

Modeling and Evaluation of Performance and Reliability of Component-based Software Systems using Formal Models

Fakhraddin Farjaminejad
Department of Computer,
Science and Research branch,
Islamic Azad University,
Ardabil, Iran

Ali Harounabadi
Department of Computer
Engineering, Central Branch,
Islamic Azad University,
Tehran, Iran

Sayed Javad Mirabedini
Department of Computer
Engineering, Central Branch,
Islamic Azad University,
Tehran, Iran

Abstract: Validation of software systems is very useful at the primary stages of their development cycle. Evaluation of functional requirements is supported by clear and appropriate approaches, but there is no similar strategy for evaluation of non-functional requirements (such as performance and reliability). Whereas establishing the non-functional requirements have significant effect on success of software systems, therefore considerable necessities are needed for evaluation of non-functional requirements. Also, if the software performance has been specified based on performance models, may be evaluated at the primary stages of software development cycle. Therefore, modeling and evaluation of non-functional requirements in software architecture level, that are designed at the primary stages of software systems development cycle and prior to implementation, will be very effective.

We propose an approach to evaluate the performance and reliability of software systems, based on formal models (hierarchical timed colored petri nets) in software architecture level. In this approach, the software architecture is described by UML use case, activity and component diagrams, then UML model is transformed to an executable model based on hierarchical timed colored petri nets (HTCPN) by a proposed algorithm. Consequently, upon execution of an executable model and analysis of its results, non-functional requirements including performance (such as response time) and reliability may be evaluated in software architecture level.

Keywords: software architecture; colored petri nets; object-oriented design; non-functional requirements; unified modeling language; performance modeling

1. INTRODUCTION

Within the recent decades, the software complexities have been increased day to day and demands for more powerful and high quality software have been increased. Therefore, software development based on principles and methodologies that in addition to reduction of costs, meet all expected features of shareholders (functional and non-functional requirements) seems to be necessary. Establishing non-functional requirements in software engineering was raised recently whilst they have considerable effect on success of software systems. Software Architecture (SA) is established at the first stages of design and has a significant effect on access to nonfunctional requirements of software system. Therefore, establishment of an executive model of SA and evaluation of nonfunctional requirements thereby is a cheap solution for prevention of time and cost waste for achieving the qualitative goals for development of software systems.

Unified modeling language (UML) is a semiformal and standard language for easy description of software, but performance and reliability of SA may not be evaluated thereby. Therefore, to evaluate the performance and reliability, pragmatic model (UML) must be transformed to formal model (HTCPN).

HTCPN is very suitable for displaying the behavior of systems with concurrent and interactive components and in addition to having a virtual structure and behavior have the capability of graphic display, hence modeling by them is easy. Moreover, these nets provide a framework for analysis, validation and evaluation of nonfunctional requirements such as performance and reliability in complex systems [10].

In this paper, we have proposed an approach, therein SA is described by UML Use Case, Activity and Component diagrams. Then, the required information related to non-functional requirements are annotated to these diagrams as stereotypes and tags. In continue, an algorithm is offered which has been assumed from UML diagrams and establishes the

executive model based on HTCPN. Ultimately, upon execution of this model and analysis of its results, nonfunctional requirements including performance (such as response time) and reliability may be evaluated in SA level.

The next sections of this paper have been organized as follows: In section 2, a general description of performance modeling in UML and hierarchical timed colored petri net models has been presented. In section 3, some works related to paper subjects have been reviewed and in section 4, procedure together with details are described. In section 5, a case study is investigated for evaluation of suggested method and ultimately in section 6, a general conclusion of suggested method is explained.

2. BACKGROUND

In this section, a general description of performance modeling in UML and timed colored petri net models is presented.

2.1 Performance Modeling in UML

The UML may describe the behavioral and structural aspects of SA. But in order to evaluate the SA, features of SA are required that UML has not the capability of their exhibition. Accordingly, a strategy has been presented by OMG including profiles consisted of stereotypes, tags and limitations that provide the capability of exhibiting these features for UML. These profiles and their complete details have been explained in [11].

2.2 Timed Colored Petri Nets

Colored petri nets are used for formal description of activities flow in the complex systems and provide the requirements of concurrency and parallelism exhibition. Classic petri nets are not suitable for modeling the systems with large space or a complex temporary behavior. In these cases, we must use a developed petri net model having color and time. This model is the base of a framework that is used for solving the problems related to design and control in complex systems.

In these networks, the concept of time is introduced by global element called global time. The values selected by this time explains the model time. This model may be an integral number that indicates the discrete time or maybe a true number explaining the continuous time. This value of time that is pertained to each token is referred to as stamp time that indicates the first time of model therein token may be used. As a result, these nets will be appropriate for evaluation of qualitative requirements (response time etc.) and reliability in SA. Specifications of colored petri nets and its different types have been explained in [7].

3. RELATED WORKS

In the past various techniques have been presented for evaluation of nonfunctional requirements of performance and reliability in component-based software systems. In general, these techniques may be divided in various groups. Some of these groups such as models based on state and path may be referred that commonly are used for evaluation of reliability. It is notable that in plenty of these methods, evaluation of performance and reliability nonfunctional requirements have not been considered at the primary stages of software systems development cycle and prior to implementation stage. For instance, in place-based models [5, 9], control graph is used to describe the software architecture, so that control graphs are commonly extracted from programs code source. Therefore, applying these techniques will be possible after implementation stage. There are some other methods that provides the prerequisites for evaluation of performance or reliability nonfunctional requirements at the design stage. It is notable that in most of these methods, no integrated executive model has been used for evaluation of performance and reliability of nonfunctional requirements and their analysis. In continue, some of these methods are raised.

Zhu and Wang [13] have introduced a platform for evaluation of software systems performance by UML and hierarchical timed colored petri net. In this method, the software system is modeled by UML Use Case, Collaboration and Deployment diagrams and then these models are transformed to a hierarchical timed colored petri net model. Consequently, an evaluation of software system performance is provided.

Fukuzawa and Saeki [4] presented a method therein software architecture is described by UML Component diagram. Then, the above algorithm has been transformed to colored petri net by an algorithm and ultimately the performance is evaluated, so that the own component and its connector are transformed to a colored petri net but its interface is transformed to a place of colored petri net.

Balsamo and Marzolla [2] presented a method therein software architecture is described by UML Use Case, Activity and Deployment diagrams, then operational profiles related to performance are annotated therein. Ultimately, to evaluate the performance, UML diagrams are transformed to an executive model based on Queuing Networks.

Balsam et al [1, 3] presented a method therein software architecture is described by UML Use Case, Collaboration and State diagrams, in this method, to evaluate the performance of software architecture, an executable model based on generalized stochastic petri nets is established. The main objective in this method is transforming the State diagram and each one of objects of Collaboration diagram to Petri Net that a stochastic petri net is established upon combination thereof and indicates the whole system.

In general, whereas the mentioned methods may provide better basic techniques for evaluation of performance or reliability of software system, but a few ones may be used at the primary stages of software systems development cycle and prior to implementation. In addition, in these methods, there is no unified model for evaluation of performance and reliability simultaneously. Therefore, the techniques will be appropriate and important that use an executive model for evaluation of several nonfunctional requirements (such as performance, reliability).

Therefore the main objectives of technique presented in this paper are as follows:

- Development of a method based on probability for evaluation of reliability in SA Level and prior to implementation stage;
- Capability in studying the performance and reliability of components and connectors so that the system architect is enabled to use an implementer (collection of activities) superseding the components, in case of non-complying with appropriate performance and reliability features;
- Establishing a unified model for evaluation of performance and reliability of nonfunctional requirements simultaneously.

4. THE PROPOSED METHOD

The main approach in this paper is establishing the HTCPN-based execution model of UML diagrams and evaluation of performance (such as response time) and reliability of nonfunctional requirements of software architecture.

For this purpose, firstly the SA is described by UML, later operational profiles related to performance and reliability features are annotated therein. In continue, an algorithm is offered for transformation of UML model to HTCPN model and ultimately the said nonfunctional requirements are evaluated by suggested techniques at the SA level.

4.1 Description of Software Architecture by UML Diagrams

Different methods has been presented for description of SA by UML. In this paper, to describe the SA, a UML-based method is used, therefore UML Use Case, Activity and Component diagrams are applied for description of SA structure and behavior. In continue, the effect of diagrams and annotations related to performance and reliability therein is explained.

4.1.1 The Role of Use Case Diagram and Annotation of Performance Specification Therein

Use case diagram describes the functional requirements of system and interaction between system and environment [12]. In this paper, this diagram is used for exhibition of functional requirements and working load applied to the system in SA description. Annotations related to performance in this diagram are related to actors that requesting service from system.

The actors indicating a sequence of unlimited requests out of system are annotated by "PAopenLoad" stereotype and actors indicating a fixed population of requests from system are annotated by "PAClosedLoad" stereotype.

“PAClosedLoad” stereotype has a tag called PAoccurrence that indicates the interarrival time between two subsequent requests. “PAClosedLoad” stereotype has two tags named PApopulation and PAextDelay that respectively indicates “the number of requests” and “the time spent by each completed request before the next interaction with the system”. An annotated use case diagram is exhibited in figure 1.

4.1.2 The Role of Activity Diagram and Annotation of Performance Specifications Therein

Activity diagrams are used for description of further details of contents of each use case. This diagram specifies the sequence, order and conditions of operation execution. The operation sequence from beginning to the end is referred to as an activity performed by the system [12]. In this paper, this diagram is used for exhibition of activities flow inside and out of components. Showing the boundaries of each component in activity diagram is not possible but it may be shown by Swimline of activity diagram.

Annotations related to performance and reliability in this diagram are related to actions. In this diagram, each action is annotated by “PAstep” stereotype that indicates the service demand from an active source of system.

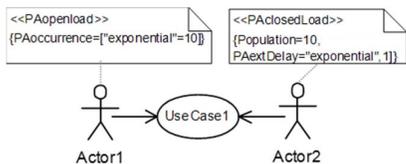
“PAstep” stereotype has two tags named PADemand and failprob that respectively indicates “service demand” and “failure probability in actions”. It is notable that failprob tag has not been defined based on OMG standard for “PAstep” stereotype, but here “PAstep” stereotype tag has been placed for easy to show that. Transitions of activity diagrams are annotated by PAprob that indicates the probability of applying a specified transition and is signified when we have several output transitions from an action; in this mode, sum total of transitions probability outputted from same action must be equal to 1. In figure 1, an annotated activity diagram has been shown.

4.1.3 The Role of Component Diagram and annotation of Performance Specifications Therein

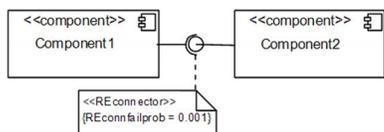
The component diagram indicates the logical structure of a software system. In addition, each component may use the services provided by other components. Services provided by each component is accessible by its interfaces [12]. In this paper, this diagram is used for describing the logical structure of software system, relationship between components and show the interfaces of each component.

Annotations related to reliability in this diagram are related to the connectors. In this diagram, each connector is annotated by “REconnector” stereotype that indicates the reliability specifications in connectors.

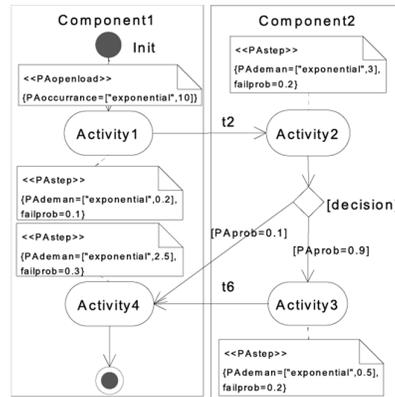
“REconnector” stereotype has a tag named REconnfailprob that indicates the probability failure in connector. An annotated component diagram is shown in figure 1.



(a) Annotated UML Use Case Diagram



(b) Annotated UML Component Diagram



(c) Annotated UML Activity Diagram

Figure 1. Annotated UML Diagrams

4.2 Suggested Algorithm for Transformation of Annotated UML Model to HTCPN Model

Our suggested algorithm has been assumed for transforming annotated UML model to HTCPN-based executive model according to a three stage design including as below:

First step: Transformation of component diagram to CPN model

At the first stage, the component diagram is transformed to a cpn model. For this purpose, each component is transformed to a sub cpn, each interface to a place and each connector to two transitions, so that one of these transitions is used for transmitting the request and the other for receiving. In table 1, transformation maps are shown.

According to table 1, cpn model related to components consists of two places and one transition. The places are applied as component interfaces, and transition as component implementer.

Second stage: Determination of hierarchical structure

At this stage, the activities to be performed in the components are specified; in other word, at this stage, each component is implemented. The activity diagram of each component is transformed as per procedure provided at third stage.

Third stage: Transformation of activity diagram to CPN model

At this stage, the activity diagram is transformed to a CPN model. Procedure of transforming activity diagram to CPN model will be in accordance with method presented in [8]. Therefore, each action and transition in activity diagram is transformed to one transition and place respectively in cpn. Procedure of transforming branching, joint and fork nodes has been provided in table 2.

Ultimately, considering the stages of suggested algorithm, the final HTCPN model established from UML model as per figure 1 will be as HTCPN model shown in figure 2. It is notable that the different demand classes of a component are related to different colors in HTCPN model.

Table 1. The general notation mapping in UML component diagram to cpn model

Item Name	Component Diagram notation	Association CPNs notation
Component		
Connector		

Table 2. The general notation mapping in UML activity diagram to cpn model

Item Name	Activity Diagram Notation	Association CPNs Notation
Action node / Transition		
Initial / Final node		
Fork / Join node		
Branching node		

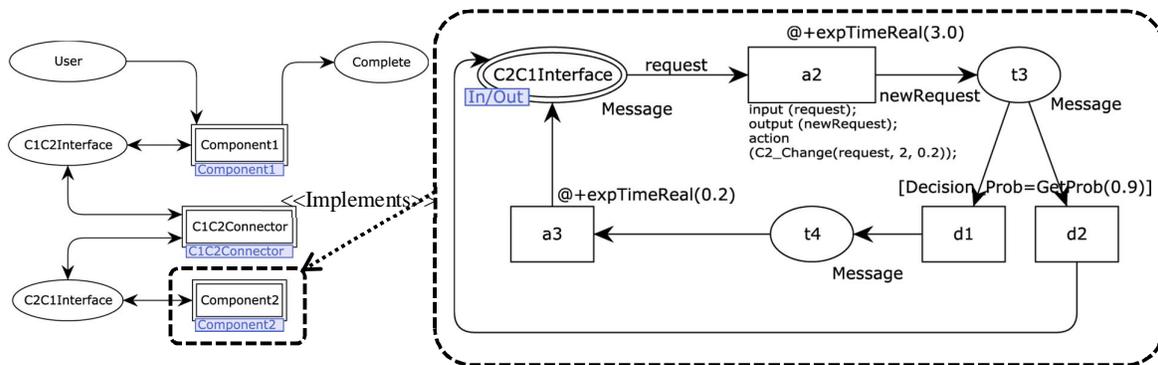


Figure 2. Generated HTCPN model from UML model

4.3 Evaluation of reliability in software architecture Level

In this study, to evaluate the reliability, extension of method presented in [4] is used so that therein the failure probability has not been considered in the internal activities of each component. Accordingly, to evaluate the reliability of SA, failure probability

in components connectors and actions performed in the components are used. Tokens of colored petri nets carry a f value as reliability. Figure 3 shows the manner of calculating reliability for reaching to points that failure probability may occur therein (e.g. components connectors and actions).

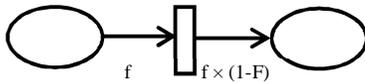


Figure 3. Reliability

According to figure 3, variable F indicates the failure probability and its values in UML diagrams are specified by REconnfail and failprob tags that respectively indicates failure probability in connectors and actions. Finally, the total reliability is equal to f value in final place of HTCPN model.

5. CASE STUDY

In this section, we use an internet sale system for evaluation of suggested method. In this system, that a part of which is examined, we consider the candidate scenario of products information display. It is notable that for drawing UML diagrams, Enterprise Architecture (Version 9) and for establishment of HTCPN models, CPNTools have been applied. Figures 4, 5 and 6 respectively show UML use case, component and activity diagrams of internet sale system.

According to figure 6 that represents the activity diagram for products information display scenario, five components are

involved therein. In this scenario, firstly the user requests for display of products information in Client section, from system. Then, the products information display requests sent to Webserver component via Client component. In continue, as per activity diagram of figure 6, products information display operation is continued and consequently the products information is displayed for the user in Client section.

Here, to evaluate the nonfunctional requirements (such as performance and reliability) for product information display scenario in SA level, UML model shown in figures 4,5 and 6 must be transformed to HTCPN model. Therefore, considering our suggested algorithm in this paper, the final HTCPN model will be as per figure 7.

For evaluation of performance (such as response time) and reliability, it is assumed that 20 requests are inserted to the system for execution of products information display scenario. Therefore, upon execution of HTCPN model that indicates the system behavior in servicing for the products information display requests by the users, we can achieve a series of valuable results related to evaluation of nonfunctional requirements in SA level. Table 3 represents the mean response time and reliability of internet sale system for providing services to 20 requests of products information display scenario.

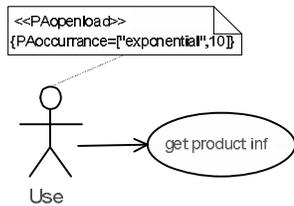


Figure 4. Annotated Use Case Diagram for Internet Sale System

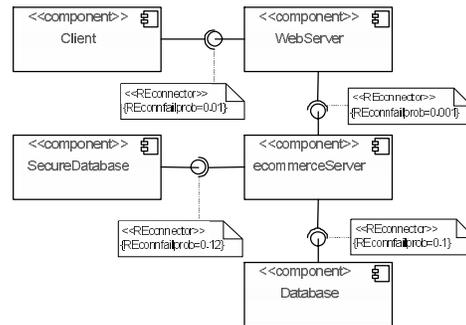


Figure 5. Annotated Component Diagram for Internet Sale System

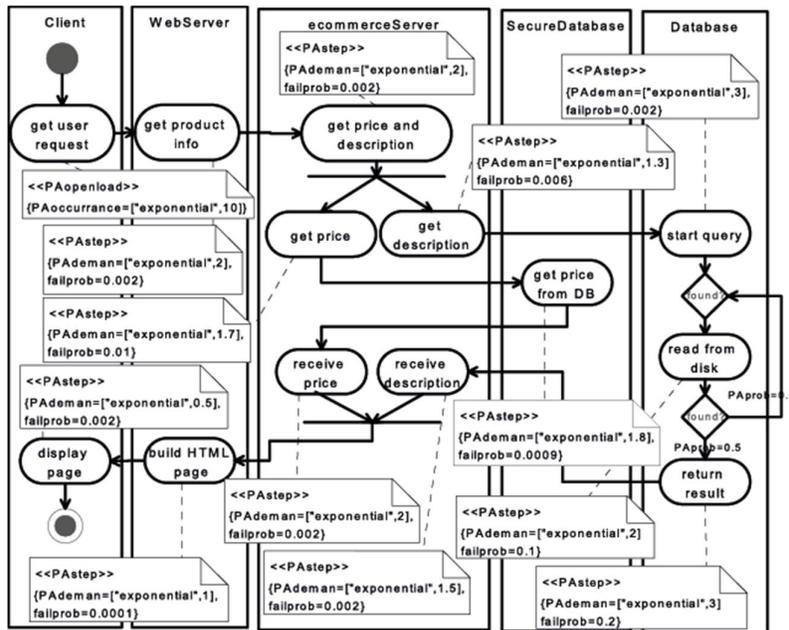


Figure 6. Annotated Activity Diagram for products information display

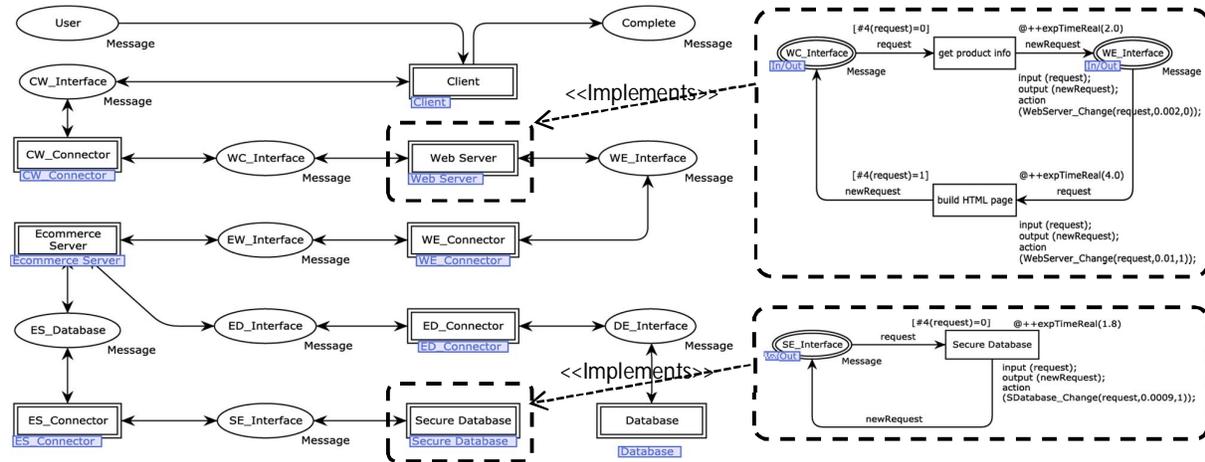


Figure 7. The full HTCPN model

Table 3. The mean response time and reliability

Number of Requests	Response Time (ms)	Reliability
20	21.058	0.956

6. CONCLUSION

In this paper, we have presented a strategy for evaluation of performance and reliability of nonfunctional requirements in software architecture modeled by UML diagrams. So, the software system may be validated for meeting or not meeting the nonfunctional requirements of case at the primary stages of software systems development cycle. The general analysis framework in this method is formed based on formal models (HTCPN) that accordingly is free of ambiguity. Whereas in this method, UML diagrams are used for description of SA, therefore description of SA by means of achievements of analysis and design stages will be very reasonable and low-cost. On the other side, a transformation algorithm has been presented for establishment of a HTCPN-based executive model from UML model for description of SA, hence the gap between architect and analyzer is removed and this process is performed easily. In this strategy, further researches are also possible. There are a lot of tools for working with UML models and UML models may be transformed to HTCPN-based executive model automatically. In addition, other nonfunctional requirements may be evaluated by means of other architectural specifications.

7. REFERENCES

- [1] Balsamo, S., Marco, A. D., Inverardi, P. and Simeoni, M. 2004. Model-Based Performance Prediction in Software Development: A Survey. IEEE Transaction On Software Engineering, Vol. 30, NO. 5.
- [2] Balsamo, S. and Marzolla, M. 2005. Performance Evaluation of UML Software Architectures with Multiclass Queueing Network Models. ACM Workshop on Software and Performance (WOSP).
- [3] Balsamo, S. and Simeoni, M. 2001. Deriving Performance Models from Software Architecture Specifications. In

Proceedings of the 15th European Simulation Multi Conference (ESM2001) on Computer Simulation.

- [4] Fukuzawa, K. and Saeki, M. 2002. Evaluating Software Architectures by Coloured Petri Nets. in SEKE02 14th International Conference on Software Engineering and Knowledge Engineering, ACM, Ischia, Italy.
- [5] Ghokale, S., Lyu, M. and Trivedi, K. 1998. Reliability simulation of component based software systems. In Proceedings of the 9th International Symposium on Software Reliability Engineering (ISSRE'98).
- [6] Gyarmati, E. and Strakendal, P. 2002. Software Performance Prediction-Using SPE. Master Thesis Software Engineering, Department of Software Engineering and Computer Science Blekinge Institute of Technology, Sweden.
- [7] Jensen, K. and Kristensen, L. 2009. Coloured Petri nets: modeling and validation of concurrent systems. Springer-Verlag.
- [8] Lai, Chien-Yuan., Shih, Dong-Her., Chiang, Hsiu-Sen. and Chen, Ching-Chiang. 2010. Transformation of UML activity diagrams into analyzable systems and software blueprints construction. WSEAS Transactions on Information Science and Applications, vol. 7, no. 3.
- [9] Littlewood, B. 1979. Software reliability model for modular program structure. IEEE Transactions on Software Engineering, Vol. 28, NO. 3.
- [10] Murata, T. 1989. Petri Nets: Properties, Analysis, and Applications. In Proceedings of the IEEE, Vol. 77, NO. 4.
- [11] Object Management Group (OMG). 2002. UML Profile for Reliability, Schedulability, Performance and Time Specification.
- [12] Object Management Group (OMG). 2005. Unified Modeling Language (UML). Version 2.0,
- [13] Zhu, L. and Wang, W. 2012. UML Diagrams to Hierarchical Colored Petri Nets: An Automatic Software Performance Tool. Science Direct International Workshop on Information and Electronics Engineering (IWIEE).