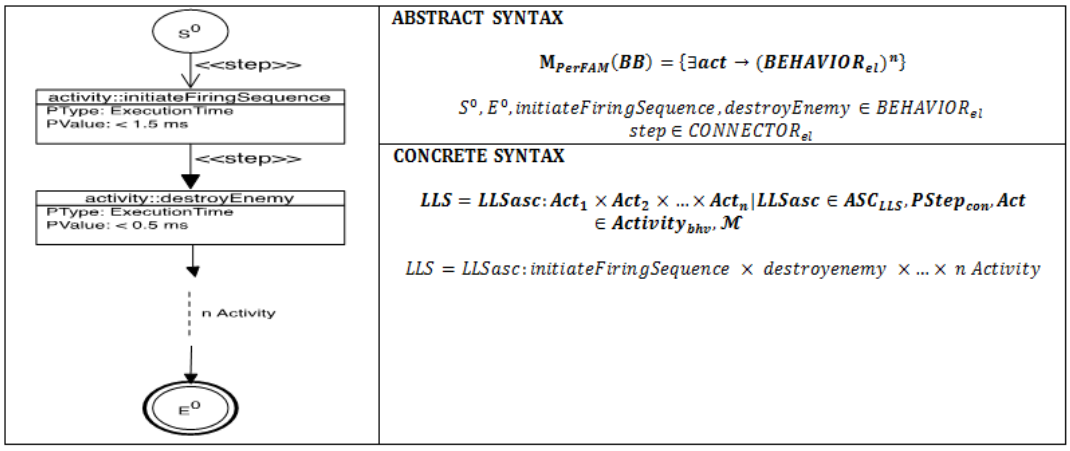

Figure 6. Mapping from Behavior to Behavior Element

Based on the description in Section 5.2.1, the *LFS* operation consists of composite operations for failure behavior interpretation called *LFS*. As shown in Figure 7, the *Activity* elements, *initiateFiringSequence* will express its failure behavior, *InitiatingFiring* with the support of *System* resource, where this operation is connected

to <<*express*>> and <<*usage*>> connectors respectively. The semantic of this action conforms with its abstract and concrete syntaxes as denoted in Figure 7.

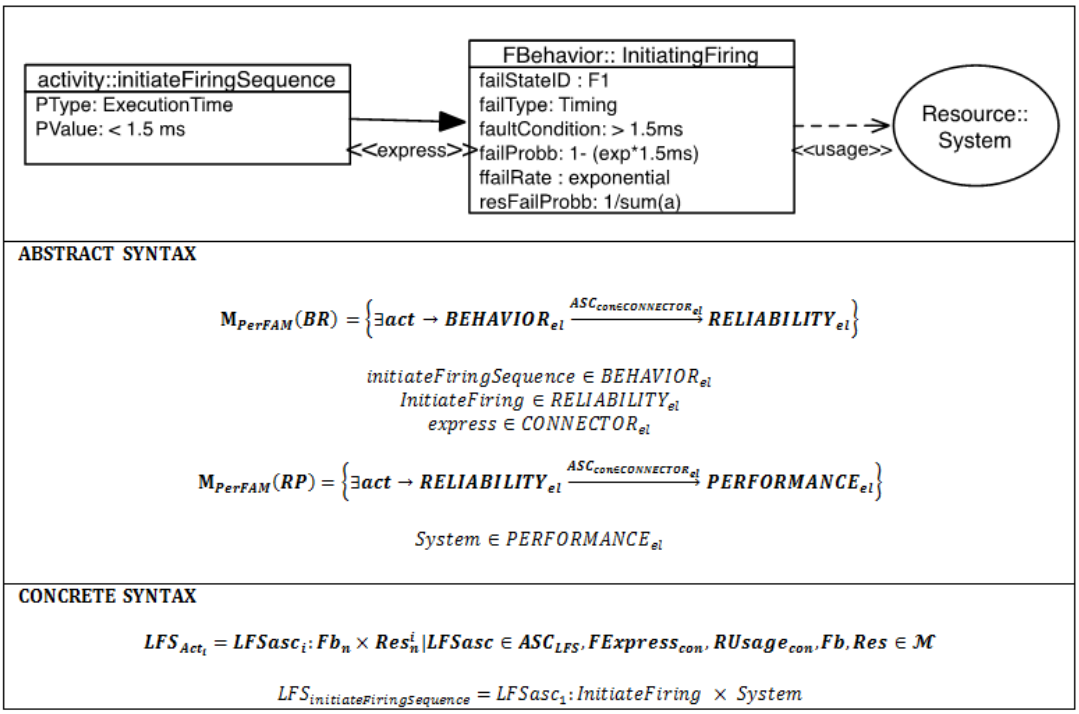

Figure 7. Mapping from Behavior to Reliability to Performance Element

From the brief explanations for one-to-one mapping regarding semantic conformity of a PerFAM model as depicted in Figure 5 to Figure 7, we can conclude that all the action in the PerFAM model in Figure 4 is semantically commensurate with the PerFAM metamodel and model semantic, as described in Section 5. Therefore, the operation state $OS_{\mathcal{M}}$ of the PerFAM model can be concluded as

$$OS_{\mathcal{M}} : LLS \bigcap_{Asc} LFS = TRUE$$

for all PerFAM models.

7. Discussion

From the example, it can be seen that once the PerFAM is formally defined in the context of metamodel and model semantics, the consistency of the model generation can easily be traced back to its corresponding metamodel through metamodel realization formal semantics. Each element in the PerFAM model in Figure 4 conforms to its corresponding metamodel in Figure 1. Also, the formal semantics defined in Section 5 are very helpful in gaining the accuracy of the developed model.

The motivation example shown in Section 6 enables consistency checking between PerFAM model inheritances towards its metamodel. This consistency checking is subjected to the model semantic through structural and behavioral characteristics of the PerFAM metamodel and model. First, we executed one-to-one mapping from the PerFAM model into the abstract and concrete syntaxes. The mapping from model to abstract syntax is

straightforward, where the behavior of the PerFAM model is mapped into the Action semantic, as described in Section 5.1.2, along with its structural semantic. Similarly, the mapping from model to concrete syntax investigates the behavior of the PerFAM model which is subjected to LLS and LFS behavior.

By applying the semantic framework [3] in our model for a formal semantic declaration, we can easily trace the conformance of a model and its metamodel in terms of its structural and behavioral compliance, as well as ensuring the accuracy of the development of the model.

8. Conclusion

Defining a precise semantic for a metamodel based on the MOF is a crucial task. The MOF is a generic framework offering development for a wide range of modeling languages. However, the MOF approach does not provide a formal semantics declaration for each developed metamodel.

The main contribution of the paper is a systematic approach for defining the PerFAM formal semantics by partitioning the PerFAM metamodel, the PerFAM model and its model conformance. In this respect, we use the algebraic semantics for MOF framework to define a formal semantic for PerFAM where this framework is established within three semantic notions, namely, metamodel, model and model conformance. The metamodel semantic describes a set of model types through metamodel realization in a mathematical representation. In order to ensure the consistency of model generation from its corresponding metamodel, a model conformance is defined as checking the consistency of its syntax and semantic. The main advantages of our contributions are the formal semantics that we proposed which can be used to check the correctness of the developed model, as well as its model conformance. In particular, these formal semantics open up a wide opportunity for model-driven development in terms of tools support for model development.

In a future work, we plan to develop a method to formalize the consistency checking of PerFAM model developments based on its metamodel in terms of model inheritance and attributes mapping. In addition, we plan to develop a tool to systematically check the consistency between a PerFAM metamodel and models during the system design process.

Acknowledgements

We would like to thank the Malaysia Ministry of Higher Education for sponsoring the research.

References

[1] P. -A. Muller, F. Fondement, B. Baudry and B. Combemale, “Modeling modeling modeling”, Software & Systems Modeling, no. Computer Science, (2010) August, pp. 1-13.
[2] Meta Object Facility (MOF) 2.0 Core Specification, **(2006)** January, www.omg.org.
[3] A. Boronat and J. Meseguer, “An algebraic semantics for MOF”, Formal Aspects of Computing, vol. 22, no. 3, **(2010)** February, pp. 269-296.
[4] M. Broy and M. V. Cengarle, “UML formal semantics: lessons learned”, Software & Systems Modeling, vol. 10, no. 4, **(2011)** June, pp. 441-446.
[5] M. A. Isa, D. N. A. Jawawi and M. Z. M. Zaki, “An intermediate metamodel for failure-based behavior of performance and reliability”, Software Engineering (MySEC), **(2011)**, pp. 234-239.
[6] L. Dan and L. Danning, “Towards a Formal Behavioral Semantics for UML Interactions”, 2010 Third International Symposium on Information Science and Engineering, **(2010)** December, pp. 213-218.
[7] C. Eichner, H. Fleischhack and R. Meyer, “Compositional semantics for UML 2.0 sequence diagrams using Petri nets”, in SDL 2005: Model Driven, Lecture Notes in Computer Science, **(2005)**, pp. 1260-1263.
[8] J. Kong, K. Zhang, J. Dong and D. Xu, “Specifying behavioral semantics of UML diagrams through graph transformations”, Journal of Systems and Software, vol. 82, no. 2, **(2009)** February, pp. 292-306.

[9] M. Lund, "A fully general operational semantics for UML 2.0 sequence diagrams with potential and mandatory choice", FM 2006: Formal Methods, **(2006)**, pp. 380-395.
[10] M. S. Lund, A. Refsdal and K. Stølen, "Semantics of UML Models for Dynamic Behavior A Survey of Different Approaches", in MBEERTS, LNCS 6100, **(2010)**, pp. 77-103.
[11] C. S. Karol, "Applying attribute grammars for metamodel semantics", Proceedings of the International, no. 1, **(2010)**, pp. 1-5.
[12] S. Kuske, M. Gogolla, H. -J. Kreowski and P. Ziemann, "Towards an integrated graph-based semantics for UML", Software & Systems Modeling, vol. 8, no. 3, **(2008)** August, pp. 403-422.
[13] J. Kohlmeyer, "Unifying the semantics of UML 2 state, activity and interaction diagrams", Perspectives of Systems Informatics, **(2010)**, pp. 206-217.
[14] S. Staton, "General structural operational semantics through categorical logic", in Logic in Computer Science, 2008, 23rd Annual IEEE Symposium on LICS'08, **(2008)**, pp. 166–177.
[15] T. Clark and A. Evans, "The metamodelling language calculus: foundation semantics for UML", Fundamental Approaches to Software, **(2001)**, pp. 17-31.
[16] B. P. Douglass, "Real Time UML: Advances in the UML for Real-Time Systems", Addison Wesley Publishing Company, Reading, MA, **(2004)**.

Authors

Mohd Adham Isa received the M.Sc in Computer Science from University Teknologi Malaysia in 2009. He is pursuing the Ph.D degree in the same institution. His current research interests are software quality analysis, software modeling and software measurement.

Dayang Norhayati Abang Jawawi received the Ph.D in Computer Science from University Teknologi Malaysia in 2006. She is now a senior lecturer at Software Engineering Department at University Teknologi Malaysia. Her research interest includes real-time system, component-based system, software quality and software modeling.

Mohd Zulkifli Mohd Zaki received the B.Sc in Computer Science from University Teknologi Malaysia in 2008. He is pursuing the M.Sc degree in the same institution. His current research interests include software quality analysis, software modeling and software measurement.