Tool Support for Software Performance Risk Assessment

Archana Radhakrishnan

Problem Report submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Hany H. Ammar, Ph.D., Chair
Katerina Goseva Popstanjanova, Ph.D.,
Tim Menzies, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2007

Keywords: Performance Risk, Software Performance Risk Assessment Tool, UML,
Software Architecture, Software Metrics

ABSTRACT

Tool Support for Software Performance Risk Assessment

Archana Radhakrishnan

The Software Architecture Risk Assessment (SARA) tool is a utility to compute and analyze different architectural risk factors of software architecture modeled using Unified Modeling Language (UML). The different architectural risk factors are maintainability, requirements, reliability, and performance. The problem report focuses on risk assessment of software performance. Performance risk is a non-functional attribute which is generally assessed during late life cycle of software architecture design and has high impact on safety-critical systems in case of performance failure. The risk is a combination of probability of performance failure and severity and they are estimated using a Software Performance Risk Assessment (SPRA) methodology. A tool is developed supporting this methodology and added as one of the risk assessment features of SARA tool. It performs scenario based performance risk assessment of a model by analyzing annotated UML diagrams. The output is expressed as scenario risk factor and overall system risk factor. It provides a descriptive explanation of the results obtained in each step of the methodology. The results help in identifying the risk factor and the bottleneck component causing high risk of scenarios in the software model. The tool is illustrated with various case studies.

Acknowledgements

I would like to place on record my sincere gratitude to **Dr. Hany Ammar**, my advisor, for the support, guidance, and invaluable inputs that he had consistently provided me throughout my project. No sooner Dr. Hany Ammar assigned me the problem report, I realized that the project would be really challenging, since it involves resolving real time problems.

I had the privilege to enroll in "Advanced Real Time System**"** course under Dr. Hany Ammar which gave me an opportunity to learn in depth on software designing using UML. As I learnt that Dr. Hany Ammar's research topic is more related to software risk assessment using UML, I approached him to be my advisor for my problem report. He provided me opportunity to explore and learn his research projects and also helped me inventing new ideas in developing a tool in Java. It was a great pleasure working with him and his supervision and guidance helped me in completing my project work on time.

I also place on record my sincere thanks to **Dr. Katerina Goseva-Popstojanova** and **Dr. Tim Menzies** for their support, review, and for serving as a member for my graduate committee. The "Software Performance Engineering" course that I took under Dr. Katerina also motivated me in working on my problem report.

I like to thank my family members especially my **mom and dad**, and my friends for being a great support through out my research. I am very grateful to my friend, **Rajesh** who was my mentor and colleague in India. Without his support in helping me in programming concepts, it would have been difficult for me to complete my project. He helped me throughout my project irrespective of the time difference between India and US. Also, I like to thank Randy Hudak, Director of Facilities Management (FM), for funding my masters program, Joe Patten and Barbara Angeletti, whose supervision and support has shaped me into a professional person.

Last but not the least, I like to specially thank my husband, Mr. Anand Chandrashekaran and his parents, Mrs & Mr. Chandrashekaran for their love and motivation throughout my problem report. My husband being an Industrial Engineer helped me in developing my tool more user-friendly by asking me intriguing questions more from user perspective. He suggested me different approaches to solve my technical problems. His involvement while preparing my slides for presentation helped me to represent my ideas in such a way that even a layman could understand my technical project. Incidentally, I must thank him for his taking over the complete household activities that helped me in focusing on my project without any distraction.

I always would hesitate to call Anand’s parents as my in-laws as they bestowed so much of affection and love on me. They motivated me to take up the masters and extended all the moral support to me to complete it. Though they are miles apart, they called me every day and night to know about my problem report updates. Their love and encouragement cannot be matched with anything else in this world. Their sweet words and positive attitude helped me in completing my project.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Performance is defined as the amount of resources needed by the software or the final product to provide full functionality under all possible environmental conditions. Risk is defined as an undesired event or any uncertainty in a system. Performance risk is a combination of two factors: probability of performance failure and the severity due to this failure. Performance failure occurs when a system or system component does not meet or perform the required function of the performance requirements. Though the software system is functionally correct, it may lead to performance failure. Risk assessment of non-functional attributes such as performance is an essential process for every software risk management. Cortellessa et. al., (2005) introduced a SPRA methodology to find the performance risk of a software model. A tool is developed based on the methodology to perform software performance risk assessment and also scenario level and system level sensitivity analysis.

1.1 Problem Statement

The impact of performance has larger significance in software systems especially is safety-critical domain. Non-functional requirement describes not what software will do but how software will do it (Ebert, 1997). In real time environment, there is a wide gap between software developers and performance validation as performance is a non-functional attribute. The developers following either a 'fix-it-later' approach during the software development process that belongs to late-life cycle or devoting very less time and effort for performance validation is still dominant. Work et al. (1995) explains the usual practices where metrics are applied only to software functional requirements. This explains that metrics has to be defined for non-functional requirements as failures are detected in performance. The failure and its probability, its severity, and the components causing risk should be identified during software design that belongs to early life cycle to avoid unpredicted or undesirable consequences in the late life cycle. Early identification and intervention of performance metrics such as response time, utilization etc., are key steps to manage risks of software system. Performance validation performed at the end of development process or in the late life cycle can lead to using more expensive and powerful hardware than originally proposed, time consuming tuning measures, or in extreme cases, completely redesigning the software model (Dimitrov, 2002).

SARA tool is developed as a solution to perform risk management of non-functional attributes in the early life cycle. The problem stated above regarding performance risk is also addressed by extending SARA tool to support SPRA. The SPRA helps in identifying the scenario with high risk factor in a software model and also the component causing the high risk. The tool also performs sensitivity analysis of system risk factor and probability of occurrences of the scenarios in the system as even scenarios with low risk but with high probability of occurrence can also lead to high system performance risk. This helps in improving the system performance as performance failure is identified and avoided in the early life cycle.

1.2 Related Work

Recently, Software Performance Engineering (SPE) has largely been active in software development life cycle as responsiveness of software systems under different workload conditions has become an important factor. The response time is a key attribute of software performance. Performance risk analysis performed early in the software life-cycle facilitates in making drastic changes in the system design so that the final product performs better.

There are many approaches and tools to perform performance risk analysis. These approaches are broad and do not propose any formal methods to evaluate performance. Some of the approaches are direct representation of performance aspects using UML and either the tools support for these approaches are very limited or that the software to performance models must be transferred manually (Dimitrov et al., 2002). There are also other approaches performing performance analysis by combining UML with formal description technique such as Message Sequence Chart (MSC) and Specification and Descriptive Language (SDL) (Dimitrov et al., 2002). This requires specialized skills and in-depth knowledge of the description techniques to perform performance modeling. There are approaches to analyze performance but the sensitivity of the performance risk factor towards its input parameters are not experimented and tested. The risk analysis together with the sensitivity analysis can be used to access the effects of changes in the software model.

Most of the software systems adopt two principal methods for performance analysis: simulation and analytical methods. The analytical methods are faster in generating results; however they produce less accurate results due to their methodology and cannot be used to plan real networks. Some approaches are based on analytical methods that require knowledge on queuing theory, Petri Net, and Markov chain. The simulation methods are slower in generating results than analytical methods but are the only means of planning and dimensioning the real networks (Molkdar et al., 2002). Sharma (2006) also discusses about various approaches and tools for performance attributes such as Software PErformance Evaluation and moDeling (SPEED) based on SPE approach, Use Case Maps to Layered Queuing Networks (UCM2LQN) and LQN solver based on LQNs, Client/Server Software Performance Evaluation (CLIPPSE) for client server systems, evaluates the performance characteristics of the specified software systems for given workload. The results of these approaches and tools are performance predictions and the performance engineers should carry out further investigation of the performance risk to improve the software system.

These real time approaches and tools required extensive knowledge and this spawned research and development of a light-weight approach which is a step-by-step methodology for software performance risk assessment. The major motivation for a new approach and tool are the following characteristics:

- Simulation modeling technology to identify and predict performance failure, bottleneck components and scenarios causing high risk of the software system.
- A user-friendly tool that requires less performance modeling techniques and knowledge from the user.

- An integrated software performance tool that is mostly automatic and the users should not perform actions or computations by hand.
- Detailed picture of performance prediction or the results that helps the user to analyze the cause for performance failure of the system
- A proactive performance engineering solutions tool that focus on all aspects of product life cycle including design, development, and production.

1.3 Research Objectives

The main objectives of the tool supporting SPRA are:

- To automate the SPRA methodology and add as a feature to the SARA tool to support model based performance risk.
- To support UML that is the standardized specification language for object modeling in software performance engineering and provides standards for performance profile (extensions or annotations).
- To understand the UML graphical notation and performance annotations to extract data from the design diagrams (usecase diagram, sequence diagram, and deployment diagram) by accepting input files in XMI format.
- To support software design that has multiple scenarios. The software performance engineering plays a predominant role in most of the real time applications and the tool is developed and can be used for performance risk assessment of these extensive real time applications.
- Ability to find the scenario and system risk factors and also identify the bottleneck components causing the risk.

- To automate sensitivity analysis in the scenario level and system level that helps in assessing the effects of changes in system.
- User friendly tool with good look and feel, performance, portability, and scalability.

1.4 Preview of Chapters

Chapter 2 briefly explains the background of the tool supporting SPRA that includes SPRA, its architectural level, UML performance profile, StarUML tool that is used to develop UML diagrams, the SPRA methodology and the SARA tool. Chapter 3 explains the different UML diagrams and its performance annotations used for SPRA methodology and the tool supporting SPRA with a detailed explanation of it Graphical User Interface (GUI), its development environment, the open-source packages used by the tool and the XMI parser, and an introduction about system level and scenario level Sensitivity analysis performed by the tool. Chapter 4 demonstrates the tool supporting SPRA by testing it with various real time applications such as ecommerce application and earth observation system. Chapter 5 discusses the conclusion of the report and about its future work.

CHAPTER 2

BACKGROUND

2.1 Software Performance Risk Assessment (SPRA)

The goal of many organizations today is a light-weight approach that helps in improving system performance by reducing performance risk. The reason being the proactive assessment of performance risk will lead to problem identification, analysis, and resolution activities before the development process. This facilitates in reducing additional development costs, avoiding time-consuming performance tuning, and also helps in time-to-market performance critical software systems. The result is a heuristic approach of model-driven SPRA methodology introduced by Cortellessa et al. (2005) based on quantitative analysis of system performance. It also requires thorough understanding of all elements of system performance to perform performance risk assessment. The SPRA focuses on Model Driven Architecture (MDA) that is used in object-oriented design of software systems. MDA is a technology introduced by Object Management Group (OMG) built using UML that provides a set of guidelines for structuring specifications expressed as models. SPRA is applied on any platform-independent models built using MDA and UML technology.

The performance requirements of SPRA is classified based on two types: a) time-related where, for e.g., the completion time of a specific operation must be less than a certain threshold and b) resource-related where, for e.g., the utilization of a specific device must fall into a certain range (Cortellessa et al., 2005). The performance risk analysis focuses on the time-related requirement. The SPRA is also based on architectural level performance-based risk analysis and UML Performance Profile.

2.1.1 Architectural level Performance based Risk Analysis

The architectural level performance-based risk is obtained from two types of models that provide information of architectural assessment: the software execution model and system execution model (Smith et al., 2001).

The software execution model represents key aspects of the software execution behavior and they are used to identify performance problems due to poor architectural decisions. That is, the software execution model provides a preliminary understanding of the software performance. The model accommodates current execution of several instances of the scenarios of the software system in a single instance. The single instance is identified as the worst case analysis of the scenario and is provided as input to system execution model. The software execution model is implemented by extending the UML sequence diagram.

The system execution model is a dynamic model that characterizes software performance in the presence of factors, such as multiple users or other workloads that could cause contention for resources. The results obtained by solving the software execution model provide input parameters for the system execution model. Solving the system execution model provides the following additional information (Guedem, 2004):

1 more precise metrics that account for resource contention

2 sensitivity of performance metrics to variations in workload composition
3 identification of bottleneck resources
4 comparative data on options for improving performance through workload changes, software changes, and hardware upgrades

2.1.2 UML Performance Profile

Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system. It covers the complete software life cycle of software performance modeling, from requirements, analysis through implementation. The UML diagrams that are used to estimate the performance failure of a system are represented as annotations or extensions. The diagrams are annotated or extended using stereotypes and arguments to annotate performance requirements. The stereotype in UML is represented with << >> symbol. The extended UML is based on the UML profile for performance that is specified by OMG UML profile for schedulability, performance, and time (SPT) (UML Profile, 2005).

The UML diagrams such as use case diagram, sequence diagram and deployment diagram with extended annotations are used for the SPRA methodology. The performance domain/requirement model is represented as UML use case diagrams and it is a composition of one or more use cases. The use case describes scenarios that are formal, sequential descriptions of steps taken to carry out the use case, or the flow of events that occur during a use case instance. That is, the scenario is a description of the interactions between the system and its environment or between the internal objects involved in particular use of the system under development (Smith et al., 1997). The annotated sequence diagram is used to describe the scenarios of the model. The hardware environment of the scenarios is represented as annotated deployment diagrams.

2.2 StarUML Tool

The UML diagram representing the software model is developed using StarUML tool which is an open source project. Features supported by StarUML are:

- Latest UML 2.0 standard: The tool provides support to latest UML standards. The current standard for UML introduced by OMG is UML 2.0. This helps the user to take advantage of the latest UML standards through an open source tool.
- MDA: It is designed to support MDA which is a modeling technology used by SPRA and provides customization variables such as UML performance profile. This helps in extending UML with performance annotations and extensions for SPRA.
- User-friendly: The tool is user-friendly and helps in developing fast, flexible annotated UML diagrams. The tool is implemented with easy-to-use dialogs, diagram overview such as Model Explorer and Diagram Explorer. It does not require any expertise knowledge to use the tool other than UML.
- XMI file format: The UML diagrams developed can be exported as XML Metadata Interchange (XMI) files which is the recommended interchange format for UML models.
- Plug-in Architecture: Many users require more and more functionalities to plug into the software modeling tools. To meet the requirements, the tool must have

well-defined plug-in platform. StarUML provides simple and powerful plug-in architecture.

- A compelling replacement of commercial tools such as Rational Rose, Visio, and Together

Though many software modeling tools are available in the market, StarUML provides the features almost supported by most commercial tools and is also an open source project.

2.3 SPRA Methodology

The SPRA methodology is applied for various scenarios in a model to estimate a) probability of performance failure of a scenario in a software model, b) its severity, and c) its bottleneck component. The steps are explained in brief.

STEP 1 - Assign demand vector to each action/Interaction in Sequence Diagram; build a Software Execution Model

STEP 2 – Add hardware platform characteristics on the Deployment Diagram; conduct stand-alone analysis

STEP 3 – Devise the workload parameters; build a System Execution Model; conduct contention-based analysis and estimate probability of failure as a violation of a performance objective

STEP 4 - Conduct severity analysis

STEP 5 – Estimate the performance risk of the scenario; Identify high-risk components

In **Step 1**, the demand vectors are obtained for each action of a scenario from the annotated UML sequence diagram. The amount of resources required for each action is a) CPU demand expressed in terms of $CPU_{work\ units}$ which is a measure of CPU required to perform this action and b) Disk demand expressed in terms of $Disk_{data}$ which contains the number of bytes that are read or written to disk. The amount of resources required for each interaction is network demand expressed in terms of MSG which contains the size of data being exchanged. Then, the demand vectors assigned to the sequence diagram is translated into Execution graph which is called the Software execution model. The difference between sequence diagram (SD) and execution graph (EG) is that the SD describes the sequence of action (internal to the components) and interactions (among components) that are triggered from an external event and EG is a structure describing all possible sequence of actions that a system performs in response to external triggers. To calculate the response time of the system, the completion times of all the actions in the longest path, where the demand is highest, is calculated. In case of concurrent executions, a worst case analysis which assumes that concurrent branches serialize (i.e. when one branch is completed, the next begins) gives the longest path.

To calculate time-based demands, the demand vectors are combined with hardware device characteristics because the same demands may take considerably different time depending on the hardware devices. In **Step 2**, the hardware platform information is obtained from annotated deployment diagram. An annotated UML diagram for client/server interconnection is defined in Gomma et al. (2000). Also, the service time or the speed of each hardware device is obtained as input. Based on the software execution model, the total demand for each hardware device expressed in work units for CPU, KB for disks and KB for networks is calculated. Then, this is multiplied with service time of corresponding hardware devices to find the service demands, Di, for each hardware device in time units. Adding the service demands (completion time) of all demand vector provide the total elapsed time (response time). This is called the standalone analysis that provides elapsed time of the scenario executed on a hardware platform with single user or workload.

Then, the performance objective is obtained as input to analyze whether the completion time value of the scenario meets the performance requirement. Failure to meet the requirement would mean that the probability of performance failure would occur in the software system. So, either the software design or the hardware configuration or the performance objective has to be reconsidered. But if the time value meets the performance objective, then system behavior can be analyzed under different workload conditions. This is called Content-based Analysis that considers any delay due to contention of resources. Thus, the value obtained from the standalone analysis helps to find whether the software system has the probability of performance failure without further investigation. Smith et al. (2002) explains a complete analysis of an in-range data reading scenario which includes, modeling a software execution model for the scenario, analyzing in different number of hardware machines and comparing the throughput values with the performance objective. In this methodology, instead of throughput, the response time of the scenario is calculated and is compared with the performance objective.

In **Step 3**, the batch workload is considered which is parameterized with population N. The information obtained as input from the software execution model is mapped with the parameters of deployment diagram to form a complete system execution model. Here, the worst-case situation is considered and the completion time of the scenario is calculated by summing the service demands of the longest path. To estimate the probability of performance failure, the asymptotic bound analysis on response time is used (Guedem, 2004). There are many advantages of bounding analysis, few of which can be listed as follows:

1 The influence of system bottleneck is highlighted and quantified
2 These bounds can be computed simply and quickly (even by hand)
3 It takes very less computation to determine response time with respect to system workload intensity.

The asymptotic bound analysis, as explained in Lazowska et al. (1984), provides optimistic (upper) and pessimistic (lower) bounds on system response time. The parameters required for the bound analysis are:

1 D_{max}, the largest service demand in the scenario
2 $D = \sum Di$, sum of all demands in the scenario
3 N, number of customers or workload

and the equation defining the bounds on response time is:

$$max(D, N * D_{max}) \leq R(N) \leq N * D$$

where $\max(D, N * D_{max})$ is the upper bound and N * D is the lower bound.

With the performance objective provided as input to the methodology and from the asymptotic bound on the performance, the probability of performance failure at a particular workload is obtained by partitioning the plot into 3 partitions as follows:

1. Zone Z1: The failure probability is zero, i.e, Z1 = 0.
2. Zone Z2: The failure probability, Z2 = (Upper bound – Performance Objective) / (Upper bound – Lower bound).
3. Zone Z3: The failure probability, Z3 = 1

The advantage of using this methodology in finding the probability of performance failure using asymptotic bound is

1 The equations requires only a few arithmetic operations

2 The computation is independent of number of resources in the model and also range of customers which scales the methodology well

3 The methodology does not require the knowledge of Queuing Network which is complex.

In **Step 4**, a Functional Failure Analysis (FFA) is performed that is a technique used to estimate and assess severity of the scenario. This high level FFA gives a comprehensive view of the ways in which the system can fail (Hassan, 2004). It uses system parameters to describe the effects of failure. The FFA analyses each event and the guidewords such as 'Late' and 'Early' for performance failure (more are discussed in Hassan (2004)), and decides whether hypothetical failure mode is possible and if so, what are their consequences or effects. The severity is classified as a)catastrophic, b)critical, c)major, and d)minor. The input to the methodology is the severity of different events based on

FFA. This methodology combines the probability of performance failure with severity to estimate the risk factor of the scenarios.

In **Step 5**, the bottleneck component causing the high performance risk in the scenario is estimated by calculating residence time of each component in a scenario. For a component Ci in Sj scenario, the residence time is given by

Rci in Sj = Overall residence time of Ci in Sj/Response time of Sj [3]

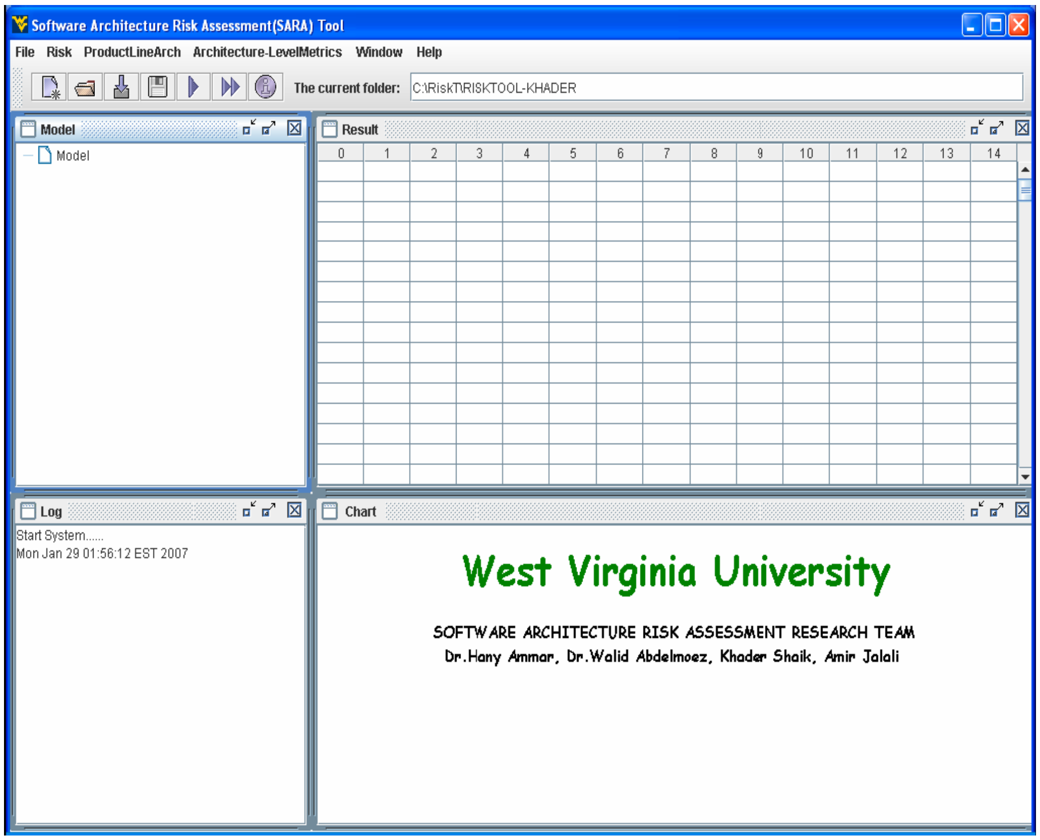
The component with the highest residence time is the cause for performance failure in the scenario and is the bottleneck component that requires more time and resources which can be spent on other devices for improving performance. The detailed performance risk assessment methodology is explained in (Cortellessa, 2005).

2.4 SARA Tool

Clements et al. (2002) explains Software Architecture as "*Software Architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them*".

The SARA tool is developed as an automated tool to perform different architectural risk factors by providing software architecture as input. The software architectures are developed as UML models developed using software modeling tools such as StarUML, Rational Rose, and Java Understand static analysis files. The SPRA methodology is implemented in SARA tool as one of its features and accepts input from StarUML. The GUI of SARA tool is easy-to-use and has file menuitems that facilitates user to easily provide the input, select the risk factors, and perform different actions. The tool performs the calculation based on user's command and displays the output in numerical value and using graphs and bar charts. The different frames in the GUI of the tools are expandable upto the maximum size for best visibility and clarity of data.

There is no special requirement for the hardware. Any desktop with common or average configuration in the market could be used to run SARA tool, For instance, 1.0-2.0 GHZ PIII CPU with 256 MB Memory, 20 GB hard disk etc. The user interface of SARA tool is shown in figure 1.


Software Architecture Risk Assessment(SARA) Tool
File
Risk
ProductLineArch
Architecture-LevelMetrics
Window
Help
The current folder:
C:\RiskT\RISKTOOL-KHADER
Model
Model
Result
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Log
Chart
West Virginia University
SOFTWARE ARCHITECTURE RISK ASSESSMENT RESEARCH TEAM
Dr.Hany Ammar, Dr.Walid Abdelmoez, Khader Shaik, Amir Jalali

```
Start System......
Mon Jan 29 01:56:12 EST 2007
```

Figure 1 User Interface of SARA Tool

CHAPTER 3

TOOL FOR SPRA

3.1 UML Performance Diagrams and Annotations

In the UML performance profile, an extension to UML notation to performance annotation is proposed. The performance domain model is modeled using pa-UML diagrams where pa is performance analysis. The performance domain model is classified into workload, scenario, and resource based analysis. A framework that does performance modeling using UML where the UML performance model are then transformed into stochastic queuing theory with simultaneous resource possession is presented in Kakhipuro (1999). Then the queues are derived from the class diagram and workload from the collaboration diagram. A different type of performance annotation on UML diagrams is used in the model based on a heuristic SPRA methodology defined in Cortellessa et al. (2005).

SPRA methodology focuses on scenario based assessment using UML diagrams such as Use case diagrams, Sequence diagrams, and Deployment diagrams. The UML diagrams of SPRA is developed using StarUML tool. The performance-annotated UML diagram is mapped to a software model which can be used to analyze the performance of the model. This is provided as input to the tool supporting SPRA and the figure 2 provides a clear picture of the input requirements to the tool.

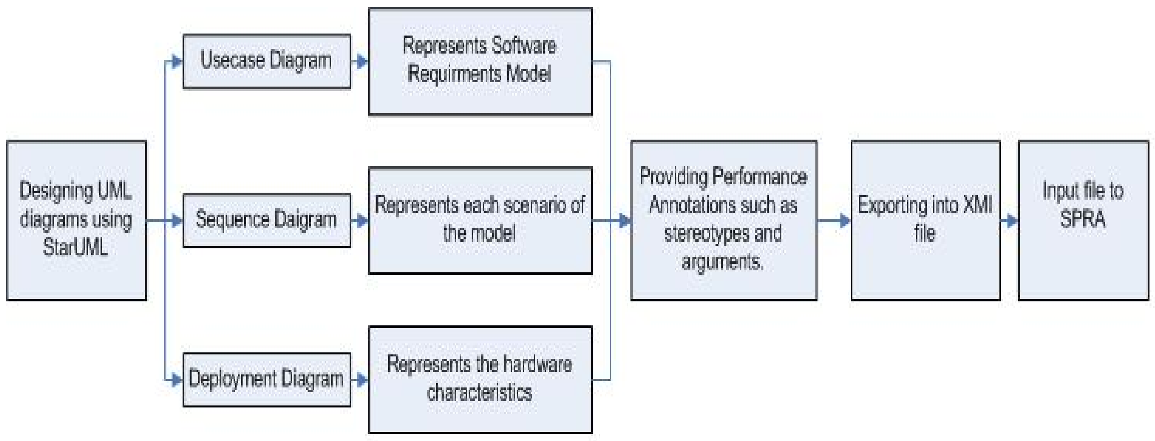

Designing UML diagrams using StarUML
Usecase Diagram
Sequence Daigram
Deployment Diagram
Represents Software Requirments Model
Represents each scenario of the model
Represents the hardware characteristics
Providing Performance Annotations such as stereotypes and arguments.
Exporting into XMI file
Input file to SPRA

Figure 2 Input Requirements to the Tool supporting SPRA

StarUML performs software modeling by creating 'Project' that is the main management unit and can manage one or more software models. Project is the top-level package and is split into multiple software model elements. In general, one project is saved as a file with extension '.UML' format or as XMI format. A project file contains the following information:

- Information of all the models
- Information of all the diagrams and views
- UML performance profile used in the project

The default model elements while creating a project are a) Use Case Model, b) Design Model, c) Analysis Model, d) Implementation Model, and e) Deployment Model. The models used for SPRA methodology are a) Use Case and b) Deployment Model.

To create a new diagram for the project,

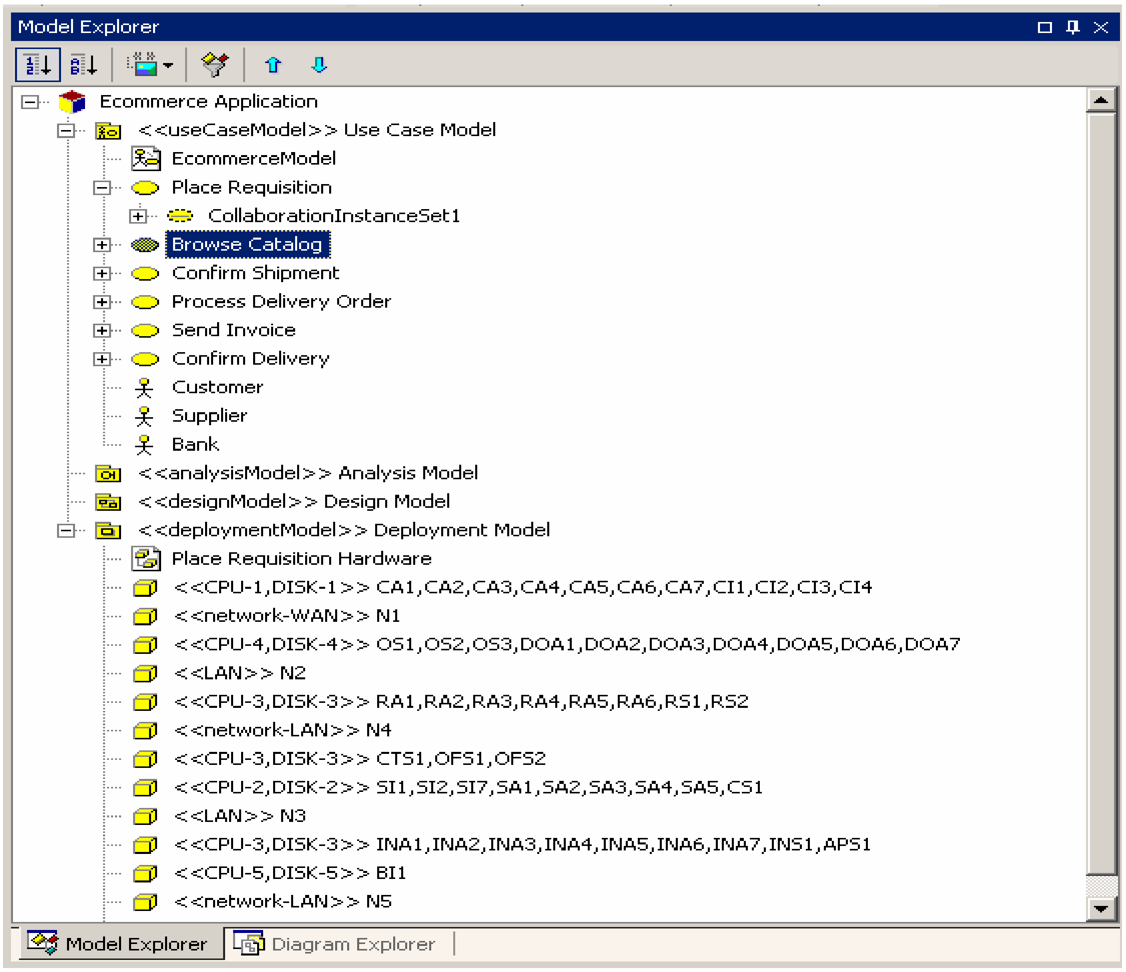
- Select from the model explorer, a model element (e.g., Use Case Model) to contain new diagram

- Right-click and select the [Add Diagram] menu.

To create element for the diagram,

- Select an element type to create from the pallet
- Drag the mouse to select an area to specify the size of the new element

The figure 3 shows the Model Explorer of ecommerce application where ‘Ecommerce Application’ is the project file and ‘Use Case Model’ and ‘Deployment Model’ are the model elements.



```
Ecommerce Application
    <<useCaseModel>> Use Case Model
        EcommerceModel
        Place Requisition
            CollaborationInstanceSet1
        Browse Catalog
        Confirm Shipment
        Process Delivery Order
        Send Invoice
        Confirm Delivery
        Customer
        Supplier
        Bank
    <<analysisModel>> Analysis Model
    <<designModel>> Design Model
    <<deploymentModel>> Deployment Model
        Place Requisition Hardware
        <<CPU-1,DISK-1>> CA1,CA2,CA3,CA4,CA5,CA6,CA7,CI1,CI2,CI3,CI4
        <<network-WAN>> N1
        <<CPU-4,DISK-4>> OS1,OS2,OS3,DOA1,DOA2,DOA3,DOA4,DOA5,DOA6,DOA7
        <<LAN>> N2
        <<CPU-3,DISK-3>> RA1,RA2,RA3,RA4,RA5,RA6,RS1,RS2
        <<network-LAN>> N4
        <<CPU-3,DISK-3>> CTS1,OFS1,OFS2
        <<CPU-2,DISK-2>> SI1,SI2,SI7,SA1,SA2,SA3,SA4,SA5,CS1
        <<LAN>> N3
        <<CPU-3,DISK-3>> INA1,INA2,INA3,INA4,INA5,INA6,INA7,INS1,APS1
        <<CPU-5,DISK-5>> BI1
        <<network-LAN>> N5
```

Figure 3 Model Explorer of Ecommerce Application in StarUML

3.1.1 Use Case Model

A Use Case Model describes the functionality to be built in the proposed system. It represents a discrete unit of interaction between user or actor (human or machine) and the system. The model can include multiple use cases or scenarios and each scenario is extended to an annotated Sequence Diagram. An example of use case model supported by StarUML is explained with an ecommerce application as shown in figure 4.

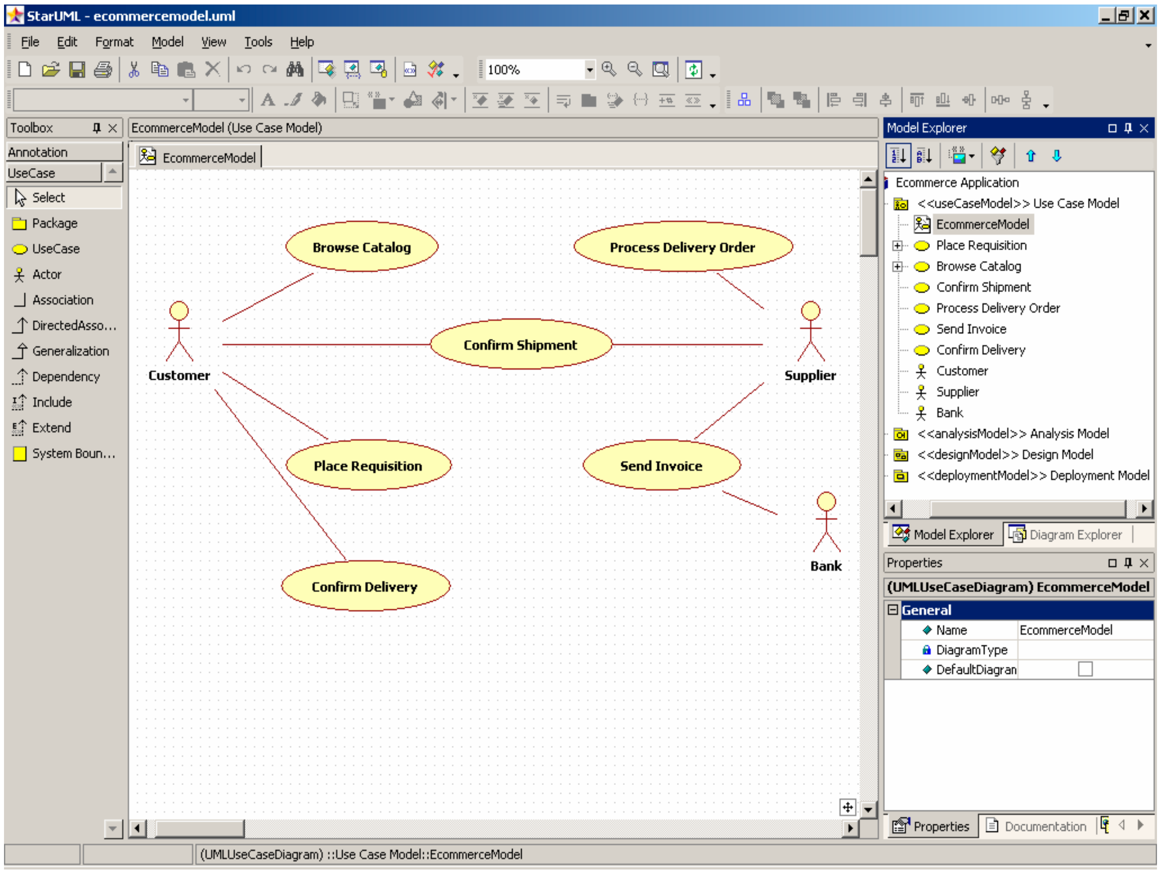
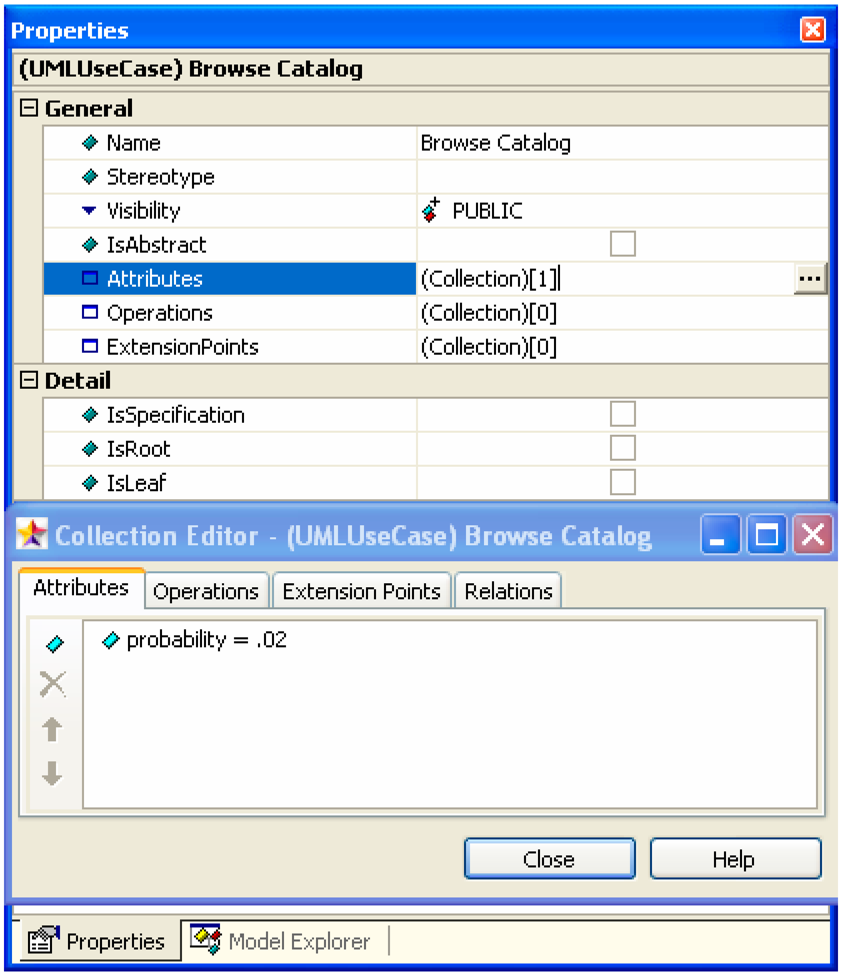


Figure 4 Use Case Diagram in StarUML

The Use Case Diagram shows an ecommerce application where the Customer (actor) is interacting with the Supplier (actor) through internet (system). The model has multiple scenarios such as Browse Catalog, Place Requisition, Process Deliver Order, Confirm

Shipment, Confirm Delivery, and Send Invoice. Each scenario is annotated to provide information about its probability of occurrence that is used to calculate system risk factor. The annotation of use case diagram is optional as either the probability of occurrence can be provided as input through the diagram or in the user interface of the tool. If the use case diagram is annotated, the tool automatically uses the input and the user can change it, if required, in the user interface of the tool.


Properties

(UMLUseCase) Browse Catalog

General

Name	Browse Catalog
Stereotype	
Visibility	PUBLIC
IsAbstract	☐
Attributes	(Collection)[1]
Operations	(Collection)[0]
ExtensionPoints	(Collection)[0]

Detail

IsSpecification	☐
IsRoot	☐
IsLeaf	☐

Collection Editor - (UMLUseCase) Browse Catalog

Attributes | Operations | Extension Points | Relations

probability = .02

Figure 5 Annotations for Use Case Diagram in StarUML

Annotation Definition for Use Case Diagram:

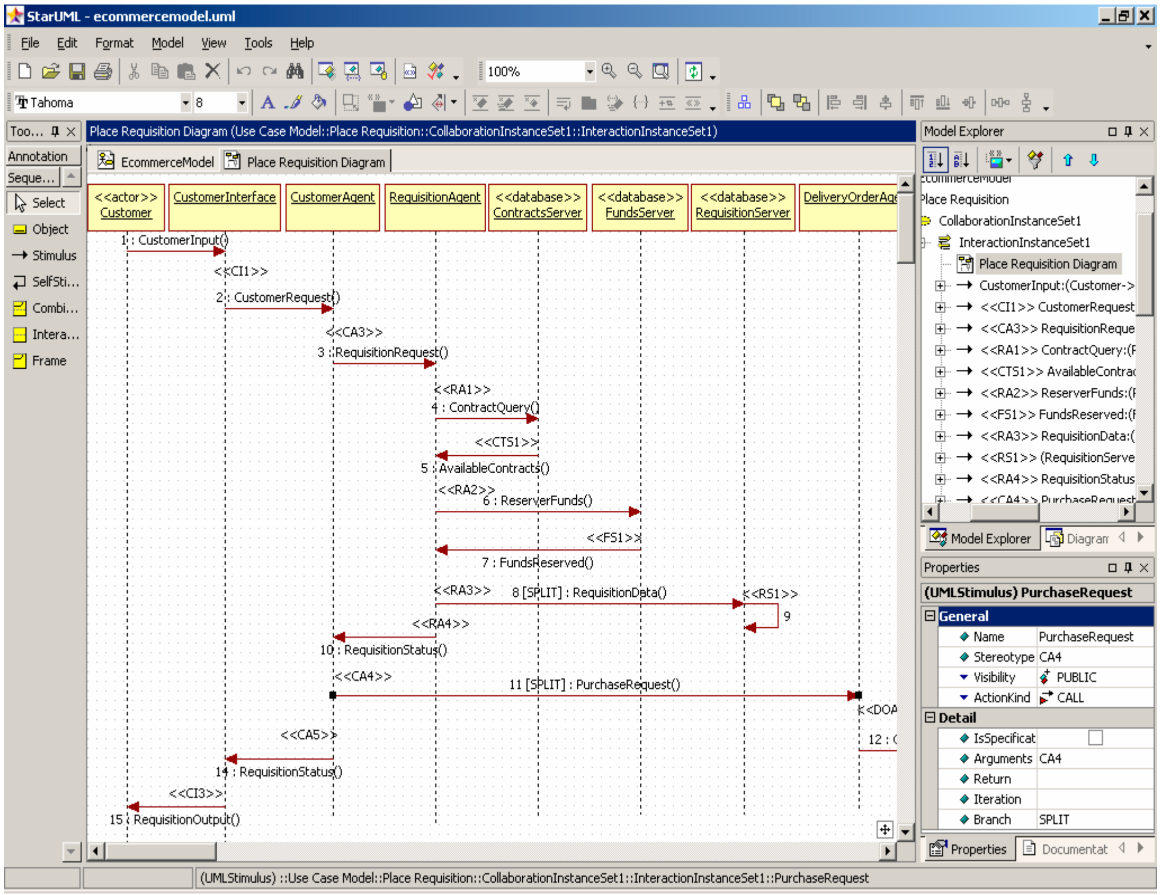
The figure 5 shows the annotation of 'Browse Catalog' scenario in the use case diagram. Select the scenario from the use case diagram and select the *Attributes* section of General dialog. Click "…" to open the Collection Editor window. In the 'Attributes' tab, click the add attribute symbol, to add an attribute. Then add an attribute with 'probability' in the *Name* section and its value in the *InitialValue* section of *Properties* window of Attributes. This annotates the probability of occurrence of the scenario in the system model.

Each scenario is further extended to annotated sequence diagram. Sequence diagrams provide a graphical representation of object interactions over time. Each sequence diagram typically represents a single Use Case 'scenario' or flow of events. The sequence diagram in use case model of StarUML is represented as objects and their interaction (using Stimulus or SelfStimulus). Each interaction is denoted by demand vectors which are represented as stereotypes and tagged values.

Annotation Definition for Sequence Diagram:

Demand Vectors

The annotation of Sequence diagram basically involves defining the demand vectors that is a unique representation of the objects/components of the scenario in each interaction between them. This means that the same object/component is used in different scenarios and the demand vectors helps in identifying the components of the particular scenario. The heuristic SPRA methodology has defined an annotation for the demand vectors where the first letter of each word of the object appended by a number is used for its unique representation. The figure 6 shows an annotated sequence diagram of *Place Requisition* scenario in StarUML.



(UMLStimulus) PurchaseRequest

General

Name	PurchaseRequest
Stereotype	CA4
Visibility	PUBLIC
ActionKind	CALL

Detail

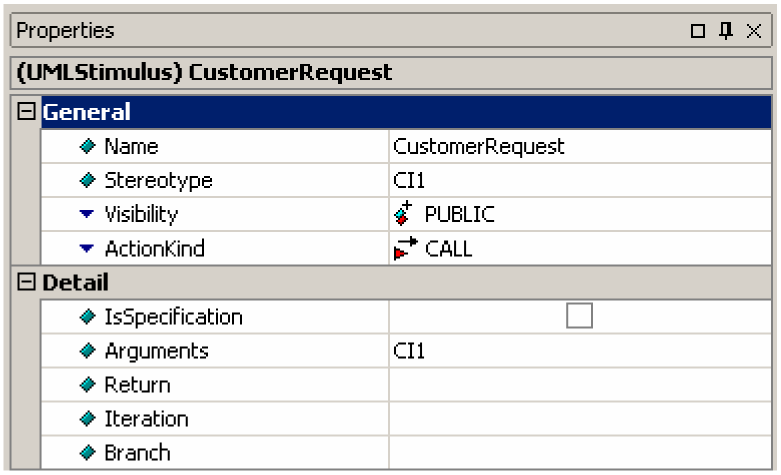
IsSpecificat	☐
Arguments	CA4
Return	
Iteration	
Branch	SPLIT

(UMLStimulus) ::Use Case Model::Place Requisition::CollaborationInstanceSet1::InteractionInstanceSet1::PurchaseRequest

Figure 6 Sequence Diagram in StarUML

The Place Requisition scenario in figure 6 shows interaction between its objects/components. Please note that the scenario names mentioned in the use case diagram and the annotations, interactions/steps, and component names in sequence diagram MUST not have **space** (“ “) in between them. The interaction between components is mapped to demand vectors. For e.g., the *CustomerRequest()* which is the step 2, in figure 6, is an action that is initiated from the *CustomerInterface* object/component, and is assigned to a demand vector, *CI1* (as 'C' from Customer and 'I' from Interface of Customer Interface and appended to '1') . Also, the *RequisitionOutput()* in step 14 is an action initiated from the same component and is assigned to a demand vector, *CI3*. Thus the demand vectors represent the objects/component but they are unique for every interaction in a scenario.

The demand vectors MUST be provided in the 'Stereotype' and 'Arguments' of *Detail* section in the *Properties* of the interaction as shown in figure 7.



Properties

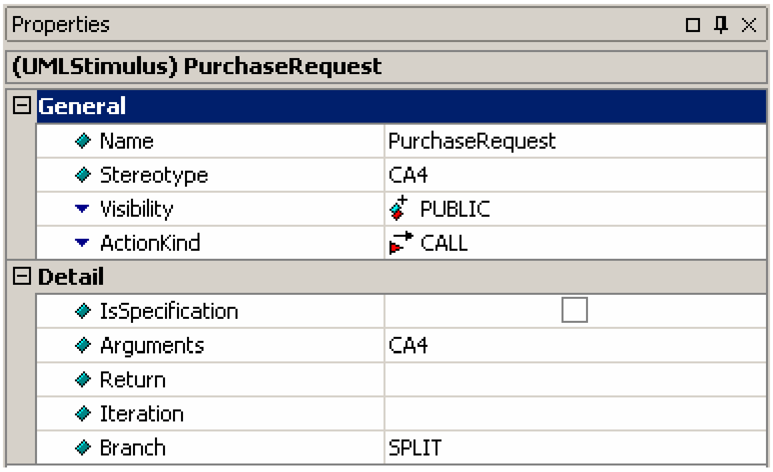
(UMLStimulus) CustomerRequest

General	
Name	CustomerRequest
Stereotype	CI1
Visibility	PUBLIC
ActionKind	CALL
Detail	
IsSpecification	☐
Arguments	CI1
Return	
Iteration	
Branch	

Figure 7 Properties section of an interaction in StarUML

Concurrent Actions

In case of concurrent actions, the annotations used are 'SPLIT' or 'FORK' or 'JOIN'. SPLIT denotes the sequence of actions that does not have a join and FORK-JOIN denotes the sequence of actions that splits into parallel actions and finally joins into a single action. The 'SPLIT', 'FORK', and 'JOIN' are the keywords that MUST be provided in the 'Branch' of *Detail* section in the *Properties* of the interaction as shown in figure 8.


Properties

(UMLStimulus) PurchaseRequest

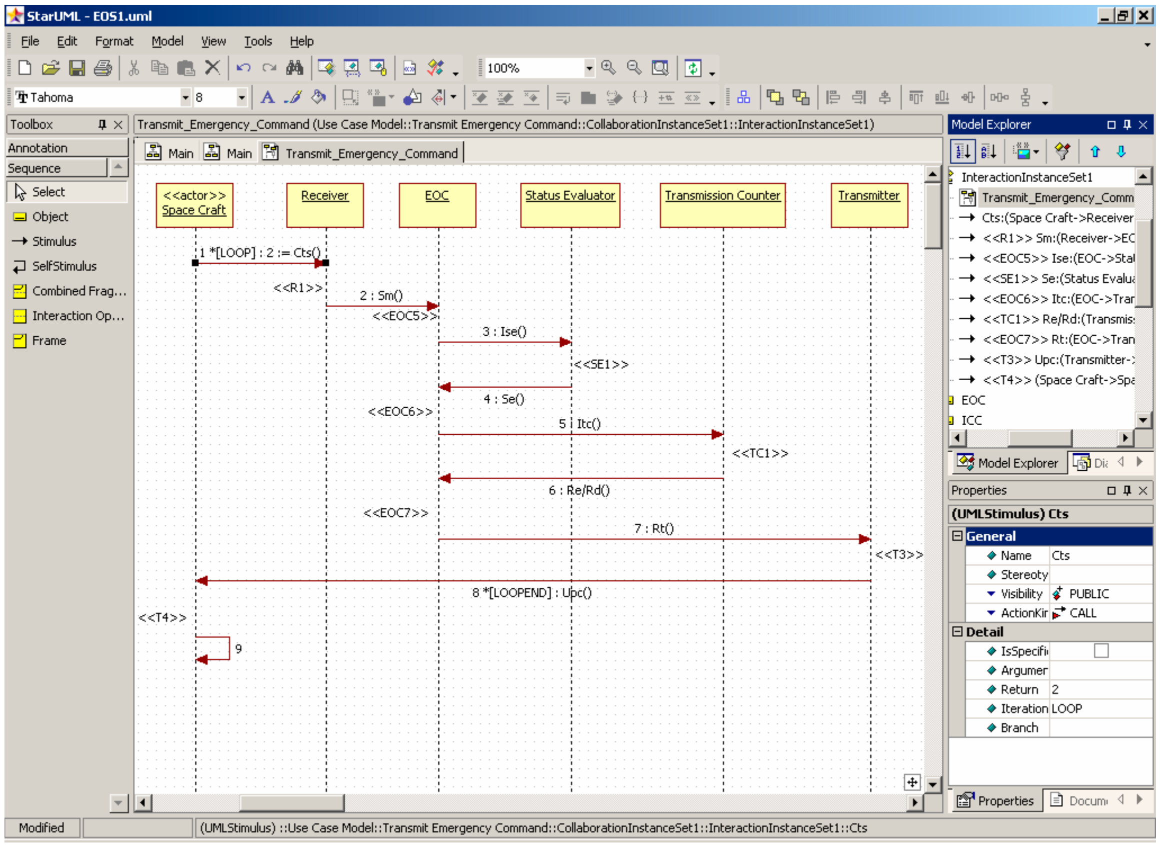
General	
Name	PurchaseRequest
Stereotype	CA4
Visibility	PUBLIC
ActionKind	CALL
Detail	
IsSpecification	☐
Arguments	CA4
Return	
Iteration	
Branch	SPLIT

Figure 8 Properties section of interaction with concurrent action

Looping

The looping of interaction in sequence diagram is supported or annotated using 'LOOP' and 'LOOPEND' keyword. Though StarUML supports modeling of sequence diagram with looping by choosing 'Combined Fragment' from the pallet of Sequence diagram, the XMIAddIn.dll does not support it and StarUML throws 'Access Violation' while exporting sequence diagram with combined fragment into XMI file. To support looping in the tool supporting SPRA, the sequence diagram is annotated with 'LOOP' keyword in the *'Iteration'* of *Properties* section of the interaction where the looping starts and providing the number of times the loop has to be executed in the '*Return'* of *Properties* section of the interaction. To end the loop, the interaction where the loop ends is annotated with 'LOOPEND' keyword in the *Iteration'* of *Properties* section of the interaction.

Figure 9 shows that the annotated sequence diagram of ‘Transmit Emergency Command’ scenario of Earth Observation System model. The sequence diagram is annotated with looping starting from interaction, Cts() to interaction, Upc() that has to be executed for 2 times. In the Properties section of the Cts() interaction as shown in the right bottom of the below figure, it is observed that the Cts() is annotated with ‘LOOP’ keyword in the *‘Iteration’* and to denote that the loop has to be executed 2 times, the Cts() is annotated with ‘2’ in the *‘Return’*. The tool supporting SPRA multiplies the amount of resources provided for each demand vector with the number of times the loop has to be executed.



Transmit_Emergency_Command (Use Case Model::Transmit Emergency Command::CollaborationInstanceSet1::InteractionInstanceSet1)

Model Explorer:
- InteractionInstanceSet1
- Transmit_Emergency_Comm
- Cts:(Space Craft->Receiver
- <<R1>> Sm:(Receiver->EC
- <<EOC5>> Ise:(EOC->Stat
- <<SE1>> Se:(Status Evalua
- <<EOC6>> Itc:(EOC->Trar
- <<TC1>> Re/Rd:(Transmis
- <<EOC7>> Rt:(EOC->Tran
- <<T3>> Upc:(Transmitter->
- <<T4>> (Space Craft->Spa
- EOC
- ICC

Properties: (UMLStimulus) Cts

General	
Name	Cts
Stereoty	
Visibility	PUBLIC
ActionKir	CALL
Detail	
IsSpecifi	☐
Argumer	
Return	2
Iteration	LOOP
Branch	

(UMLStimulus) ::Use Case Model::Transmit Emergency Command::CollaborationInstanceSet1::InteractionInstanceSet1::Cts

Figure 9 Looping Annotated in Sequence Diagram

3.1.2 Deployment Model

The Deployment Model shows the hardware platform for your system, the software that is installed on the hardware, and the middleware (network connection) used to connect the disparate machines to one another. The deployment diagram explores the architecture of embedded system and shows how the hardware and software components work together. The software installed on the hardware and the hardware devices are represented as 'Nodes' where the software components are represented as node names and hardware devices are represented as stereotypes. The network connection (network or LAN or WAN) between the nodes is represented as 'Association' in pallet of deployment model. The type of network connection is represented as stereotypes and name of the network connection is represented as node names.

Annotation Definition for Deployment Diagram:

The demand vectors used in the sequence diagram MUST be used as software installed in the hardware devices. They are provided in the node names of the deployment diagram. The hardware devices such as the processor and disk MUST be represented as 'CPU' appended with a number/text such as 'CPU-1' or 'CPU-DB' and 'DISK' appended with a number/text such as 'DISK-1'or 'DISK-DB' respectively. The hardware devices are represented using << >>. The network connection between hardware devices is also represented using nodes in the deployment diagram where the name of the network is represented in the node names and the type using << >>. The type MUST be represented using 'network' appended with number/text such as 'network-LAN' or using 'LAN' appended with number/text or using 'WAN' appended with number/text. The figure 10 shows the Deployment model of ecommerce application in StarUML.

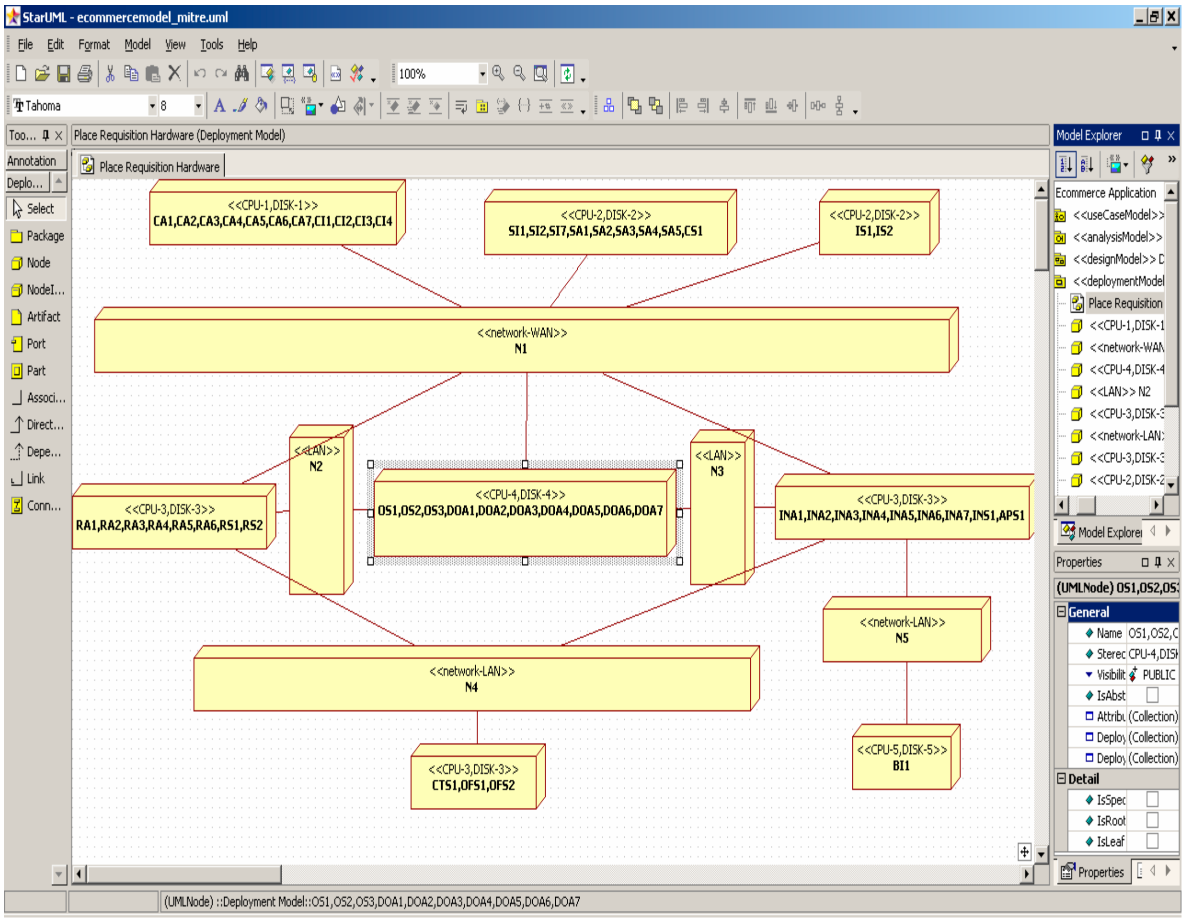

StarUML - ecommercemodel_mitre.uml
File Edit Format Model View Tools Help
Place Requisition Hardware (Deployment Model)
Place Requisition Hardware
<<CPU-1,DISK-1>> CA1,CA2,CA3,CA4,CA5,CA6,CA7,CI1,CI2,CI3,CI4
<<CPU-2,DISK-2>> SI1,SI2,SI7,SA1,SA2,SA3,SA4,SA5,CS1
<<CPU-2,DISK-2>> IS1,IS2
<<network-WAN>> N1
<<LAN>> N2
<<LAN>> N3
<<CPU-3,DISK-3>> RA1,RA2,RA3,RA4,RA5,RA6,RS1,RS2
<<CPU-4,DISK-4>> OS1,OS2,OS3,DOA1,DOA2,DOA3,DOA4,DOA5,DOA6,DOA7
<<CPU-3,DISK-3>> INA1,INA2,INA3,INA4,INA5,INA6,INA7,INS1,APS1
<<network-LAN>> N5
<<network-LAN>> N4
<<CPU-3,DISK-3>> CTS1,OFS1,OFS2
<<CPU-5,DISK-5>> BI1
Model Explorer
Ecommerce Application
Place Requisition
Properties
(UMLNode) OS1,OS2,OS
General
Name OS1,OS2,C
Sterec CPU-4,DISK
Visibilit PUBLIC
Detail
(UMLNode) ::Deployment Model::OS1,OS2,OS3,DOA1,DOA2,DOA3,DOA4,DOA5,DOA6,DOA7

Figure 10 Deployment Diagram in StarUML

The components of the Place Requisition scenario such as CI1 and CI3 are installed in the hardware devices CPU-1, DISK-1 in a node as shown in the above figure where the hardware device is provided in the ‘Stereotype’ and software components are provided in the ‘Name’ of *General* section in the *Properties* of the node. Similarly, the type of network connection such as network-WAN is provided in the ‘Stereotype’ and name of the network such as N1 is provided in the ‘Name’ of *General* section in the *Properties* of the node representing network. Please note that the hardware devices and list of components mentioned in deployment diagram MUST not have **space** (“ “) in between them. This explains the complete development of annotated UML diagrams of a software performance model using StarUML tool. The model is converted to a XMI file using StarUML and provided as input to the tool supporting software performance risk assessment.

3.2 Tool Support for SPRA

The SPRA methodology is a more extensive approach in performing performance risk assessment and to bring it to practice in real time and also to be used by stakeholders, a tool is developed supporting the methodology. The basic development requirements of the tool are: to understand the input file in the XMI format using an XMI parser explained later in this chapter, perform SPRA computations and provide the required output in numerical data or in graphical form or using bar charts. Apart from performing risk analysis, the tool also performs sensitivity analysis where the changes in software performance risk factor are observed by changing its input parameters. The tool provides scenario level and system level sensitivity analysis that helps in user for to make performance based decisions.

Figure 11 shows an overall picture of the tool supporting SPRA as a flow graph where the blocks highlighted in blue represent inputs from the user, the blocks in green represent the SPRA methodology, blocks in orange represents the output, and the blocks in pink represents the scenario-level sensitivity analysis. The annotated UML diagrams are designed using StarUML as per the performance annotations proposed in the section 3.1.

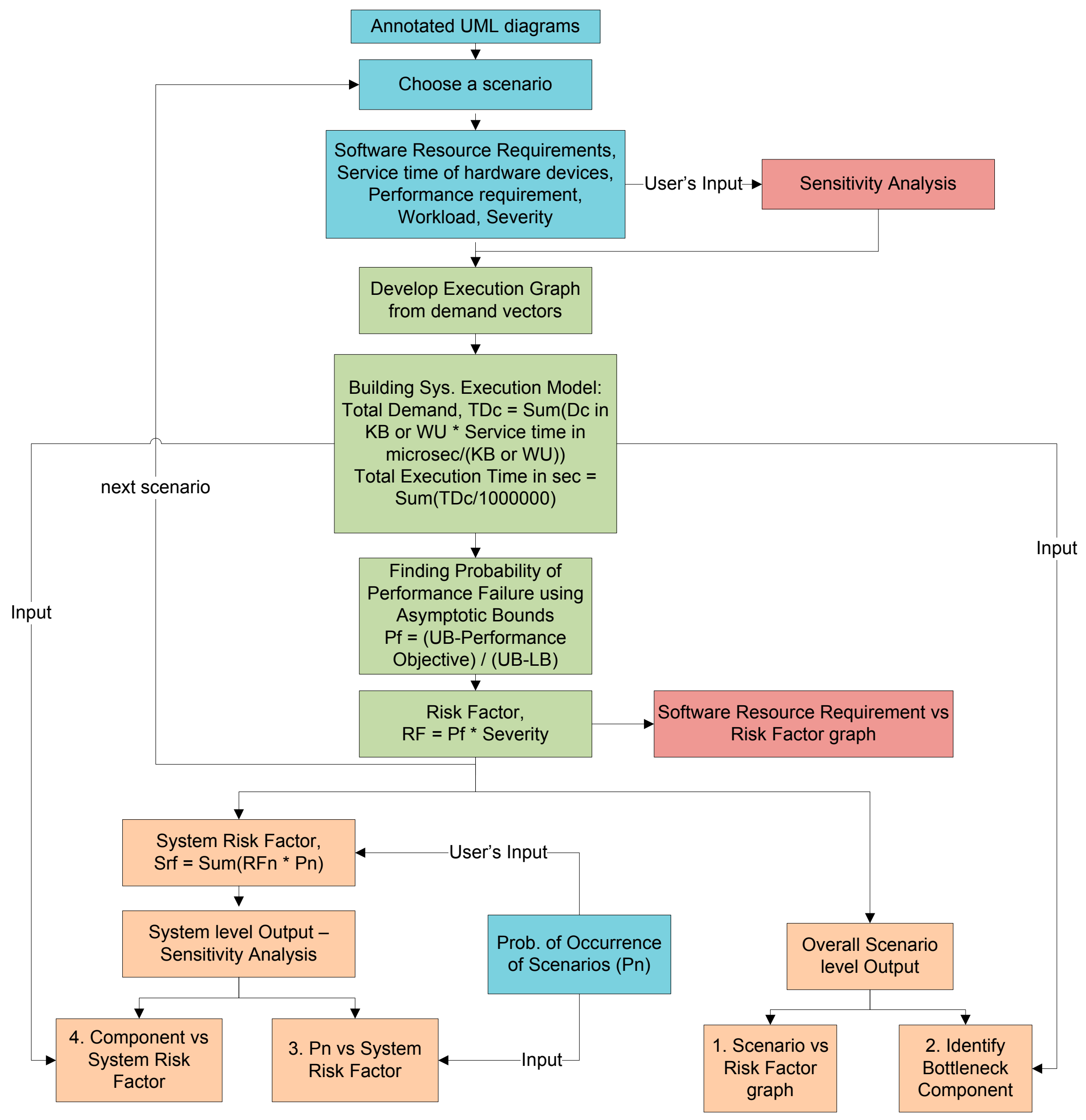

Annotated UML diagrams
Choose a scenario
Software Resource Requirements, Service time of hardware devices, Performance requirement, Workload, Severity
User's Input
Sensitivity Analysis
Develop Execution Graph from demand vectors
Building Sys. Execution Model: Total Demand, TDc = Sum(Dc in KB or WU * Service time in microsec/(KB or WU)) Total Execution Time in sec = Sum(TDc/1000000)
next scenario
Input
Input
Finding Probability of Performance Failure using Asymptotic Bounds Pf = (UB-Performance Objective) / (UB-LB)
Risk Factor, RF = Pf * Severity
Software Resource Requirement vs Risk Factor graph
System Risk Factor, Srf = Sum(RFn * Pn)
User's Input
System level Output – Sensitivity Analysis
Prob. of Occurrence of Scenarios (Pn)
Overall Scenario level Output
4. Component vs System Risk Factor
3. Pn vs System Risk Factor
Input
1. Scenario vs Risk Factor graph
2. Identify Bottleneck Component

Figure 11 Overview of Tool supporting SPRA

The above figure shows the scenario level SPRA and the output of the scenarios are used for the further output analysis. The output of the tool is further categorized as scenario level and system level outputs. The inputs for the different output are also represented in the above figure and are explained in the Chapter 4 with case studies.

The tool is build as one of the features of SARA tool along with other features such as maintainability, reliability, and requirement risk assessment. The figure 12 shows UML class diagram that represents the performance risk assessment and its association with SARA tool as part of other risk factors and Shaik (2007) explains the UML class diagram for SARA tool.

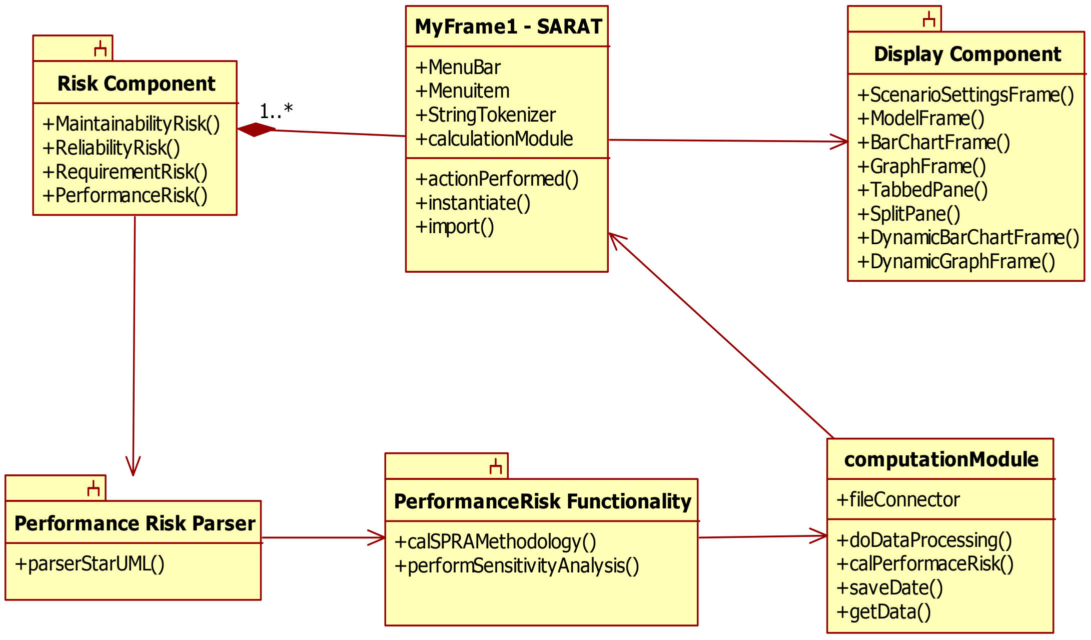
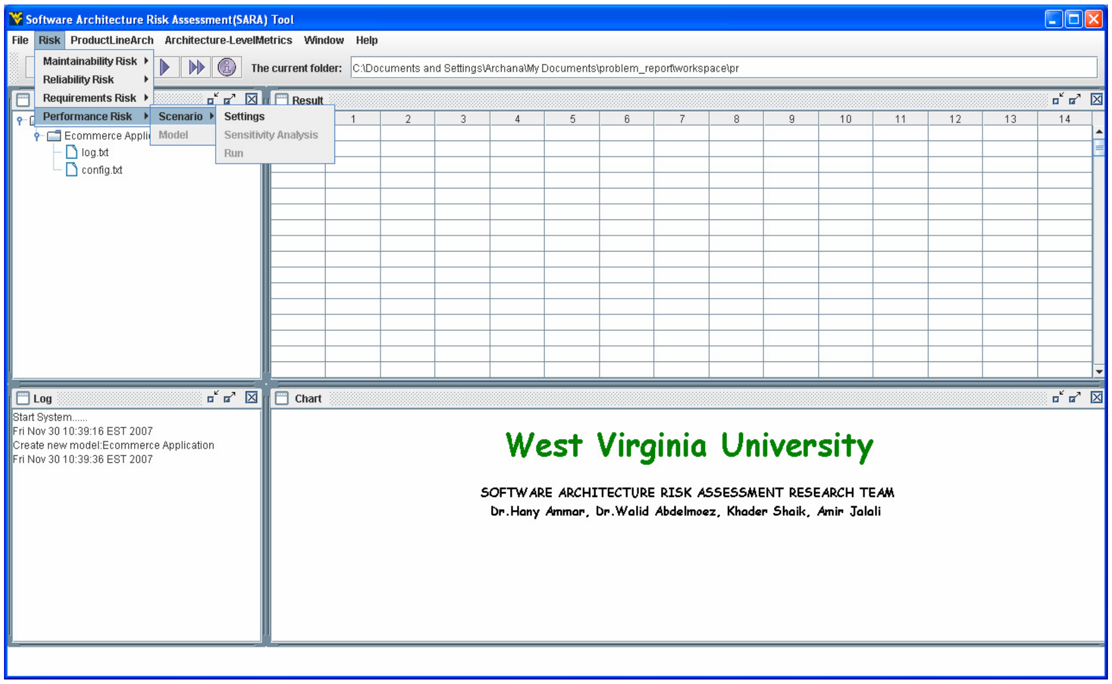


Figure 12 UML Class Diagram of SPRA in SARA tool

Figure 13 shows the tool supporting SPRA as a feature of SARA tool.



The current folder: C:\Documents and Settings\Archana\My Documents\problem_reportworkspace\pr

```
Start System......
Fri Nov 30 10:39:16 EST 2007
Create new model:Ecommerce Application
Fri Nov 30 10:39:36 EST 2007
```

Figure 13 Tool supporting SPRA as part of SARA tool

The **Performance Risk** menu in the SARA tool denotes the tool supporting SPRA. The steps to work with the tool are explained as below:

i) Create a model by using *File->New Model* option

ii) Import the XMI file of the software requirements model developed using StarUML by using *File -> Import Arch.Desc File -> XMI file for StarUML* option as shown in figure 5 below.

iii) Go to *Risk -> Performance Risk -> Scenario -> Settings* for Performance Risk Assessment Settings. There are 2 Settings dialog and click *Next* button to go to next/last dialog and *Back* button to make modifications to the previous dialog.

iv) Provide the inputs and click *OK* button to save the input values for the scenario.

v) Go to *Risk -> Performance Risk -> Scenario -> Sensitivity Analysis* to perform sensitivity analysis of the scenario. This step is optional.

vi) Go to *Risk -> Performance Risk -> Scenario -> Run* for executing SPRA.

vii) Follows the same steps for other scenarios.

viii) Click *Risk -> Performance Risk -> Model* for the overall results about the software model. The tab 'Risk Factor vs Scenario' shows the risk factor vs scenario graph.

ix) Select the next tab 'Component vs Normalized Time' that shows the different scenarios and their bottleneck component. Select the scenario from the left to view its bottleneck component.

x) Select the next tab 'Sys. Risk vs Probability' that shows the overall system risk factor and also the sensitivity analysis graph that represents the changes in the system risk factor based on the probability of occurrences of the scenarios.

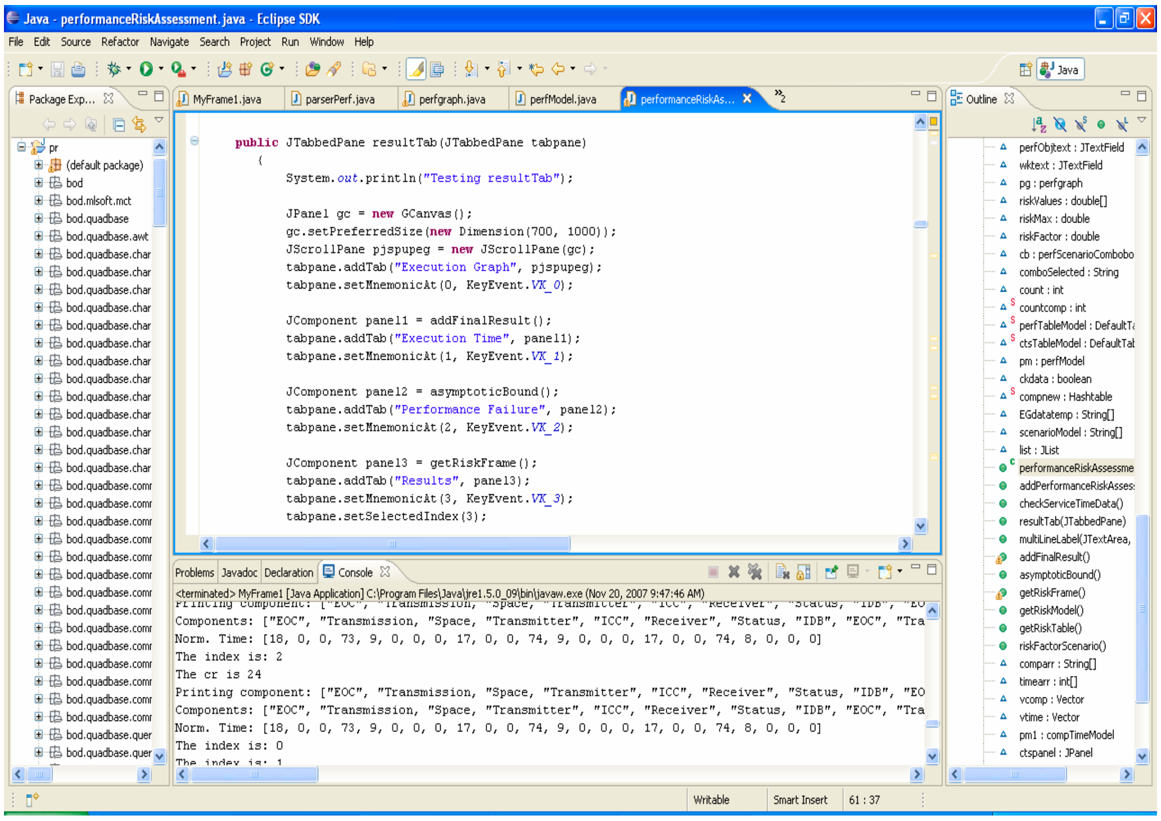
xi) Select the next tab 'Sys. Risk vs Component' that shows the sensitivity analysis graph that represents the changes of System risk factor based on the components of the scenario. Select the scenario to view the sensitivity analysis of system risk factor vs component of that scenario.

3.2.1 Development Environment

The easy-to-use GUI of the tool supporting SPRA provides better user interaction such as obtaining user input, displaying calculated results clearly and precisely, and also good exception handling. The Integrated Development Environment (IDE) used to develop the tool is the open source software, Eclipse 3.2 from IBM Corporation. The advantages of using Eclipse IDE are

- create and edit source code files, compile, and debug java projects seamlessly in a single IDE.
- automatic syntax check, automatic brace generation, parentheses and quote check etc.
- fits development projects in any scale
- open standard – a Worldwide standard of IDE
- IDE with many features such as optional plug-ins offered as freeware or commercial basis.

The figure 14 shows the snapshot of Eclipse IDE.



```
    public JTabbedPane resultTab(JTabbedPane tabpane)
       {
           System.out.println("Testing resultTab");

           JPanel gc = new GCanvas();
           gc.setPreferredSize(new Dimension(700, 1000));
           JScrollPane pjspupeg = new JScrollPane(gc);
           tabpane.addTab("Execution Graph", pjspupeg);
           tabpane.setMnemonicAt(0, KeyEvent.VK_0);

           JComponent panel1 = addFinalResult();
           tabpane.addTab("Execution Time", panel1);
           tabpane.setMnemonicAt(1, KeyEvent.VK_1);

           JComponent panel2 = asymptoticBound();
           tabpane.addTab("Performance Failure", panel2);
           tabpane.setMnemonicAt(2, KeyEvent.VK_2);

           JComponent panel3 = getRiskFrame();
           tabpane.addTab("Results", panel3);
           tabpane.setMnemonicAt(3, KeyEvent.VK_3);
           tabpane.setSelectedIndex(3);
```

```
<terminated> MyFrame1 [Java Application] C:\Program Files\Java\jre1.5.0_09\bin\javaw.exe (Nov 20, 2007 9:47:46 AM)
Printing component: ["EOC", "Transmission, "Space, "Transmitter", "ICC", "Receiver", "Status, "IDB", "EO
Components: ["EOC", "Transmission, "Space, "Transmitter", "ICC", "Receiver", "Status, "IDB", "EOC", "Tra
Norm. Time: [18, 0, 0, 73, 9, 0, 0, 0, 17, 0, 0, 74, 9, 0, 0, 0, 17, 0, 0, 74, 8, 0, 0, 0]
The index is: 2
The cr is 24
Printing component: ["EOC", "Transmission, "Space, "Transmitter", "ICC", "Receiver", "Status, "IDB", "EO
Components: ["EOC", "Transmission, "Space, "Transmitter", "ICC", "Receiver", "Status, "IDB", "EOC", "Tra
Norm. Time: [18, 0, 0, 73, 9, 0, 0, 0, 17, 0, 0, 74, 9, 0, 0, 0, 17, 0, 0, 74, 8, 0, 0, 0]
The index is: 0
The index is: 1
```

Java - performanceRiskAssessment.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Package Exp...
MyFrame1.java
parserPerf.java
perfgraph.java
perfModel.java
performanceRiskAs...
Outline
Problems Javadoc Declaration Console
Writable
Smart Insert
61 : 37

Figure 14 Eclipse IDE Snapshot

The tool is build using Java programming language and J2SDK1.5.2 compiler, which is the latest version of Java compiler introduced by Sun Corp. This makes the tool to make complete use of the highly object oriented java classes and create simple and efficient source code. The look & feel of the tool improves the usability and also performance as they use light weight components such as the Swing classes in javax package. Some of the swing components used are JComboBox, JList, JTabbedPane, JSplitpane etc. Finally, the JVM compiler provides both rapid memory allocation for objects and also performs garbage collection of seldom used code, thereby improving performance. Any platform with Java Virtual Machine (JVM) 1.2 or higher can run the feature in SARA tool.

The java files used for developing the tool supporting SPRA is compiled and integrated with the SARA development kit package. The different java files used for the tool supporting SPRA are:

- MyFrame1: This is the main class executing the different architectural risk factors and is used to call the performanceRiskAssessment class to perform SPRA.
- performanceRiskAssessment: This is the frame that forms the GUI of the tool supporting SPRA. It performs the complete computations of SPRA methodology by obtaining the input, parsing it using parserPerf and perfScenarioCombobox class and displays the results using perfGraph, perfModel, compTimeModel class files.
- parserPerf: This is a file that is used for parsing the information required for the SPRA methodology from the XMI file generated using StarUML and is explained later in this chapter.

- perfGraph: The file is used to display the results of asymptotic bound analysis (for finding the probability of performance failure) in a graphical format. The file uses classes packaged with JFreechart for graphical representation and is explained later in this chapter.
- perfModel: This file is used to display the risk factor vs scenario graph and uses JFreechart package for the graphical representation.
- compTimeModel: This file is used to display the component vs normalized time graph and uses JFreechart package for the graphical representation.
- perfScenarioCombobox: This file is used to get and set the values selected from the combobox containing the scenarios parsed from the XMI file. The get method of the class is used by the performanceRiskAssessment class to quantify the input parameters based on the selected scenario. The set method is used by the MyFrame to store the selected scenario in a variable used for displaying in the GUI.
- perfSensitivity: This file performs the sensitivity analysis of the scenario and is called from the main class, MyFrame. The MyFrame calls the performanceRiskAssessment to perform the computations for sensitivity analysis and calls the perfSensitivity to display the results. The file also performs system level sensitivity analysis explained later in the chapter. The perfSensitivity file uses JFreechart package to display the output in graphical format.

The libraries or jars used for JFreehcart that are packaged with the SARA tool are jfreechart-1.0.6.jar and jcommon-1.0.10.jar. The jar files are available under jfreechart_graph folder in SARA tool.

3.2.2 Integrating JFreeChart Package

The tool supporting SPRA shows the results in a more presentable format such as graphs and 3D bar charts by using JFreechart libraries. JFreechart is an open source java project that helps to display professional quality charts in any Java based application. JFreechart's extensive features are:

- supports wide range of chart types.
- A flexible design that is easy to extend and integrate with any application.
- support for many output types including Swing components

JFreeChart requires the Java 2 platform (JDK version 1.3 or later). It is a class library used by developers and is not an end user product. It has well documented API that helps in utilizing or extending its API to support wide range of chart types. The JFreeChart package is bundled with sample applications that help in developing different 3D bar charts and graphs. Each sample application extends an ApplicationFrame class which opens as a frame. The frame is called in the panel used for displaying results in the tool supporting SPRA. The other different classes used for charts are JFreeChart, ChartPanel and ChartFactory and for dataset are XYSeries, XYDataCollection, and XYDataset.

3.2.3 XMI Parser

The SARA tool is packaged with an XMI parser to obtain input from the XMI file developed using StarUML. The XMI parser understands the different model elements and the UML performance annotations to provide input to the tool supporting SPRA. The following are the model elements that are supported by the XMI parser of the tool.

- UML Use case diagram
- annotated UML Sequence diagram
- annotated UML Deployment diagram

Please note that

- The scenario names mentioned in the use case diagram and the annotations, interactions/steps, and component names in sequence diagram and the hardware devices and list of components mentioned in deployment diagram MUST not have **space** (“ “) in between them. The XMI parser obtains each value required as input to the tool by using StringTokenizer(“ “) function in Java. This parses the string containing information of different model elements by splitting them using space.
- The names of the class/components used in all the diagrams MUST be the same.

Parser Design:

- Take XMI file as input
- Read the input file line by line and store it in an temporary storage area
- Extracting the scenario names to list them in a combobox:
    1. Search for the tag that begins with “<UML:UseCase xmi.id” and store the names, that represent the names of the scenario, in an array to display in the combobox
    2. The starting occurrence and ending occurrence (using “</UML:UseCase”) of the scenario name is stored in another string arrays to display the details of only that scenario selected. Please note that the scenario(s) without any

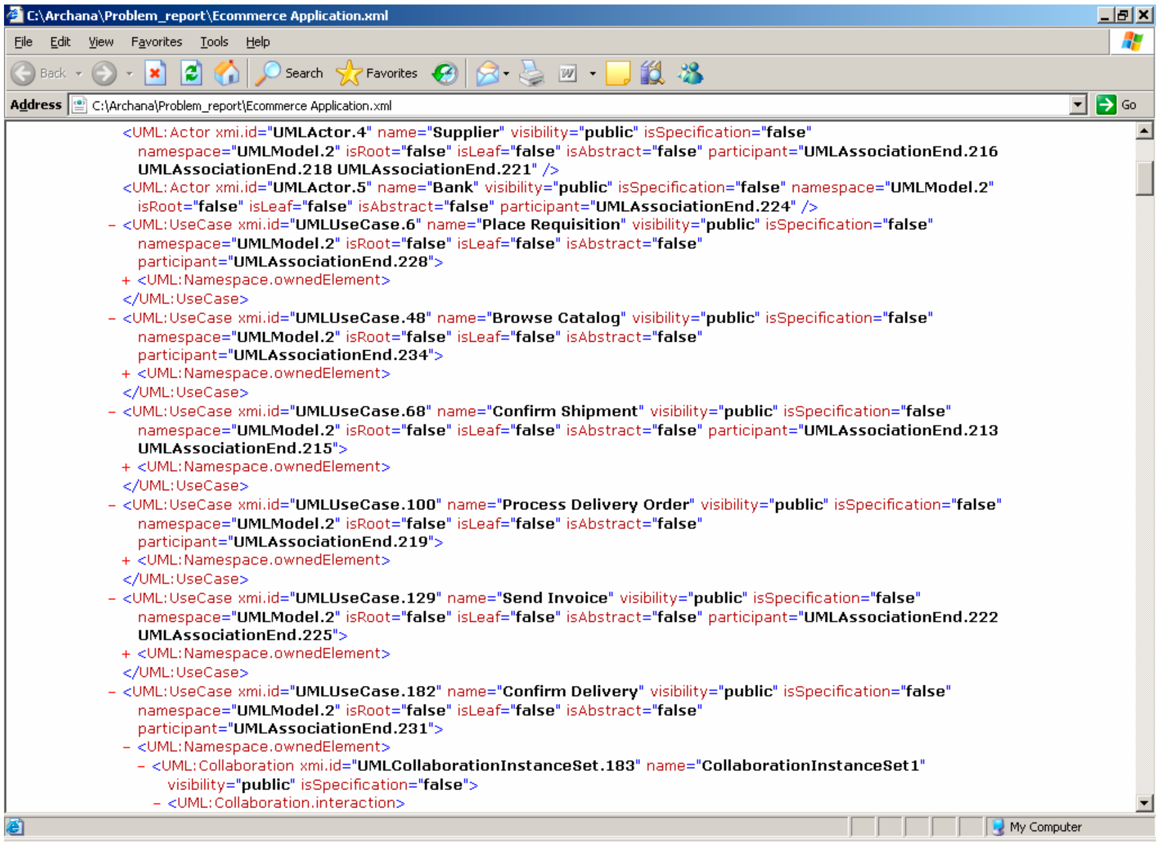
sequence diagram MUST be created finally in the Use Case model.

- Extracting the messages between the objects from sequence diagrams:
    1. Search for the tag which begins with “<UML:Message xmi.id” and store the names of the messages in an array
    2. Store the sender ID and receiver ID in separate arrays. Note: in sequence diagrams, the objects are represented with a unique ID.
- Extracting the component names installed in the hardware devices from deployment diagram
    1. Search for the tag which begins with “<UML:Node xmi.id” and store the names of the component in an array
- Extracting the association information between the nodes from the deployment diagram
    1. Search for the tag which begins with “<UML:AssociationEnd xmi.id” and store them in an array. The array is used to find the components installed in the hardware device.
- Extracting the node names that represent the hardware devices from the deployment diagram
    1. Search for the tag which begins with “<UML:Stereotype xmi.id" and store the names in the ‘extendedElement’ in an array. If the array contains ‘network’ or ‘LAN’ or ‘WAN’, they are stored in a separate array that represents the network devices.
- Extracting the component names from the sequence diagram

1. Search for the tag which begins with “<UML:TaggedValue xmi.id” and store its value in an array. The value with tags such as ‘SPLIT’, ‘FORK’ and ‘JOIN’ are stored in a separate for computation of the longest path with highest demand.

- Extracting the probability of occurrence of the scenario from the use case diagram
    1. Search for the tag which begins with “<UML:Expression xmi.id” and store its value in an array.

Figure 15 shows an XMI file of ecommerce software model that is generated using StarUML tool.

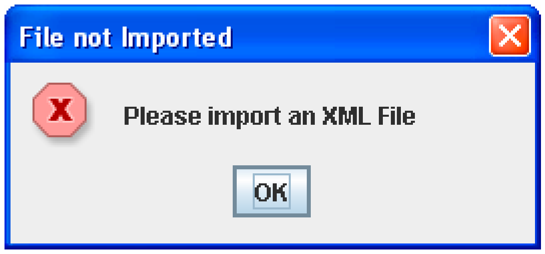

C:\Archana\Problem_report\Ecommerce Application.xml
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
C:\Archana\Problem_report\Ecommerce Application.xml
Go
My Computer

```
        <UML:Actor xmi.id="UMLActor.4" name="Supplier" visibility="public" isSpecification="false"
          namespace="UMLModel.2" isRoot="false" isLeaf="false" isAbstract="false" participant="UMLAssociationEnd.216
          UMLAssociationEnd.218 UMLAssociationEnd.221" />
        <UML:Actor xmi.id="UMLActor.5" name="Bank" visibility="public" isSpecification="false" namespace="UMLModel.2"
          isRoot="false" isLeaf="false" isAbstract="false" participant="UMLAssociationEnd.224" />
      - <UML:UseCase xmi.id="UMLUseCase.6" name="Place Requisition" visibility="public" isSpecification="false"
          namespace="UMLModel.2" isRoot="false" isLeaf="false" isAbstract="false"
          participant="UMLAssociationEnd.228">
        + <UML:Namespace.ownedElement>
        </UML:UseCase>
      - <UML:UseCase xmi.id="UMLUseCase.48" name="Browse Catalog" visibility="public" isSpecification="false"
          namespace="UMLModel.2" isRoot="false" isLeaf="false" isAbstract="false"
          participant="UMLAssociationEnd.234">
        + <UML:Namespace.ownedElement>
        </UML:UseCase>
      - <UML:UseCase xmi.id="UMLUseCase.68" name="Confirm Shipment" visibility="public" isSpecification="false"
          namespace="UMLModel.2" isRoot="false" isLeaf="false" isAbstract="false" participant="UMLAssociationEnd.213
          UMLAssociationEnd.215">
        + <UML:Namespace.ownedElement>
        </UML:UseCase>
      - <UML:UseCase xmi.id="UMLUseCase.100" name="Process Delivery Order" visibility="public" isSpecification="false"
          namespace="UMLModel.2" isRoot="false" isLeaf="false" isAbstract="false"
          participant="UMLAssociationEnd.219">
        + <UML:Namespace.ownedElement>
        </UML:UseCase>
      - <UML:UseCase xmi.id="UMLUseCase.129" name="Send Invoice" visibility="public" isSpecification="false"
          namespace="UMLModel.2" isRoot="false" isLeaf="false" isAbstract="false" participant="UMLAssociationEnd.222
          UMLAssociationEnd.225">
        + <UML:Namespace.ownedElement>
        </UML:UseCase>
      - <UML:UseCase xmi.id="UMLUseCase.182" name="Confirm Delivery" visibility="public" isSpecification="false"
          namespace="UMLModel.2" isRoot="false" isLeaf="false" isAbstract="false"
          participant="UMLAssociationEnd.231">
        - <UML:Namespace.ownedElement>
          - <UML:Collaboration xmi.id="UMLCollaborationInstanceSet.183" name="CollaborationInstanceSet1"
              visibility="public" isSpecification="false">
            - <UML:Collaboration.interaction>
```

Figure 15 XMI file of Ecommerce application

3.2.4 Exception Handling

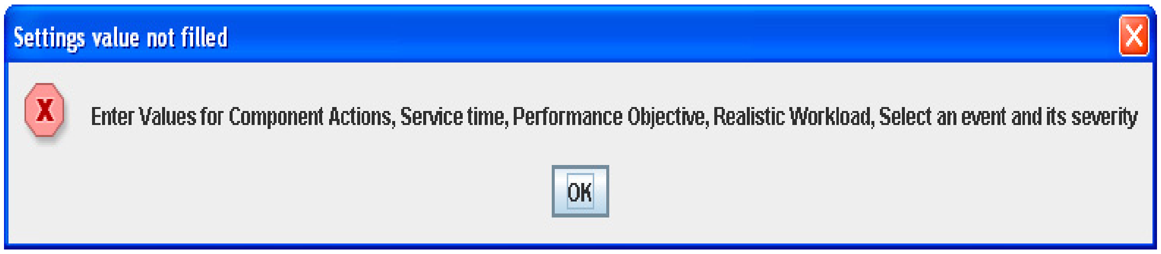
The tool supporting SPRA has a user-friendly interface that helps easy user interaction. It also warns the users using messages in a pop-up window. While running the tool, the user might have problems such as missing some input parameters. The tool should not crop up in the normal course of operations. The tool handles such exceptions by performing exception handling using 'try' and 'catch' in java and throwing the exception using 'JOptionpane' class. The figure 16 shows the pop-up window if the input XMI file is missing


File not Imported

Please import an XML File

Figure 16 Error Message if XMI file missing

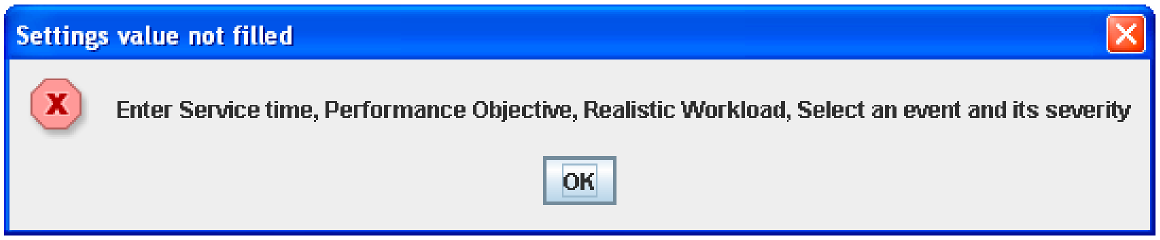
Figure 17 shows the pop-up window if the input values in the 'Settings' dialog of the tool is missing


Settings value not filled

Enter Values for Component Actions, Service time, Performance Objective, Realistic Workload, Select an event and its severity

Figure 17 Error Message if Settings Dialog input parameters are missing

If any of the input parameter of the Settings Dialog is missing, the pop-up window shows that the corresponding messages are missing. Figure 18 shows the pop-up window if the input values of the software requirements alone in 'Settings' dialog is provided and the rest of them are missing.


Settings value not filled

Enter Service time, Performance Objective, Realistic Workload, Select an event and its severity

Figure 18 Error Message if few of Settings Dialog input parameters are missing

3.3 Sensitivity Analysis

The Sensitivity Analysis is a measure to analyze the uncertainty in output of a system model due to small perturbation in its input parameters. With the increased attention to quantitative risk analysis of software systems, there is a need to quantitatively evaluate the behavior of the models used to assess the risk. This is an important component for mathematical, computational, and simulation models as the model output is determined by uncertainty of input variables, computations, and other parameter values. Sensitivity analysis helps in identifying the input parameter that has the greatest impact on the output and aid in developing priorities for risk mitigation. The analysis can play an important role in verification and validation of a model, as well as to provide insight into the robustness of model results, when making decisions. The results can be used for various purposes, e.g., ranking the inputs in order of their impact on the output, assessing changes in the output due to parameter and input variations, improving the quality of the computations, or limiting the use of a program to appropriate parts for the input domain.

The sensitivity analysis can either be performed using mathematical or computational methods. The mathematical sensitivity analysis of a model is performed by calculating the variability of output (dependent variable) with respect to perturbations in the input (independent variable). It becomes difficult if an automatic computation of the mathematical equations involved in finding the output is not available because the input variables can be related indirectly or in a complicated fashion with the parameter of the model. The limitation of mathematical sensitivity analysis is that it becomes more time consuming and complex as the model can evolve over time with more modifications. To overcome the limitation, the computational sensitivity analysis is used that assess the changes in output for any small change in input by automatically performing the calculations.

Liebrock (2005) explains that the behavior of uncertainty of system model could be any of the following:

For a small change in input parameter,

i) The output is a relatively small change which is acceptable or

ii) The output is a drastic change which in turn can lead to two results:

  - This change is acceptable or
  - This change is not acceptable

The above theory helps in analyzing the behavior of uncertainty due to performance risk factor in software system. The sensitivity analysis of software performance risk assessment means that

For a small or drastic change in input parameter, the output is

i) *acceptable*, if the value of performance risk factor is between 0 and 1.

ii) *not acceptable*, if the value of performance risk factor is greater than 1.

The tool supporting SPRA also supports sensitivity analysis of the software model assessed using SPRA methodology. It performs computational sensitivity analysis of performance risk factor that automatically performs computations used for SPRA methodology in estimating risk factor. This helps in reducing the pain to perform the procedure of SPRA methodology repeatedly for every small change in the input parameters. The objective of performing the sensitivity analysis in the tool is to analyze the variation of performance risk factor of a scenario in a software model based on small changes in its input parameters. The different ways or cases to perform computational sensitivity analysis to obtain risk factor for a scenario are:

1. Base Case: In this case, run the code with the original inputs and record the outputs.
2. Comparison case: Vary selected input parameters one at a time, rerun the code with the changes, and record the corresponding changes in the outputs calculated by the code.
3. Compute Sensitivity: Represent the sensitivity for the parameter under test in a graphical form as risk factor vs the input parameters.

Case 2 and 3 are used for sensitivity analysis of SPRA. In SPRA, two types of sensitivity analysis are performed. The following section explains the scenario level and system level sensitivity analysis.

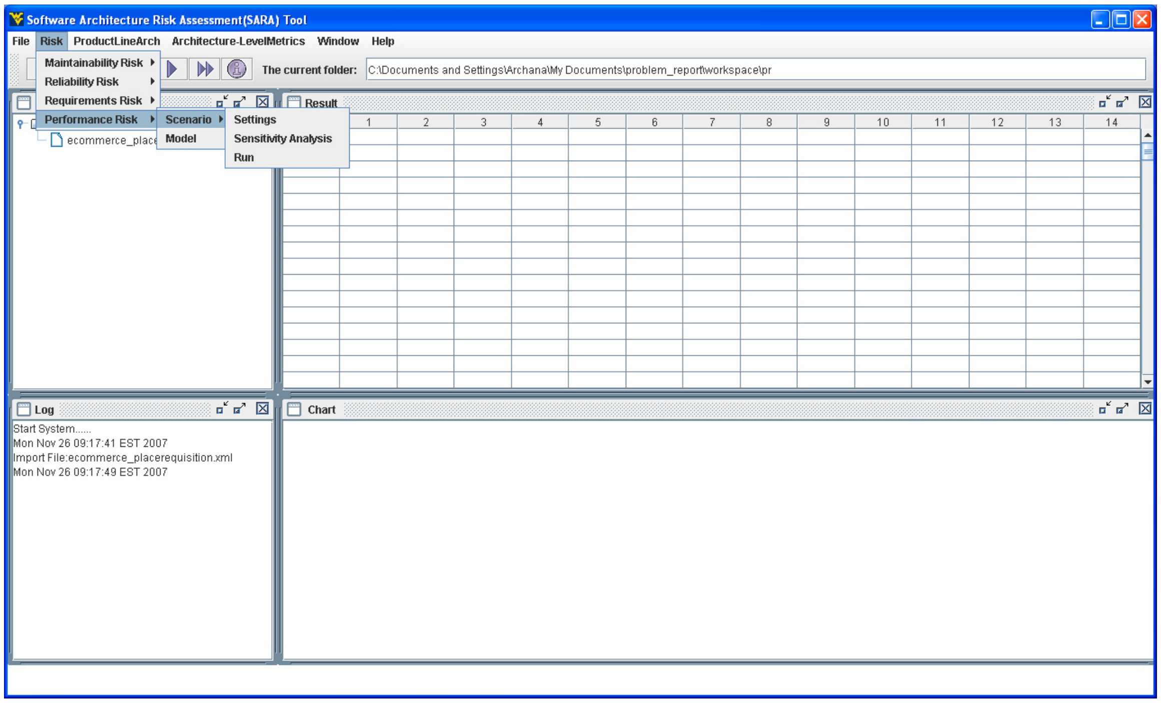
3.3.1 Scenario level Sensitivity Analysis

The SPRA methodology is a scenario-level simulation method that performs computations for performance risk assessment. To perform the scenario level sensitivity analysis, the most critical input parameter of the scenario that has greater impact on its risk factor should be identified. The different input parameters to the SPRA methodology are:

- Software resource requirements
- Service time or speed of hardware devices
- Performance Objective
- Realistic Workload and
- FFA table for severity analysis

The service time or speed of hardware devices are the characteristics of the hardware platform used by the software model and this cannot be changed as the functionality of the software model will be affected. The performance objective is the inability of the system to meet its performance requirements and this is already defined by the developer for a software model and hence cannot be changed. The impact due to different workload conditions is already implemented in the SPRA methodology using asymptotic bound analysis. Also, the severity analysis is performed using FFA in the methodology and hence cannot be changed. The input parameter that can be changed to assess the variation of performance risk factor is by changing each input to the software resources and is considered for performance sensitivity analysis of the tool supporting SPRA methodology. The main goal of the analysis is to understand which aspect of the input is limiting the performance or causing performance risk of the scenario.

Once the XMI input is provided as input to the SARA tool, the SPRA is performed by using *Risk->Performance Risk->Scenario->Settings*. The input parameter is provided in the Settings Dialog and *Scenario->Sensitivity Analysis* is selected to perform the sensitivity analysis. Performing sensitivity analysis is optional and is added as a feature for the user to get an insight of the impact of his input parameters on the performance risk assessment. The figure 19 shows the Sensitivity Analysis as a feature of performance risk in SARA tool.



The current folder: C:\Documents and Settings\Archana\My Documents\problem_report\workspace\pr

```
Start System......
Mon Nov 26 09:17:41 EST 2007
Import File:ecommerce_placerequisition.xml
Mon Nov 26 09:17:49 EST 2007
```

Figure 19 Sensitivity Analysis as menuitem of Performance Risk

Once the Sensitivity Analysis is selected, the software requirements input parameter is displayed along with the range. The user has to choose the software requirement for which he has to perform the sensitivity analysis testing, select the range, and click the 'Perform Sensitivity Analysis' button to view the change of performance risk factor based on the change of the software requirements between the given range. The risk factor vs software resources presented in a graphical format helps the user to assess the values provided as input to the software model.

3.3.2 System level Sensitivity Analysis

The system risk factors are estimated using the scenario risk factors and is given as follows in Popstojanova et. al., (2003):

$$rf = \sum_{\forall U_k} rf_k \cdot p_k,$$

Where rf is the overall system risk factor, rfk is the risk factor of the use cases and pk is the probability of occurrences of use case. The risk factor of use case and scenario is identical as one scenario per use case is considered.

From the above expression, it is obvious that the overall system risk factor is dependent on probability of occurrence of the scenarios (use case). So the system level sensitivity analysis can be performed by varying the probability of occurrences of each scenario. This provides a closer view of the changes in system risk factor based on probability of occurrences of the scenarios in the system. Also, the system level sensitivity analysis can be performed by observing the changes of system risk factor based on the components of each scenario. The system level sensitivity analysis helps in experimenting the system risk factor with input parameters that are more critical and can cause unpredictable changes in the system. The tool facilitates in performing sensitivity analysis not only in scenario level but also in system level which requires more computations as the number of scenarios in a system increases, repetition of mathematical calculation, and time-consuming manual analysis. Once the complete scenario level SPRA is performed using the SARA tool, select Performance Risk->Model and view ‘Sys. Risk vs Probability’ tab and ‘Sys. Risk vs Component’ tab to view system level sensitivity analysis.

CHAPTER 4

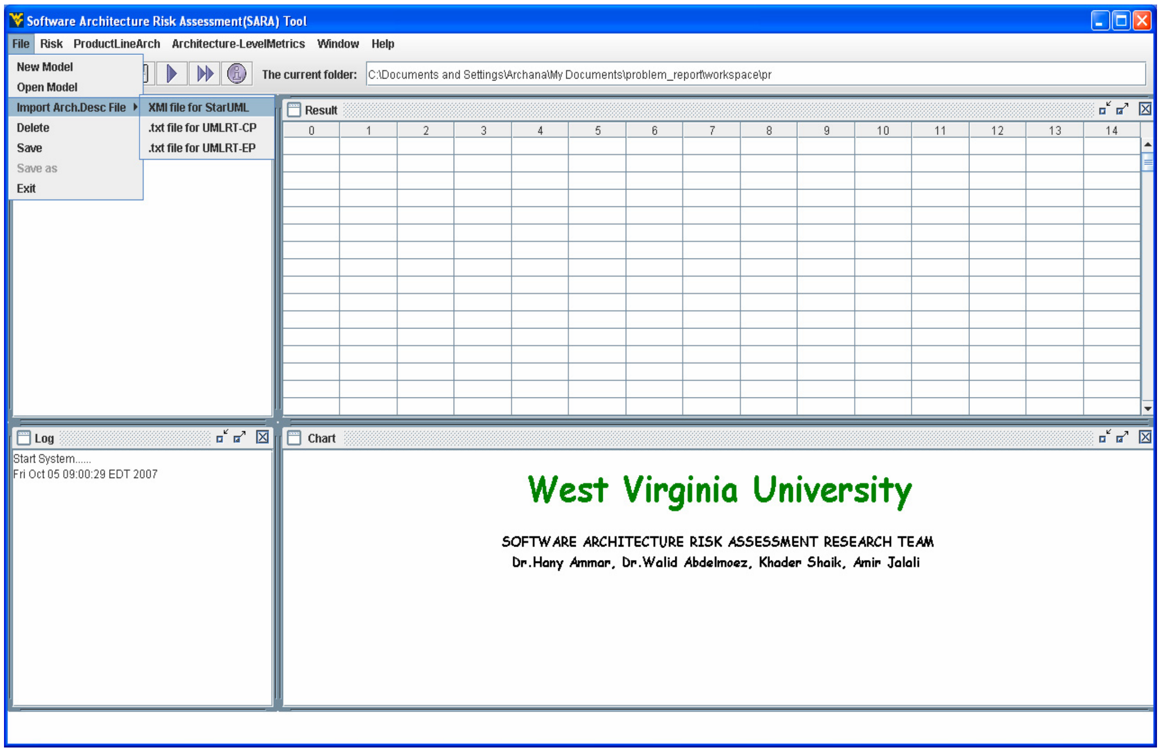
TESTING AND RESULTS

The tool supporting SPRA is integrated with SARA tool and functionally tested on multiple case studies. This section explains the tool on ecommerce application and Earth Observation System (EOS).

4.1 Ecommerce application

An ecommerce application allows a customer to interact with supplier through internet system. The customer browses through various catalogs provided in the internet to select an item and place a requisition to the supplier. The supplier then processes the order and proceeds with shipment and delivery information by validating the customer's account with the supplier and with one or more banks through which the payment can be made. Then the supplier checks the availability of the product and if it is available, ships the product to the customer. The customer confirms the delivery and the supplier sends an invoice statement to the bank. The bank transfers the funds from the customer's account to the supplier's account. The Use case model and Deployment model of the ecommerce application is shown in figure 1 and 3 respectively.

Figure 20 shows the SARA tool to which the XMI file with the defined UML performance specifications as explained in Section 3.1 is imported.



The current folder: C:\Documents and Settings\Archana\My Documents\problem_report\workspace\pr

```
Start System......
Fri Oct 05 09:00:29 EDT 2007
```

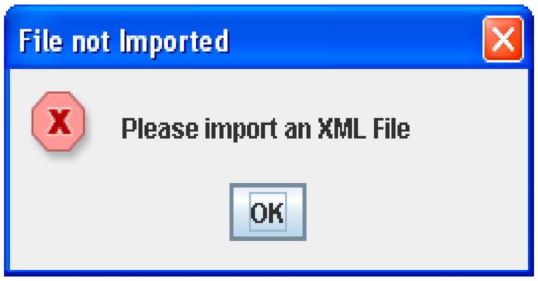
West Virginia University

SOFTWARE ARCHITECTURE RISK ASSESSMENT RESEARCH TEAM

Dr.Hany Ammar, Dr.Walid Abdelmoez, Khader Shaik, Amir Jalali

Figure 20 Importing XMI file for Performance Risk Assessment

If the XMI file is not imported, the SARA tool reports an error notification as shown in figure 21.

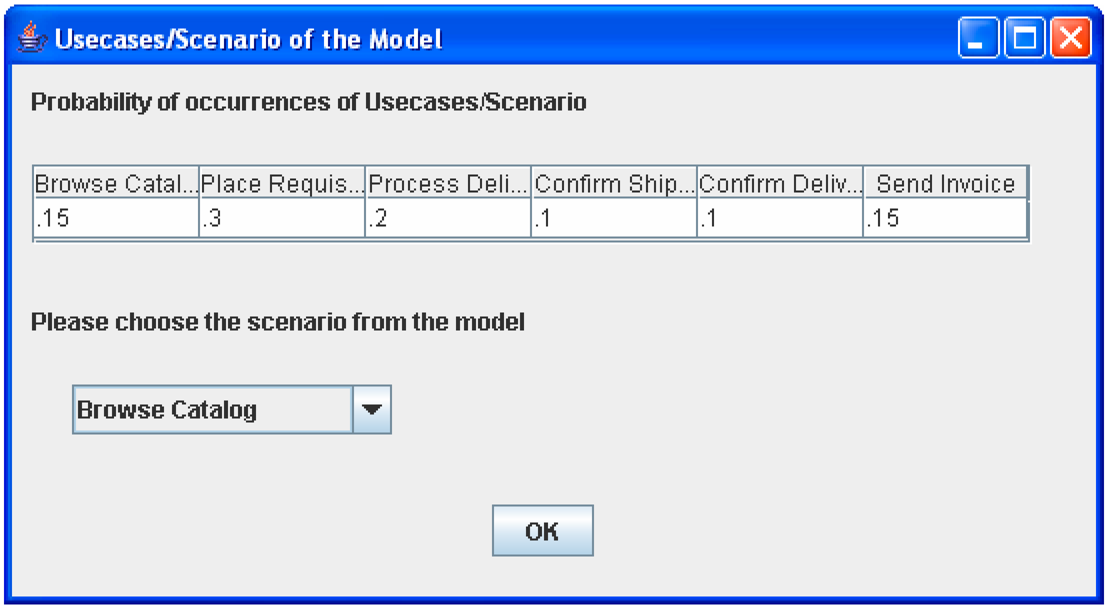


Please import an XML File

Figure 21 Error Notification for not importing XMI file

Once the XMI file is imported, the Performance Risk Assessment is performed using *Risk -> Performance Risk ->Scenario -> Settings*. The 'Sensitivity Analysis' and 'Run' menitem will be enabled once the Settings dialog is selected. The 'Model' menuitem will be enabled after performing performance risk assessment for atleast one scenario. Figure

22 shows the window that displays the values of probability of occurrences of scenario if annotated using the use case diagram or else the user can input the values. It also lists the scenarios of the software model.



Probability of occurrences of Usecases/Scenario

Browse Catal...	Place Requis...	Process Deli...	Confirm Ship...	Confirm Deliv...	Send Invoice
.15	.3	.2	.1	.1	.15

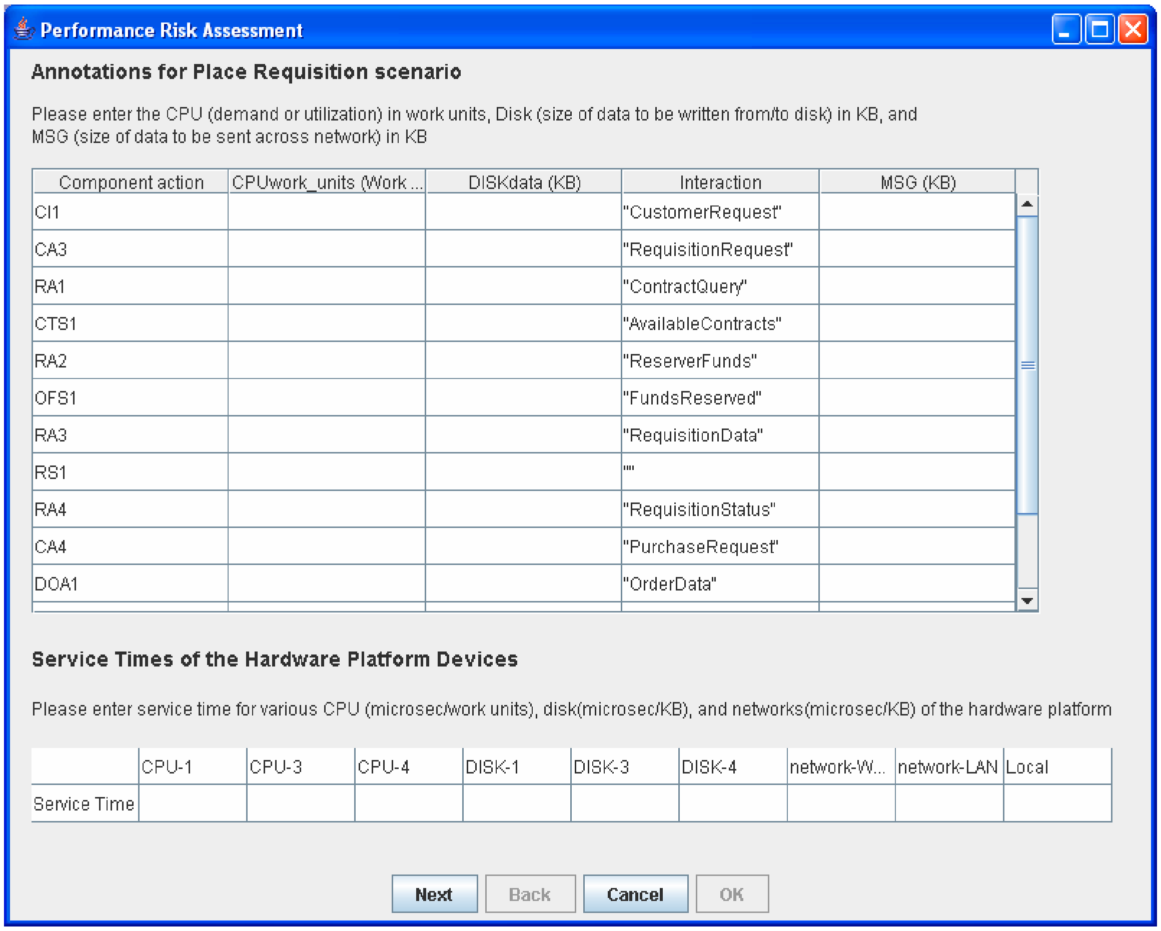
Please choose the scenario from the model

Browse Catalog

Figure 22 Listing Scenarios of the model

The scenario for which the performance risk assessment is to be performed is selected from the combo box. The GUI of SPRA consists of 2 settings dialog or the input to be obtained from the user for performance risk assessment. The Settings dialog lists the software components and the interaction between them in a scenario. The software components are the demand vectors annotated in the sequence diagram of the scenario. The amount of resources such as CPU_{work_units} and $Disk_{data}$ for software components and MSG_{data} for interactions between components are required as input. It also displays the hardware devices in which the software components are installed and the type of network connection. The hardware devices and the type of network connection are derived from the annotations of the deployment model. The service time for each hardware device and its network that is the speed of processor, disk, and network in terms of time units (µsec) is required as input.

Figure 23 shows the settings dialog of *Place Requisition* scenario where the components and their interaction are listed in a table and the hardware devices in which the components are installed in listed in another table. The amount of resources required for the components and network and the service time of hardware devices for the scenario is provided as input as shown in figure 24.



Annotations for Place Requisition scenario

Please enter the CPU (demand or utilization) in work units, Disk (size of data to be written from/to disk) in KB, and MSG (size of data to be sent across network) in KB

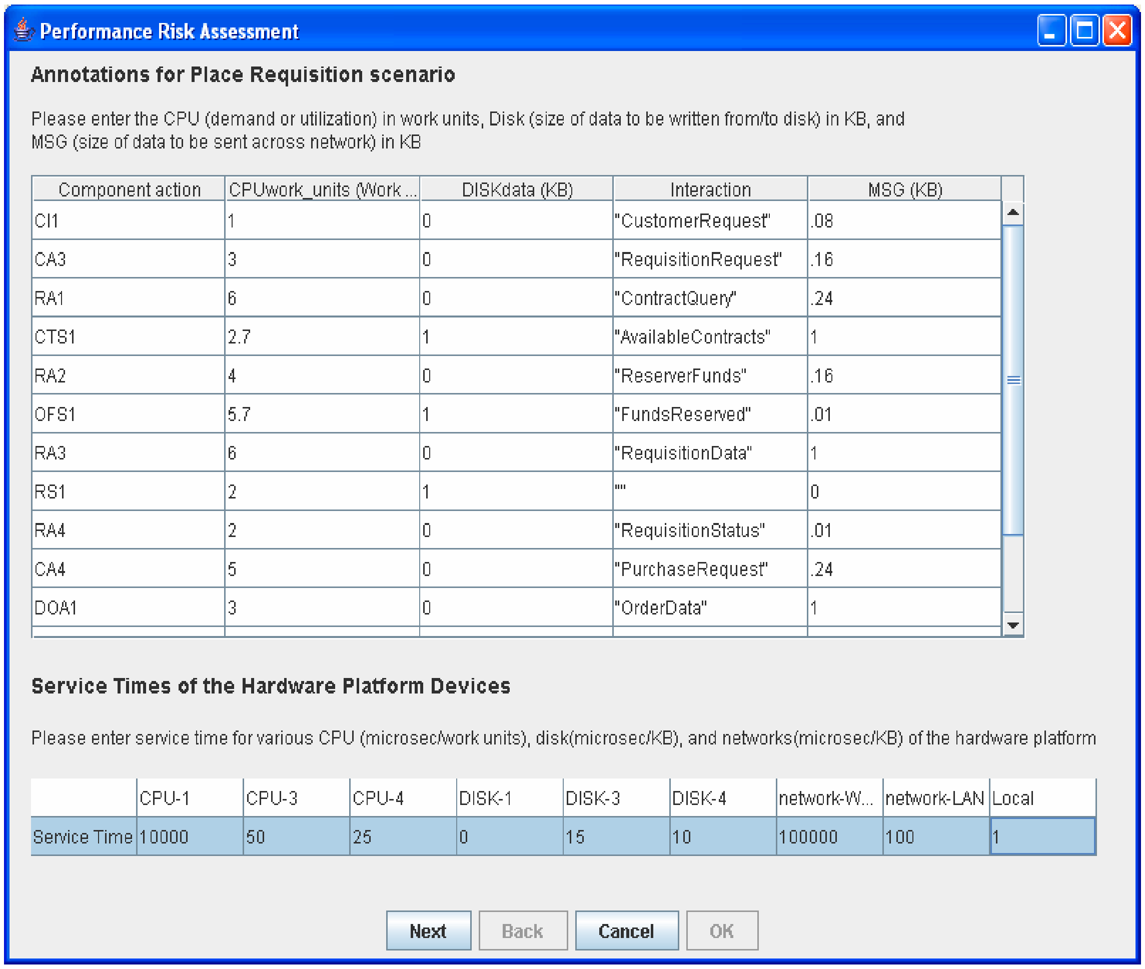
Component action	CPUwork_units (Work ...	DISKdata (KB)	Interaction	MSG (KB)
CI1			"CustomerRequest"	
CA3			"RequisitionRequest"	
RA1			"ContractQuery"	
CTS1			"AvailableContracts"	
RA2			"ReserverFunds"	
OFS1			"FundsReserved"	
RA3			"RequisitionData"	
RS1			""	
RA4			"RequisitionStatus"	
CA4			"PurchaseRequest"	
DOA1			"OrderData"	

Service Times of the Hardware Platform Devices

Please enter service time for various CPU (microsec/work units), disk(microsec/KB), and networks(microsec/KB) of the hardware platform

	CPU-1	CPU-3	CPU-4	DISK-1	DISK-3	DISK-4	network-W...	network-LAN	Local
Service Time									

Figure 23 Settings Dialog for Performance Risk Assessment



Annotations for Place Requisition scenario

Please enter the CPU (demand or utilization) in work units, Disk (size of data to be written from/to disk) in KB, and MSG (size of data to be sent across network) in KB

Component action	CPUwork_units (Work ...	DISKdata (KB)	Interaction	MSG (KB)
CI1	1	0	"CustomerRequest"	.08
CA3	3	0	"RequisitionRequest"	.16
RA1	6	0	"ContractQuery"	.24
CTS1	2.7	1	"AvailableContracts"	1
RA2	4	0	"ReserverFunds"	.16
OFS1	5.7	1	"FundsReserved"	.01
RA3	6	0	"RequisitionData"	1
RS1	2	1	""	0
RA4	2	0	"RequisitionStatus"	.01
CA4	5	0	"PurchaseRequest"	.24
DOA1	3	0	"OrderData"	1

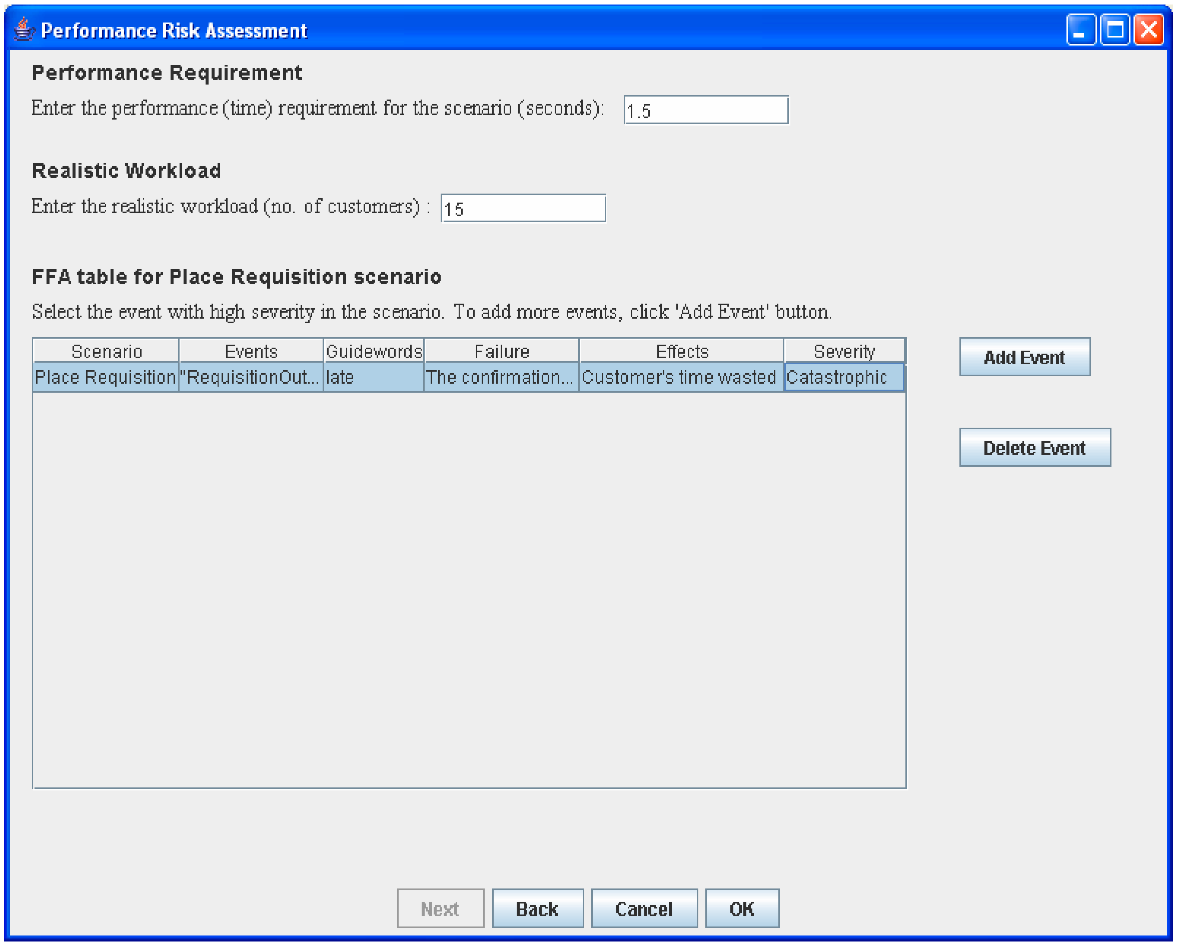
Service Times of the Hardware Platform Devices

Please enter service time for various CPU (microsec/work units), disk(microsec/KB), and networks(microsec/KB) of the hardware platform

	CPU-1	CPU-3	CPU-4	DISK-1	DISK-3	DISK-4	network-W...	network-LAN	Local
Service Time	10000	50	25	0	15	10	100000	100	1

Figure 24 Settings Dialog with Inputs

Once the required input is provided, the *Next* button will display the next Settings dialog as shown in figure 25 that requires information such as performance objective, realistic workload, and the FFA table for severity analysis. The FFA table displays the scenario name and requires the input such as selecting the events with high severity, stating its reason to fail and the effect of failure and selecting its severity. Any number of events can be added using the *Add Event* button and can be deleted using the *Delete Event* button. The event with the maximum severity rate will be taken as the severity for the scenario.



Performance Requirement

Enter the performance (time) requirement for the scenario (seconds): 1.5

Realistic Workload

Enter the realistic workload (no. of customers) : 15

FFA table for Place Requisition scenario

Select the event with high severity in the scenario. To add more events, click 'Add Event' button.

Scenario	Events	Guidewords	Failure	Effects	Severity
Place Requisition	"RequisitionOut...	late	The confirmation...	Customer's time wasted	Catastrophic

Figure 25 Additional Settings Dialog for Performance Risk Assessment

The *Back* button is used to go to the previous settings for any modification in the input. In brief, the tool requires the following input:

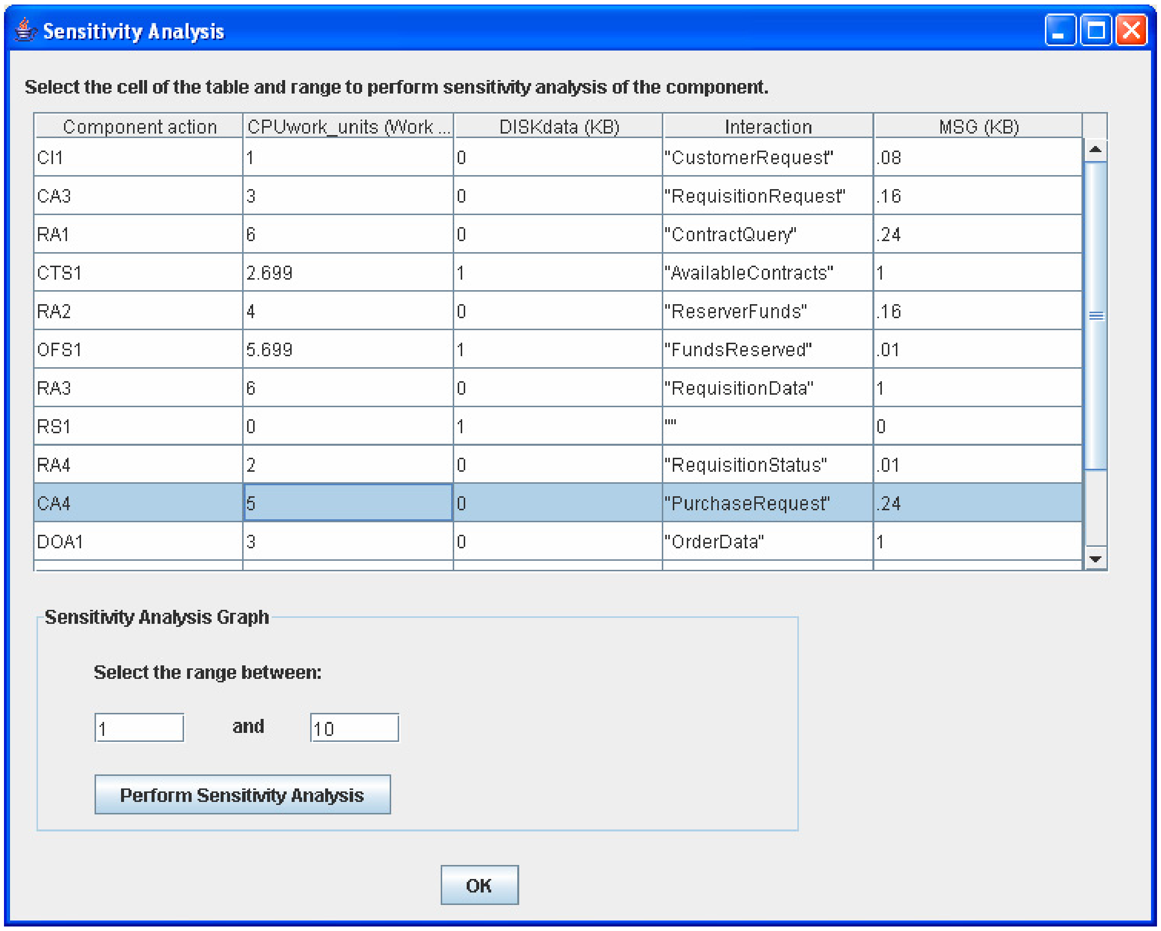
- XMI file containing details of the software model
- amount of resources required for software components and its interaction
- service time of hardware devices and network
- performance objective
- realistic workload
- Severity using FFA

The *OK* button saves the input values of the scenarios.

Then, the user can perform sensitivity analysis to assess the impact of the input values to performance risk factor. Once sensitivity analysis is selected, it displays the software resources with the software requirements provided as input by the user in the *Settings* dialog. It also displays the range of software requirements. The user can select the range to experiment or analyze the risk factor of the scenario by varying the software requirement within that range. The Sensitivity Analysis feature is applied on this scenario. Kamavaram et.al. (2003) explains the sensitivity analysis of ecommerce application to perform software reliability.

Figure 26 shows the Sensitivity Analysis dialog of Place Requisition Scenario that performs software performance. The following are the steps to perform sensitivity analysis of software performance risk:

- Choose a cell or value of a component whose software requirement is to be experimented for sensitivity analysis. Please note that the sensitivity analysis is performed for testing the change of a software requirement of a component with the scenario risk factor. Hence, multiple selections of software requirements are not supported.
- Provide the input range. This is the range of values for the software requirements to be experimented and is the values of x-axis.
- Click ‘Perform Sensitivity Analysis’ button.


Sensitivity Analysis

Select the cell of the table and range to perform sensitivity analysis of the component.

Component action	CPUwork_units (Work ...	DISKdata (KB)	Interaction	MSG (KB)
CI1	1	0	"CustomerRequest"	.08
CA3	3	0	"RequisitionRequest"	.16
RA1	6	0	"ContractQuery"	.24
CTS1	2.699	1	"AvailableContracts"	1
RA2	4	0	"ReserverFunds"	.16
OFS1	5.699	1	"FundsReserved"	.01
RA3	6	0	"RequisitionData"	1
RS1	0	1	""	0
RA4	2	0	"RequisitionStatus"	.01
CA4	5	0	"PurchaseRequest"	.24
DOA1	3	0	"OrderData"	1

Sensitivity Analysis Graph

Select the range between:

1 and 10

Figure 26 Sensitivity Analysis Dialog

The figure 27 shows the graphical representation of risk factor vs the range of software resources where the value of CPUworkunits of CA4 component is changed within a range of 1 to 10. The figure 25 shows that there is a linear increase of scenario risk factor with the increase of CPU of CA4 component.

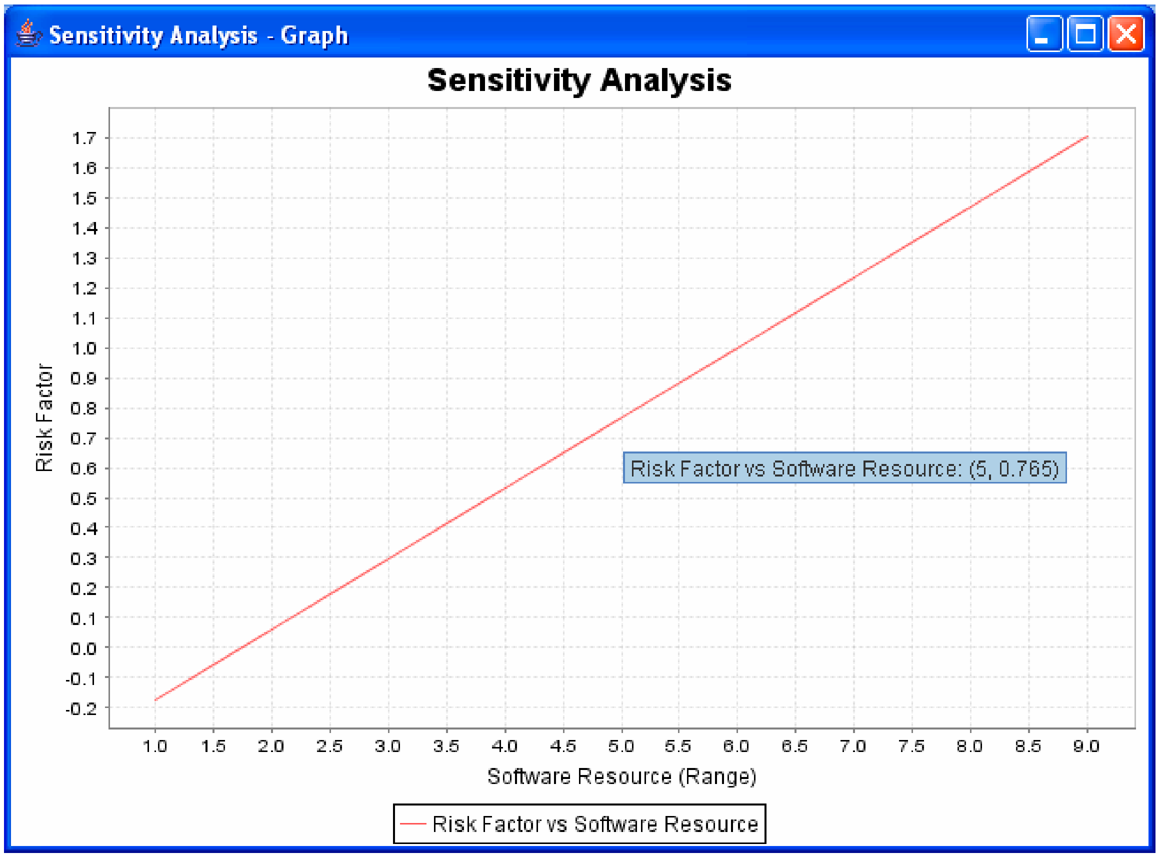
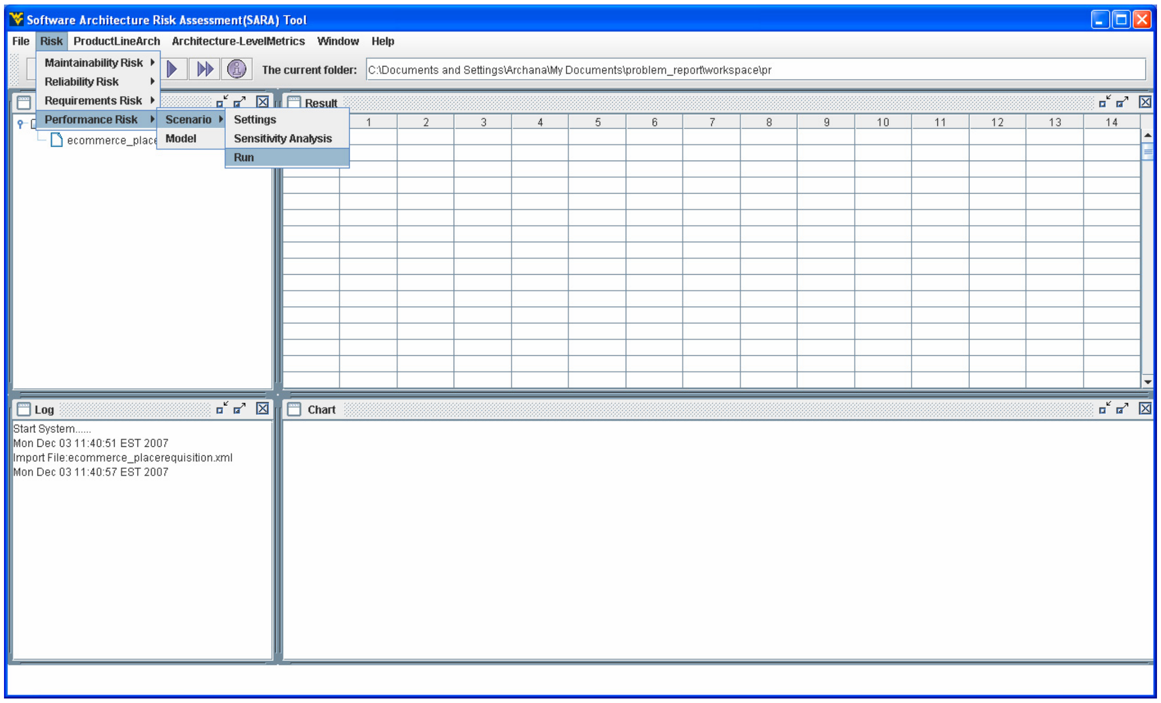


Figure 27 Risk Factor vs Software Resource

From the graph, it is seen that the risk factor for 5 workunits of CPU for CA4 component is 0.765 which is acceptable as it lies between 0 and 1. The risk factor is greater than 1 if the CPUworkunit is greater than 6 and this leads the Place Requisition scenario to have performance risk. Thus, the sensitivity analysis helps the user to verify and validate his input parameters to avoid performance risk. The tool supporting SPRA performs the computations required to calculate the risk factor automatically by getting the software requirement value and its range as input. This facilitates the user to run SPRA methodology as many times to estimate the risk factor of the software model.

Then *Scenario - > Run* button is selected that performs the performance risk assessment as shown in figure 28.

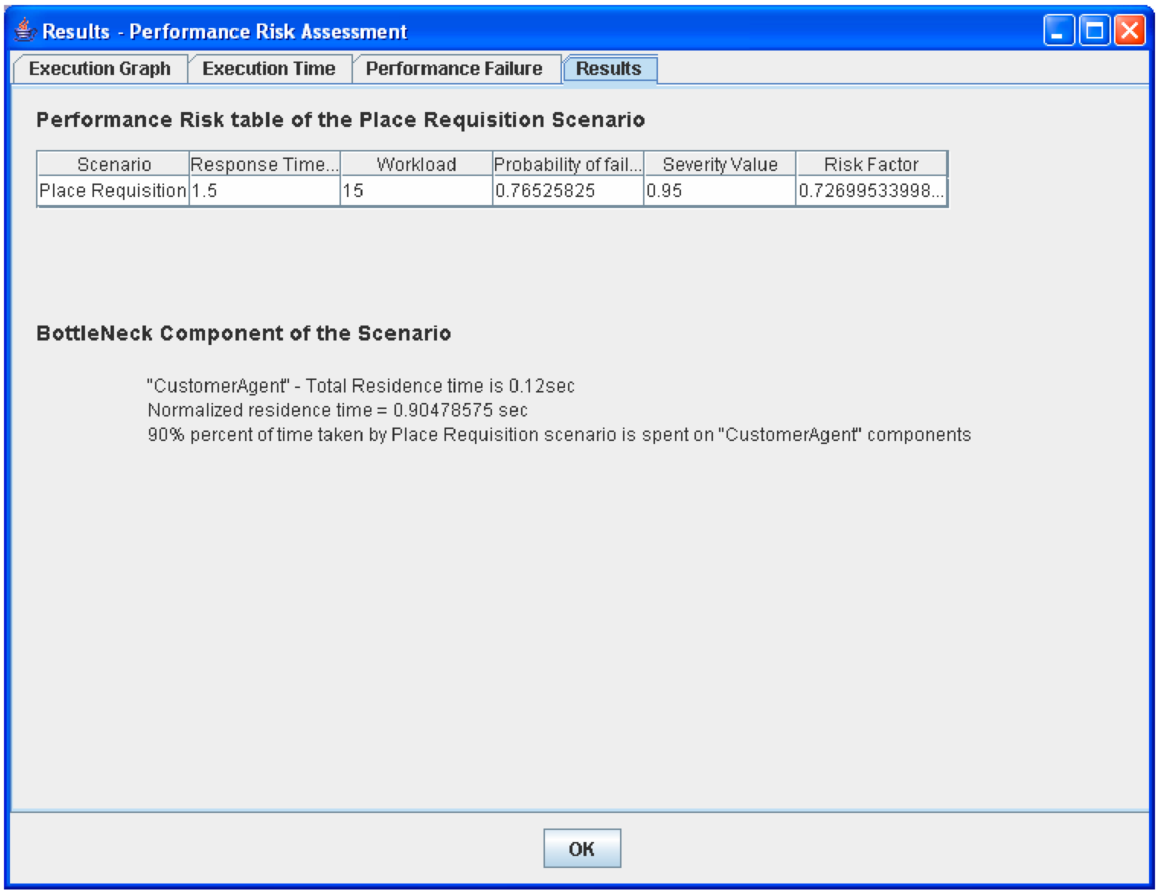


The current folder: C:\Documents and Settings\Archana\My Documents\problem_report\workspace\pr

```
Start System......
Mon Dec 03 11:40:51 EST 2007
Import File:ecommerce_placerequisition.xml
Mon Dec 03 11:40:57 EST 2007
```

Figure 28 Executing Scenario to perform SPRA using SARA tool

Figure 29 shows a tabbed window with a ‘Results’ tab displaying the results: Risk factor of the scenario and the bottleneck component causing the performance risk. The Risk factor is obtained by multiplying probability of performance failure and the severity. The probability of performance failure is obtained from the asymptotic bounds which are represented as a graph in figure 32 and the severity is obtained from the FFA table provided in the Settings Dialog. The categories of severity is assigned to a numerical value as 0.95 to catastrophic, 0.75 to critical, .5 to major, and .25 to minor.



Performance Risk table of the Place Requisition Scenario

Scenario	Response Time...	Workload	Probability of fail...	Severity Value	Risk Factor
Place Requisition	1.5	15	0.76525825	0.95	0.72699533998...

BottleNeck Component of the Scenario

"CustomerAgent" - Total Residence time is 0.12sec
Normalized residence time = 0.90478575 sec
90% percent of time taken by Place Requisition scenario is spent on "CustomerAgent" components

Figure 29 Results Tab displaying the SPRA

The other tab displays the output obtained in the step-by-step performance risk assessment methodology. The *Execution Graph* tab displays the execution graph or the software execution model of the scenario. It is a flow graph that represents the sequence of demand vectors and the longest path (worst case analysis) with the highest demand. The longest path considered for the input to system execution model is marked in red color and the demand vectors that are not considered are marked in blue color. If the scenario does not have concurrent actions, the longest path is the sequence of its demand vectors. The figure 30 shows the execution graph of *Place Requisition* scenario where there are two concurrent paths and the path with the highest demand is represented in red.

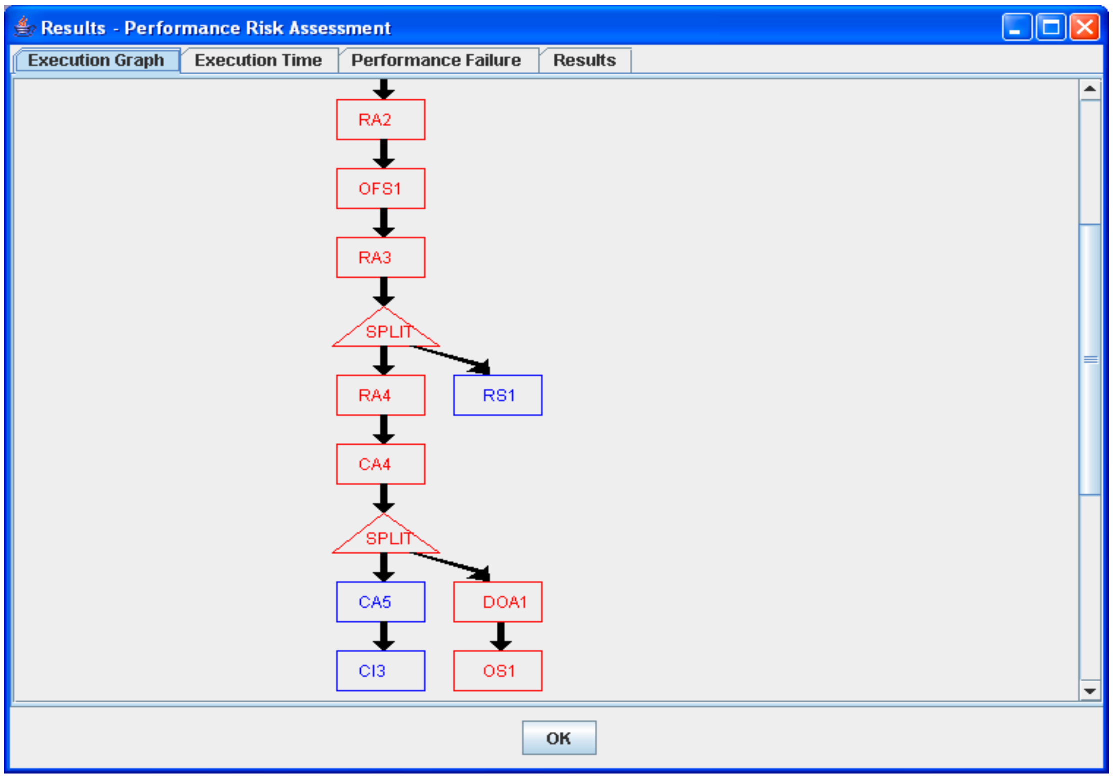
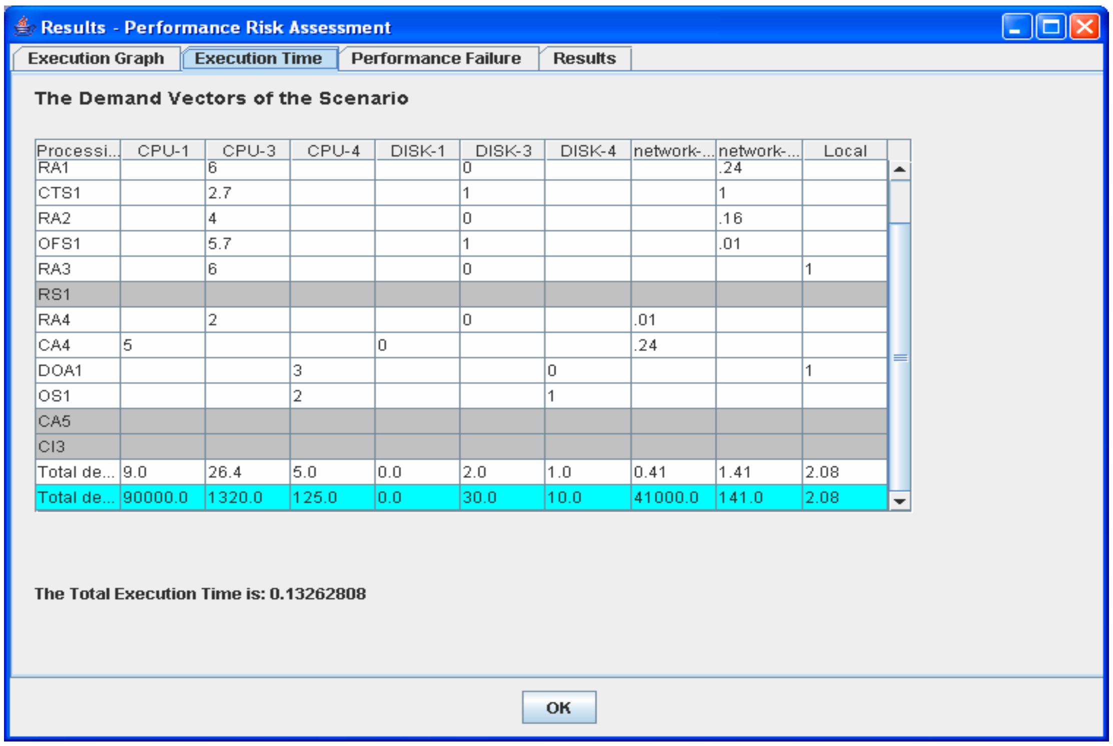


Figure 30 Execution Graph Tab



The Demand Vectors of the Scenario

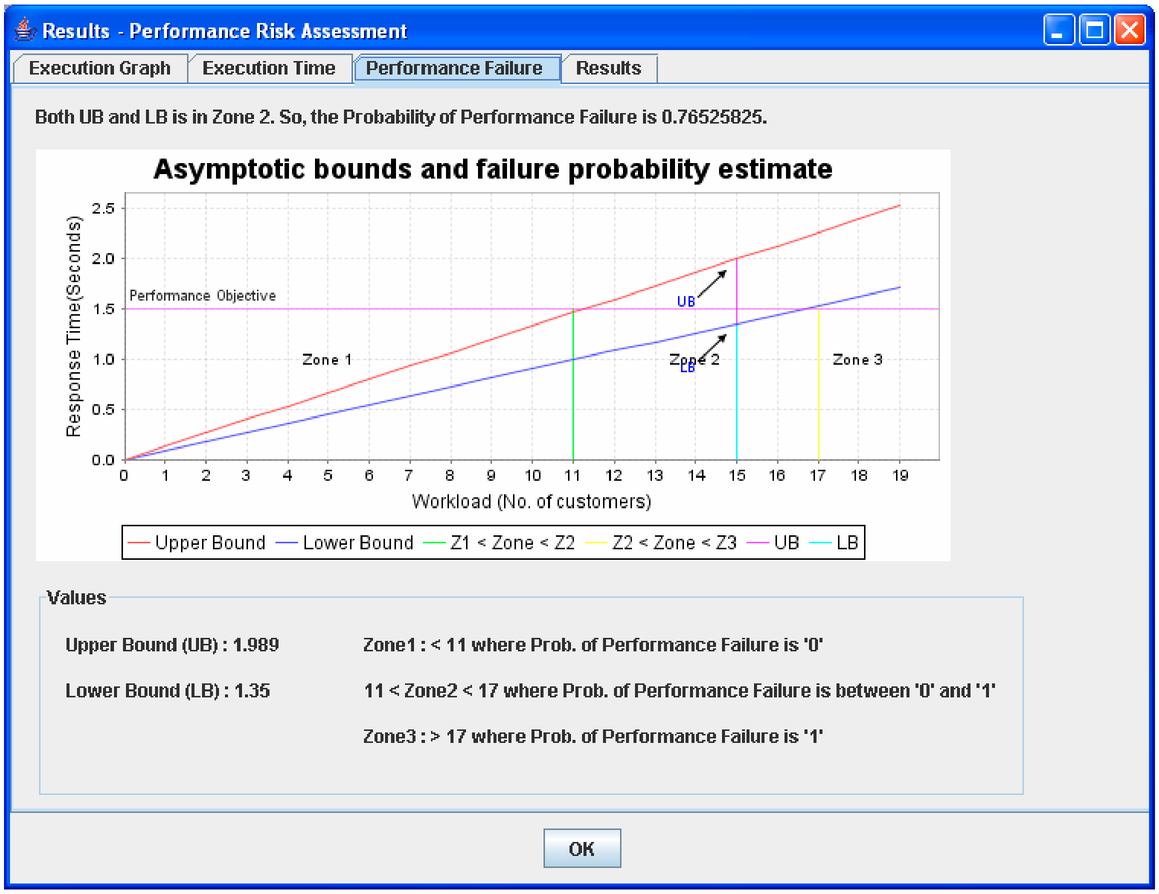
Processi...	CPU-1	CPU-3	CPU-4	DISK-1	DISK-3	DISK-4	network-...	network-...	Local
RA1		6			0			.24	
CTS1		2.7			1			1	
RA2		4			0			.16	
OFS1		5.7			1			.01	
RA3		6			0				1
RS1									
RA4		2			0		.01		
CA4	5			0			.24		
DOA1			3			0			1
OS1			2			1			
CA5									
CI3									
Total de...	9.0	26.4	5.0	0.0	2.0	1.0	0.41	1.41	2.08
Total de...	90000.0	1320.0	125.0	0.0	30.0	10.0	41000.0	141.0	2.08

The Total Execution Time is: 0.13262808

Figure 31 Execution Time Tab

The *Execution Time* tab, in figure 31, displays the total execution time of the scenario. In figure 30, the components shaded with gray represent the components that are not considered for the performance risk assessment as they do not belong to the longest path. This result in the tab clearly explains the total demand in KB units in each hardware device and they are converted to seconds units by multiplying with the speed or service time provided as input in the Settings dialog. The total demand in seconds unit is displayed in cyan whose sum provides the total execution time of the scenario.

The *Performance Failure* tab, in figure 32, displays the asymptotic bound graph and probability of performance failure.



Both UB and LB is in Zone 2. So, the Probability of Performance Failure is 0.76525825.

Upper Bound (UB) : 1.989

Lower Bound (LB) : 1.35

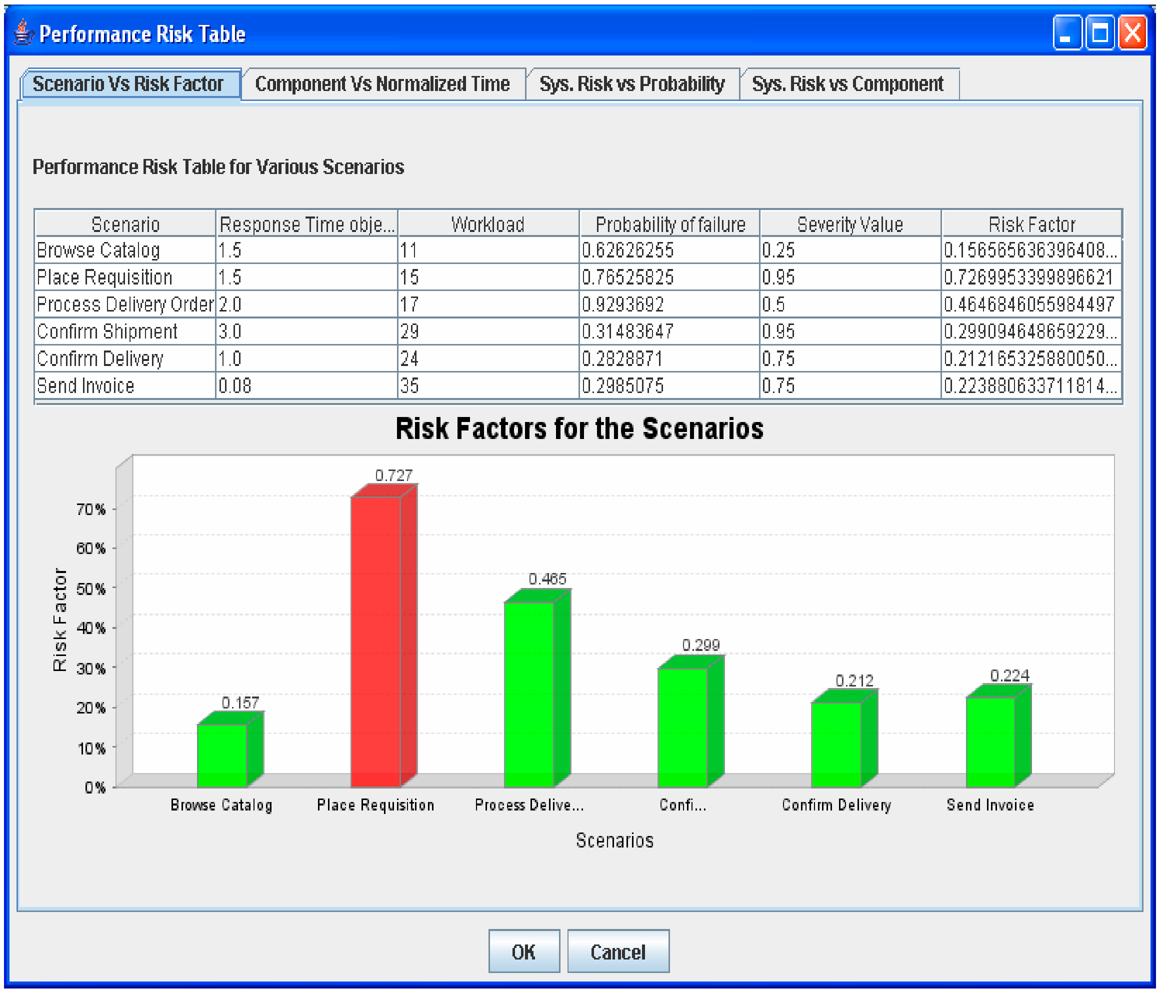
Zone1 : < 11 where Prob. of Performance Failure is '0'

11 < Zone2 < 17 where Prob. of Performance Failure is between '0' and '1'

Zone3 : > 17 where Prob. of Performance Failure is '1'

Figure 32 Performance Failure Tab

Once the performance risk assessment is performed for a scenario, the similar steps are followed for different scenario in the model. The risk factor of the scenario obtained by performing software performance risk assessment is a numerical value and the overview of Risk factor vs Scenario graph helps in understanding the impact of the scenario in the model. By navigating to *Risk -> Performance Risk -> Model*, the overall details about the scenarios in the model can be viewed. Figure 33 gives a clear picture of the scenario with high risk factor from the risk factor versus scenario graph. The Place Requisition scenario is identified as the scenario causing high risk.



Performance Risk Table for Various Scenarios

Scenario	Response Time obje...	Workload	Probability of failure	Severity Value	Risk Factor
Browse Catalog	1.5	11	0.62626255	0.25	0.156565636396408...
Place Requisition	1.5	15	0.76525825	0.95	0.7269953399896621
Process Delivery Order	2.0	17	0.9293692	0.5	0.4646846055984497
Confirm Shipment	3.0	29	0.31483647	0.95	0.299094648659229...
Confirm Delivery	1.0	24	0.2828871	0.75	0.212165325880050...
Send Invoice	0.08	35	0.2985075	0.75	0.223880633711814...

Figure 33 Model Output of Place Requisition Scenario

The next tab shows a Component vs Normalized time for each scenario. This helps in identifying the bottleneck component that consumes most of the residence time of the scenario thereby reducing the response time of the scenario. This identifies the scenario causing high risk in the system model. Figure 34 shows the component vs normalized time graph of Place Requisition scenario that shows that *CustomerAgent* component takes 90% of the total residence time of Place Requisition scenario and is the bottleneck component of the scenario.

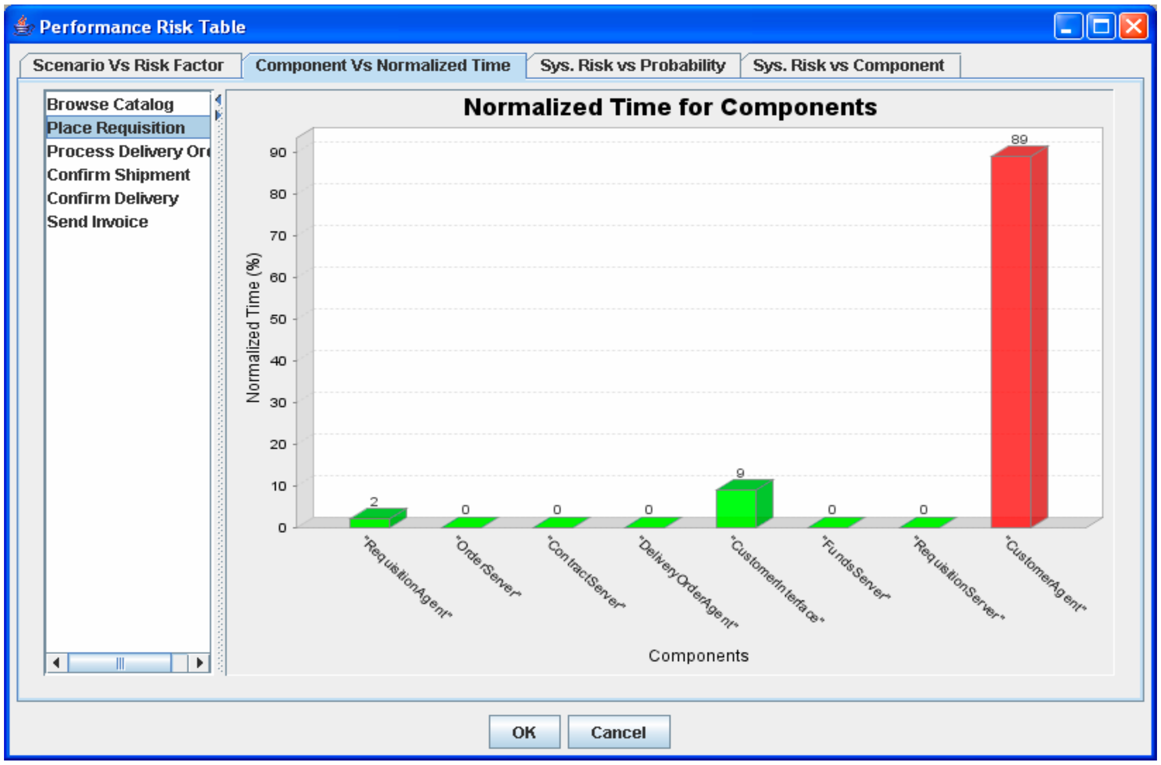


Figure 34 Component Vs Normalized Time Graph of Place Requisition Scenario

Figure 35 shows the component vs normalized time graph of Browser Catalog scenario that shows that *CatalogServer* component takes 85% of the total residence time of the scenario and is identified as the bottleneck component.

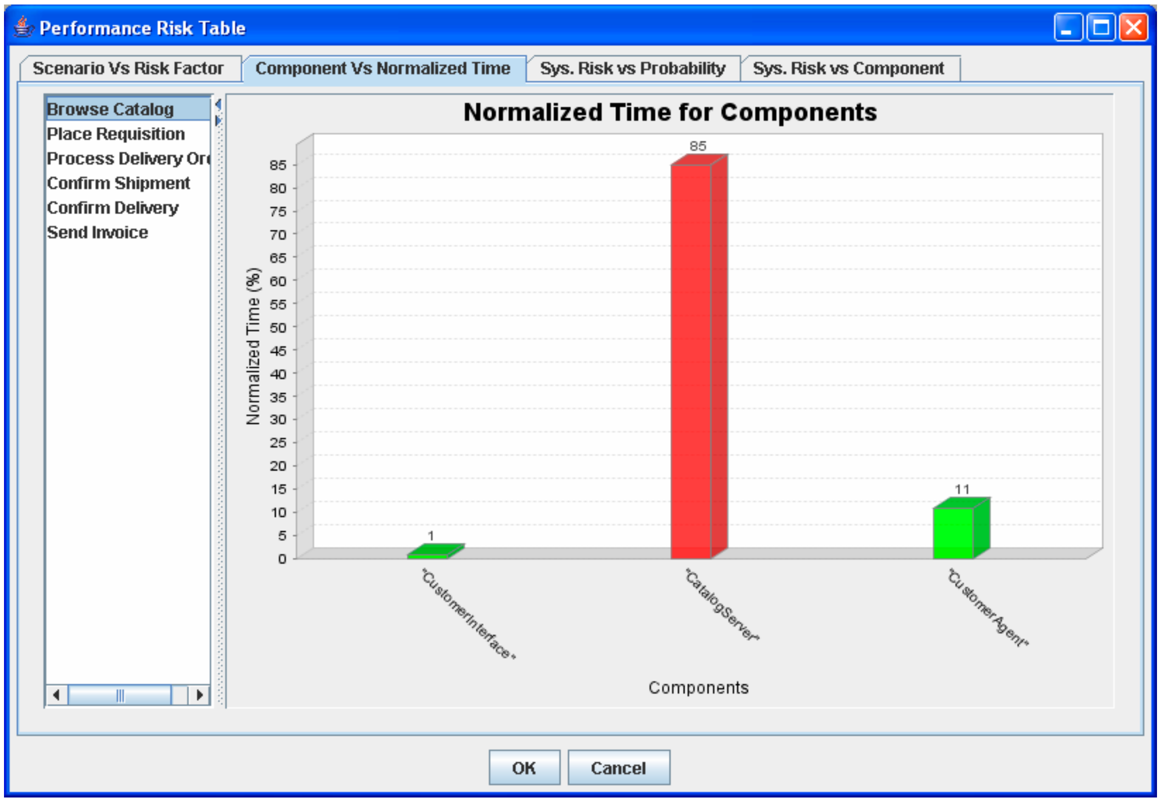


Figure 35 Component Vs Normalized Time Graph of Browse Catalog Scenario

The next two tabs represent the system level sensitivity analysis. The figure 36 shows the overall system risk factor and the graph represents the sensitivity analysis that shows the change in the overall system risk factor based on probability of occurrence of scenarios. It is observed from the figure 36 that even though the Place requisition scenario has high risk factor, if the probability of occurrence of the scenario is less, then the scenario does not have any major impact on the overall system risk factor. But, if the probability of occurrence of the Process Delivery Order that has less risk factor is high, then it has an impact on the overall system risk factor.

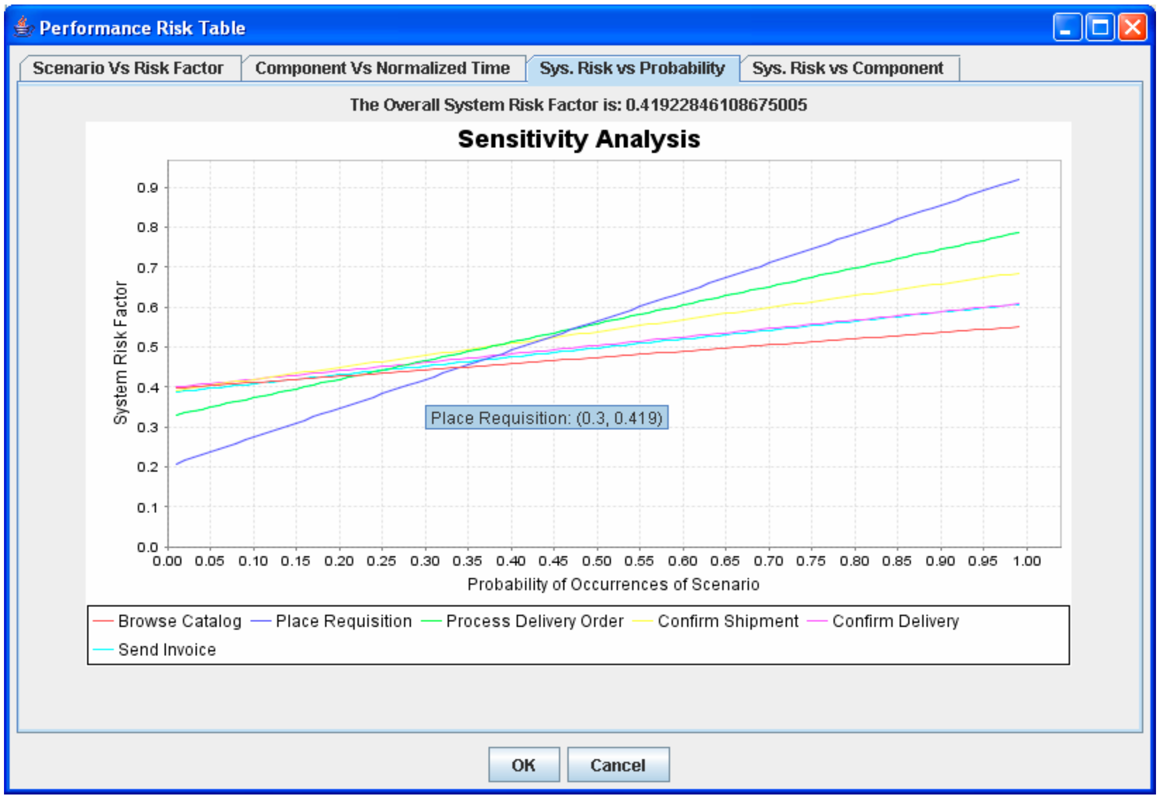


Figure 36 Overall System Risk Factor vs Probability of Occurrence of Scenario

The next tab represents the sensitivity analysis between overall system risk factor and the components of a scenario. The scenario is selected to view the changes of system risk factor based on changing the total demands of the component in that scenario. Figure 37 and 38 shows the sensitivity analysis for Place Requisition and Browse Catalog scenarios respectively.

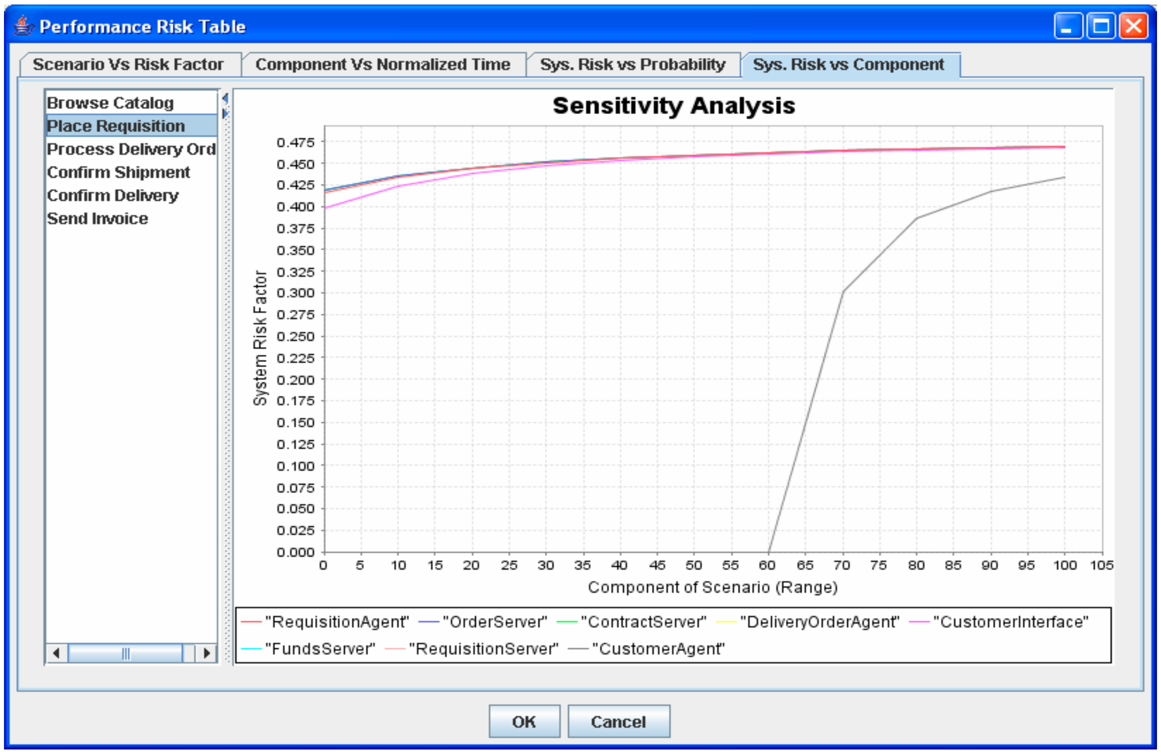


Figure 37 System Risk Factor vs Components of Place Requisition Scenario (Range)

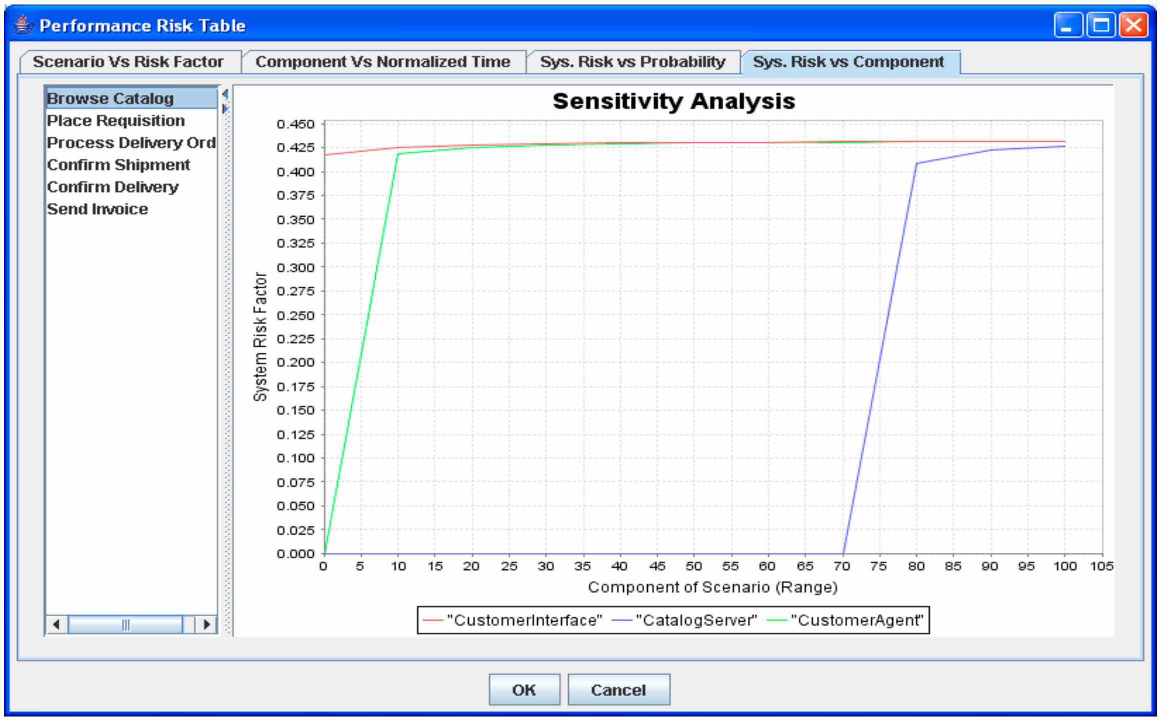


Figure 38 Sys. Risk Factor vs Components of Browse Catalog Scenario (Range)

4.2 Earth Observation System - Case Study

The SPRA methodology is applied on NASA's Earth Observation System (EOS) which is the first observing system to offer integrated measurements of the Earth's processes. The Flight Operations Segment (FOS) of EOS is responsible for the planning, scheduling, commanding, and monitoring of the spacecraft and the instruments on board. The SPRA is performed based on commanding service. The figure 39 shows the use case diagram of commanding service of EOS.

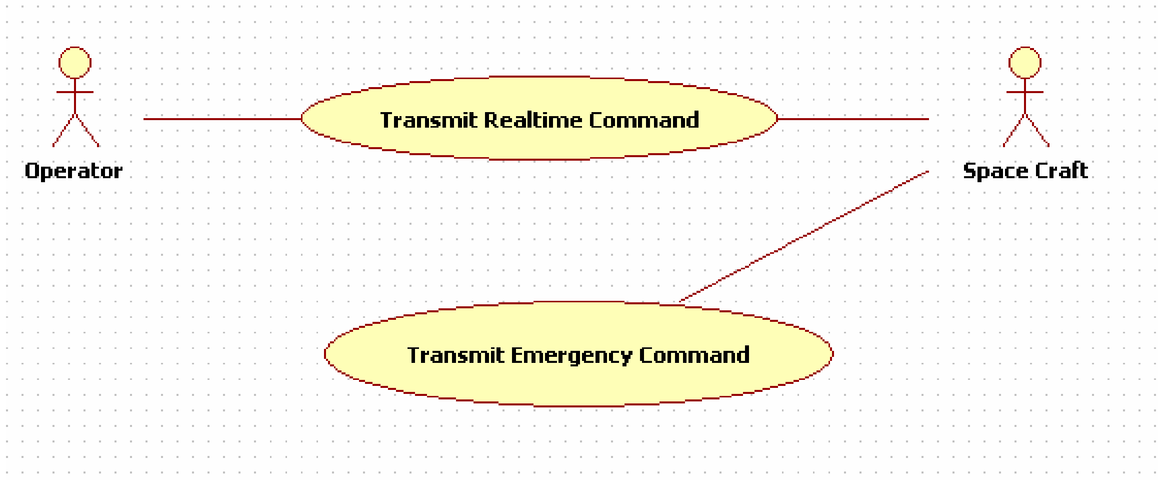


Figure 39 Use Case Diagram of EOS

The Preplanned emergency scenario comprises of two sequence diagrams:

- Preparation of command groups that are to be uplinked (SD1)
- Handling the transmission failure during uplink (SD2)

It is assumed for the purpose of illustration that SD1 is executed once and SD2 i.e. the retransmission twice before there is a mission failure. The figure 40 shows the sequence diagram of the *Preplanned Emergency Command Transmission (SD1)* scenario and figure 41 shows the sequence diagram of *Handle Transmission Failure (SD2*) scenario.

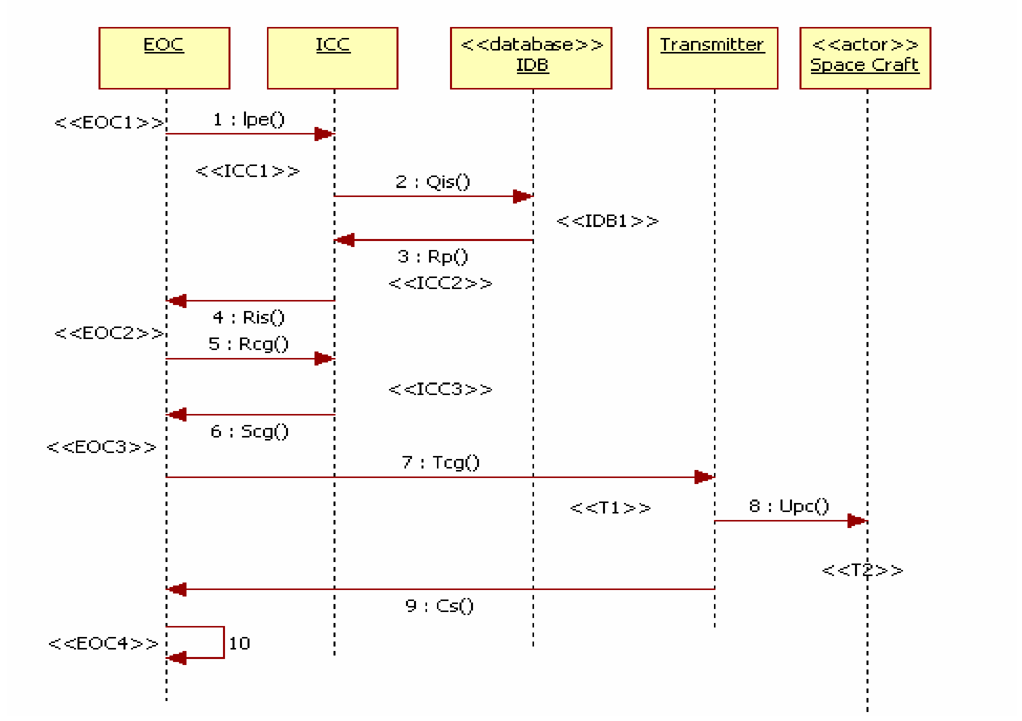


Figure 40 Preplanned Emergency Command Transmission (SD1) scenario

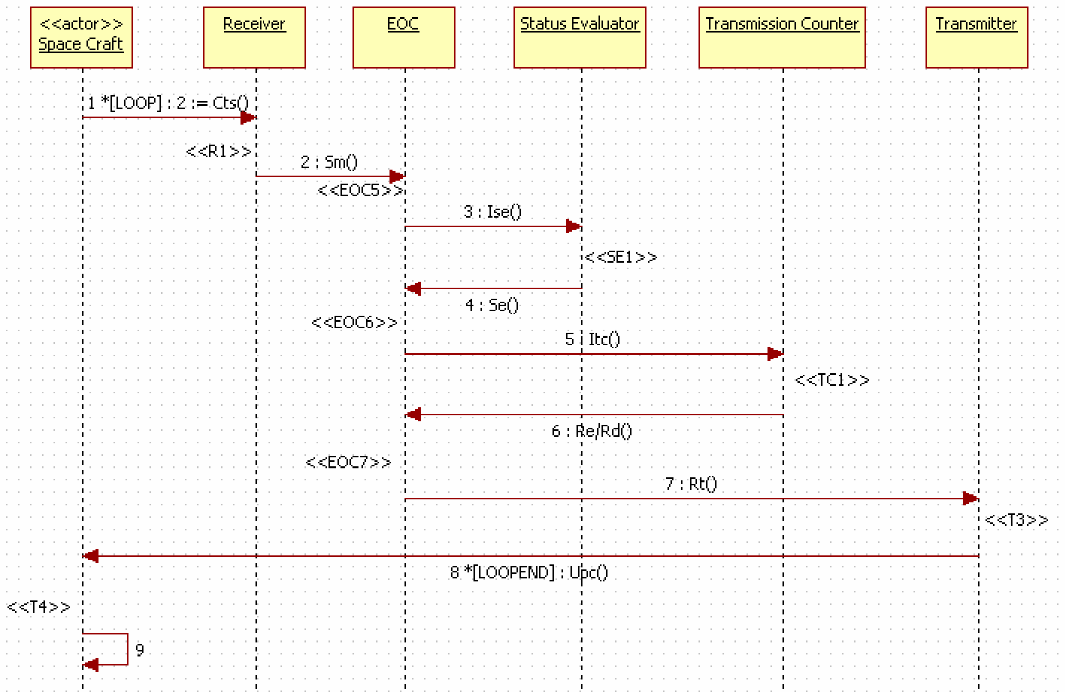


Figure 41 Handle Transmission Failure (SD2) scenario

Figure 42 shows the Execution graph of Preplanned Emergency Scenario where SD2 is run in a loop that executes twice.

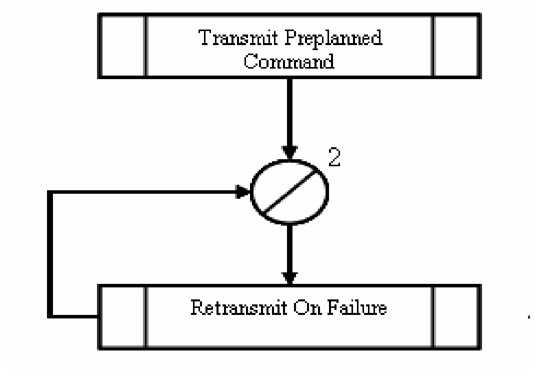


Figure 42 Execution graph of Preplanned Emergency Scenario

The sequence diagram of both the scenarios (SD1 and SD2) can be represented in the same sequence diagram with looping for SD2. With looping, the tool multiplies the amount of resources provided for each demand vector with the number of times the loop has to be executed. The figure 43 shows the deployment diagram of EOS.

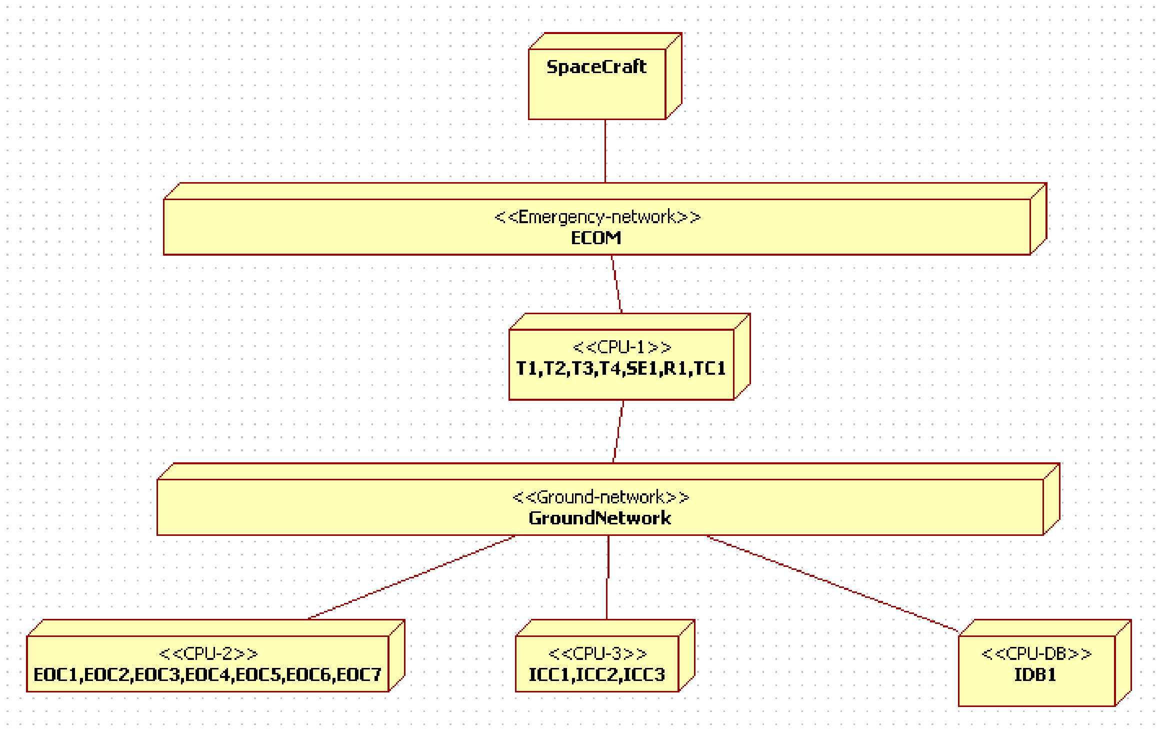
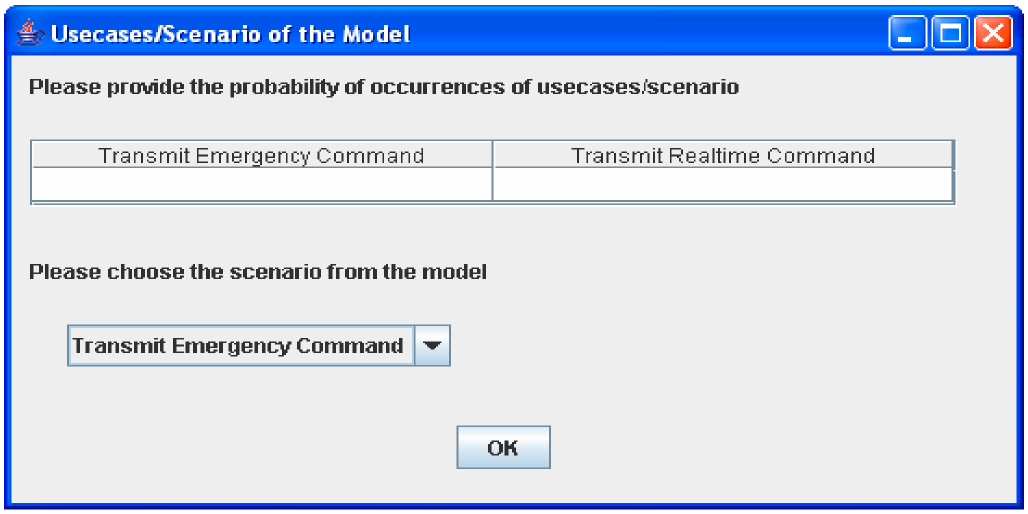


Figure 43 Deployment Diagram of EOS

The EOS model developed using StartUML is provided as input to the tool developing SPRA. Figure 44 shows the window that pops-up where the probability of occurrences should be provided by the user as it is not annotated in the use case diagram and figure 45 shows the window with its input. Then choose the scenario to perform SPRA from the combobox.



Usecases/Scenario of the Model

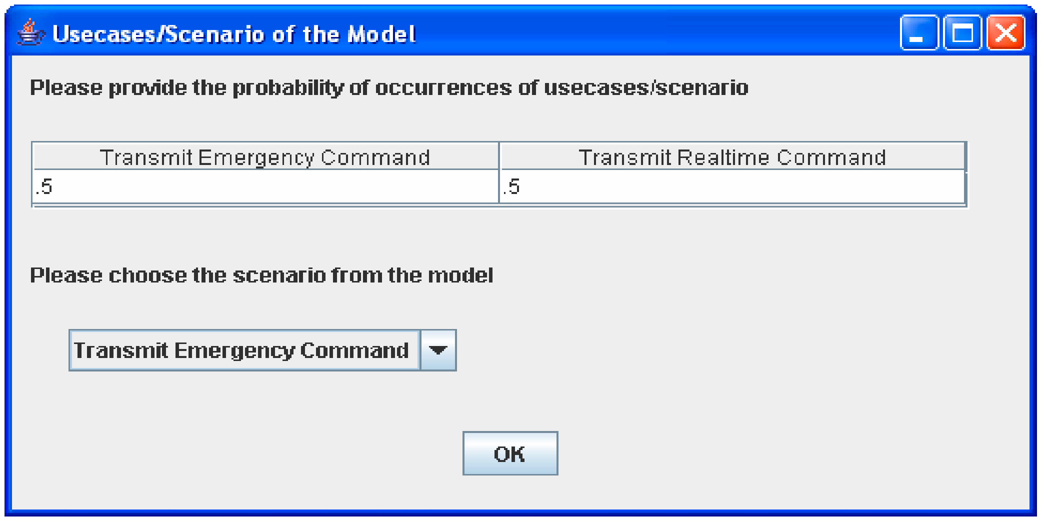
Please provide the probability of occurrences of usecases/scenario

Transmit Emergency Command	Transmit Realtime Command

Please choose the scenario from the model

Transmit Emergency Command

Figure 44 Dialog to enter the Probability of Occurrences and choose the Scenario



Usecases/Scenario of the Model

Please provide the probability of occurrences of usecases/scenario

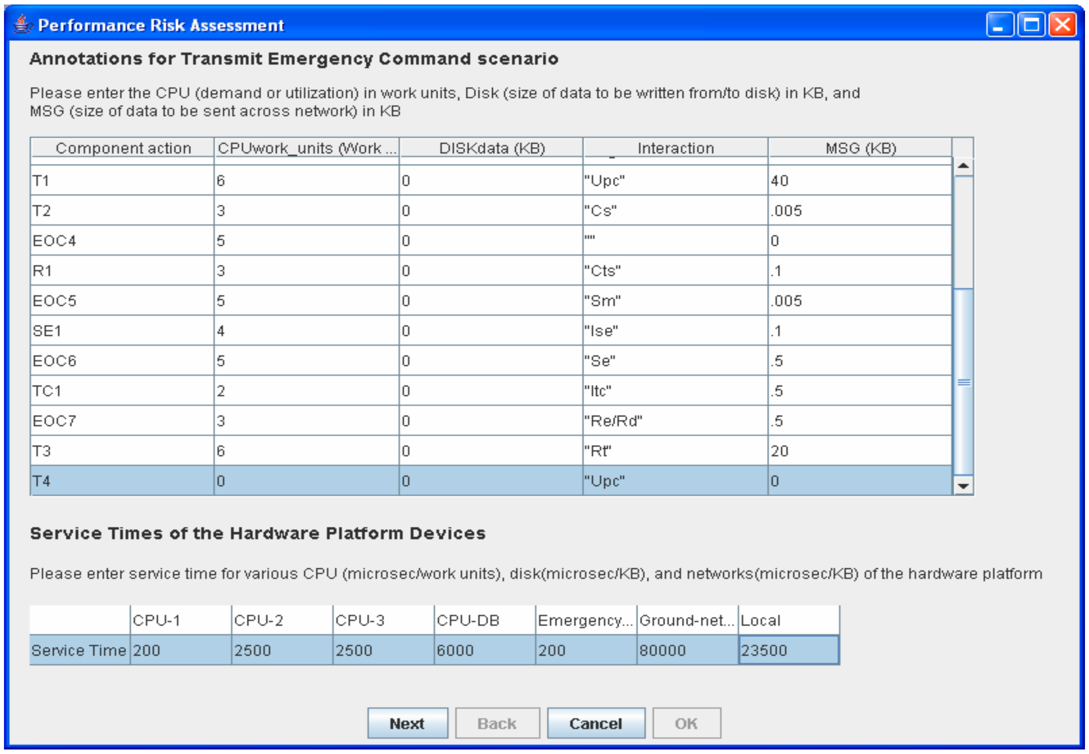
Transmit Emergency Command	Transmit Realtime Command
.5	.5

Please choose the scenario from the model

Transmit Emergency Command

Figure 45 Input Values for Probability of Occurrences of Scenario

The figure 46 and figure 47 shows the Settings Dialog where the amount of resources required by the components, speed of hardware devices, performance objective, realistic workload, and severity obtained from FFA is provided as input.



Annotations for Transmit Emergency Command scenario

Please enter the CPU (demand or utilization) in work units, Disk (size of data to be written from/to disk) in KB, and MSG (size of data to be sent across network) in KB

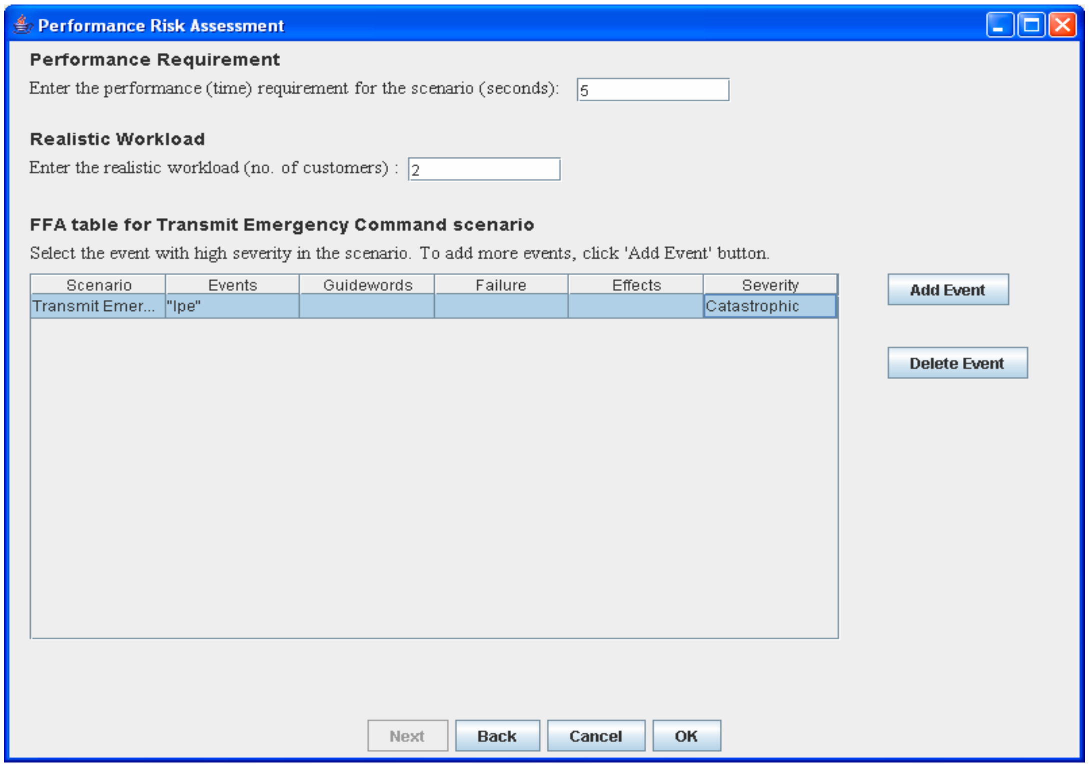
Component action	CPUwork_units (Work ...	DISKdata (KB)	Interaction	MSG (KB)
T1	6	0	"Upc"	40
T2	3	0	"Cs"	.005
EOC4	5	0	""	0
R1	3	0	"Cts"	.1
EOC5	5	0	"Sm"	.005
SE1	4	0	"Ise"	.1
EOC6	5	0	"Se"	.5
TC1	2	0	"Itc"	.5
EOC7	3	0	"Re/Rd"	.5
T3	6	0	"Rt"	20
T4	0	0	"Upc"	0

Service Times of the Hardware Platform Devices

Please enter service time for various CPU (microsec/work units), disk(microsec/KB), and networks(microsec/KB) of the hardware platform

	CPU-1	CPU-2	CPU-3	CPU-DB	Emergency...	Ground-net...	Local
Service Time	200	2500	2500	6000	200	80000	23500

Figure 46 Settings Dialog of Transmit Emergency Scenario



Performance Requirement

Enter the performance (time) requirement for the scenario (seconds): 5

Realistic Workload

Enter the realistic workload (no. of customers) : 2

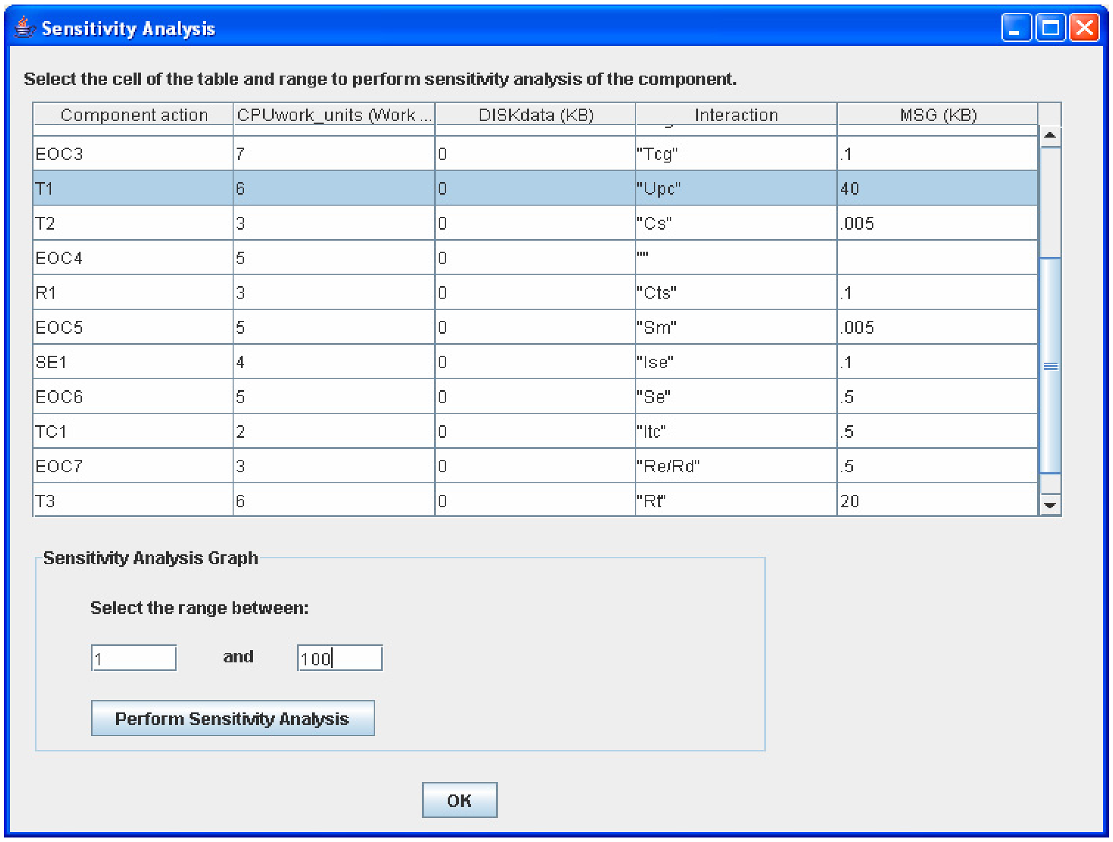
FFA table for Transmit Emergency Command scenario

Select the event with high severity in the scenario. To add more events, click 'Add Event' button.

Scenario	Events	Guidewords	Failure	Effects	Severity
Transmit Emer...	"Ipe"				Catastrophic

Figure 47 Additional Settings Dialog of Transmit Emergency Scenario

Once the input is provided, click *OK* button to save the settings. The user can perform sensitivity analysis of the input values to view its impact on the scenario risk factor. Figure 48 shows the Sensitivity Analysis dialog for Transmit Emergency Command Scenario.



Sensitivity Analysis

Select the cell of the table and range to perform sensitivity analysis of the component.

Component action	CPUwork_units (Work ...	DISKdata (KB)	Interaction	MSG (KB)
EOC3	7	0	"Tcg"	.1
T1	6	0	"Upc"	40
T2	3	0	"Cs"	.005
EOC4	5	0	""	
R1	3	0	"Cts"	.1
EOC5	5	0	"Sm"	.005
SE1	4	0	"Ise"	.1
EOC6	5	0	"Se"	.5
TC1	2	0	"Itc"	.5
EOC7	3	0	"Re/Rd"	.5
T3	6	0	"Rt"	20

Sensitivity Analysis Graph

Select the range between:

1 and 100

Perform Sensitivity Analysis

OK

Figure 48 Sensitivity Analysis Dialog of Transmit Emergency Command Scenario

Figure 49 shows the sensitivity analysis graph between scenario risk factor and changes of MSG software requirement of T1 component in the range of 1 to 100. It is seen that the value of data size of the network for the T1 component increases the risk factor as its value increases.

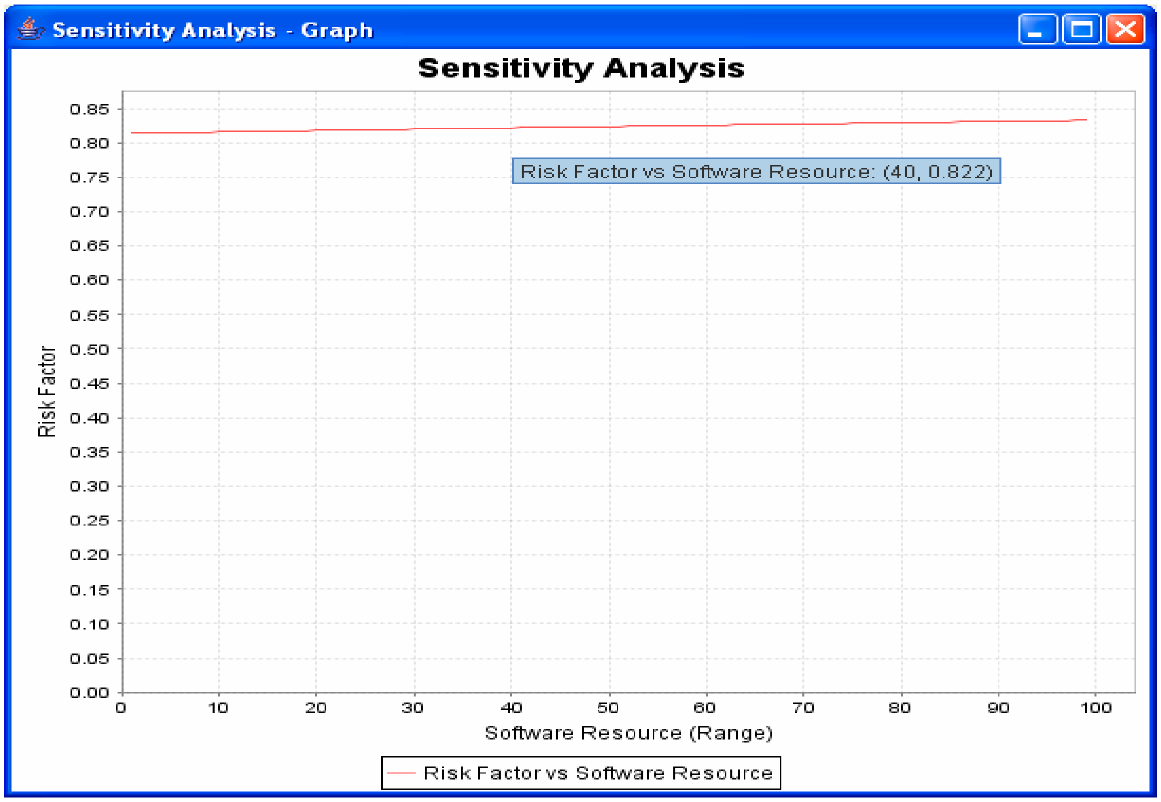
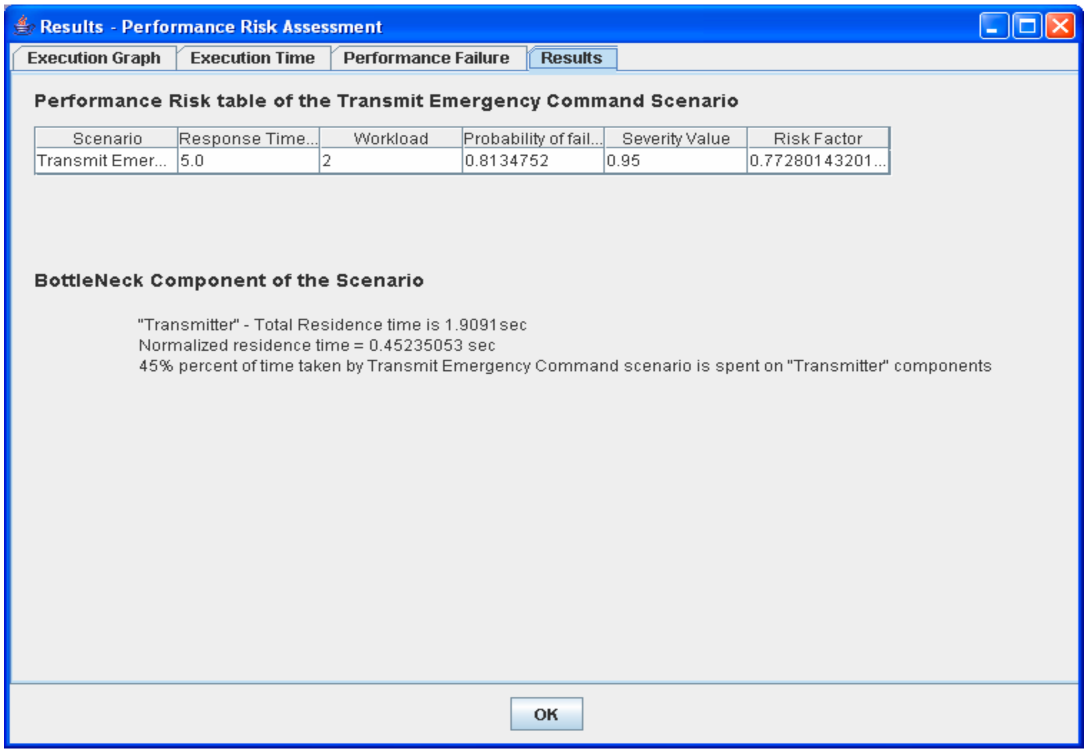


Figure 49 Sensitivity Analysis graph for T1 component

Once performing sensitivity analysis, the *Scenario* -> *Run* is clicked which performs the calculations as per the SPRA methodology and displays the results. Figure 50 shows the *Results* dialog with risk factor and bottleneck components and Figure 51 shows the Execution graph of the scenario.



Performance Risk table of the Transmit Emergency Command Scenario

Scenario	Response Time...	Workload	Probability of fail...	Severity Value	Risk Factor
Transmit Emer...	5.0	2	0.8134752	0.95	0.77280143201...

BottleNeck Component of the Scenario

"Transmitter" - Total Residence time is 1.9091sec
Normalized residence time = 0.45235053 sec
45% percent of time taken by Transmit Emergency Command scenario is spent on "Transmitter" components

Figure 50 Results Dialog of Transmit Emergency Command Scenario

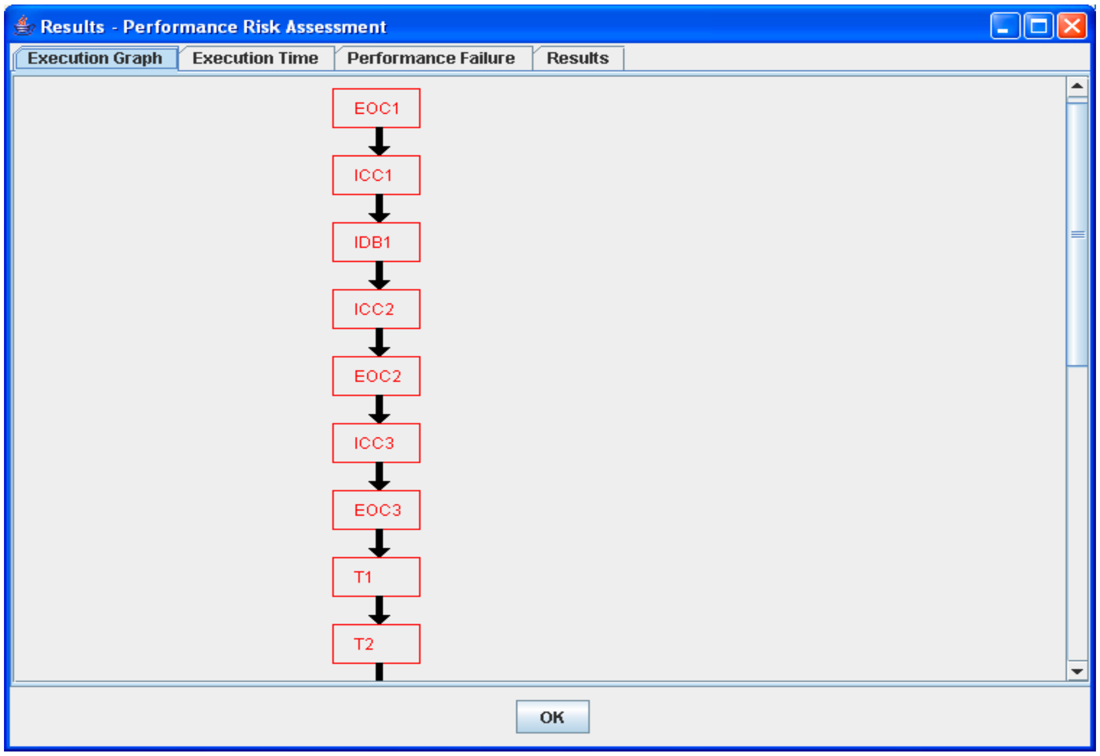
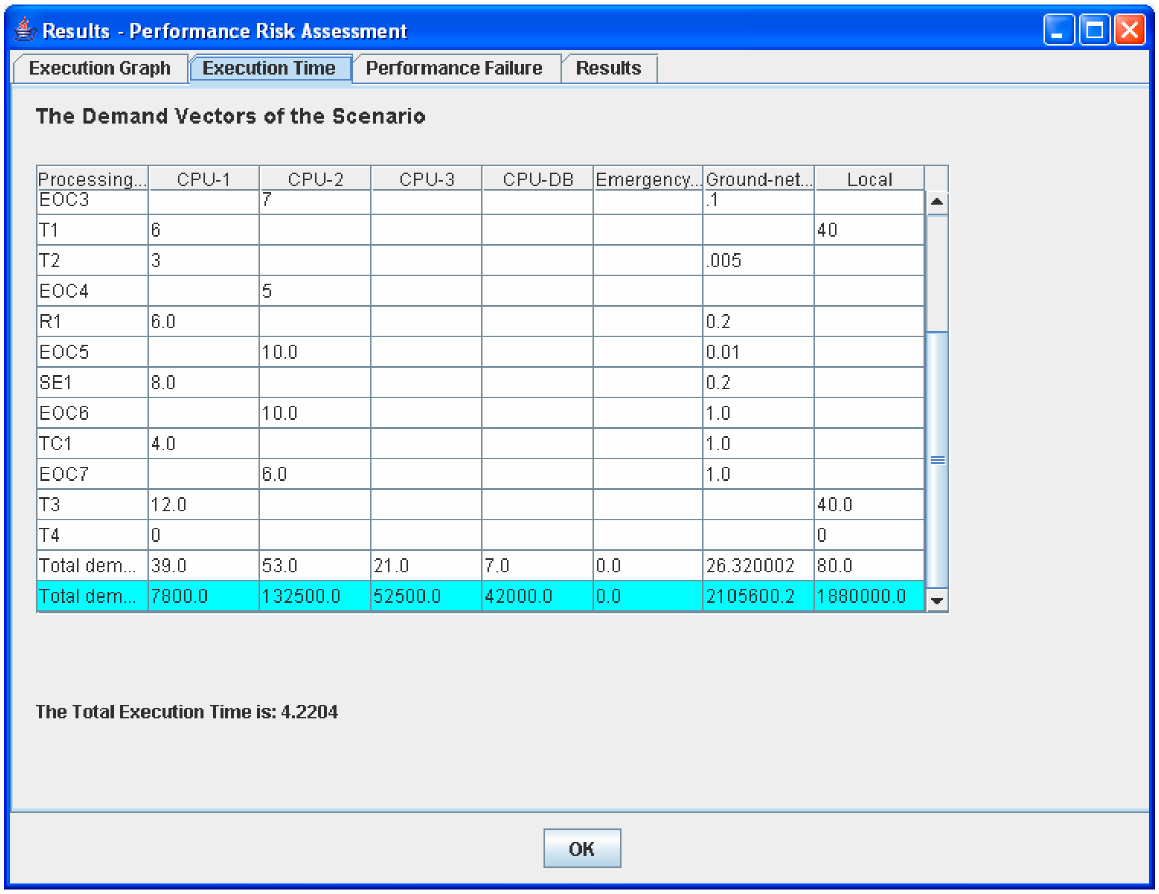


Figure 51 Execution Graph Tab of Transmit Emergency Command Scenario

Please note that the execution graph shown in figure 42 and figure 51 is different because of the limitation of the tool not supporting ‘looping’ in converting demand vectors into execution graph. The execution graph shows the longest path in red and since there are no concurrent actions in the scenario, all the components are considered for System Execution Model of the scenario.

Figure 52 and Figure 53 represent the Execution Time tab and Performance Failure tab of Transmit Emergency Command scenario.

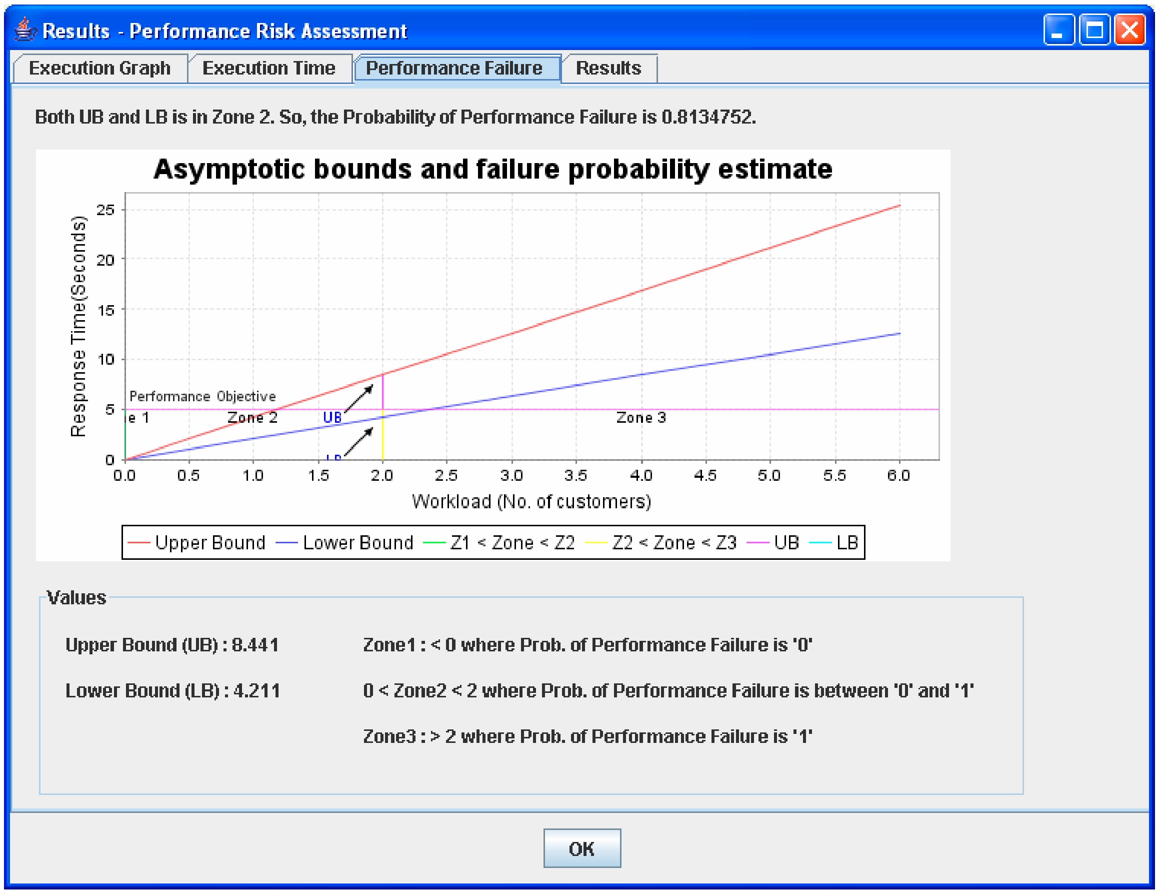


The Demand Vectors of the Scenario

Processing...	CPU-1	CPU-2	CPU-3	CPU-DB	Emergency...	Ground-net...	Local
EOC3		7				.1	
T1	6						40
T2	3					.005	
EOC4		5					
R1	6.0					0.2	
EOC5		10.0				0.01	
SE1	8.0					0.2	
EOC6		10.0				1.0	
TC1	4.0					1.0	
EOC7		6.0				1.0	
T3	12.0						40.0
T4	0						0
Total dem...	39.0	53.0	21.0	7.0	0.0	26.320002	80.0
Total dem...	7800.0	132500.0	52500.0	42000.0	0.0	2105600.2	1880000.0

The Total Execution Time is: 4.2204

Figure 52 Execution Time Tab of Transmit Emergency Command Scenario



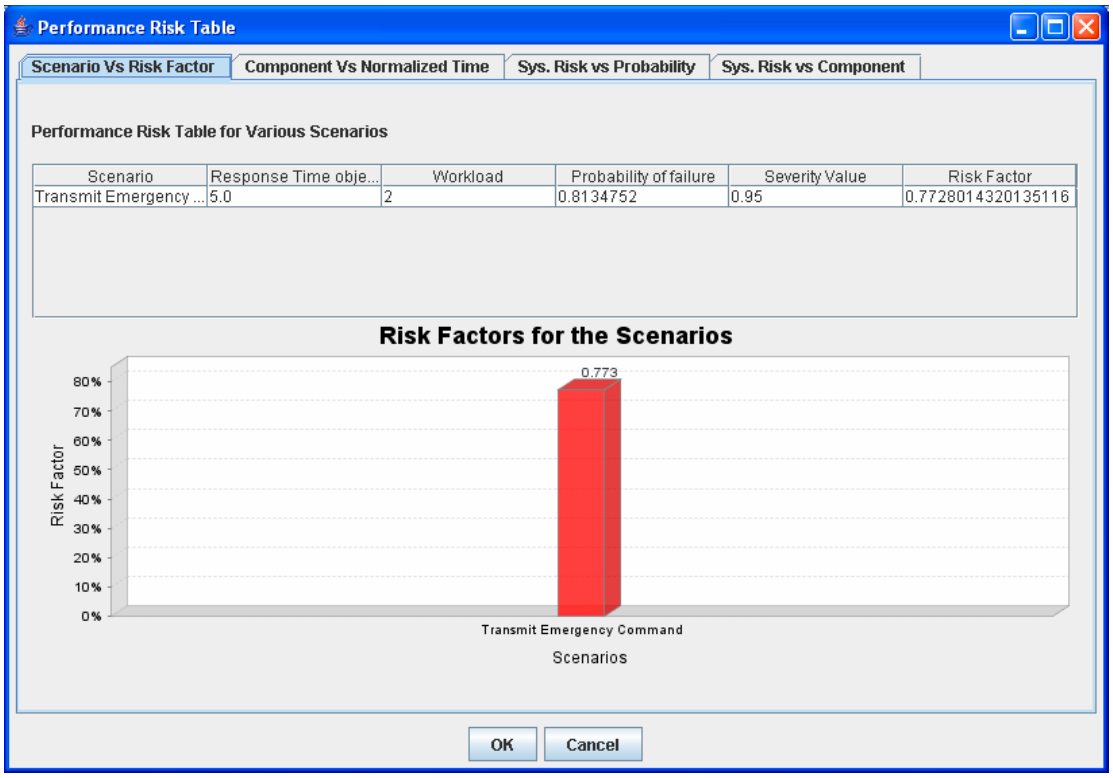
Both UB and LB is in Zone 2. So, the Probability of Performance Failure is 0.8134752.

Values

Upper Bound (UB) : 8.441	Zone1 : < 0 where Prob. of Performance Failure is '0'
Lower Bound (LB) : 4.211	0 < Zone2 < 2 where Prob. of Performance Failure is between '0' and '1'
	Zone3 : > 2 where Prob. of Performance Failure is '1'

Figure 53 Performance Failure Tab of Transmit Emergency Command Scenario

Once the scenario based SPRA is performed using the tool, the overall picture of the EOS model can be obtained by selecting the model output of the tool. The figure 54 shows the risk factor vs scenario graph, figure 55 shows the component vs normalized time of each scenario, and figure 56 and figure 57 shows the system level sensitivity analysis. Since the tool is implemented for commanding service functionality of EOS model, the model output displays output for Transmit Emergency Command scenario.


Performance Risk Table

Scenario Vs Risk Factor | Component Vs Normalized Time | Sys. Risk vs Probability | Sys. Risk vs Component

Performance Risk Table for Various Scenarios

Scenario	Response Time obje...	Workload	Probability of failure	Severity Value	Risk Factor
Transmit Emergency ...	5.0	2	0.8134752	0.95	0.7728014320135116

Figure 54 Model Output – Risk Factor vs Scenario Graph

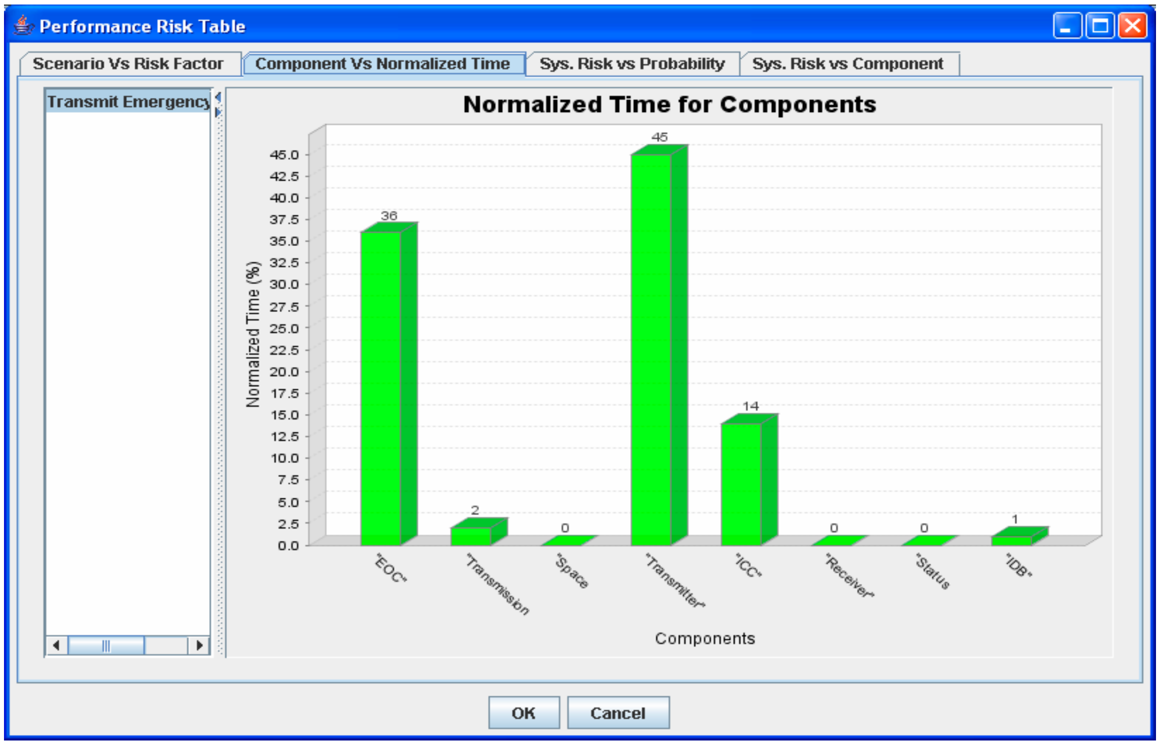
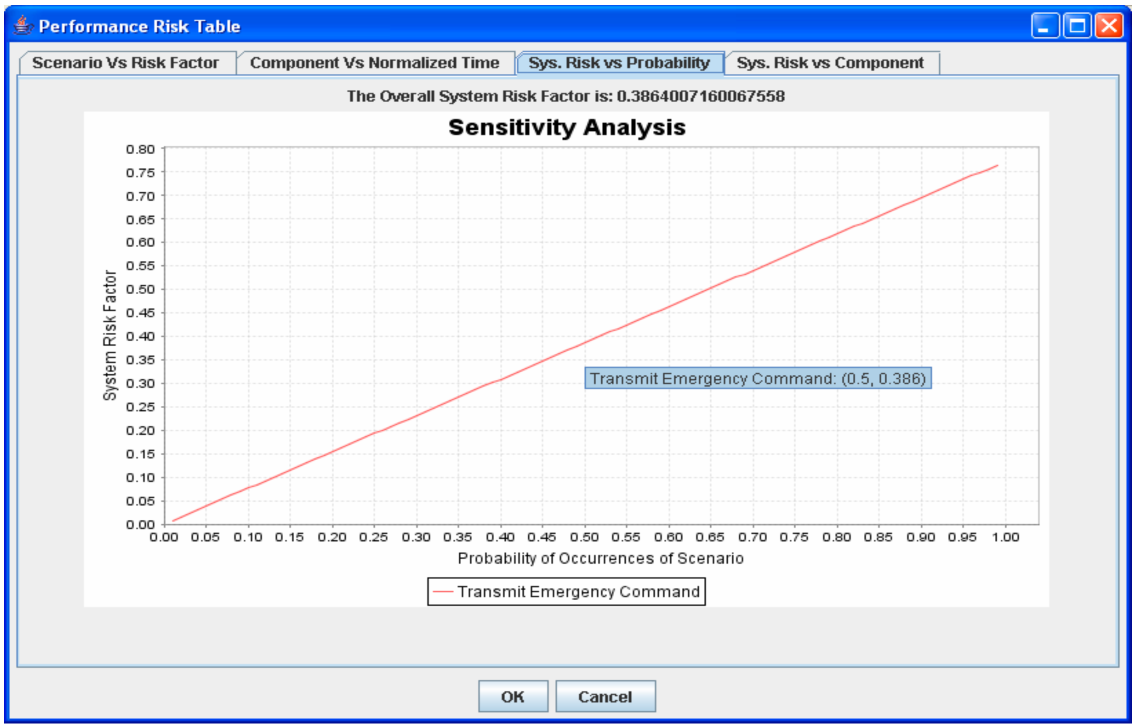


Figure 55 Model Output – Component vs Normalized Time Graph



The Overall System Risk Factor is: 0.3864007160067558

Figure 56 System Risk vs Probability of Occurrence of Scenario

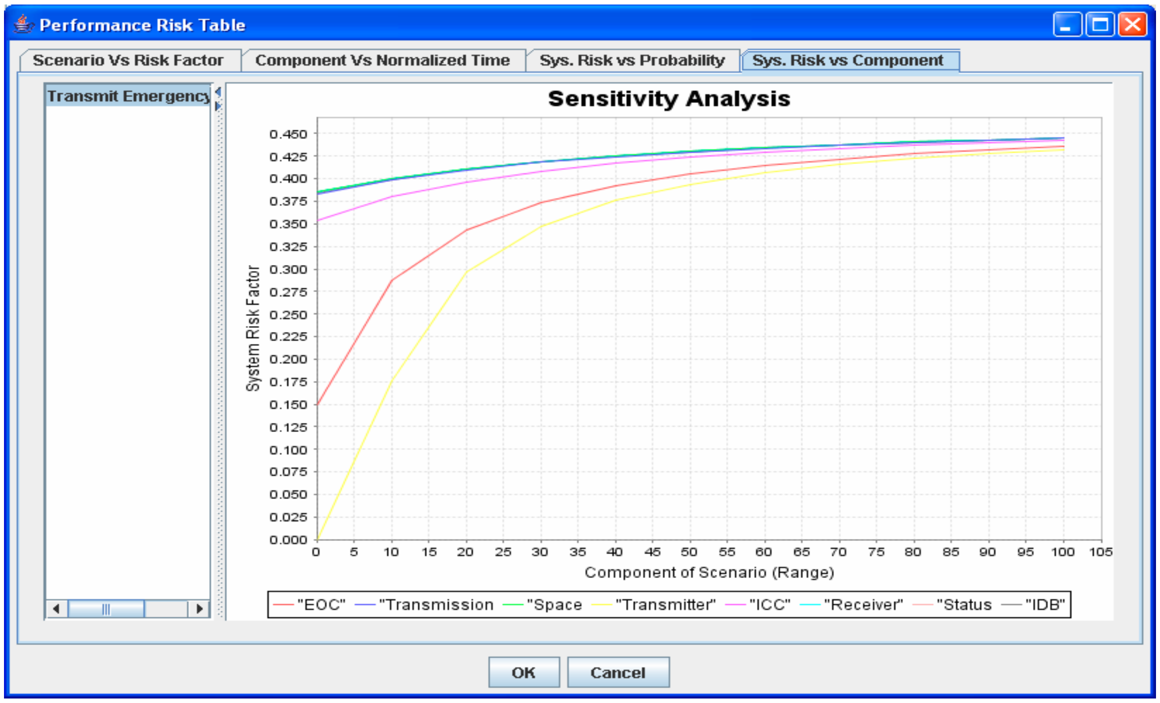


Figure 57 System Risk vs Component of Scenario (Range)

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The step-by-step methodology for Software Performance Risk Assessment is an extensive and light-weight approach to perform performance modeling and it can be applied in real time by use of an easy-to-use and automated tool. The problem report describes the tool developed for implementing the SPRA methodology in SARA tool as one of its features. The tool supporting SPRA is an automated tool that provides 3 outcomes: a) the probability of performance failure, b) estimating risk factor, and c) identifying the high-risk or bottleneck component causing the performance risk The tool performs the calculation in the background from the annotated UML sequence diagram that describes the scenario of a software architectural model and annotated UML deployment diagram that provides hardware information of the model as input files to the tool.

The tool helps software developers and designer in early identification of performance risk and intervention to rectify the changes which is more cost-effective. These outcomes are an important feedback to the designers as they can devise more effort to the design and implementation of the most critical components.

5.2 Future Work

The future work of the problem report is

- The tool should be compatible to inputs developed using other software modeling tools such as Rational Rose.
- The tool should support use case with multiple scenarios in a system model.

REFERENCES

Cortellessa, V., Popstojanova, K., Appukkutty, K., Guedem, A., Hassan, A., Elnaggar, R., Abdelmoez, W., Ammar, H. (2005), *Model-Based Performance Risk Analysis*, IEEE transactions on software engineering, Vol. 31, No. 1.

Clements, P. and Northrop, L. (2002), *Software Product Lines Practices and Patterns*, Addison-Wesley.

Dimitrov, E. and Schmietendorf**,** A.**,** *Deutsche Telekom,* Dumke, R., University of Megdeburg (2002), *UML-Based Performance Engineering Possibilities and Techiniques.*

Ebert, C. (1997), *Dealing with nonfunctional requirements in large software systems***,** Annals of Software Engineering 3.

Fullam, K., Lam, D. (2003), *Performance Simulations from UML Performance Work Product*.

Gomma, H., Menasce, D. (2000), *Design and Performance Modeling of Component Interconnection Patterns for Distributed Software Architectures,* ACM, NY, pp. 117 – 126.

Guedem, A. (2004), *Software Architectural Risk Assessment*, Masters Thesis, Department of Computer Science and Electrical Engineering Morgantown, WVU.

Hassan, A. (2004), dissertation, *Architectural Level Risk Assessment***,** College of Engineering and Mineral Resources, WVU.

Kakhipuro, P. (1999), *UML based performance modeling of object-oriented distributed systems*, Proc. Second Int’l Conf. the Unified Modeling Language.

Kamavaram, S. and Popstojanova, K. (2003), *Sensitivity of Software Usage to Changes in the Operational Profile*, Lane Department of Computer Science and Electrical Engineering West Virginia University, Morgantown, WV

Liebrock, L. (2005), *Empirical Sensitivity Analysis for Computational Procedures,* New Mexico Institute of Mining and Technology Computer Science Department Socorro, NM, pp 32-35.

Lazowska, D., Zahorjan, J., Graham, S., and Sevcik, K. (1984), *Quantitative System Performance: Computer System Analysis Using Queuing Network Models*, Prentice-Hall, Inc.

Molkdar, D., Burley, S., and Wallington, J. (2002), *Comparison Between Simulation and Analytical Methods of UMTS Air Interface Capacity Dimensioning*, Motorola Systems Engineering Analysis, United Kingdom, pp. 1596 – 1601.

Popstojanova, K., Hassan, A., Guedem, A., Abdelmoez, W., Nassar, D., Ammar, H., Mili, A. (2003), Architectural-Level Risk Analysis Using UML, IEEE.

Shaik, K., (2007), *Software Architecture Risk Assessment Tool*, Masters Problem Report, College of Engineering and Mineral Resources, WVU.

Sharma, V., Jalote, P., Trivedi, K. (2006), *A Performance Engineering Tool for Tiered Software Systems*, Proceedings of the 30th Annual COMPSAC.

Smith, C., and Williams, L. (1997), *Performance Engineering Evaluation of Object-Oriented Systems with SPE.ED*, Computer Performance Evaluation: Modelling Techniques and Tools, Berlin.

Smith, C., and Williams, L. (2001), *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software,* Addison-Wesley, ISDN 0-201-72229-1

Smith, C. and Williams, L. (2002), *PASA^{SM}: An Architectural Approach to Fixing Software Performance Problems*, Software Engineering Research and Performance Engineering Services.

UML^{TM} Profile for Schedulability, Performance, and Time Specification (2005), An Adopted Specification of the Object Management Group (OMG), Inc,Version 1.1.

Work, P. and Johnson, H.E., Jr. (1995), *Risk Management in Computer-Based Systems Development by Use of Performance and Reliability Metrics,* Systems Engineering of Computer Based Systems, pp. 367-373.