

A Survey of Design Model for Quality Analysis: From a Performance and Reliability Perspective

M. A. Isa¹, Mohd Z. M. Zaki¹ & Dayang N. A. Jawawi¹

¹ Software Engineering Department, Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Johor, Malaysia

Correspondence: M. A. Isa, Software Engineering Department, Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia. Tel: 60-12-338-4246. E-mail: mohdadham@utm.my

Received: January 15, 2013 Accepted: February 26, 2013 Online Published: March 20, 2013

doi:10.5539/cis.v6n2p55

URL: <http://dx.doi.org/10.5539/cis.v6n2p55>

Abstract

The use of a model for the analysis of the software quality attributes during the design phase has been gaining more attention in recent years. These models, which are peripheral in system design, are the center of quality analysis. The system design is the central focus in representing the structure and behavior of the system. Therefore, the goal of the software architecture performance and reliability analysis is to discover possible quality problems that may violate the quality requirements, which have been stated in the design. The use of an intermediate model to correlate the performance and reliability specification from the UML model and which is then transformed into an analysis model could facilitate the analysis process. This paper provides a survey of the existing intermediate metamodels from a performance and reliability perspective, through the evaluation and discussion on the similarities and differences focusing on the aspects of general concepts, modelling and analysis. The purpose of the discussion is to offer guidelines on which intermediate metamodel is appropriate for the use of quality analysis at design time as well as outline the possible space for improvement by making classifications and comparisons studies.

Keywords: software quality, performance, reliability, modeling, meta-object facility

1. Introduction

1.1 Introduce the Problem

As there are now more complex systems for almost every possible tasks being developed for the purpose of accommodating users in terms of simplifying domestic activities, there is now more need for software quality assurance. Software qualities including reliability and performance are inevitably required as to prevent undesired problems caused by unreliability and lack of performance (Rivers & Kudva, 2009). Unreliable software systems could cause inconvenient phenomenon such as failure to access the Internet or even more disastrous affects including death. In addition, underperformance of software systems could contribute to the delay of certain operations when executing a process and consequentially cause waste of money and time and even trust among members of the society.

Sufficient analysis on the reliability and performance of software can prevent possible negative effects to the software systems (Gokhale, Eric Wong, Horgan, & Trivedi, 2004). Adequacy of analysis *should* cause **positive effect**, and when positive effect manifests in the design, the architecture *should not* be compromised. The analysis of the software systems, which can be done, in terms of reliability and performance, ranges from prediction to risk analysis (Chise & Jurca, 2009; Lyu, 2007; Singh, 2011). The complexity of software systems can be treated as a reason as to why additional attention is required in ensuring sufficient software reliability and performance. Therefore, it is much more desirable for analysis to be done as early as in the design phase instead of waiting until the implementation phase.

The system design is representing the abstraction of the system as well as to define the systems' architecture and behavior elements (Tang, Tran, Han, & Vliet, 2008). These elements are the major factors in determining how the software systems may perform and behave. Therefore, early analysis of software performance and reliability which can be performed at the design phase could help determine the potential problems in which early preventive action could be taken to ensure that the software systems are working as intended and performing

well before the implementation phase.

In recent years, several approaches have been introduced, to quantify software performance and reliability, which can be used as early as during the system design phase (Grunske, 2007; Krishna, 2010). This phase defines the process of designing the systems in the context of structure and behavior identification, platform and environment definition and technological space implementation and presenting this conceptual context into design models using appropriate modelling languages. Some approaches analyze the software systems for performance (Becker et al., 2006) while others, for reliability (Vittorio et al., 2002). These approaches were applied with different techniques specifically from analytical-based models such as the Markov Chain (Haverkort, 2001), Petri Nets (Emadi, 2010) and Layered Queuing Network (Smith et al., 2010) to represent software systems with the elements of performance and reliability. However, with the emergence of UML language (OMG, 2010) as the de facto modeling language for various applications nowadays, software practitioners often concentrate on using this language to represent software systems functionality.

As mentioned before, in order to ensure that the software systems can perform well and reliably, accommodating between the software systems functionalities and software performance and reliability is inevitable. Therefore, in order to bridge the gap between UML-based models and analysis models, several researchers have introduced a MOF-based intermediate metamodel, which acts as the model formalism for generating analysis model from UML model, as illustrated in Figure 1. The intermediate metamodels that will be evaluated include the resource-based scenario, Core Scenario Model (CSM) (Petriu & Woodside, 2006), inspired from the Performance Domain Model (PCM) from UML SPT (OMG, 2005), KLAPER (V Grassi, Mirandola, & Sabetta, 2007; Vincenzo, Mirandola, Randazzo, & Sabetta, 2008), PCM (Distefano, Scarpa, & Puliafito, 2011), component-based system, Palladio (Becker, Koziol, & Reussner, 2009), enhancement of Palladio for reliability, Palladio-Reliability (Brosch, Koziol, Buhnova, & Reussner, 2010), and a reliability analysis based for fault tree analysis, MBDA (Lauer, German, & Pollmer, 2011).

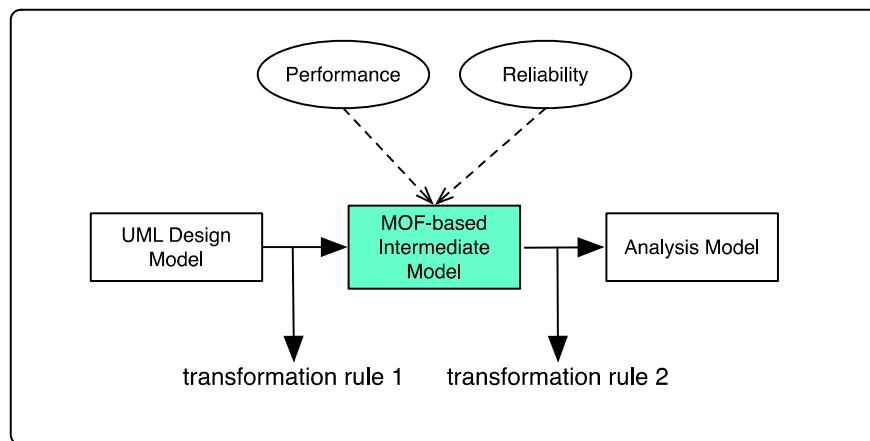


Figure 1. Intermediate model for quality analysis

This survey presents the intermediate metamodels in the same view by reviewing the basis of model-based quality analysis. The key of this paper is that it is directed to the basic definitions of the concepts, which are fundamentally used in the intermediate metamodels. Based on these definitions, the comparative evaluation framework for comparison and discussion are defined in the perspective of 1) the general concepts, 2) the modelling structure and approach and 3) the analysis approaches. The contributions of this paper are (i) a classification scheme for intermediate metamodel in generating performance and reliability models, (ii) a critical comparative evaluation of the existing intermediate metamodels to determine their advantages and shortcomings, and (iii) an analysis of possible future works on how intermediate metamodel specifically for attributes regarding performance and reliability can accommodate analysis on software quality trade-offs.

This paper is structured as follows: Section 2 provides the details of the main concepts. Section 3 describes the comprehensive comparative evaluation of intermediate metamodels. In Section 4, various intermediate metamodels are described and discussed and the differences and similarities are summarized in a comparison table. Section 5 is where the comprehensive discussion on the intermediate metamodels is carried out and the last section presents some future works and improvements, which can be done for further enhancement.

2. Definition of the Main Concepts

The definition of the main terminology for model-based quality analysis, which includes UML, model definition, quality attributes and quality analysis method is introduced.

2.1 UML Profile for Performance and Reliability

Since the Object Management Group (OMG) has been accepted as the final specification for various UML profiles such as Schedulability, Performance and Time Specification (SPT) (OMG, 2005), Modeling and Analysis of Real-Time Embedded Systems (MARTE) (OMG, 2009) and Modeling Quality of Service and Fault Tolerance Characteristics and Mechanism Specification (QoS/Fault Tolerance) (OMG, 2008), software engineers are starting to consider the use of stereotypes and tags value when modelling the system. The capability of stereotypes and tagged values would enable the extension of an incompleteness profile for certain domain concepts. With the current need for reliable software, existing UML profiles such as UML MARTE and UML QoS/Fault Tolerance have defined software quality attributes in their specification. Figure 2 shows the basic modelling layer based on MOF.

The UML MARTE Profile is a UML standard profile specifically for Embedded Real Time System (ERTS). This profile was developed based on the UML SPT profile with the introduction of two important aspects; domain-specific concepts for design and real-time analysis of embedded systems. Through the UML MARTE design model, a new component concept is derived from UML2, which further defines the details of the concept and notation for components in ERTS. Additionally, the UML MARTE analysis model enables the analysis of software behavior, specifically for the package of performance and schedulability namely the Performance Analysis Model (PAM) and the Schedulability Analysis Model (SAM) which were previously available in the UML SPT profile. These analysis packages are based on two important new features of MARTE, which are specification for quantitative and qualitative non-functional properties (NFP), and General Quantitative Analysis Model (GQAM). The NFP package allows the developers to express the non-functional annotations onto modelling elements. The GQAM package defines the Workload, Resources and Observers sub-packages that share some of the foundation concepts from NFP, Generic Resource Model (GRM) and MART Library packages.

The UML QoS/Fault Tolerance Profile is another standard profile provided by OMG, which directly associates the software quality attributes in the UML model elements. Generally, this profile consists of three major metamodels; QoS Characteristics, QoS Constraints and QoS Values. These metamodels support different software quality concepts, which are further expressed in the QoS Catalogs. For the purpose of modelling, this profile defines the two-step annotation processes. The first step is to produce a quality model for each of the developed system's class and the second step, annotate the UML model with QoS Constraints and QoS Values. However, these processes might require additional effort and may induce the whole process of software development life cycle.

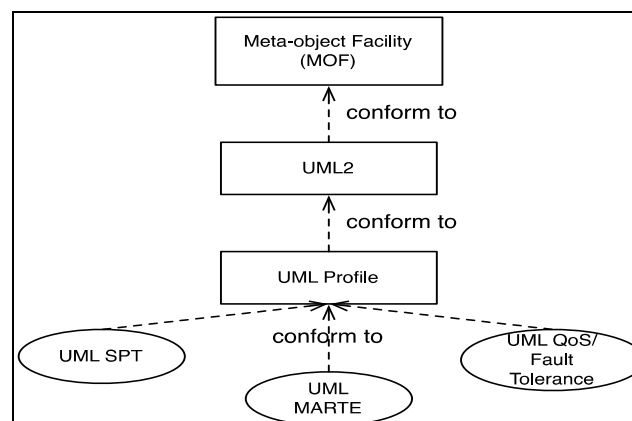


Figure 2. UML-based model level

2.2 Quality Analysis at Design Level

Most formal efforts are usually interested in the evaluation of the performance and reliability analysis attributes during system runtime (Yang et al., 2009). Thus, the aim of the evaluation of a final system is to establish a precise estimation of system's quality. However, if the result of quality estimation is uncertain, the process of

rectifying the uncertainties of the system might be too costly and time consuming. Therefore, it is suggested that the software quality analysis should be done at an early phase, as early as during the software development process.

There are two effective approaches in analyzing performance and reliability attributes during the early phase of system development, which are during the design time (Balsamo et al., 2004) and through simulations (Arpinen et al., 2009; Vittorio et al., 2008). Throughout the years, software architecture (SA) has been used when dealing with software quality analysis (Dobrica & Niemela, 2002). SA acts as the boundary in incorporating the software qualities towards the more reliable and highly performing resulting system. The important aspect of SA such as separating the view of the model implies that the model allows more abstraction evaluation when SA investigates both the functionality and the software quality (Briand et al., 1993).

Even though quality evaluation is more accurate during runtime and through simulations (Yang et al., 2009), those approaches lack of certain important criteria including scale, feedback, predictable, time and cost of execution (Balsamo et al., 2004). In addition, if the quality problems are discovered during runtime evaluation, it is almost not possible for those problems to be rectified in a short time. Quality evaluation during design time is more convenient in many aspects as it provides more space for rectification should the quality problems exist (Balsamo et al., 2004; Dobrica & Niemela, 2002).

3. Comparative Evaluation Framework

The comparative evaluation framework introduced in Figure 3 is used for the model comparison. The framework explains the list of criteria needed for the performance and reliability analysis of the metamodel at design level. The list of criteria is divided into a three categories, which are general concepts, modelling and analysis. These categories of the framework are explicitly based on the NIMSAD (Normative Information Model-based Systems Analysis and Design) framework (Jayaratna, 1994). NIMSAD generally classifies the criteria into four classes; context, elements, user and validation. In the context criteria, the method is investigated from the views of the general concepts, while in the elements criteria; the method is viewed for the essential contents itself. As for user criteria, the method is described in the center of the users targeted for the method, whereas the validation criteria are for the validation of the result and the maturity of the method against the target domain problems.

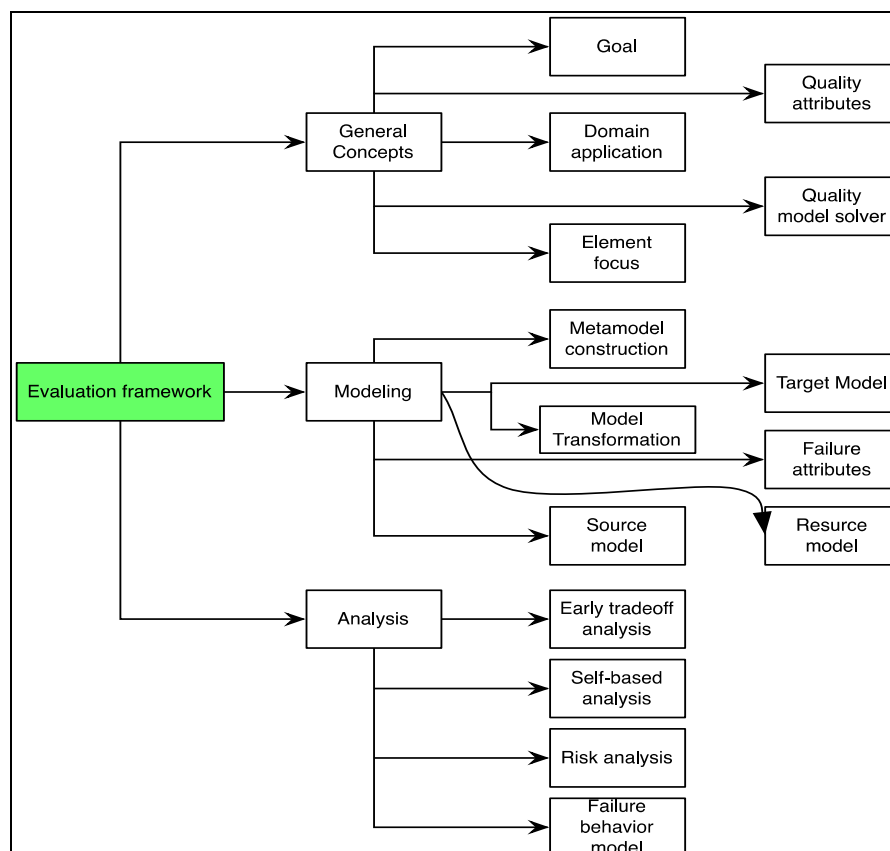


Figure 3. Comparative evaluation feature model

This framework is generally composed of two criteria from the NIMSAD framework; context and elements. In the context criteria, we renamed it as general concept, which is used to describe the general information of the intermediate metamodel from the viewpoint of its general information. As for the element criteria, we divided it into two other sub-criteria; modelling domain and analysis domain. These domains provide detailed information on what are the essential elements in the intermediate metamodel. The modelling and analysis criteria provide advantages to some specific information for performance and reliability analysis at design level. Table 1 shows the details of the framework criteria.

Table 1. Comparative evaluation framework description

Criteria	Sub-Criteria	Questions
General Concepts	Goal	What is the primary goal of the intermediate metamodel?
	Domain application	Which area is the intermediate metamodel applicable to be applied?
	Central elements focus	What are the main problems that the intermediate metamodel is willing to solve?
	Quality attributes	Does the intermediate metamodel define the quality concepts?
	Quality model solver	Does the intermediate metamodel provide the automation of numerical quality model solver or simulation-based quality solver?
Modeling	Approach for metamodel construction	What are the approaches usable for developing intermediate metamodel?
	Model transformation	What are the approaches that can be used for model transformation?
	Source model	What is the input model?
	Target model	What is the output model as in the resulting analysis model?
	Details failure attributes	How are the details of the failure behaviour explained in the intermediate metamodel?
	Resource model	Does the intermediate metamodel provide a resource model for analysis?
Analysis	Early trade-off analysis	Does the intermediate metamodel provide for early trade-off analysis?
	Self-based analysis	Does the intermediate metamodel provide an analysis for each of the quality attribute?
	Risk analysis	Does the intermediate metamodel express an early risk analysis in detecting any conflict between software qualities?
	Failure behaviour model	What type of failure behaviour that can be expressed in the intermediate metamodel?

3.1 General Concepts

A general concept addresses the basic information of the intermediate metamodel, which gives an overview to some specific goal of the model. The goal must be clearly defined as to distinguish it from the other metamodel goals. The primary goal is accompanied by the primary focus of the intermediate metamodel in which a problem domain needs to be solved. In order to solve the intended problem domain, the intermediate metamodel might be directed to the specific elements such as resource context, service-based solution and workload utilization (Koziolek & Happe, 2008).

3.2 Modeling

The model description is capable of addressing the space of concerns, which is related to the representation of the domain problems (Muller, Fondement, Baudry, & Combemale, 2010). As the modelling of the performance and reliability analysis for the intermediate metamodel remains descriptive, the conventions of constructing the metamodel must exploit the performance and reliability information that reflects the system architecture. The way the metamodels are constructed is entirely independent and scalable.

3.2.1 Approach for Metamodel Construction

Three basic approaches, which can be used for the construction of the metamodel, include extending the MOF metamodel, extending the UML2 metamodel and extending the UML profile, and are available at the MOF-based level as was defined in (Volter & Stahl, 2006). The extension of the MOF metamodel generates a new instance for the MOF metamodel and independent with UML metamodel. This approach provides more room for the exploration of modelling and analysis concepts and explicitly not tied to the domain application and domain knowledge. Through extension of the UML2 metamodel, not only is the metamodel is extended from the UML2 metamodel but it also inherits the complexity and ambiguity of UML. This will cause other disadvantages such as loss of design familiarity among the software developers, increased lacking of tool supports and absence of design standard. Another approach is extending the UML profile. This approach is the best way to introduce a new modelling concept for a specific need of a domain application. Profiles such as UML MARTE and UML SPT provide an extension of the embedded system application domain through stereotypes and tagged values.

3.2.2 Model Transformation

Model transformation enables one model to be translated or rephrased into another (Mens & Gorp, 2006). The intermediate model offers many-to-one model transformation from the UML design model to an analysis model. Once the intermediate metamodel is developed to bridge the gap between those two models, the model transformation definition will be essential in providing transformation rules and guides on how the source model can be transformed into the target model (Czarnecki & Helsen, 2003).

3.2.3 Source Model

The process for quality analysis at the design level is applicable to different kinds of abstraction models. A source model is an input model used in generating the other level of abstraction model. The source model can be a formal modelling language such as UML or non-UML diagram such as block diagram.

3.2.4 Target Model

The target model, which is used for analysis, should embody the common features or data, which was in the source model. The target model can be produced with a single model or multiple models, depending on the suitability of the transformation rules for the intermediate metamodel.

3.2.5 Detailed Failure Attribute

In the case of performance and reliability analysis, information on failure attributes can be essential for quality prediction. An intermediate metamodel may need failure information, which may include information on what are the characteristics that can cause failure. Failure attribute representation would increase the visibility of failure information for a scenario in the system (Goseva-Popstojanova, 2001).

3.2.6 Resource Model

The representation of the resource model as a sub-model in the intermediate metamodel could be crucial, as the resource usage is the compulsory attribute for performance analysis (Woodside, 2007). The resource model may be descriptive, monitoring the resource usage and utilization as to show how the resource is being used.

3.3 Analysis

The analysis of the multiple quality attributes in a single system could be difficult especially at the design level. The analysis might be inaccurate and sometimes-undesired equality can be produced between the quality attributes. The quality analysis paradigm is shifted from self-based analysis to tradeoff analysis as to offers for the *analysis to conform to a benchmark to ensure equality in the analysis* of quality attributes. Different analysis results represent different states of system quality. The results may be from the *individual analysis*, which is responsible of predicting a single quality attribute, or the *tradeoff analysis*, which is responsible of assuming the equality measure among software quality (Yang, Huang, Zhu, Cui, & Mei, 2009) or the *risk analysis*, which is used to facilitate the conflict among quality attributes (Cortellessa, 2005) or from the *failure behavior model*, which is used as a parameter for failure prediction analysis.

3.3.1 Early Trade-off Analysis

It is possible to execute an early tradeoff analysis, which can be done at the design level. Early tradeoff analysis of quality attributes offers the facilities for early design decisions to be made, model refinement to be scaled and early quality attribute conflicts to be managed (Andrews, 2005). These facilities could be realized through the model by adding other information regarding quality attribute.

3.3.2 Self-Based Analysis

An individual quality analysis does not only provide the boundary for the resulting system, but also embodies the potential risks of the systems. Each quality analysis result poses a self-based goal, which encompasses the need for the quality requirements to be fulfilled. This analysis will help clarify the result for each quality attribute so that, further action can be taken to refine the model.

3.3.3 Risk Analysis

A potential risk might occur during the tradeoff analysis process. Risk analysis is the process of investigating the conflicts among the quality attributes and outlining the severity of the risk caused by the conflicts (Cortellessa, 2005). Unexpected results could be predicted at the design level, which will prove helpful in preventing unwanted problems during deployment. The advantage of the risk analysis process is that it is possible to accommodate it with a tradeoff analysis decision so that a possible tradeoff solution can be made through risk analysis result.

3.3.4 Failure Behavior Model

Failure behavior model is a model used for the prediction process in the analysis of performance and reliability. This model poses its own parameter, which depends on the level of prediction accuracy and the method for the analysis process at the system architecture level. Some of the existing failure behavior models include *the probability of failure*, *time-dependent failure intensity* and *constant failure rate*.

3.4 Scope of the Comparison

There are a large number of model-based quality analyses in previous works and it is difficult for all the known analysis to be included in this paper. Our scope of interest for model-based quality analysis is limited to the following criteria:

- 1) Design level quality analysis
- 2) Particularly for performance and reliability attributes
- 3) Focus on the uses of intermediate metamodel for generating performance/reliability model from UML-based model to analytical-based model.

As described in Section 2.3, conducting a quality analysis at the design level would provide some advantages for the purpose of model refinement. Refining the model at the early stages of development could help facilitate in making decision regarding the design. On the other hand, we only focus on two quality attributes, namely, performance and reliability. Throughout our study on model-based quality analysis, there are some approaches, which use the standard UML profile (UML SPT, UML MARTE) to annotate performance and reliability onto the design models. Other approaches define informal UML annotations for quality analysis in the design model annotation. The model should be capable of expressing both the UML-based model and the analytical-based model accordingly to support significant representation of the performance and reliability domain model.

4. Overview of the MOF-Based Intermediate Metamodel for Performance and Reliability Analysis

4.1 Core Scenario Model (CSM)

CSM, which was developed by Petriu and Woodside (Petriu & Woodside, 2006), corresponds with the trend for model-based performance analysis and was developed with the purpose of providing better understanding of resource usage, which plays a role in system performance evaluation. CSM is an intermediate model that acts as performance model formalism and is capable of generating a number of performance models from UML behavior model (Sequence diagram, State machine diagram, etc.). Thus, in the design phase of a system's development, the development of performance model by the CSM model is still acceptable without causing excessively high effort from software developers. Figure 4 shows the details of the input elements in CSM as well as the main CSM process.

4.1.1 General Aspects

The main goal of CSM is to act as performance model formalism, which can generate a number of performance models from the annotated UML design in order to express the resource context in the system scenario. The fundamental of CSM is the illumination of the performance attribute in the context of scenarios. Additionally, CSM offers a view to assess resource usage in order to predict its performance. CSM guides the evaluation of the system scenario at the design level by focusing on the resource usage. The capability of CSM to assess the utilization of the resource with respect to the intended action of a system can also be used to various application domains.

4.1.2 Modeling

CSM was developed based on the MOF-based extension approach. In this approach, the metamodel was constructed within the performance domain and is not self-defined but in fact motivated from the Performance Domain Context (PDC) in the UML SPT profile. The concepts in PDC such as resource, workload and scenario are reused in CSM as an appreciation to the UML SPT contribution to the performance analysis domain. Prior to the generation of the performance model from the UML design model by CSM, the description of the model transformation rules should first be in understandable form. The model transformation rules, the source model and the target model definition are the three elements, which are essential and need to be defined when defining the model transformation rules. The source model (UML) is the input model and should be presented in a dynamic representation such as in the form of sequence diagrams or activity diagrams to show how the system behaves. The target model, which can be in the form of Petri Nets or Queuing Network, is somewhat the reflection of the source model, which can identify how the system execution is realized to determine the resource utilization in the system. CSM uses the XML schema to define its transformation rules as the main input is the UML model and the output is in the form of Petri Nets or Queuing Network model.

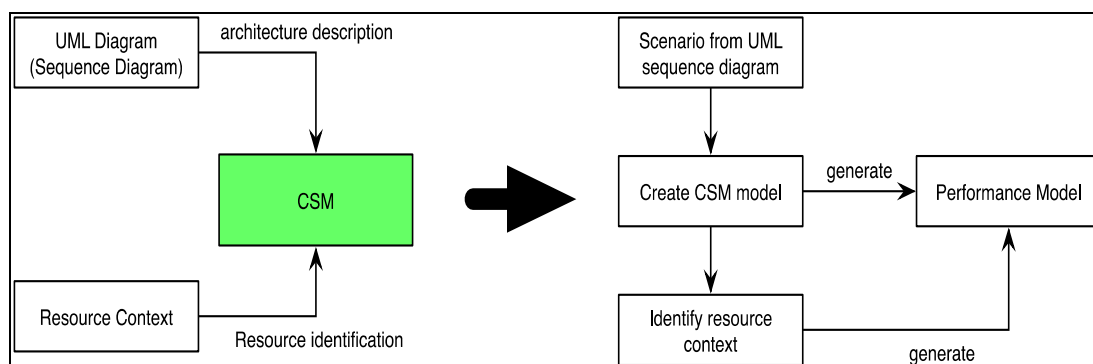


Figure 4. Input and process for CSM

4.1.3 Analysis

CSM provides an analysis of resource usages, which is useful for the performance analysis. This analysis increases the understanding of the resource usage in the context of system scenario and can be very practical in answering questions regarding the acquisition and release of the resource in the given scenario.

4.2 Kernel Language for Performance and Reliability Analysis (KLAPER)

KLAPER treats the necessary details regarding performance and reliability as a kernel language, which allows the performance and reliability analysis to be done onto component-based systems. KLAPER was developed in 2007 by Grassi et al. (Grassi et al., 2007) and it contributes by bridging the gap between design-based models and analysis-based models and is a way of reducing model transformation complexity and how the component's services are evaluated in regarding to the aspects of performance and reliability. Figure 5 shows a general transformation framework for KLAPER.

4.2.1 General Aspects

KLAPER, which is applied in the model-based performance and reliability analysis, considers only the problem of model transformation complexity and model heterogeneity. The main goal of KLAPER is of twofold; to reduce the complexity of model transformation caused by heterogenous models for both design-based and analysis-based models and to analyze the performance and reliability of the list of services provided by the components.

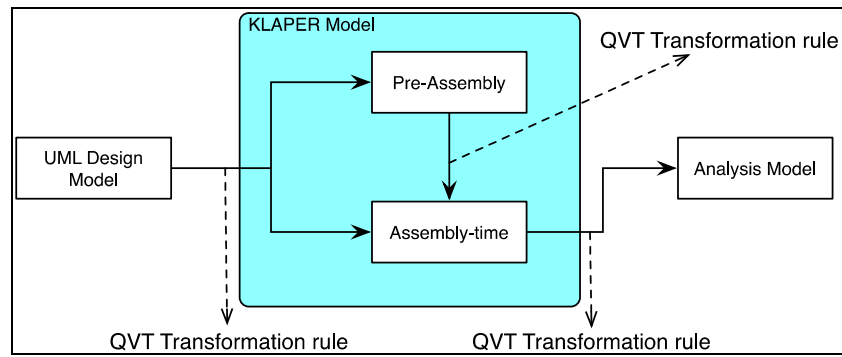


Figure 5. KLAPER transformation framework

4.2.2 Modeling

The metamodel is constructed based on the MOF extension approach. This approach provides more room for the exploitation of the model transformation rules as the OMG have proposed a model transformation technology called the Model Driven Architecture (MDA). MOF-MDA facilitates the transformation from the source model to the target model as long as both models conform to the MOF. KLAPER uses QVT relational as its model transformation technology as to accommodate the transformations from the design model (UML) to KLAPER, to Markov Chain model and EQN.

4.2.3 Analysis

A failure behavior model is included in the KLAPER model and KLAPER provides an early tradeoff analysis at a low degree. Failure behavior is defined as a failure of the component's service to be provided upon request.

4.3 Performance Context Model (PCM)

PCM interposing between UML design model and performance domain model and the metamodel with performance domain knowledge is presented in the intermediate model representation. This metamodel which was first introduced in (Distefano, Paci, Puliafito, & Scarpa, 2004) is for the purpose of performance analysis in UML design for general application. Figure 6 shows the execution steps in generating a PCM model.

4.3.1 General Aspects

The main goal of PCM is to enable smooth transition of the performance analysis from the UML model to the performance model using the intermediate model. PCM uses both the structure and behavior diagrams in UML for modelling purposes performance annotations, which are available in the UML SPT profile. Generally, PCM can be executed in five steps; Step 1 is to model the system in UML model with annotated performance tags. Step 2 addresses the process of mapping the UML model to the PCM model, which is then executed in Step 3 where the PCM model is mapped to the corresponding performance model. Step 4 is devoted to the performance analysis and the predicted result would be compared against the performance requirements and finally, Step 5 covers the process of reverse engineering for the purpose of model refinement. These steps are fully automated by tools such as the *AgroPerformance* except for Steps 1 and 5. The details of the execution are as shown in Figure 4.

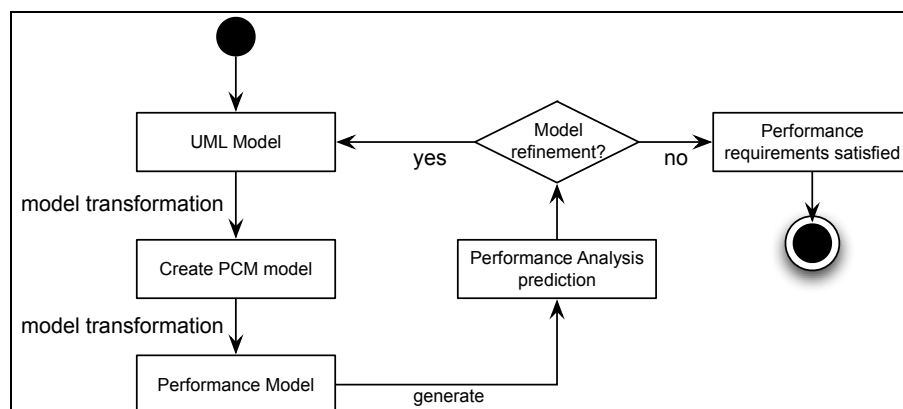


Figure 6. PCM execution steps

4.3.2 Modeling

The space solution for PCM is entirely based on the performance information. Therefore, the intermediate metamodel for PCM was developed based on the MOF-based extension approach. PCM provides a model transformation rule of transforming from UML-based model to PCM. This model transformation rule would accelerate the process of transformation. The visualization of the resource dependency in PCM is rather explicit, abstracting away the details for the resource usage model and focusing more on the steps for the performance analysis execution.

4.3.3 Analysis

PCM provides an analytical analysis for the purpose of performance prediction from the performance model. The performance analysis is entirely for the performance attributes, which considers the workload distribution and the time to process the resource.

4.4 *Palladio Component Model (PCM)*

For component-based systems, components are the central artifacts that produce a workable system. Any absence in the quality of the components, especially in terms of the performance attribute, would result in undesired problems. Therefore, (Becker et al., 2009) proposed the Palladio component model with the main purpose of assisting in early performance analysis of components, which can be done during the design time. Figure 7 shows the elements in Palladio that represents the source and target elements.

4.4.1 General Aspects

The Palladio component model aims to investigate the performance of the components including the components' structure, resource usage and resource allocation. A new metamodel based on the MOF-based extension was introduced to assist the component model instance from the UML model before transforming it into a performance model. The fundamental of the Palladio component model is the concretization of the performance analysis of the components. The performance indices are obtained from the simulation and analytical result.

4.4.2 Modeling

A new metamodel is developed to ease the use of the Palladio component model in terms of performance analysis. The metamodel includes a repository, an interface, relation, structure and resource environment, allocation and usage. This metamodel create an instance for the purpose of performance model development. Therefore, an informal model transformation technology called ad-hoc java is used as to enable the transformation from the Palladio component model to the performance model.

4.4.3 Analysis

Palladio offers implicit performance analysis at design level, leaving behind an early tradeoff analysis as well as risk analysis. The limitation that has been stated in the paper is that the mathematical assumption is rather explicit in terms of reducing the model's complexity.

4.5 *PCM-Reliability*

This work was introduced as an enhancement of the Palladio component model in terms of reliability analysis for system's component (Brosch, Koziolok, & Buhnova, 2011). The purpose of enhancing the model to include the reliability analysis is motivated by the need to obtain a reliable system through concrete design decisions.

4.5.1 General Aspects

The main goal of this work is to enable a reliability analysis for component-based systems and introduce an enhancement of the Palladio component model for said purpose. The new notions added were *software failures*, *communication link failures* and *unavailable hardware*. Software failure offers the probability of service failures during internal execution. Communication link failure describes the failures that can occur during the message transaction among the components, which will later lead to the failure of service. The unavailable hardware failure contributes to the reliability metrics on deciding when the available system's resources may fail in a certain period of time and condition. The metrics include *Mean Time To Failure (MTTF)* and *Mean Time To Repair (MTTR)*.

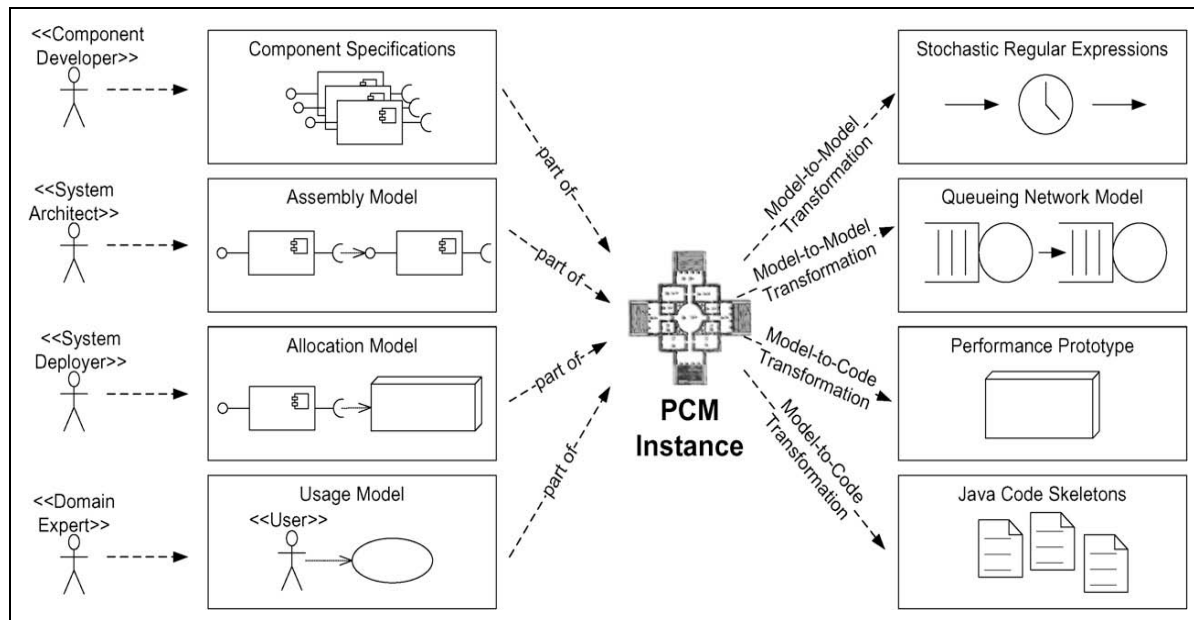


Figure 7. Palladio component model transformation process (Steffen et al., 2009)

4.5.2 Modeling

The modelling construction is rather straightforward as this model was extended from the Palladio component model. The additional notions, which were added, and the source and the target models are the UML and the Markov Chain model, respectively.

4.5.3 Analysis

The guidelines for reliability analysis provided by this model are based on the MTTF and MTTR metrics. Similar to Palladio, there is no facility for early tradeoff for performance and reliability even though the Palladio model is enhanced with reliability aspects.

4.6 Model-Based Dependability Analysis (MBDA)

The MBDA (Lauer et al., 2011) metamodel uses a semi-formal analysis language called Fault Tree Analysis (FTA) (Reza, Pimple, Krishna, & Hildle, 2009) as the analysis model. The idea is to produce the FTA from the annotated UML model particularly for reliability analysis during design time.

4.6.1 General Aspects

The main goal of this model is to identify the fault propagation and fault containment of the system. In order to assist the current UML2 model, a new UML profile was developed for fault tree synthesis, which includes information content and application specific information. A universal model transformation algorithm is developed with the sole purpose of ensuring smooth transformation from the annotated UML model with the new profile to the FTA model.

4.6.2 Modeling

There are two layers of the metamodel for fault tree synthesis; the architecture model and the application model. These models enable information to be separated from the underlying architecture as to minimize the re-modelling of the applications for fault tree analysis.

4.6.3 Analysis

The analysis of the system's reliability is straight from the FTA model. The capability of the FTA model in providing a design decision results in a risk analysis for the system. The result of risk analysis is used to obtain the initial tradeoff analysis among the fault propagation responsibilities.

5. Discussions

The current trend in designing a system shows the widespread use of UML as the main modelling language. The lack of an analysis method in this language causes some problems which leads to the introduction of an intermediate metamodel by several researchers with the intention of bridging the gap between UML-based

models and analysis models. This discussion provides some guidelines for the selection of the best intermediate model for the purpose of performance/reliability analysis. Table 2, Table 3 and Table 4 shows the comparative results in the views of *General concepts*, *modelling* and *analysis*.

5.1 Specific Goal

Each intermediate metamodel poses a defined set of goals. Viewed from the goal of a model are considered as fundamental in determining which metamodel is suitable for performance/reliability analysis during design time. In this paper, each intermediate metamodel was critically evaluated and the model were compared to each other to provide different perspectives and views. The perspectives are divided according to: applicable for general domain application (CSM, PCM), for component-based systems (KLAPER, Palladio, PCM-Reliability) and safety critical systems (MBDA); resource oriented analysis (CSM, KLAPER, Palladio, Palladio-Reliability); multi-state quality analysis (KLAPER); establishment of model quality solver using simulation (PCM, KLAPER) or simulation and analytical, (Palladio, Palladio-Reliability). These perspectives lead to the discovery of explicit similarities and differences, which can be used for comparison in order to visibly, determine the suitability of the metamodels for specific purposes.

5.2 Classification of the Modeling Aspects

Based on the metamodel construction, we can conclude with the assumption that the construction of the metamodel considers the thing from which it was based on. From this view, some of the intermediate metamodel such as CSM, PCM, KLAPER and Palladio—are known to use MOF-based extension while others like PCM-Reliability reuses an existing intermediate metamodel. Another metamodel, MBDA, creates a new UML profile for fault tolerance. The concept of performance is treated as the basic information from the Performance Domain Model (PAM), which is obtained from the UML SPT profile (CSM, PCM, and KLAPER). Palladio offers flexibility in performance concepts for components without depending on any existing UML performance concepts while MBDA tries to manipulate the stereotypes and tagged values' flexibility by introducing a new UML profile for fault propagation.

Based on the model transformation method, some of the intermediate metamodel used transformation languages such as XML, XMI and Java code (CSM, PCM, Palladio), while others use relational transformation approaches such as QVT (KLAPER) and another approach, the ad-hoc algorithm approach, is used by MBDA. Although the involvement of model transformation is essential, not all intermediate models consider its presence as mandatory especially for information consistency and model refinement. CSM, PCM and Palladio do no stress on information consistency during the model transformation process and differ from KLAPER, which takes information consistency into consideration as the QVT approach promotes consistency and model refinement through bi-directional model transformation.

Based on the use of source model and target model, all the intermediate metamodels share a common understanding that the source model should be from an UML-based model and the target model should be in the form of an analytical model; for the purpose of performance and reliability analysis.

Table 2. Comparative evaluation for general concepts criteria

<i>Criteria: General Concepts</i>				
Sub-Criteria Methods	Domain Application	Focus Element	Quality Attribute(s)	Quality Model Solver
CSM	General	Resource Centric	Performance	N/A
PCM	General	General Performance Concepts	Performance	Simulation
KLAPER	Distributed System (Component-Based)	Service	Performance and Reliability	Simulation
Palladio	Distributed System (Component-Based)	Component of Resource Usage	Performance	Analytical and Simulation
Palladio-Reliability	Distributed System (Component-Based)	Usage Profile and Hardware	Reliability	Analytical and Simulation
MBDA	Safety Critical System	Fault Origin, Fault Detection and Fault Propagation	Reliability	Algorithm

Table 3. Comparative evaluation for modeling criteria

<i>Criteria: Modeling</i>					
Sub-Criteria Methods	Construction	Model Transformation	Source/Target Model	Resource Dependency	<i>Details Failure Attribute</i>
CSM	Extention of MOF	XML Schema	UML/Petri Nets, QN	High	<i>N/A</i>
PCM	Extention of MOF	XMI for interchange metadata	UML/Petri Nets	Low	<i>N/A</i>
KLAPER	Extention of MOF	QVT	UML/Markov Chain, EQN	Average	<i>Low</i>
Palladio	Extention of MOF	Ad Hoc JAVA	UML/EQN, LQN	High	<i>N/A</i>
Palladio-Reliability	Extention of Palladio metamodel	N/A	UML/DTMC	Low	<i>N/A</i>
<i>MBDA</i>	<i>Extention of UML2</i>	<i>Ad Hoc Algorithm (Build Tree)</i>	<i>UML/Fault Tree Analysis</i>	<i>Low</i>	<i>Average</i>

Table 4. Comparative evaluation for analysis criteria

<i>Criteria: Analysis</i>					
Sub-Criteria Methods	Early Trade-Off Analysis	Individual Analysis	Risk Analysis	Failure Behaviour Model	Multi-State Analysis
CSM	No	High	N/A	No	No
PCM	No	High	N/A	No	No
KLAPER	Low	High	N/A	Yes	Yes
Palladio	No	High	N/A	No	No
Palladio-Reliability	No	High	N/A	No	No
MBDA	Low	High	Low	No	No

5.3 Resource Usage Views

One important characteristic, which needs to be taken into account in the performance/reliability domain, is how the resource in the system is being utilized. The activities regarding resource utilization in the intermediate metamodel differ between the *dependency level* and the *context*. *Dependency* describes how the resource in the intermediate metamodel behaves; whether it is independent or is dependent on one another. The independent resource link is considered to have low resource dependency and the representation of the resource utilization in the model level is considered low (PCM, MBDA). The dependent resource link provides more accurate resource utilization analysis as this relationship offers loose couplings among the resources (CSM, KLAPER, Palladio). For example, CSM provides concrete resource usage in terms of when to acquire and when to release the resource for a specific action. *Context* means that the intermediate metamodel offers guidelines that represent the resource context model. Another example in CSM is that the respective CSM model implicitly shows the resource context for each scenario action in order to differentiate the resource utilization views of the system scenarios.

5.4 Analysis Coverage

The analysis coverage of the intermediate metamodel is treated as an additional facility for the model. The analysis in the modelling level is considered as the preliminary analysis, which will assist in the future works focusing on in-depth quality analysis. A special case for multi-attribute analysis is observed in KLAPER. Considering the event of performance failure, KLAPER enables an investigation of the performance level for service execution of components and predicts its probability of failure during the execution. Unlike the rest of

the metamodels, which focuses on a single quality attribute, KLAPER manages to execute analysis for both performance and reliability but in return, offers a little bit assumption for the tradeoff analysis. Unlike KLAPER, MBDA describes the early tradeoff analysis among events that may possibly experience fault.

6. Conclusions

This paper has shown a survey on MOF-based meta-models for performance and reliability analysis by presenting and discussing six metamodels. The analysis of system performance and reliability fundamentally requires both design models and analysis models. The current trend of using UML modelling language had led to the standardization in the process of designing the system. This language however suffers in the aspect of formal analysis particularly in terms of performance and reliability. Therefore, it is essential for some mechanism to be introduced in order to bridge-the gap between the design model in UML and the analysis model in the analytical model. In this paper, a comparison framework was developed to compare the previous works on performance and reliability analysis, particularly for MOF-based approaches between UML-based models and analytical-based models for intermediate metamodels.

The result of the comparison shows that there are a few intermediate metamodels, which can provide the unified model of both performance and reliability. However, the metamodels still suffer some form shortcoming in terms of details attribute for failure behavior definition. Another shortcoming is the lack of rules description for a systematic model transformation of the intermediate metamodel from the UML-based model to the intermediate metamodel and from the intermediate metamodel to the analytical-based model. The lack of model transformation rules results in the lack of a proper guide for the users to use the metamodel. As for the analysis part, none of the metamodels provide early quality analysis especially for the purpose of risk analysis. Early risk analysis could reveal any possible performance failure and the consequences of the failure, which can lead to the prediction of the outage of the system.

In summary, it can be assumed that the possible future works for improvement could be done and some of the open problems are as follow:

6.1 Unified Modeling for Multi-Quality Attribute

6.1.1 Progress in Unified Modeling

The purpose of quality analysis to be done at the design level is to provide a possible response to the risks that might occur after deployment. Various quality attributes have been distinguished to be contributed to the analysis of the problems for the purpose of identifying possible risks. For example, the progress of the system performance may cause some degradation to the system's reliability, which may expose the system to various security issues. The lack of security may speed up the system's performance but at the same time cause availability problems to be unavoidable. Inter-related analysis needs to be incorporated into a single model and represented by another modelling approach to show how the model can support multi-attribute analysis for unified model. Possible risks which may originate from related quality attributes could be presented more profoundly as each representation of the quality attributes in the model can help in determining which part needs to be analyzed more deeply in the software development lifecycle.

6.1.2 Progress in Unified Modeling for Model Transformation

The existing intermediate metamodels, which have been evaluated, seem to neglect the importance of model transformation. The completion a model transformation technique with fine-level consistency checking could possibly provide improved output. The model transformation technique should offer a specific analysis for consistency checking to assist the process of model refinement. For instance, a source model can be transformed into the target model with each element in the source model correctly transformed into the target model. Then, the model transformation approach would be capable of checking the consistency level of the information, which in return, ensures that no information will go missing.

6.1.3 Progress in Unified Analysis

A complete analysis specification must contain the identified analysis for a quality attribute, an early tradeoff analysis and a risk analysis. The adaptation of an early tradeoff analysis should be considered as a topic for future researches. Another important thing is the risk analysis. Obtaining early risk analysis could help in determining how to organize the tradeoff analysis. The tradeoff analysis and the risk analysis are equally useful in the process for formalizing the relationship between quality attributes.

References

- Andrews, A. (2005). A Framework for Design Tradeoffs. *Computer*, 377-405.
- Becker, S., Koziolok, H., & Reussner, R. (2009). The Journal of Systems and Software The Palladio component model for model-driven performance prediction. *The Journal of Systems & Software*, 82(1), 3-22. <http://dx.doi.org/10.1016/j.jss.2008.03.066>
- Brosch, F., Koziolok, H., Buhnova, B., & Reussner, R. (2010). Parameterized reliability prediction for component-based software architectures. *QoSA 2010, LNCS 6093*, 36-51.
- Brosch, F., Koziolok, H., Buhnova, B., & Reussner, R. (2012). Architecture-Based Reliability Prediction with the Palladio Component Model. *IEEE Transactions on Software Engineering*, 38(6), 1319-1339. <http://dx.doi.org/10.1109/TSE.2011.94>
- Chise, C., & Jurca, I. (2009). Towards early performance assessment based on UML MARTE models for distributed systems. *2009 5th International Symposium on Applied Computational Intelligence and Informatics*, 521-526. <http://dx.doi.org/10.1109/SACI.2009.5136304>
- Cortellessa, V. (2005). Model-based performance risk analysis. *IEEE Transactions on Software Engineering*, 31(1), 23-20. <http://dx.doi.org/10.1109/TSE.2005.12>
- Czarnecki, K., & Helsen, S. (2003). Classification of model transformation approaches. *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture* (pp. 1-16). Citeseer.
- Distefano, S., Paci, D., Puliafito, A., & Scarpa, M. (2004). UML design and software performance modeling. *Computer and Information Sciences-ISCIS 2004*, 564-573.
- Distefano, S., Scarpa, M., & Puliafito, A. (2011). From UML to Petri Nets: The PCM-Based Methodology. *IEEE Transactions on Software Engineering*, 37(1), 65-79. <http://dx.doi.org/10.1109/TSE.2010.10>
- Gokhale, S. S., Eric Wong, W., Horgan, J. R., & Trivedi, K. S. (2004). An analytical approach to architecture-based software performance and reliability prediction. *Performance Evaluation*, 58(4), 391-412. <http://dx.doi.org/10.1016/j.peva.2004.04.003>
- Goseva-Popstojanova, K., & Trivedi, K. S. (2001). Architecture-based approach to reliability assessment of software systems. *Performance Evaluation*, 45(2-3), 179-204. [http://dx.doi.org/10.1016/S0166-5316\(01\)00034-7](http://dx.doi.org/10.1016/S0166-5316(01)00034-7)
- Grassi, V., Mirandola, R., & Sabetta, A. (2007). Filling the gap between design and performance/reliability models of component-based systems: A model-driven approach. *Journal of Systems and Software*, 80(4), 528-558. <http://dx.doi.org/10.1016/j.jss.2006.07.023>
- Grassi, V., Mirandola, R., Randazzo, E., & Sabetta, A. (2008). KLAPER: An Intermediate Language for Model-Driven Predictive Analysis of Performance and Reliability, 327-356.
- Koziolok, H., & Happe, J. (2008). Performance metrics for specific domains. *Dependability metrics LNCS 4909*, 233-240. http://dx.doi.org/10.1007/978-3-540-68947-8_22
- Lauer, C., German, R., & Pollmer, J. (2011). Fault tree synthesis from UML models for reliability analysis at early design stages. *ACM SIGSOFT Software Engineering Notes*, 36(1), 1-8. <http://dx.doi.org/10.1145/1921532.1921558>
- Lyu, M. R. (2007). Software reliability engineering: A roadmap. *2007 Future of Software Engineering* (pp. 153-170). IEEE Computer Society.
- Mens, T., & Gorp, P. V. (2006). A Taxonomy of Model Transformation. *Electronic Notes in Theoretical Computer Science*, 152, 125-142. <http://dx.doi.org/10.1016/j.entcs.2005.10.021>
- Muller, P. A., Fondement, F., Baudry, B., & Combemale, B. (2010). Modeling modeling modeling. *Software & Systems Modeling (Computer Science)*, 11(3), 347-359. <http://dx.doi.org/10.1007/s10270-010-0172-x>
- Petriu, D. B., & Woodside, M. (2006). An intermediate metamodel with scenarios and resources for generating performance models from UML designs. *Software & Systems Modeling*, 6(2), 163-184. <http://dx.doi.org/10.1007/s10270-006-0026-8>
- Reza, H., Pimple, M., Krishna, V., & Hildle, J. (2009). A Safety Analysis Method Using Fault Tree Analysis and Petri Nets. *2009 Sixth International Conference on Information Technology: New Generations*, 1089-1094. <http://dx.doi.org/10.1109/ITNG.2009.183>

- Rivers, J. A., & Kudva, P. (2009). Reliability Challenges and System Performance at the Architecture Level. *IEEE Design & Test of Computers*, 26(6), 62-73. <http://dx.doi.org/10.1109/MDT.2009.153>
- Singh, L. K. (2011). Software Reliability Early Prediction in Architectural Design Phase: Overview and Limitations. *Journal of Software Engineering and Applications*, 04(03), 181-186. <http://dx.doi.org/10.4236/jsea.2011.43020>
- Tang, A., Tran, M. H., Han, J., & Vliet, H. Van. (2008). LNCS 5281 - Design Reasoning Improves Software Design Quality. *Design*, 28-42.
- UML Profile for Schedulability, Performance, and Time Specification. (2005). *Time*. Retrieved from www.omg.org
- Woodside, M. (2007). Resource Architecture and Continuous Performance Engineering. *Architecture*, 1-14.
- Yang, J., Huang, G., Zhu, W., Cui, X., & Mei, H. (2009). Quality attribute tradeoff through adaptive architectures at runtime. *Journal of Systems and Software*, 82(2), 319-332. <http://dx.doi.org/10.1016/j.jss.2008.06.039>