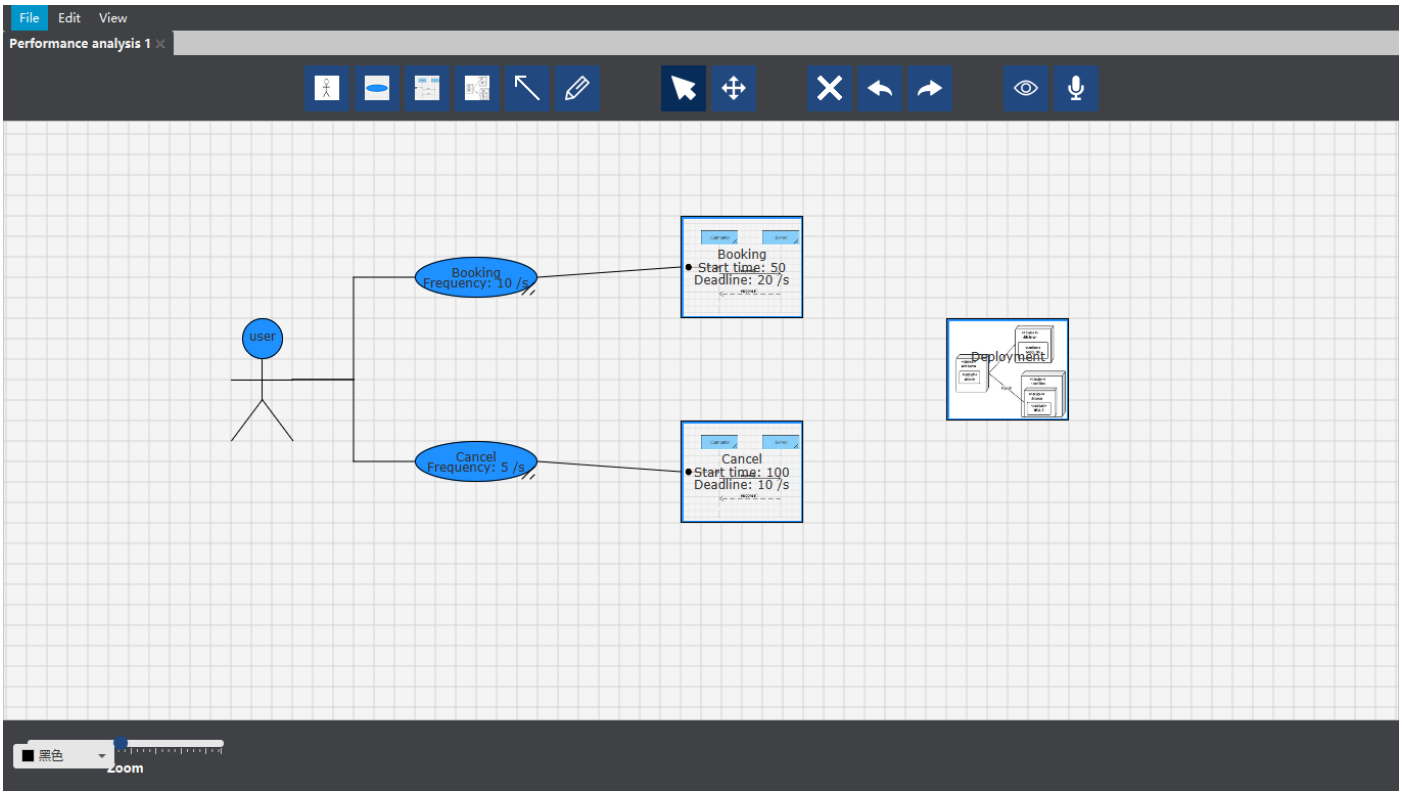
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Performance Analysis in Early Design Stage of Software System

Bachelor of Science Thesis in Software Engineering and Management

Kai Fu
Darja Linkova

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Performance Analysis in Early Design Stage of Software System

Kai Fu
Darja Linkova

Supervisor: Michel Chaudron, Rodi Jolak
Examiner: Salome Maro

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Performance Analysis in Early Design Stage of Software System

Kai Fu
Gothenburg University
Gothenburg, Sweden
guskaifu@student.gu.se

Darja Linkova
Gothenburg University
Gothenburg, Sweden
guslinkda@student.gu.se

***Abstract*—This study aims to deliver a user-friendly environment which allows architects to analyze the performance of software systems during early design stages. Design science method is used for developing the environment which consists of a UML design tool and a component for performance analysis. To evaluate the usability of the environment, a user study is conducted. All users' feedback was collected and analyzed. The results show that the perceptions regarding the usability were not that good as expected. Suggestions on future research and implementation are also reported in this study.**

I. Introduction

Systems become more and more complex and expensive every day. As a result, system architecture becomes more critical for the whole project and mistakes in it are too costly. New tools that can evaluate early architectural decisions can make a process of the system design more efficient. Smith and Williams state that performance is a critical quality for the success of today's software system. They also give a definition to performance as "the degree to which a system meets its objectives for timeliness" [1] . Performance related decisions that are made early can and should be evaluated in the early design phase. [2]

A. Statement of the problem

Smith and Browne mention that performance engineering has a quite unique problem since success in it is impossible to demonstrate and failure is obvious [2] . Concerning performance problems "fix-it-later" approach was traditionally chosen. The main emphasis is made on software correctness, performance issues are postponed till later stages of the software development, such as integration and testing [3] . At the same time, one of the conclusions made in the study "The Future of Software Performance Engineering" is that performance problems are mainly caused by bad architecture decisions, not by coding issues [4] . Neglecting performance and postponing it to the later phases can be the reason of increasing of the development costs and late deployment. Moreover, such approach may affect not only performance but other system qualities, such as understandability, maintainability, and reusability [3] . All mentioned works give an understanding of the importance of approaching of performance problems in the early stage. Smith and Williams introduce Software Performance Engineering (SPE) modelling process as a method which helps to identify potential performance problems in the stage when they can be solved in the less costly way. As a rule use cases are seen as systems' requirements. In the scopes of SPE, use cases have a bigger importance. They allow architects to identify the workloads, which are groups of user requests [1] . This approach allows to start performance analysis in the early design stage when only important use cases are identified. Figure 1 shows use case diagram with values required for performance evaluation in SPE. [5]

There is a number of tools that allow developers to evaluate system performance, they will be discussed in related work and background part. However, these tools are aimed to be used in later stages of software design or on already functioning systems. As a result, it may lead to high costs due to late problem recognition.

B. Purpose of the study

This thesis is a part of a project driven by Chalmers and Gothenburg University researching a smartboard software design environment [6] . The main goal of which is to create a user-friendly UML environment to support an effective software design process. The UML modeling tool is provided for us. Our purpose is to integrate it with performance evaluation algorithm and improve the usability of the whole system.

C. Research question

Our main research question: RQ1: Is it possible to accomplish a performance prediction system for the early software design stage? Our sub-questions are: RQ1.1: How can we integrate the performance prediction in the early design stage into a smartboard UML design environment? RQ1.2: How can we evaluate the usability of the performance prediction system? The sub-questions aim to assist answering the main research question.

D. Significance of the study

After successfully addressing all the research questions, the result is a system which is able to get use case -, sequence-, deployment diagrams as user input and introduce performance evaluation results based on those diagrams. It can help architects to predict possible performance bottlenecks at the very beginning of the system design. The earlier performance

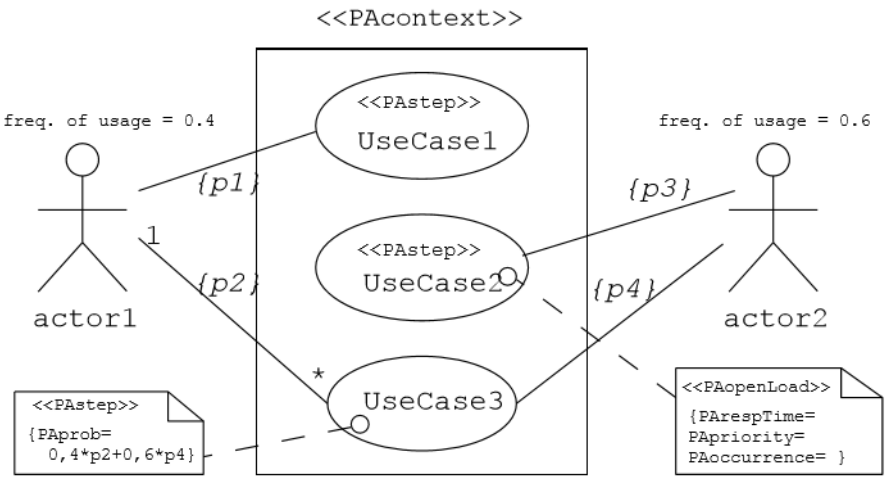


Fig. 1. Use case diagram with performance annotations

issues can be identified and addressed the fewer resources are spent to solve them [7] . The authors of this study hope that the users of the system can use it without any special preparation and studying of the interface. Usability quality of the system should allow architects to use the tool intuitively based on their previous user experience and to be able to get performance evaluation results without spending time trying to get familiar with the system. Otherwise, there is a high possibility that the users may prefer a whiteboard to UML design tool even with wide range functionality. In the early software development stage designers often use whiteboards to create a system design. The project aims to provide designers with the tool that allows them to get an early-vision of the performance of the designed system. Standard whiteboards do not provide means for data processing but Smartboards do. That is why for purposes of our report we Smardboard is needed.

E. Limitations and delimitation

Limitations: Development time in the scopes of this thesis is quite short. Delimitation: At the beginning, it was decided to use a Quality-driven optimization method for systems architectures (AQOSA). Moreover, it is assumed that AQOSA algorithm is correct based on (reference Ph.D.'s thesis). The assumption is that provided results are correct and they are never evaluated in the scopes of this research. [8]

F. Thesis structure

This thesis will include an introduction, related work part where systems and literature that are important for our study will be shortly reflected in chapter 2, chapter 3 contains methodology description, chapter 4 describes results of the study, chapter 5 contains discussion part, chapter 6 includes conclusions, then comes bibliography and appendixes.

II. Related Work and Background

A. SPE

In "Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software"C.U. Smiths and L.G. Williams provide straightforward techniques and strategies that could manage performance in software development process, especially regarding management in the software design stage. In this book, they describe the solutions to such performance issues as SPE, which is a systematic, quantitative method to the cost-effective development of a software system to meet performance requirements of software [9] . It starts in the design stage of the software development lifecycle and provides guidelines for applying performance modelling to help the software product meet its performance requirements. SPE usually uses quantitative methods to identify whether the system satisfies performance as expected. There are several SPE techniques including gathering data, coping with uncertainty, constructing and evaluating performance models, evaluating alternatives, verifying and validating results [9] .

B. Performance Analysis Method

Michel Chaudron introduced a scenario-based method for predicting run-time resource consumption in multi-task component based systems [10] . He also extends the method by offering a system model that is tailored to the domain of real-time applications. [11] In his papers, he explained the logic of the algorithm which predicts the software performance from architecture design. But all the mentioned solutions are about the real-time of performance results, the performance prediction in the early design stage is lacked.

C. Background

1) Performance Analysis Tool: Etemaadia et al. [8] created a Quality-driven optimization method for systems architectures: AQOSA, Figure 2 displays the architecture of AQOSA.

It provides the evaluation and optimization from different aspects of software architecture. The evaluation sub-system component from AQOSA provides the analysis function for the design model. It can analyze the architecture from five aspects:

1) Response time
2) Processor utilization
3) Communication line utilization
4) Safety
5) System cost

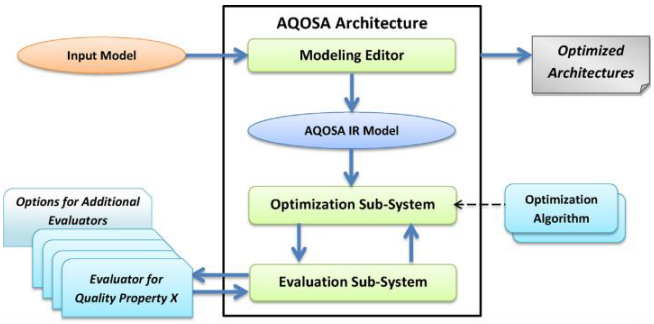


Fig. 2. AQOSA architecture

2) UML Design Environment: The smartboard UML design environment - OctoUML [12] will be used, which is a UML creation tool that allows users to create diagram elements with geometric UML notation or sketched drawings of UML. [12] At present, it only supports class diagrams, they can create a class diagram with OctoUML and save the diagram in the image or XML format. In this research, authors want to analyse the performance from the early design stage. In the scopes of this research, functionality that enables users to create use case-, sequence- and deployment diagrams will be added. Use case diagrams will be used to describe the early design of architecture, sequence- and deployment diagrams will be used to show the details of the system.

D. Similar System

There is a number of tools that allow developers to evaluate system performance, such as the Palladio Approach [13] , and the SPE·ED [14] . They will be introduced as follows:

1) Palladio: Palladio [13] is a software architecture simulation approach, which allows users to analyse system at the model level from the aspects of performance bottlenecks, scalability issues, reliability threats, users can also do subsequent optimisation with their works. Figure 3 shows the simulation Palladio system.

Palladio supports a broad range of analysis scenarios, each scenario can be analysed with performance, reliability, maintainability, and costs. Users can choose the most suitable scenarios in different design cases.

2) SPE·ED: SPE·ED [14] is a performance analysis tool which supports the SPE methods and model. It uses performance models to provide data for the quantitative assessment of the system performance. Users can easily construct software performance models with SPE·ED and get the analytical results for these models which will include end-to-end response time, the elapsed time for each processing step, the device utilization, and the amount of time spent at each computer device for each processing step. Figure 4 shows the simulation SPE.ED system.

III. Methodology

The research method will be used in this paper is Design Science Approach, Design Science tries to extend the boundaries of human and organisation capability by creating new and innovative artifacts. [15] . Figure 5 shows a framework of a research of an information system. An artifact will be developed based on the business requirements and applicable knowledge. After development, the artifact will be assessed by the evaluation method, based on results of evaluation artifact can be refined.

A. Artifact Design

An artifact is the result of Design Science Research which is created to address an important organizational problem [15] . It is the key factor of the method. In this research, the artifact is a new architecture evaluation method, which integrates a performance prediction method into the Smartboard UML design environment. Architects can get the result of performance in the architecture design process much easier.

1) System Framework: Figure 6 describes the framework of our solution. At first, OctoUML will be used to create a Performance Diagram Model which will be introduced in the following section. Then a Model Editor will translate the Diagram Model to AQOSA Model as an XML data structure. After that, the Evaluation System of AQOSA will evaluate the Model and get the result data. At last, the result data will be sent back to OctoUML and showed in diagram pane.

2) Software Engineering Performance Process: Figure 7 shows the general process of SPE [1] . This process consists of 6 steps. The first two steps are not related to this research, so they will not be discussed in this paper. The third step identifies critical use cases which can be implemented in the use case diagram. In the fourth step performance scenarios can be implemented in the scenario diagrams (such as a sequence diagram or a state machine diagram). In our case sequence diagram will be used. The fifth step - resource requirements can be implemented via giving the measurable performance attribute to the components. The last step is to design hardware infrastructure, it can be implemented with deployment diagram. So in this research use case-, sequence- and deployment diagram will be used. And the measurable performance values will be added to each diagram for a performance model.

3) Performance Model: One example of the performance diagram model is showed in Figure 8. The steps of creating a performance diagram are following:

1) Create a use case diagram, and give a frequency for each use case.

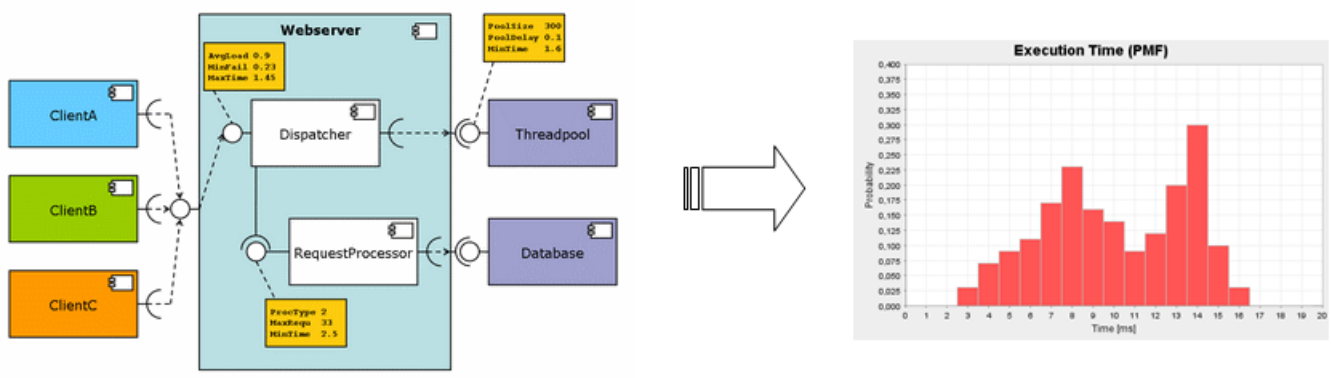


Fig. 3. Palladio Simulation [13]

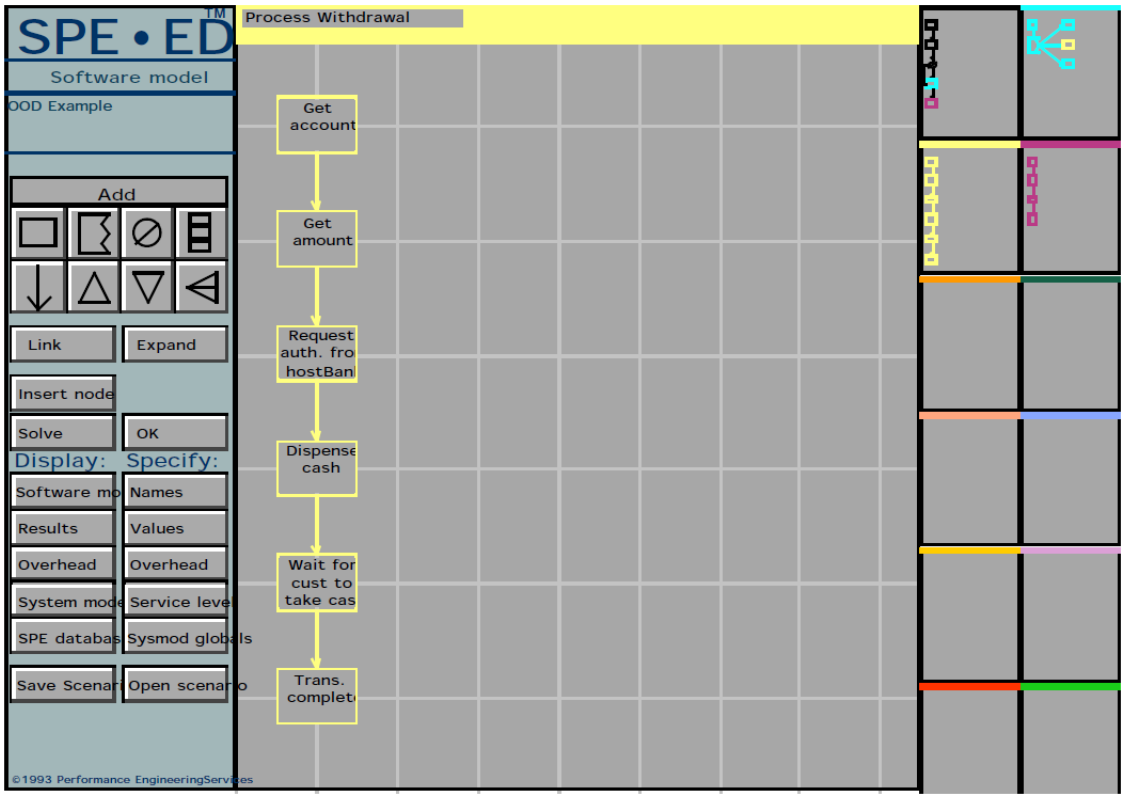


Fig. 4. SPE·ED Simulation [14]

2) Based on each use case, develop the related sequence diagram, specify memory and CPU cycles for each component and give a network bandwidth for each message line.
3) Create a deployment diagram, and get the sum claims per resource in both processor and network.

In our system, we make a performance model by creating a new diagram view which is called "Performance Analysis Diagram", Figure 9 is an example of how performance model can be implemented. Rectangle with images of sequence- and deployment diagrams are used as links. The users create such links in Performance analysis diagram and then load them (Figure 10), as a result a new tab for a sequence or deployment diagram is opened, the users need to save the diagram after it is finished(Figure 11). When the users finish every diagram in the performance model, they can analyse the performance result for the whole system(Figure 12).

4) Model Translation: After creating of Performance Model, our system will transfer the Performance Model into an AQOSA Model to fulfill the requirements of AQOSA evaluation system. AQOSA architecture modelling is defined by the Eclipse EMF. Figure 13 represents a simplified view of AQOSA metamodel. It consists of four major parts: Assembly, Scenarios, Repository, and Objectives [8] .

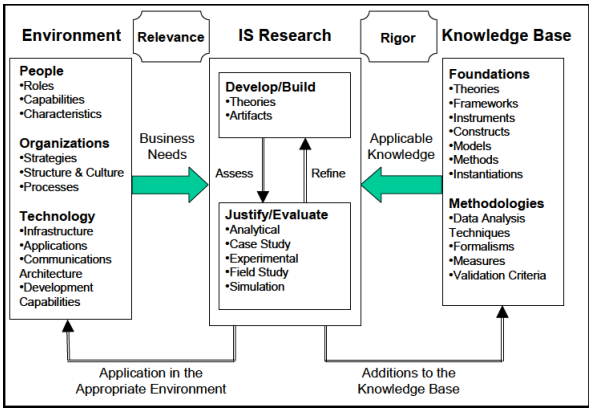


Fig. 5. Information Systems Research Framework [15]

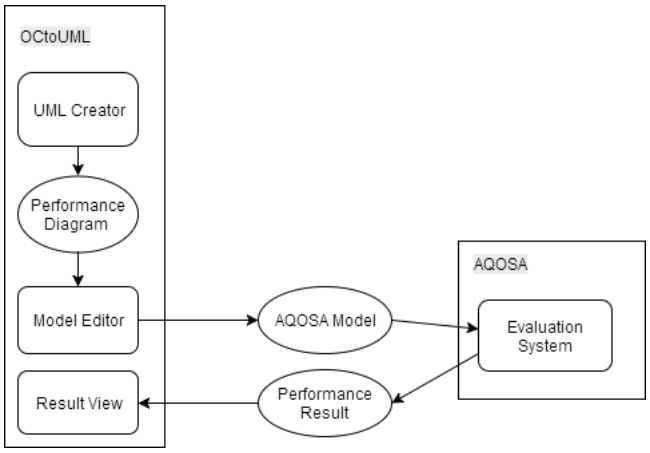


Fig. 6. Prediction System Framework

- Assembly: This part includes software components and their assembly for delivering system functionalities. The data for AQOSA model will come from input made for activation boxes and messages in the sequence diagram.
- Scenarios: This part contains scenarios and use case data, together with their input values. All this data is used for the translation into AQOSA model.
- Repository: This part includes the information from the software and hardware components, including processors, buses, and components implementation. In this part data required for AQOSA model is collected from the objects and objects input from deployment diagram and sequence diagram.
- Objectives: This part defines the objectives needed to be evaluated. Since only performance is evaluated in the scopes of this study, results will contain only Response time, Processor utilization, Communication line utilization.

Appendix D is an example of AQOSA model structure which is created in the performance analysis process, illustrated in Appendix C.

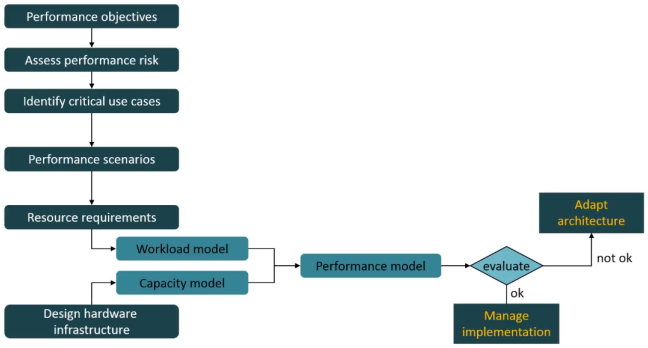


Fig. 7. SPE Process [1]

B. Evaluation of the Artifact

Evaluation is a crucial component of the research process, the research artifact must be rigorously demonstrated via well-executed evaluation methods [15] . In this paper, the adopted evaluation method is User Study [16] . User Study is an evaluation method that focuses on user-centered design. By focusing attention on the people themselves, user study gets the result from learning about people's activities, how they perform them, and what they need in the way of support [16] . Figure 14 shows the process of the User Study, the analysis and model-building process after data collection.

The data which is used in the paper is qualitative data related to the usability of the system. The User Study evaluation approach collects data via three methods: interview, observation, and questionnaire [16] . All of them are implemented in this work. The participants are required to have some architecture design experience. 10 participants were engaged in this research: 1 Lecturer with Ph.D. and 9 bachelor students. All of the bachelor students have passed the course "Technical analysis and design" and "Software architecture". The whole data collection process for each participant will take approximately 30 mins, each method will take around 10 mins to collect the data. The process of data collection are introduced as follows:

1) Interview: During the interview the researchers ask several questions related to the topic of the research and interviewee's experience. The dialogue between the researchers and the participants is guided by the questions [17] , the questions are designed by the researchers. There are three kinds of interviews: unstructured, semi-structured and fully structured interviews [17] . The semi-structured interview are used in this study. Several fixed questions are asked at the beginning of the interview as following:

1) Are you experienced in software architecture design? (Scale: 1 Novice - 5 Expert)
2) Are you experienced with Software Performance Engineering? (Scale: 1 Novice - 5 Expert)
3) Have you ever used Smartboard before?
4) In which stage of development you first evaluate performance?
5) Do you have experience in developing systems where

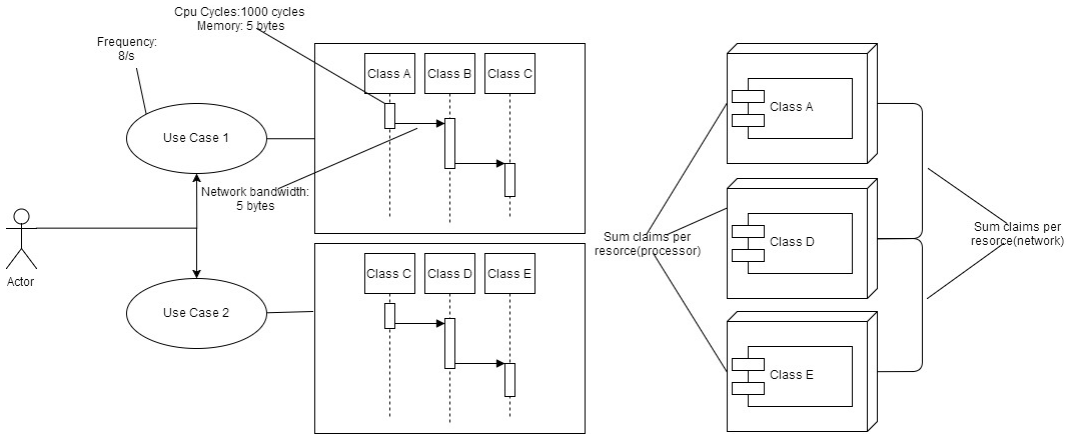


Fig. 8. Performance Model

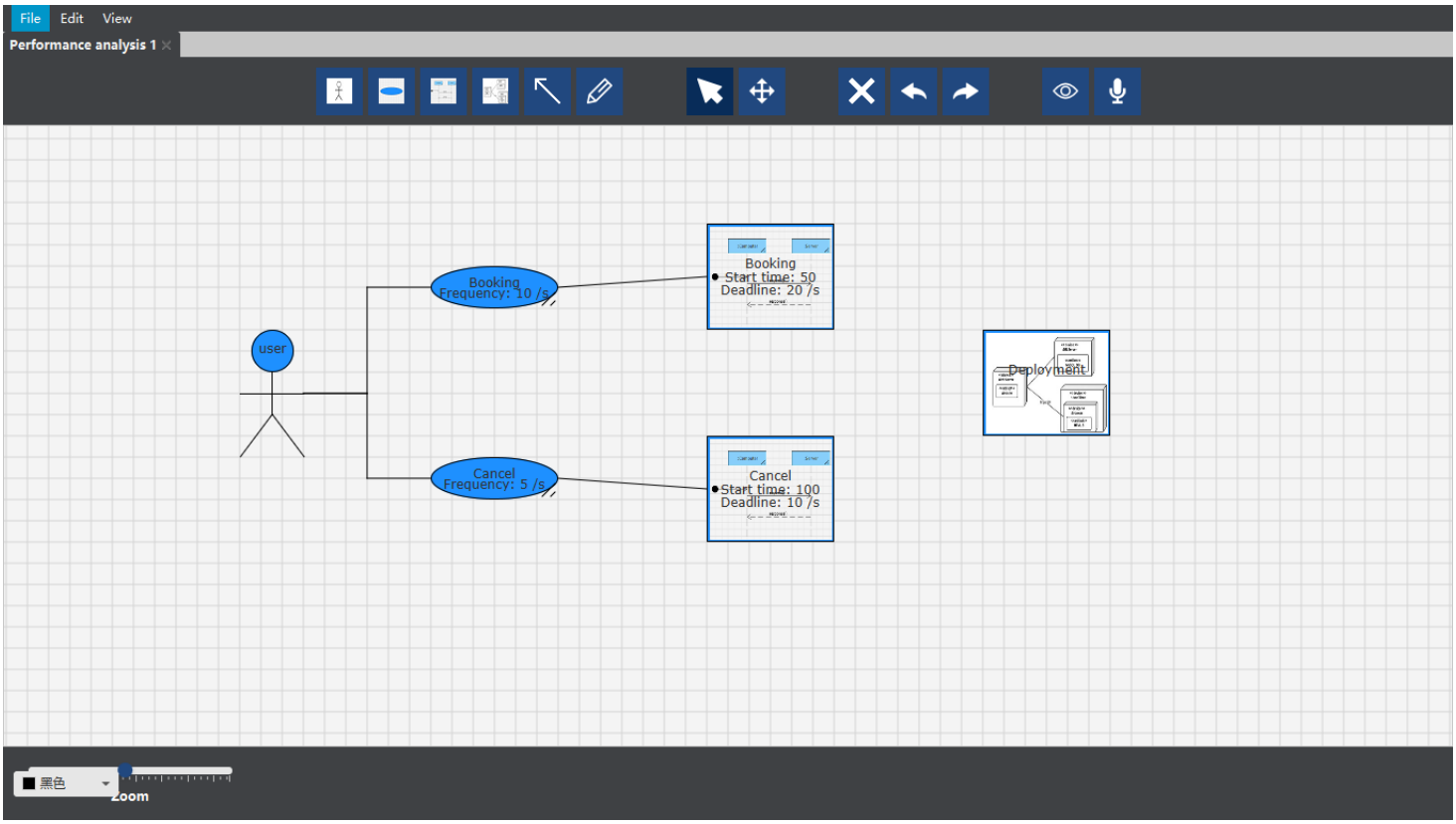


Fig. 9. Performance model view

performance was a critical quality?(if yes - What architectural decisions and in what stage were made to increase performance?)

6) What tools do you use to evaluate performance?
7) Do you think decisions made in the early design stage significantly affect system performance results? (Scale: 1 Not important - 5 Important)
8) Do you think that performance evaluation functionality within UML design environment will be helpful for architects? (Scale: 1 Not help at all - 5 Helps a lot)

After the participants answered the prepared question, the researchers would have an open discussion with them. the content of the discussion depends on the participants' answers. For example, if the participants have totally no experience in SPE some knowledge about SPE process will be given first. Then the researchers introduce the whole system in a very detailed way. If the participants already have some SPE knowledge, the researchers can simply introduce the system.

2) Observation: Observation is used for investigating how a certain task is accomplished by software engineers [17] .

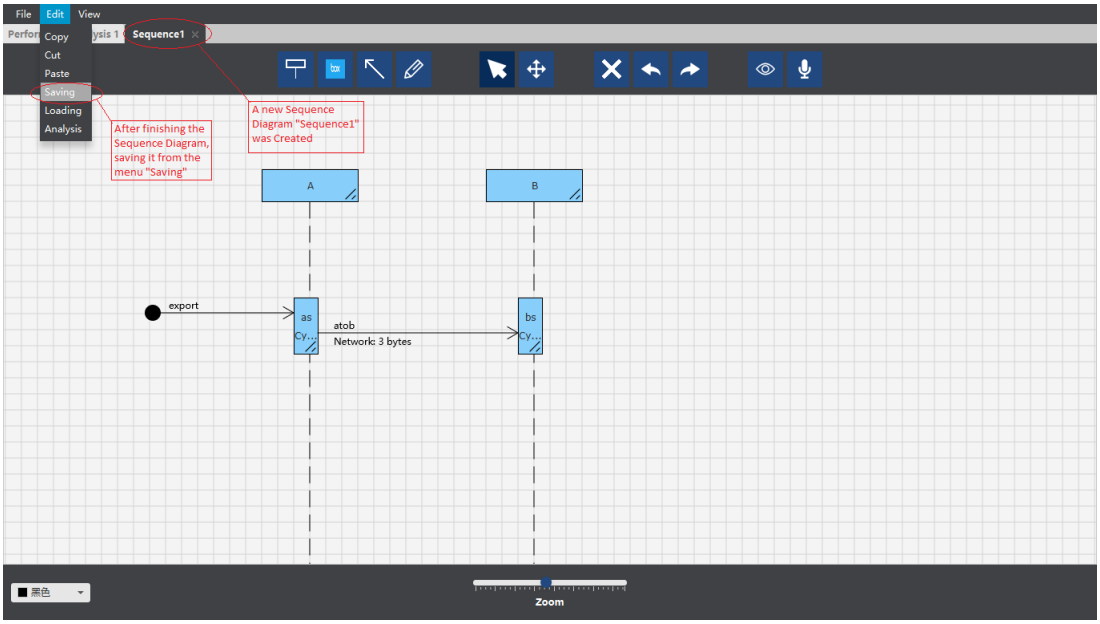


Fig. 10. Loading a new diagram from Performance Model View

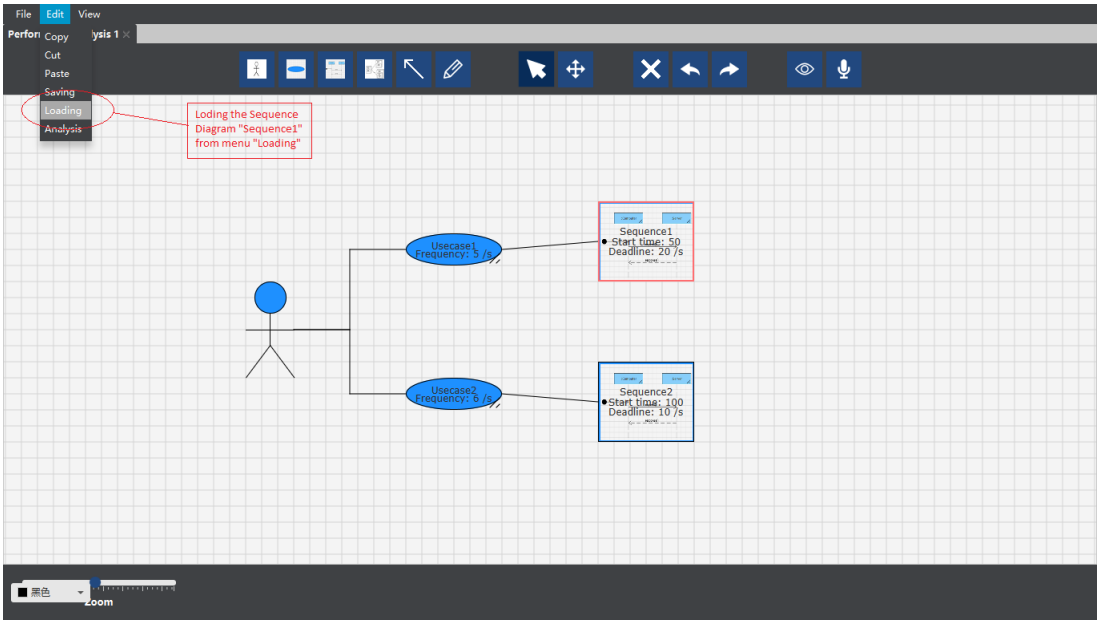


Fig. 11. Saving a Created diagram

In this study the task is to design a performance model of a specific system and to get its performance result. The participants get a paper with requirements of some system, they should design the performance model based on the requirements. They can input all the performance attributes. Their whole process is recorded using a screen recording programme. This videos are evaluated as our collected data.

3) Questionnaires: System Usability Scale(SUS) is implemented in the questionnaire for testing the usability of our system. SUS provides a "quick and dirty", reliable tool for measuring the usability [18] . It consists of a questionnaire with 10 questions. The participants should give 1-5 credits for each question from "strongly agree" to "strongly disagree". 10 questions from the questionnaire are shown in Figure 15.

The result of SUS is calculated and collected to help the researchers to evaluate the usability of the system and identify changes that need to be done to improve it.

IV. Results

A. Observation

Researchers observed participants during the task that was assigned to them. All participants were interviewed separately

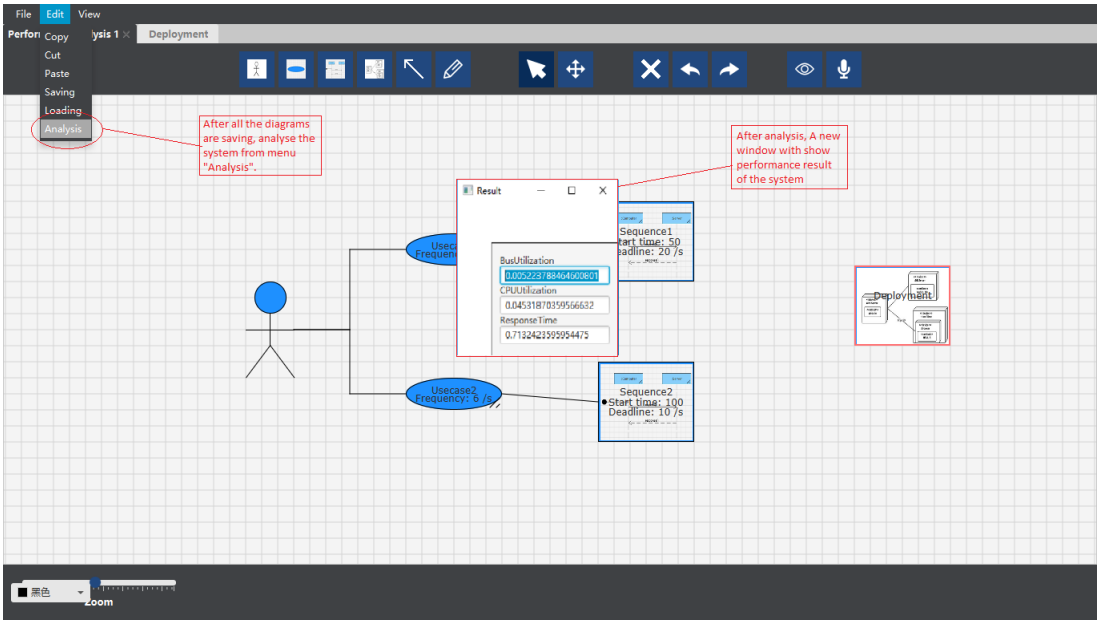


Fig. 12. Analysis in the system

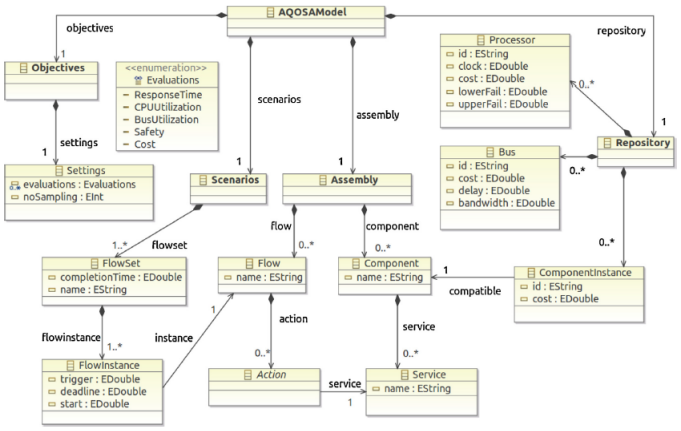


Fig. 13. AQOSA model [8]

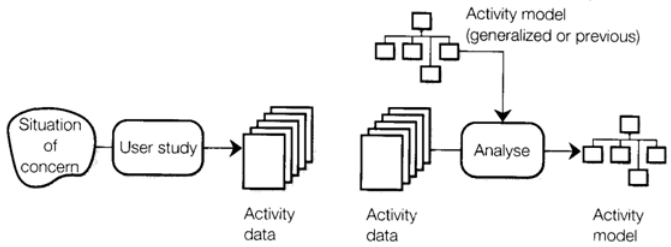


Fig. 14. The process of the user study and the analysis and model-building [16]

The task was done by one participant at a time. It is important to mention that there were no time restrictions to complete the task. The reason for this is there are certain steps needed to be done in the system to complete performance analysis. Moreover, time spent on accomplishing the task was regarded

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Fig. 15. System Usability Scale(SUS) [18]

as a usability metric. It was taking the users from 15 minutes to a half an hour to finish the task.

All interviews started with questions about the experience. The participants were asked about their experience in software architecture (SA) and SPE, experience in evaluating performance, tools that they used for it and practises they used to improve it. Also, they were specifically asked whether they had worked with the Smartboard before. After that presentation about the system's work was shown. The presentation was preferred to live demonstration because there are a lot of details important for work of the system that needed to be explained. Our system requires a lot of data to input which is essential for the performance analysis. It was possible to add supplement text with explanation for the required input in presentation materials There are several observations made during interviews and after analysis of recordings made during interviews. First of all, most of the participants are not familiar with the Smartboard, and one of them had big difficulties to work with the Smartboard. Secondly, almost all participants had problems with the feature that allows you to select several objects at the same time. Thirdly, it was difficult for all of the participants to notice a new tab which is created after loading

sequence- or deployment diagram link. Creating an object for the first time most of the users tried to drag it from the top bar. At the beginning interviewees also had problems with creating a connection between objects. Important to mention that after figuring out the right way to create objects and connections between them participants did not repeat this mistake again. During observation of the user study, the most common problems were multiple selection and loading of a new tab for sequence- and deployment diagrams. The participants usually did not notice that more than one object was selected. As a result, when they wanted to relocate one object, all selected objects were moved, and users were very confused. If they wanted to load a link to sequence diagram and something else was selected, loading of a new tab failed. In this case, participants could not identify the reason without our help.

B. Results of the interviews

This section contains the data which is collected during the interviews. There are two parts in this section. First, the participants' information, including participants' SA and SPE experience, and their experience in using Smartboards. Second, the participants' opinion about performance evaluation, including the stage in which they first evaluate performance. The participants were also asked about their view on the importance of decisions made in the early design stage and performance analysis functionality within UML design tool.

1) The participants' experience: Figure 16 shows the participants' experience on SA, SPE, and Smartboard. All participants have some experience in SA, average experience score is 3.3 out of 5. Since most of the participants are bachelor students, they do not have any SPE knowledge, only one participant with Ph.D. has SPE experience. The situation is similar with participants' Smartboard experience, there are only two persons who worked with smartboard before.

There are very few participants who used some performance evaluation tools before. The most common method they used was stopwatch while running a prototype, as well as simple observation. However, some participants used Yakindu [19] and automated tests.

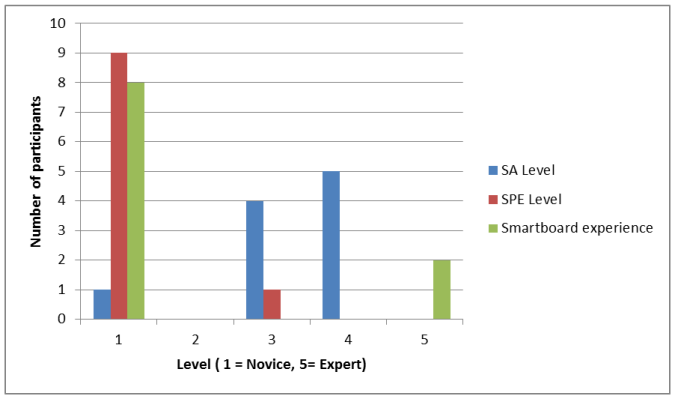


Fig. 16. Participants' experience in SA and SPE

2) The Participants' opinion: Figure 17 illustrates a comparison of the data collected during interviews. The blue bar shows how important decisions made in the early design stage for system performance results. And the red bar shows how important performance evaluation within UML design tool. All of the participants think the early stage performance analysis and UML design environment are important for the performance analysis. The lowest score on these two parts is 3 out of 5. Four participants evaluated the importance of performance analysis in the early stage with 5 out of 5, even though most of the participants do not have any experience in SPE. Most participants believe that the performance evaluation analysis in the early design stage will be helpful for architects. However, some of them still think it will depend on how the method is implemented.

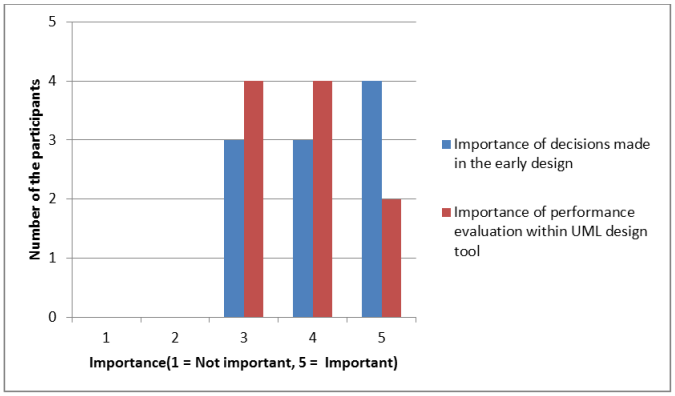


Fig. 17. The Participants' responses on importance of decisions made in the early design stage and importance of performance evaluation within UML design tool

C. Users feedback

It is significant that participants gave very similar feedback for the system. Most of the users mentioned that the system was intuitive and easy to use. Several users used "not overcomplicated" to describe the system. Half of the participants specifically said that they liked the overall look of the system. Several people participants did not like to work with Smartboard, they were sure that if they used the system on a computer they will be more efficient. One person participant liked the way of creation objects. Another participant mentioned that he liked how edges were implemented since objects are connected with the edge, it is bound to these objects. Only one person participant thought that a multiple selection option were useful, all other participants found it very confusing. Moreover, all users participants were not happy with selection feature. One of the participants even mentioned that multiple selection was the biggest problem for him. Most of the participants had offers on how to improve the system's usability. First of all, all participants mentioned that there should be a way to get system's feedback. Thus, users wanted to see information messages if saving was successful, as well as when a new tab was opened. Also, most of the participants wanted to get tips, which could clarify requested

input. In addition, it would be useful to get tips on how to work with a program (for the first session with the system). Some participants noted that it would be useful to have some template for input values, which can be used for several projects. Most of the interviewees stated that icons should be more descriptive, several people offered to add pop-up text supplement for each icon, others thought that better icons should be used. One of the participants advised us to add snap to grid feature, which allows users to align all objects on the diagram.

D. System Usability Scale results

All the participants filled in the SUS questionnaire after the interview and the task. The results are calculated with the grading system presented in "SUS: a 'quick and dirty' usability scale" by J. Brooke [18] . Based on this research, a SUS result which is higher than 68 will be considered as above average. That means that usability of this system fulfills the users' expectations. If the result is lower than 68, the system needs to be improved. The highest SUS result we got is 82.5, and the lowest score is 42.5, the average score is 63. The result shows that some improvements. are needed. It is significant that there is some dependency between SUS score and the participants' experience. Those who were more experienced in SA and SPE also gave higher SUS score. Besides, we got the lowest SUS score 42.5 from the participant who evaluated his SA experience with 1.0 out of out of 5.0.

V. Discussion

A. How performance analysis feature affected usability of OctoUML

As it was mentioned earlier our thesis is a part of Chalmers/Gothenburg University project which makes research on UML designing tool. OctoUML tool was created in the scopes of Master thesis "Combining Formal and Informal Notations in software design" [12] . Our program is based on OctoUML and developed as its part. In the Master thesis researchers also conducted usability evaluation. They also used SUS, that is why now it is possible to compare our SUS results against their results. This is needed for better understanding of performance analysis usability. Performance analysis part gained 63 SUS points. Original UML creating tool gained 78.75 points. These results indicate that performance analysis component within OctoUML has lower usability level than general OctoUML. There were three aspects that could have a significant impact on SUS results: the complexity of the performance analysis, evaluation task issues, and participants experience. First of all, the users are required to follow certain fixed steps to accomplish performance analysis and get final results. The users should create at least three diagrams and make reasonable input. Moreover, they should not leave any input field empty and save all diagrams, as well as not to forget to create one of the diagrams. Secondly, the task complexity, which is, of course, connected with performance analysis complexity. The main difference between our usability evaluation task and OctoUML usability task is that in our case there is an expected outcome in performance analysis task, which is performance result. In the scopes of our task participants needed to create three diagrams, connect them and make the required input. Of course, it took a lot of time and that is why our task did not have time restrictions, as long as the goal of the task was to complete performance analysis. On the contrary, OctoUML evaluation task was time limited and did not have any outcome. Moreover, the participants were expected to create only one class diagram during 12 minutes. Thirdly, experience of the participants may have had a huge impact on results. In our research only 1 participant had Ph.D., while all others were Bachelor students. In OctoUML research majority of the participants had Ph.D., others were Post-doc and Master students. Moreover, in our research, some special knowledge (for example, knowledge in SPE and SA) would be very beneficial for the participants and lack of this knowledge made understanding of the system and work with it more difficult. For example, only one participant had experience in SPE.

B. Smartboard usability

During the interview more than a half of the participants mentioned that they did not think it is necessary to use a Smartboard to work with the system. Some noted that if they used it with computer it would be easier for them to work with the system. One of the participants was quite experienced with Smartboard and this affected usability score, he had one of the highest results. For now, there is only one version of the system, which can be used on both a Smartboard and a computer. This may be one of the reasons of usability issues we have. In the future, the system could be customized for computer and Smartboard to fulfill different users' requirements for usability on both smartboard and computer side.

C. How software architecture experience affects usability results

The evaluation result shows that the users' SA experience could influence the system usability. The more experienced user will evaluate systems usability higher. For those users who are not so experienced in SA, it is difficult to understand the operation process of the system, especially to define the input values in performance model. Lack of SA knowledge is a big problem for the system usability improvement. The goal is to make this system user-friendly for a wide audience, including junior architects. Most participants felt that they did not really know what they were doing when they input the values. The solution could be the system to auto-generate some suitable values for AQOSA model depending on the architecture design. For example, the system response time can be generated depending on the amount of software components, the processor and bus values. We can also provide some option for input values. For example, we can give several options for the processor and bus types (e.g. processor i7-6700, bus 100M). All the other values are predefined based on processor and bus types. Users just need to choose the suitable processor and bus type for their system's design instead of inputting all

the requirement values. That could help people to work with the system and save their time, as a result it would increase usability.

D. Interview and evaluation task

Creating a questionnaire we were thinking about things that could have impact on participants' answers. For example, users experienced in SA, SPE, and Smartboard are expected to have fewer problems working with the system. Moreover, it was sufficient to know how and in what stage participants usually check system performance, what tools do they use for it. It was important to know whether they have experience in developing systems where performance was a critical quality. It was sufficient to choose an appropriate task for evaluation. While deciding on the task several aspects were taken into account. First of all, the system needed to be not very complex. The reason for this is we did not want participants to think about how to design the system. Instead, we wanted them to create a simple system design, as long as the purpose of the user study was to check how difficult it is to work with our system. Otherwise, complex task could have affected observation results. Secondly, for testing performance analysis functionality the system needs to use at least two physical tiers. All participants were asked to perform SUS test and leave the feedback about pluses and minuses of the system, as well as suggestions on how to improve usability of the system. It was important for us to know not only minuses but also sides of the system that the users liked. For example, if the users liked some aspect of our system, it could become a good pattern for the system future improvement.

E. Threats to validity

AQOSA: AQOSA can affect our validity in several ways:

1) At the beginning of this research, we intended to create a connection between physical tiers and software components in deployment diagram. Different software components would be allocated in different physical tiers, depending on an architect's decision. This would help architects to analyze the performance of the system in case it is distributed. However, for now, AQOSA evaluates the whole system by processing data about all available hardware resources and all software components without any assigning.
2) AQOSA was initially developed under the guidance of the mentor of this research. But it was never put into practice in the industry and it did not have that many users. Eventually, there are still potential risks to be verified.
3) Instructions of AQOSA system are not very clear. We could rely only on explanations and examples introduced in the Ph.D. thesis. Thus, the way the system is utilized now may differ from the initial intention of the AQOSA creator.

OctoUML: As it was mentioned earlier performance analysis component was developed as a part of OctoUML system. It took some time to understand the structure of the system's code and the purpose of its classes and functions, as far as the code was not very well commented. We also needed to follow the MVC architecture style and to be consistent with backend and frontend decisions made in OctoUML project before us. Some of these decisions affected our work. For example, the multiple selection feature was implemented in the scopes of OctoUML project, and all participants of performance analysis evaluation had problems with it. As a result, it affected interviewees' opinion about performance analysis usability.

The participants evaluated their experience themselves. The first question to all participants was about their experience in SA. They needed to scale their experience from 1 to 5. Some Bachelor students evaluated their experience higher or on the same level as Ph.D. with previous industrial experience in software architecture. This example leads to a question about validity of the users' answers concerning their experience. It would be more objective to ask participants to describe their experience in SA, and based on their answers to scale them ourselves.

F. Comparison with Palladio and SPE·ED

Our system satisfies the requirements for the performance analysis, as well as Palladio and SPE·ED. Some advantages of our system were identified while studying the documentations of Palladio [13] and SPE·ED [14] and comparing our system to those systems:

- It is easy to understand and use the system, not so much SA experience required for a user to perform an evaluation.
- The logic of our system follows steps of SPE process. It is easy to understand how all required diagrams are connected.
- Our system is open-sourced, and it has good extensibility. If the users have some special requirements for the system, it is easy to add new features.

There are also some disadvantages. It is designed to be used in the early stage of software design. If the users need some deeper functionality from the system. For example, to analyse the performance of the system from class diagram level, they have to choose Palladio and SPE·ED.

VI. Conclusion

A. Conclusion on results

There were several stages in the scopes of this research: First, OctoUML was extended with diagrams, which are essential for early performance evaluation. Functionality that enables users to create use case diagram and deployment diagram was created "from scratch", some elements of the sequence diagram were implemented before, so sequence diagram functionality was completed. For all the diagrams input windows were added for collection the data required by performance evaluation algorithm. Second, OctoUML was integrated with a performance prediction system. Based on AQOSA performance model structure, the model editor was built-in into OctoUML for creating file format which could be

analysed by AQOSA algorithm. A pop-up window was implemented for demonstrating performance results in OctoUML. Third, the User Study was conducted to evaluate usability of the system. One of the participants of the OctoUML evaluation mentioned that, if OctoUML had more functionality it would be more complex for users. Results of our evaluation confirm this assumption. Our SUS score is 63, which is considered to be lower than average (68) . However, the system can perform and almost all users got results of performance evaluation. It is significant that the users left very similar feedback. Some suggestions repeat in a half of all answers we got. As a result, the plan for the future improvement of the system can be prepared.

B. Future work

This section contains some aspects that can be considered for a future research and development:

1) Simplify the analysis process: Firstly, one of the problems of our system is that performance analysis process is too complex for the users. It takes too much time to decide on the input value. In the future developing this process can be simplified. Some input values could be auto-generated depending on the architecture design. That would significantly simplify the analysis process and improve the system usability. Secondly, most of the participants of the study noted that they needed some system feedback, such as information message that the saving was successful, loading was successful and a new tab is open, as well as error messages. Thirdly, there was one very good suggestion from the users for the problem of multiple selection - simply use control button in case when multiple selection is needed, and forbid multiple selection by default.

2) Debugging: Because of the limited time, there are still some bugs in system. For example, if after saving the users make changes, save them and analyze the performance model, sometimes the system will break down with error message. Some other small bugs influencing the usability should be fixed in the future work.

3) Improve the evaluation of distributed-systems: In section V subsection D "Interview and evaluation task", it was mentioned that we want to create a connection between physical tiers and software components in deployment diagram. It could evaluate distributed-systems in a more reasonable way. For this purposes, more research on AQOSA is needed and some new features should be added.

4) Develop a Smartboard version of the system: In section V subsection A "How performance analysis feature affected usability of OctoUML" we discussed the usability of Smartboard. Most of the participants of the use case did not see any necessity to work with the system on Smartboard. Some improvements can be made to customize the system for Smartboards.

REFERENCES

[1] L. Lavagno, G. Martin, B. Selic, S. A. (e-book collection), and S. O. service), *UML for real: design of embedded real-time systems*. Boston: Kluwer Academic Publishers, 2003.

[2] C. U. Smith and J. C. Browne, "Performance engineering of software systems: a case study," in *AFIPS National Computer Conference*, 1982.

[3] C. U. Smith and L. G. Williams, "Software performance engineering: A case study including performance comparison with design alternatives," *IEEE Trans. Software Eng.*, vol. 19, pp. 720–741, 1993.

[4] C. M. Woodside, G. Franks, and D. C. Petriu, "The future of software performance engineering," in *Future of Software Engineering (FOSE '07)*, 2007.

[5] J. Merseguer and J. Campos, "Software performance modeling using uml and petri nets," in *MASCOTS Tutorials*, 2003.

[6] M. R. Chaudron and R. Jolak, "A vision on a new generation of software design environments." in *HuFaMo@ MoDELS*, 2015, pp. 11–16.

[7] L. G. Williams and C. U. Smith, "Making the business case for software performance engineering," in *Int. CMG Conference*, 2003.

[8] R. Etemaadi, K. Lind, R. Heldal, and M. R. V. Chaudron, "Quality-driven optimization of system architecture: Industrial case study on an automotive sub-system," *Journal of Systems and Software*, vol. 86, pp. 2559–2573, 2013.

[9] C. Smiths and L. William, "Performance solutions: A practical guide to creating responsive, scalable software." Addison-Wesleys, 2003.

[10] J. Muskens and M. Chaudron, "Prediction of runtime consumption in multi-task component-based systems," in *Proceedings of 7th ICSE Symposium on Component Based Software Engineering*, 2004.

[11] E. Bondarev, J. Muskens, P. H. N. de With, M. R. V. Chaudron, and J. J. Lukkien, "Predicting real-time properties of component assemblies: A scenario-simulation approach," in *EUROMICRO*, 2004.

[12] C. V. B. Marcys Isaksson, "Combining formal and informal notations in software design: Towards a one-stop software design environment."

[13] R. H. Reussner, S. Becker, J. Happe, R. Heinrich, A. Koziolek, H. Koziolek, M. Kramer, and K. Krogmann, *Modeling and Simulating Software Architectures: The Palladio Approach*. MIT Press, 2016.

[14] C. U. Smith and L. G. Williams, "Performance engineering evaluation of object oriented systems with spe-ed," in *Int. CMG Conference*, 1997.

[15] A. R. Hevner, J. Park, and S. Ram, "Design science in information systems research 1," 2004.

[16] W. M. Newman and M. Lamming, "Interactive system design," 1995.

[17] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[18] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[19] itemis, "Yakindu," https://www.itemis.com/en/yakindu/state-machine/.

Appendix A. Interview Questions and User Study Task

Questions for pre-user study interview:

1. Are you experienced in software architecture design? (Scale: 1 Novice - 5 Expert)
2. Are you experienced with Software Performance Engineering? (Scale: 1 Novice - 5 Expert)
3. Have you ever used Smartboard before?
4. In which stage of development you first evaluate performance?
5. Do you have experience in developing systems where performance was a critical quality?(if yes - What architectural decisions and in what stage were made to increase performance?)
6. What tools do you use to evaluate performance?
7. Do you think decisions made in the early design stage significantly affect system performance results? (Scale: 1 Not important - 5 Important)
8. Do you think that performance evaluation functionality within UML design environment will be helpful for architects? (Scale: 1 Not help at all - 5 Helps a lot)

SUS questions:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

User case task

“You need to create a performance model for a self-driving car which will contain a use case diagram, a sequence diagram, and a deployment diagram. Choose one of the following use cases: follow the lane, park, overtake an obstacle. You need to create only use case diagram for one of this use cases. The system should be allocated on two boards: low level for controlling servo and ESC, and getting data from sensors; high level board for processing.”

Post-user study interview:

1. Name the pluses of the system.
2. Name the minuses of the system.
3. Give us some suggestions for the future improvement of the system.

Appendix B. Users' Answers

Table 1: Pre-users study interview Results from Participants

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Participant 1	3	1	"No"	"The latest Stage"	"Not prioritize performance, increase performance at the last stage"	"Stopwatch, manual tests"	5	5
Participant 2	4	1	"No"	"Test on prototype"	"Yes. designing components, UML designing."	"Stopwatch "	5	5
Participant 3	4	3	Yes	"1st sprint release development stage"	"No"	"Metrics, Mongo DB drivers "	3	4
Participant 4	3.5	1	"No"	"Software architecture design (State machine diagram) "	"Yes"	"Yankindu"	4	3
Participant 5	4	1	"No"	"Use case diagram"	"Yes"	"Yankindu"	4	4
Participant 6	3	1	"No"	"Prototype, testing"	"Yes, Design stage"	"Timing programs, stopwatch"	5	4
Participant 7	3	1	"Yes"	"Prototyping"	"Yes, prototype stage"	"Observation"	4	5
Participant 8	4	1	"No"	"Designing"	"No"	"Observation"	4	3
Participant 9	3	1	"No"	"Prototype"	"Yes, decoupling of sub components (design stage)"	"Stopwatch, program to check time complexity (automated texts), code observation"	5	4
Participant 10	1	1	"No"	"Usually while developing prototype, I test performance of part of the prototype or the whole thing."	"Some experience. The decision was about removing and simplifying components in the design stage of a new system. "	"No tools used. "	5	3

Table 2: SUS score from Participants

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Participant 1	2	2	3	2	3	1	4	4	4	3
Participant 2	3	4	4	2	4	2	5	3	2	3
Participant 3	3	2	4	3	4	2	4	2	4	5
Participant 4	4	3	4	5	5	2	3	2	2	3
Participant 5	3	2	3	4	4	2	4	3	3	4
Participant 6	2	2	4	1	3	2	4	3	3	2
Participant 7	3	2	4	2	3	2	5	2	4	2
Participant 8	3	1	3	2	4	2	4	2	4	2
Participant 9	4	1	4	1	4	1	4	2	4	2
Participant 10	2	3	2	4	4	2	3	4	2	3

Participant 1.
Post-user study interview
Pluses:

- “GUI”
- “There is one performance model view with links to specific diagrams.”
- “Opening new tabs for creating sequence- and deployment diagrams with loading”

Minuses:

- “Performance analysis process is complex”

Suggestions:
-

Participant 2.
Post-user study interview
Pluses:

- “Intuitive, easy to use”
- “Follows UML”
- “Easy to use with a Smartboard”

Minuses:

- “Multiple selection”
- “Have to press an icon to use selection feature”
- “Links to sequence and deployment diagrams”

Suggestions:

- “Add more features to make models more complex“

Participant 3.
Post-user study interview
Pluses:

- “Intuitive”
- “Overall look”
- “Easy to use”

Minuses:

- “Have to work with a Smartboard”
- “Saving and loading buttons location”
- “A lot of information to know”

Suggestions:

- “If it’s one person it’s smarter to do it on computer”

Participant 4.
Post-user study interview
Pluses:

- “Easy to navigate”
- “Intuitive”

Minuses:

- “Have to work with a Smartboard”
- “Saving and loading buttons location”
- “A lot of information to know”

Suggestions:

- “Tips for input needed”
- “Instructions needed”
- “Error messages needed”

Participant 5.
Post-user study interview
Pluses:

- “GUI”
- “The system is intuitive”

Minuses:

- “Wouldn’t use it just for performance analysis”

Suggestions:

- “Error messages needed”

Participant 6.
Post-user study interview

Pluses:

- “You always knew what you are doing”

Minuses:

- “Selection was inconsistent”

Suggestions:

- “Template for input values”
- “Feedback for the user while saving, loading”

Participant 7.

Post-user study interview

Pluses:

- “How connection works (if you move one of the objects connections are reallocated with it)”

Minuses:

- “Loading part was not intuitive”
- “Selection”

Suggestions:

- “Pop up for saving”
- “Easier mechanism for opening a new tab for sequence-, deployment diagrams. If you open a new tab the view should be switched to the new tab“
- “Icons need to be more descriptive (text supplement, create your own system of icons)”
- “Snap to grid (for making it more aligned)”
- “Ability to save all diagrams of performance analysis model”
- “Info about input data (some clarification)”
- “System feedback to the user (while saving, loading)”

Participant 8.

Post-user study interview

Pluses:

- “System is very simple and not overcomplicated”
- “Each feature is very useful”
- “Possibility of selection of several objects”
- “Intuitive”
- “Board is good for group discussion”

Minuses:

- “Too easy for some people who need more formal diagrams”
- “Not specific enough about the input (some fields labels sound too ambiguous)”

Suggestions:

- “Feedback while saving, loading, opening new tabs”
- “Tips for the user who first time uses the system”
- “Pop-up labels for icons”

Participant 9.
Post-user study interview
Pluses:

- “Tool bar was minimal and easy to use”
- “Don’t need to drag and drop to create an object”
- “Intuitive”

Minuses:

- “Selection of several objects”
- “The way how results were presented”

Suggestions:

- “Use ctrl button for multiple selection.”
- “Since you loading a new tab the view should switch to a new tab”

Participant 10.
Post-user study interview
Pluses:

- “Overall look”

Minuses:

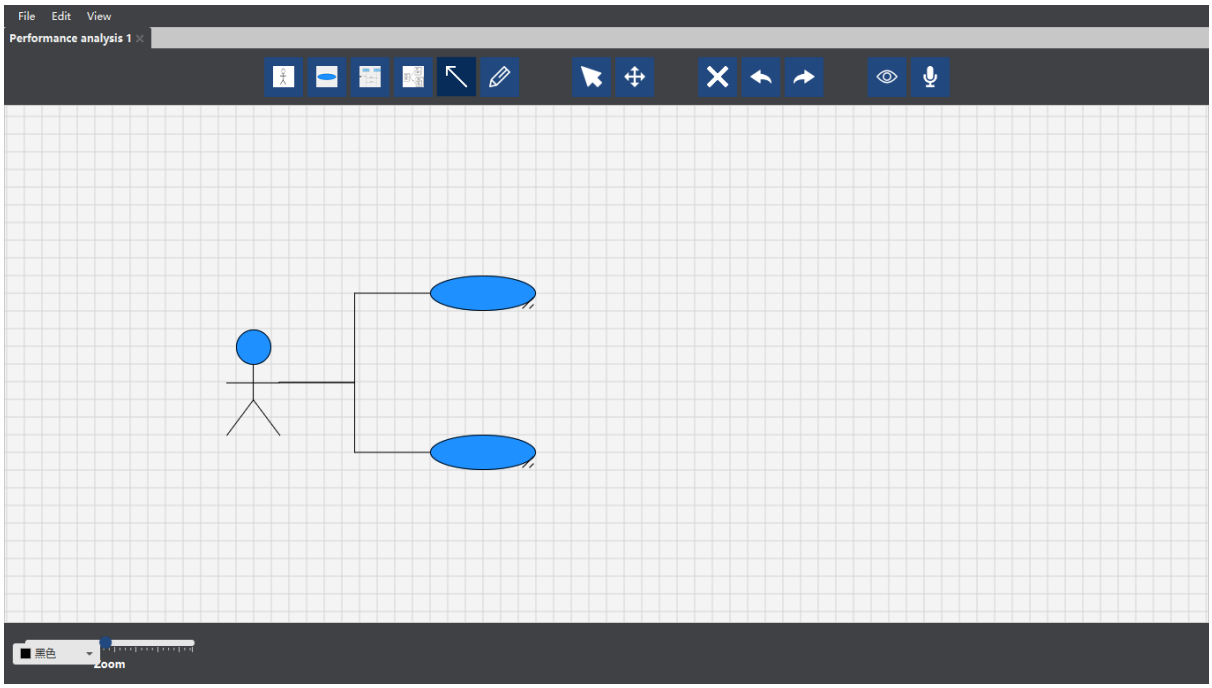
- “Multiple selection”

Suggestions:

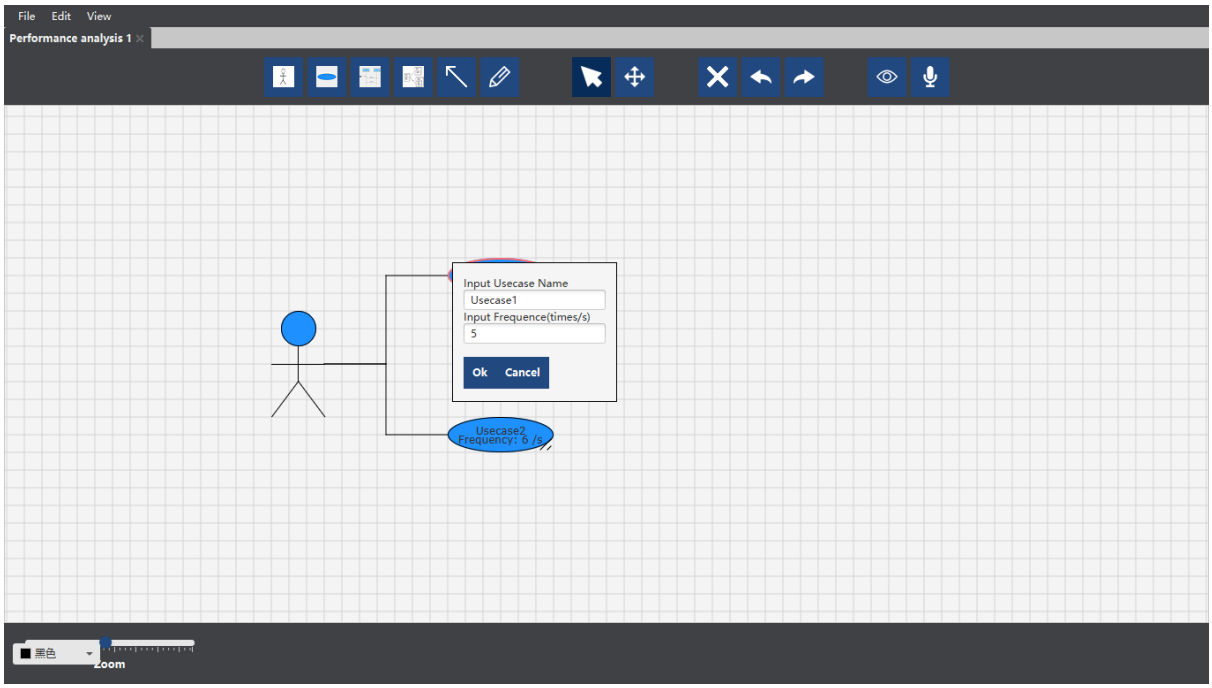
Appendix C. Performance Analysis Process

Software Github address: https://github.com/fk19841217/Performance-Analysis

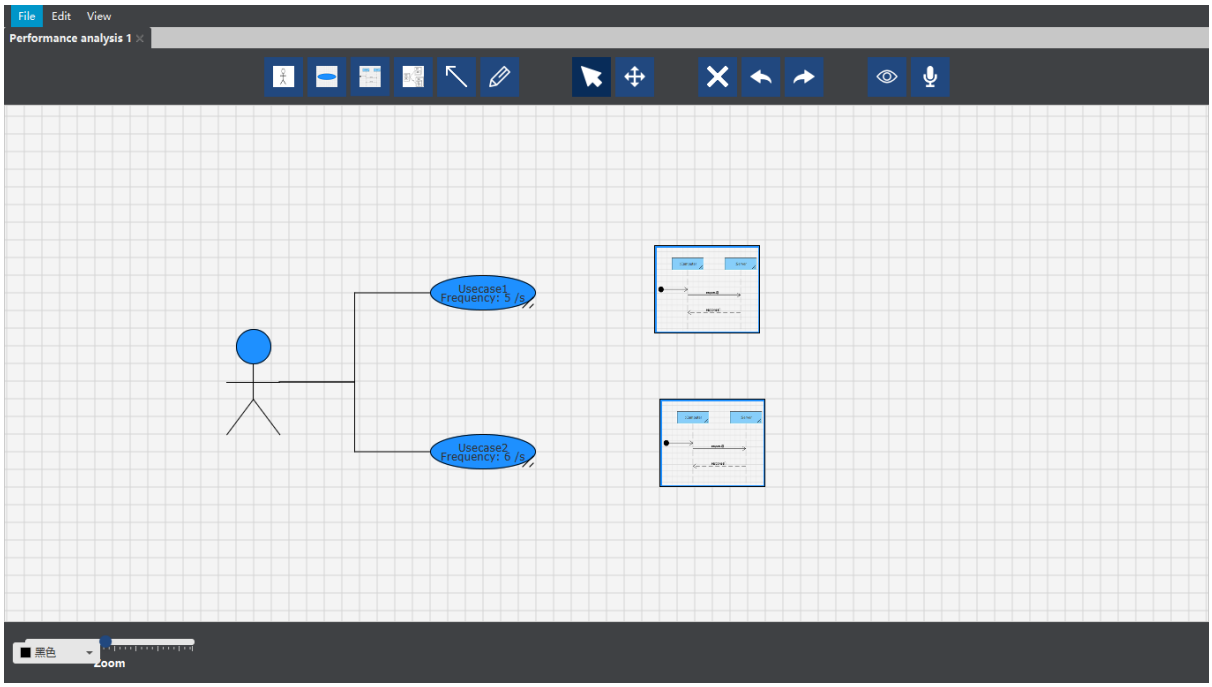
1. Create a use case diagram



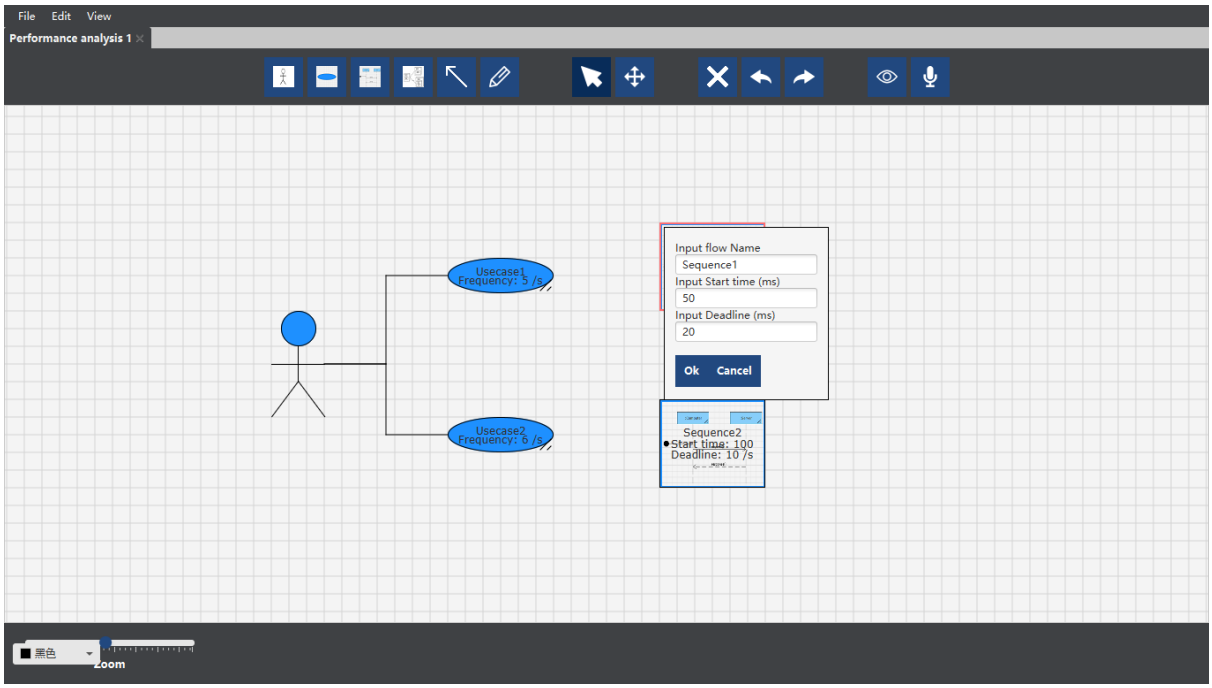
2. Input the frequency value for each use case



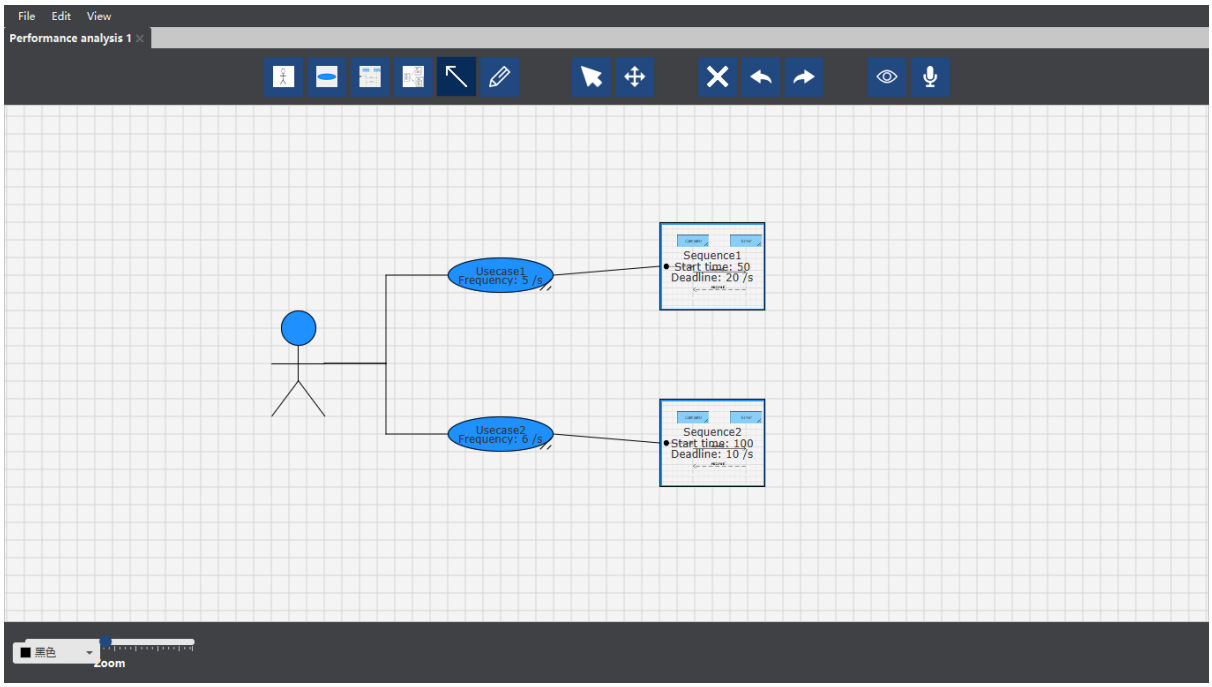
3. Create a link to a sequence diagram for each use case



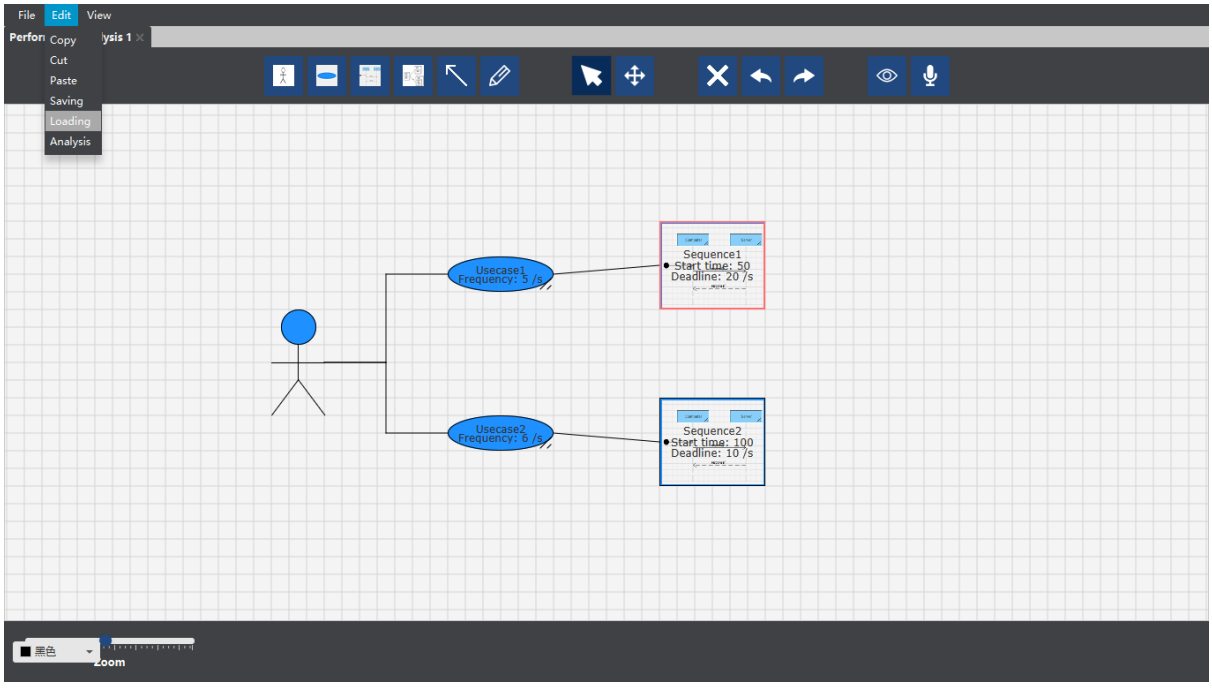
4. Input value for the link to a sequence diagram



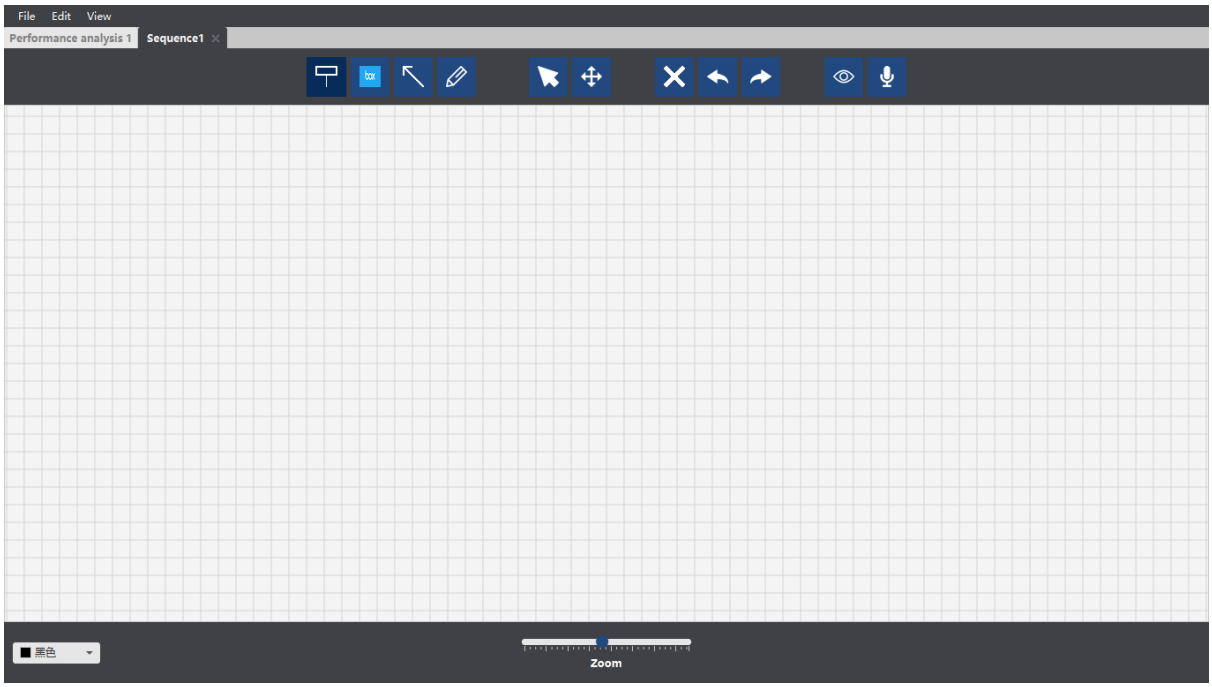
5.Connect the use case node and the link to a sequence diagram



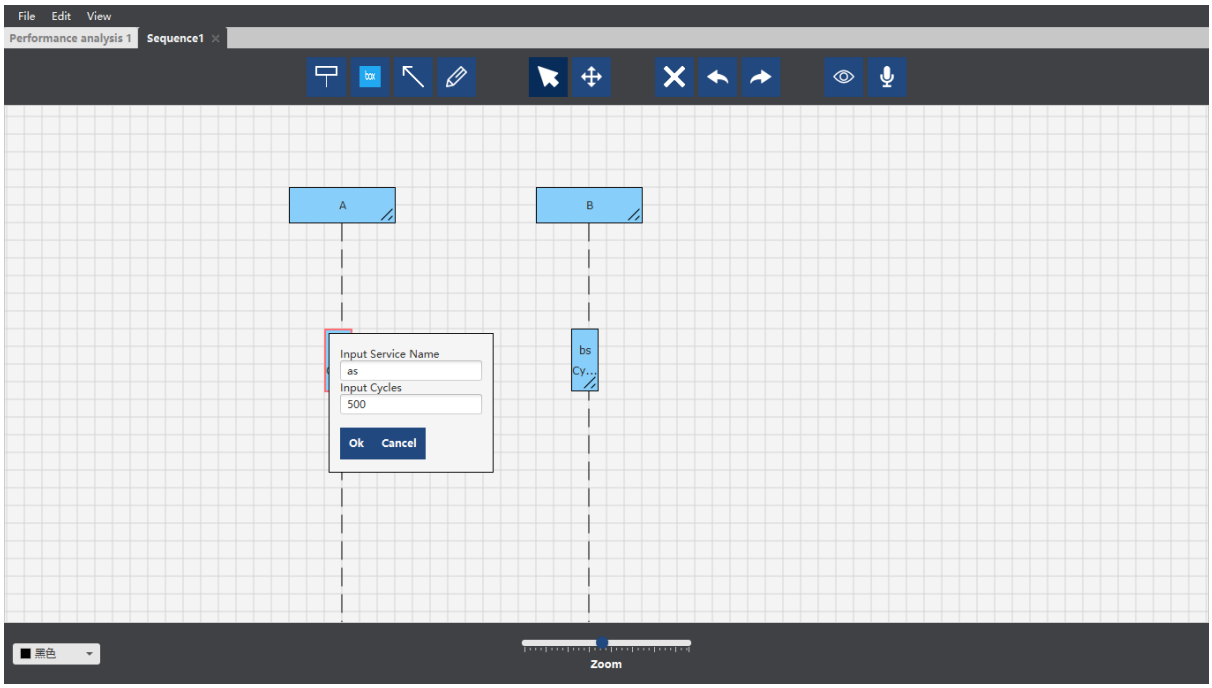
6. Select the link to a sequence diagram “Sequence1” and click “Loading” in the Edit menu

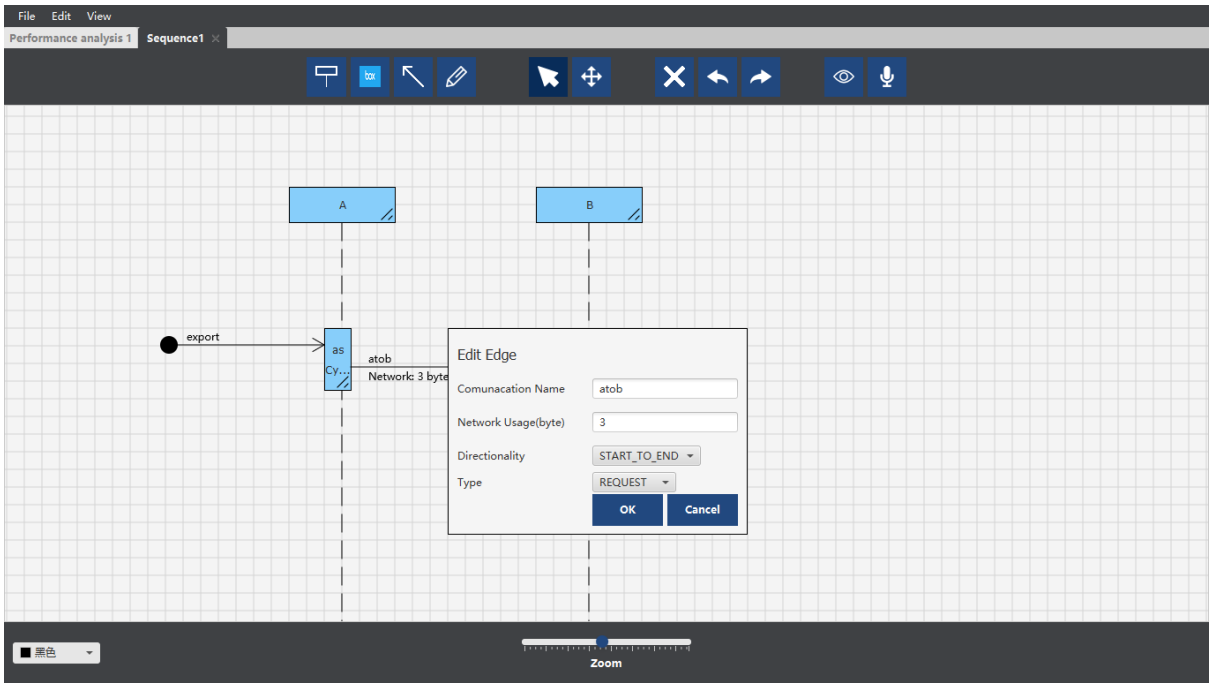


7. A new sequence diagram view named “Sequence1” will be opened for creating sequence diagram

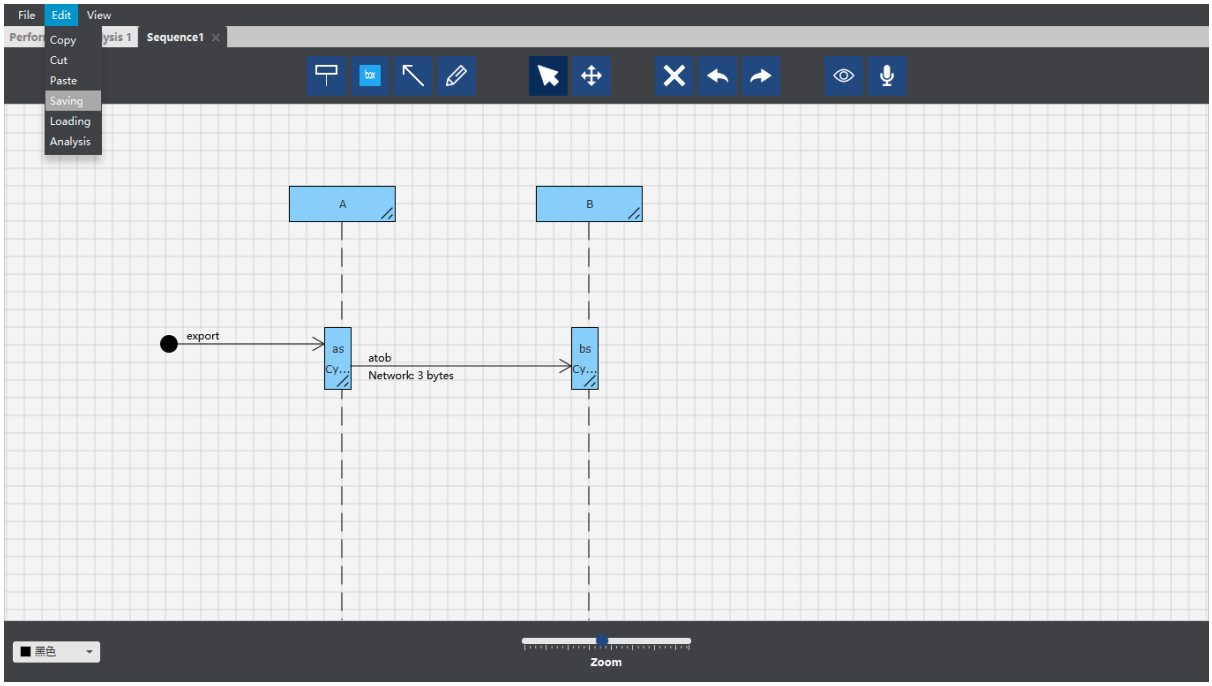


8. Create a sequence diagram, and input value for each node and edge.

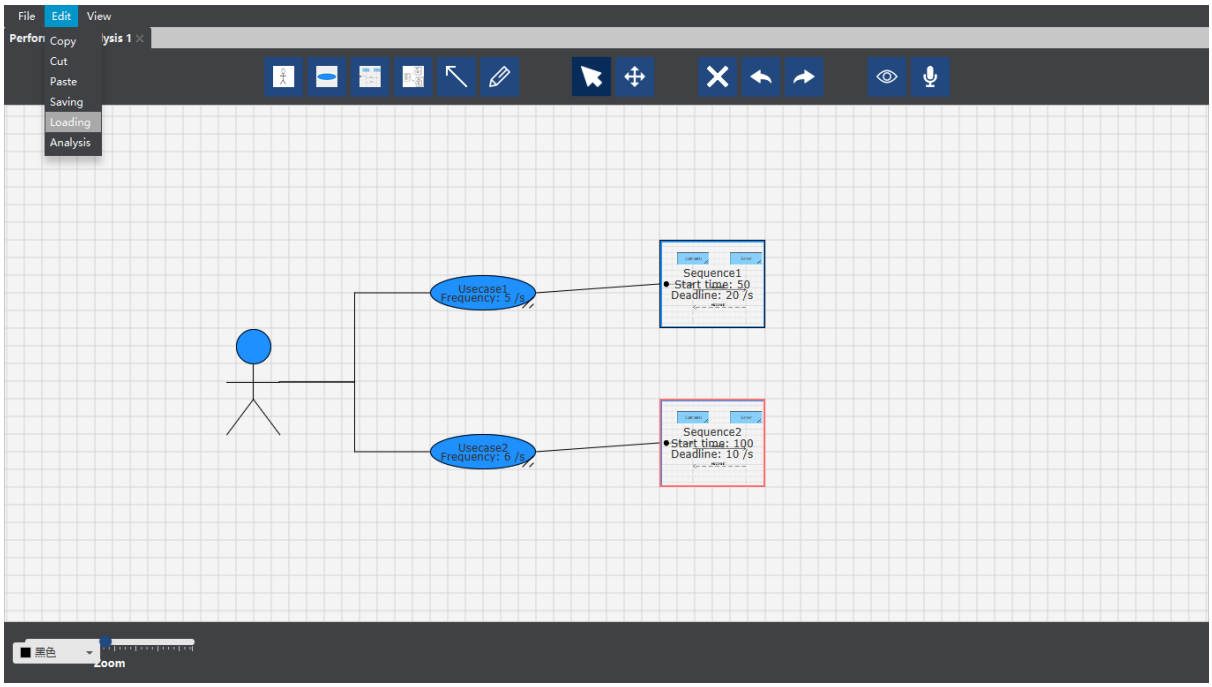


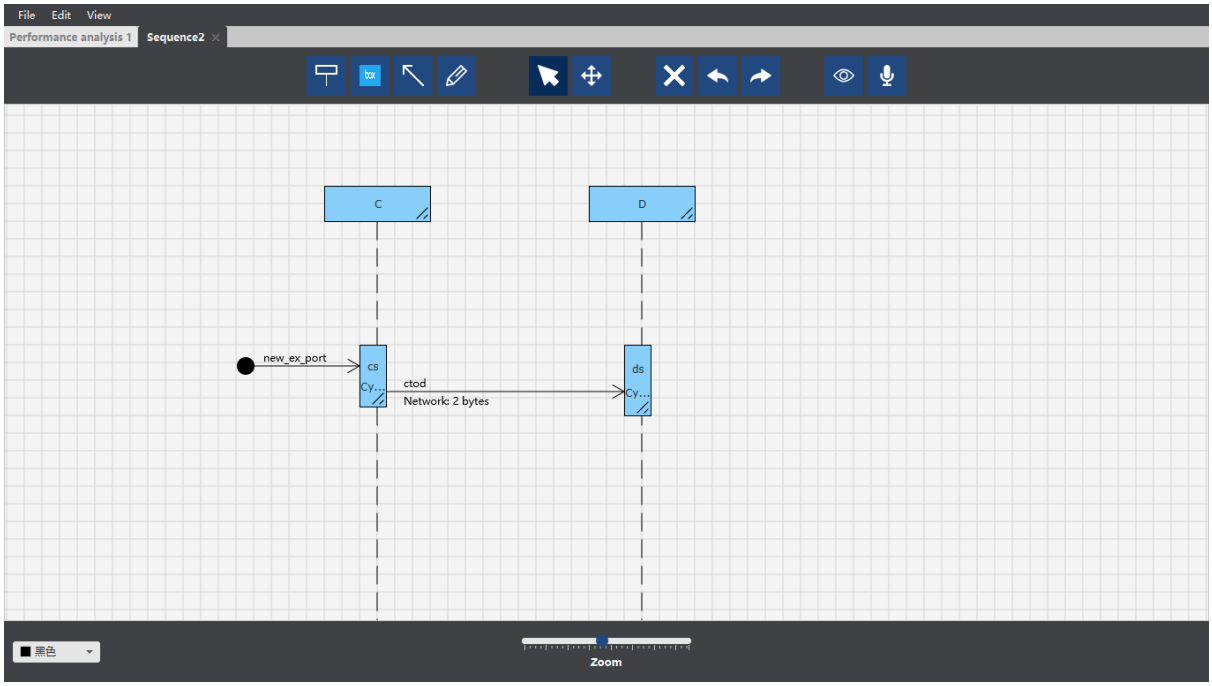


9. After finishing the sequence diagram, click the “Saving” in Edit Menu

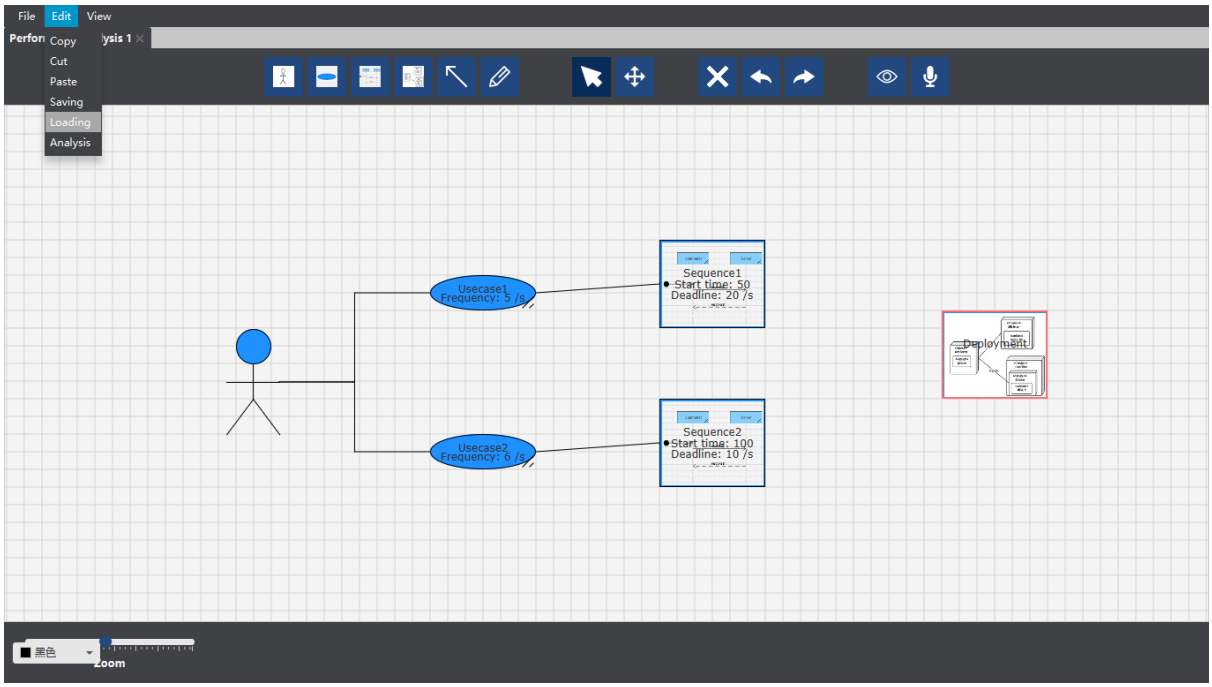


10. Go back to the Performance model view, do the same process for a node “Sequence2”

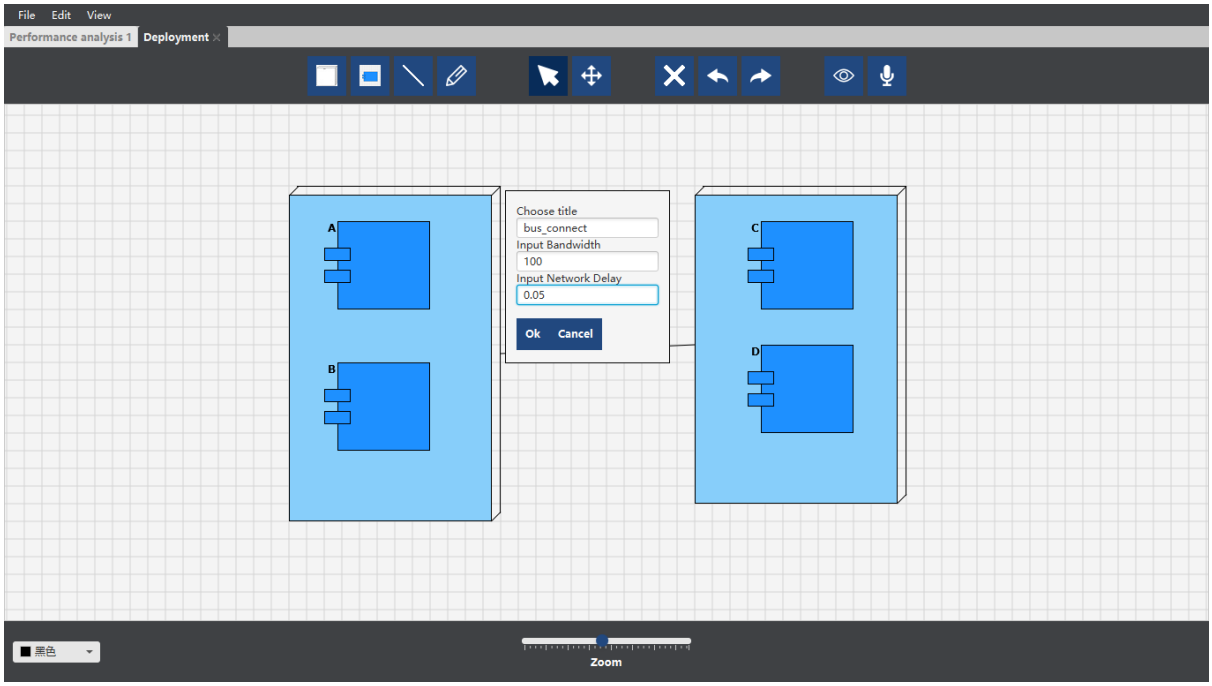


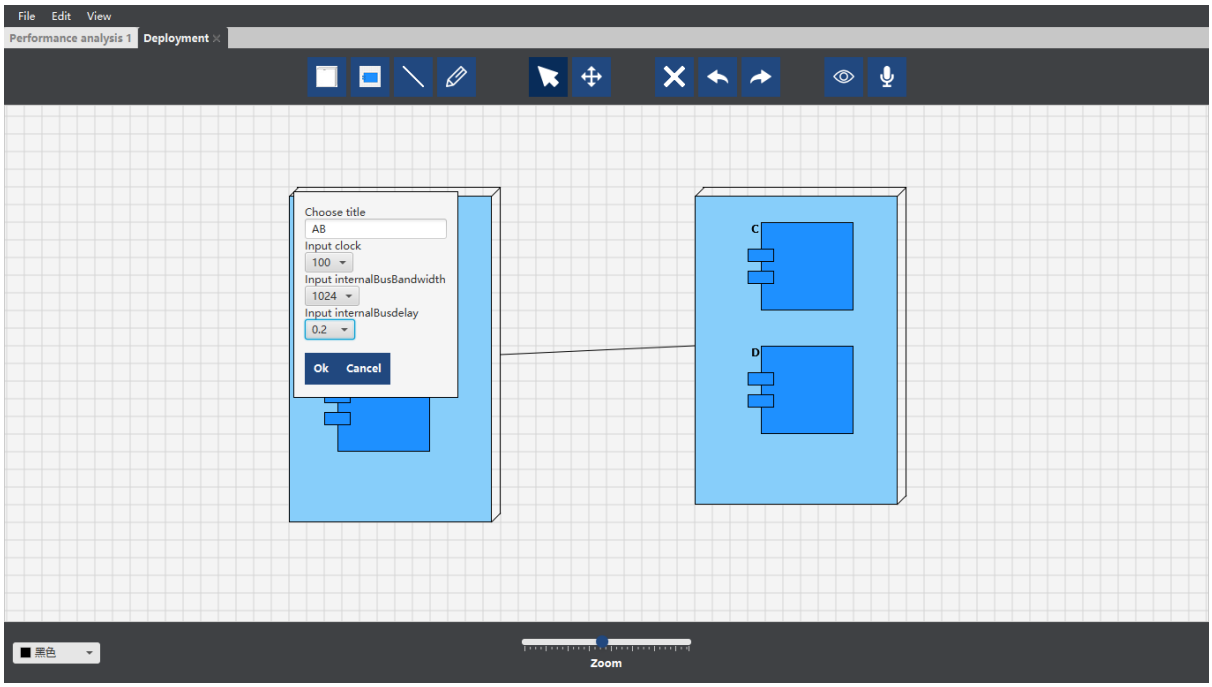


11. Create a link to a deployment diagram and load this node

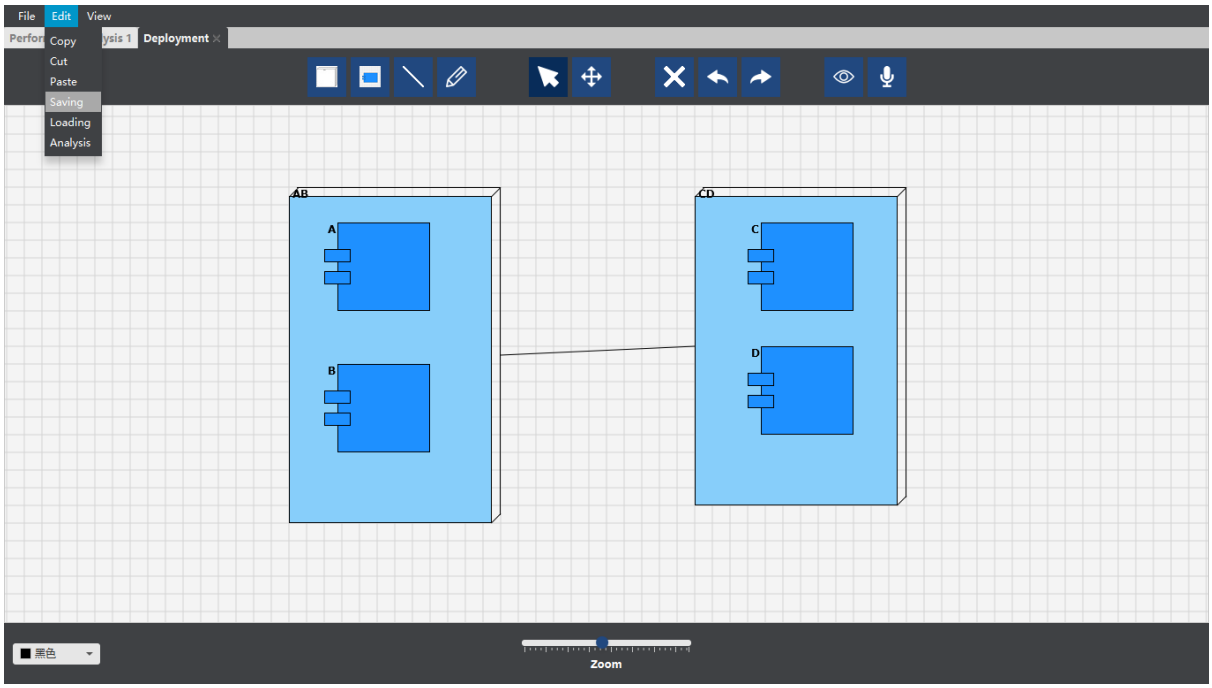


12. Create a deployment diagram in the new view, input value for each hardware node and the bus connection between them

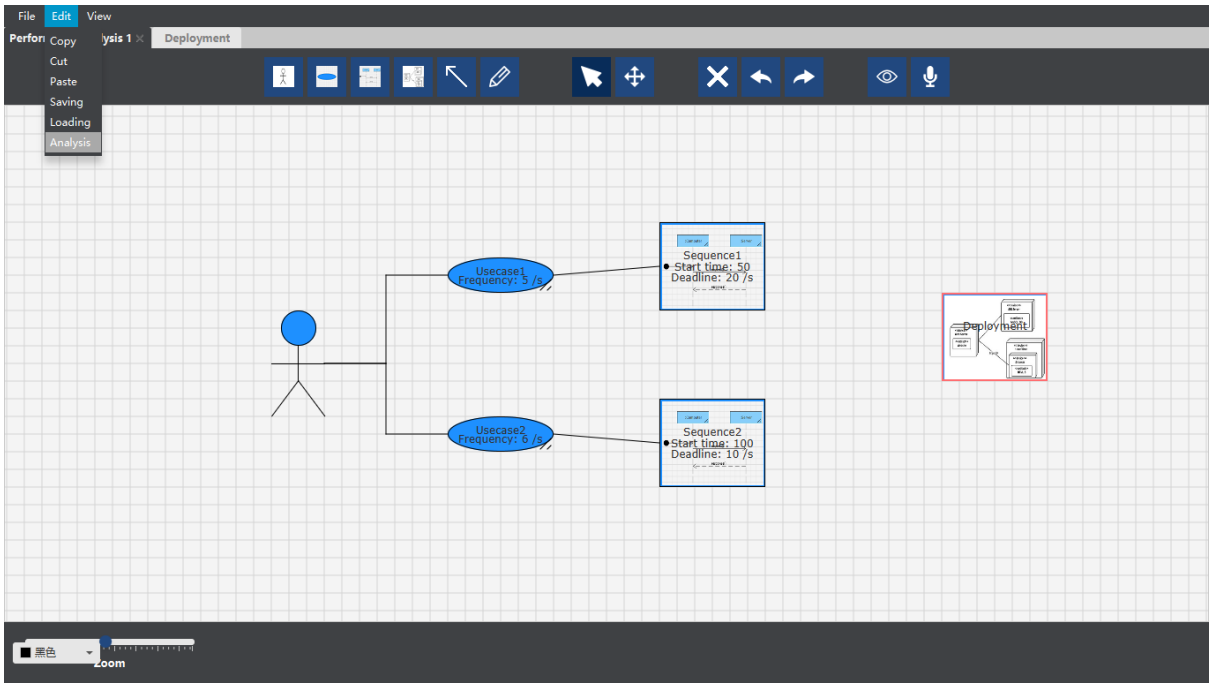




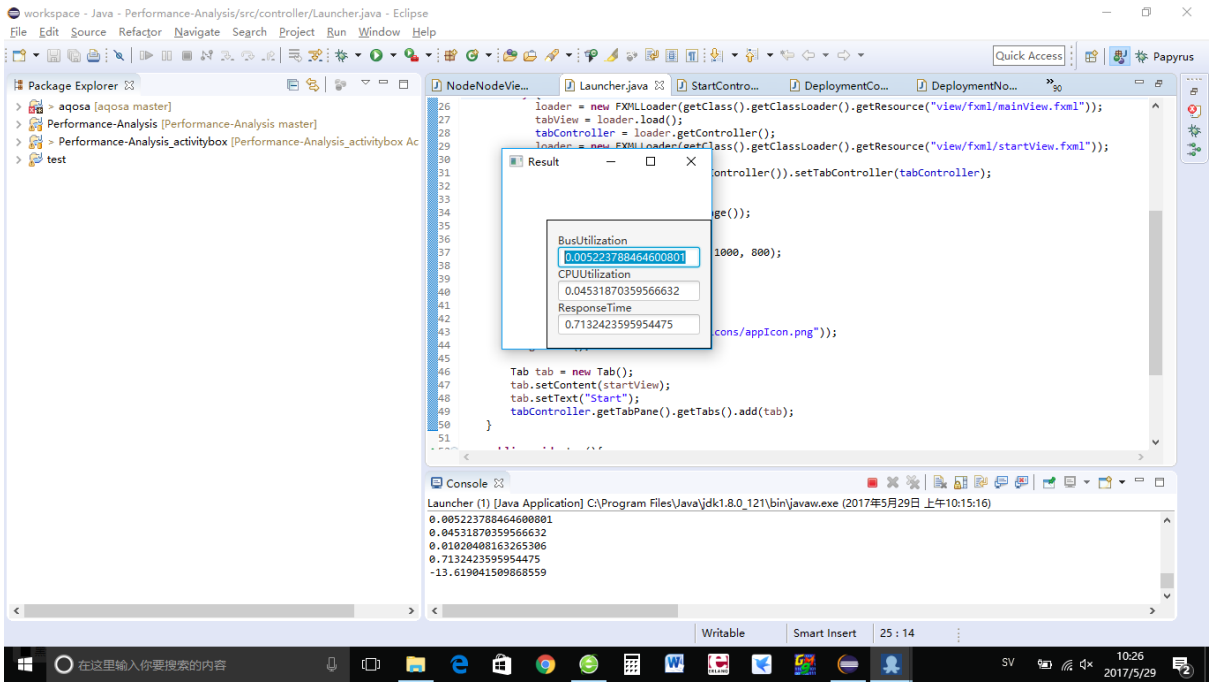
13. Save the diagram after finishing it



14, Go back to the Performance model and click “Analysis” in the Edit menu



15. After a while the performance result will open in a new window



Appendix D. AQOSA Model

```
<?xml version="1.0" encoding="UTF-8"?>
<aqosa.ir:AQOSAModel xmi:version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xmi="http://www.omg.org/XMI" xmlns:aqosa.ir="http://se.liacs.nl/aqosa/ir">
        <assembly>
            <component name="A">
                <service name="as"/>
                <inport name="as_inport"/>
                <outport name="as_outport"/>
            </component>
            <component name="B">
          <service name="bs"/>
                <inport name="bs_inport"/>
                <outport name="bs_outport"/>
            </component>
                <component name="C">
                <service name="cs"/>
                <inport name="cs_inport"/>
                <outport name="cs_outport"/>
            </component>
            <component name="D">
                <service name="ds"/>
                <inport name="ds_inport"/>
                <outport name="ds_outport"/>
            </component>
            <flow name="Sequence1">
                <action xsi:type="aqosa.ir:ComputeAction"
            service="//@assembly/@component.0/@service.1"/>
                <action xsi:type="aqosa.ir:CommunicateAction"
            source="//@assembly/@component.0/@outport.1"
            destination="//@assembly/@component.1/@inport.1"/>
                <action xsi:type="aqosa.ir:ComputeAction"
            service="//@assembly/@component.1/@service.1"/>
            </flow>
            <flow name="Sequence2">
                <action xsi:type="aqosa.ir:ComputeAction"
            service="//@assembly/@component.2/@service.0"/>
                <action xsi:type="aqosa.ir:CommunicateAction"
            source="//@assembly/@component.2/@outport.0"
            destination="//@assembly/@component.3/@inport.0"/>
```

```
                <action xsi:type="aqosa.ir:ComputeAction"
        service="//@assembly/@component.3/@service.0"/>
        </flow>
</assembly>
<scenarios>
        <flowset name="Average" missedPercentage="0.01" completionTime="10000">
                <flowinstance trigger="200" start="50" instance="//@assembly/@flow.0"
        deadline="20"/>
                <flowinstance trigger="166" start="100" instance="//@assembly/@flow.1"
        deadline="10"/>
        </flowset>
</scenarios>
<repository>
        <componentinstance variancePercentage="0.01" id="A_Instance"
compatible="//@assembly/@component.0">
                <service instance="//@assembly/@component.0/@service.0" cycles="0">
                        <depend>
                                <require internal="//@assembly/@component.0/@inport.0"/>
                        </depend>
                </service>
                <service instance="//@assembly/@component.0/@service.1" cycles="500"
        networkUsage="3000">
                        <provide connects="//@assembly/@component.0/@outport.1"/>
                        <depend>
                                <require external="//@repository/@externalport.0"/>
                        </depend>
                </service>
        </componentinstance>
        <componentinstance variancePercentage="0.01" id="B_Instance"
compatible="//@assembly/@component.1">
                <service instance="//@assembly/@component.1/@service.0" cycles="0">
                        <depend>
                                <require internal="//@assembly/@component.1/@inport.0"/>
                        </depend>
                </service>
                <service instance="//@assembly/@component.1/@service.1" cycles="600">
                        <depend>
                                <require internal="//@assembly/@component.1/@inport.1"/>
                        </depend>
                </service>
        </componentinstance>
        <componentinstance variancePercentage="0.01" id="C_Instance"
compatible="//@assembly/@component.2">
```

<service instance="//@assembly/@component.2/@service.0" cycles="200" networkUsage="2000">
<provide connects="//@assembly/@component.2/@outport.0"/>
<depend>
<require external="//@repository/@externalport.1"/>
</depend>
</service>
</componentinstance>
<componentinstance variancePercentage="0.01" id="D_Instance" compatible="//@assembly/@component.3">
<service instance="//@assembly/@component.3/@service.0" cycles="100">
<depend>
<require internal="//@assembly/@component.3/@inport.0"/>
</depend>
</service>
</componentinstance>
<processor id="CD-h" upperFail="0.03" lowerFail="0.015" internalBusDelay="0.15" internalBusBandwidth="2048" cost="100" clock="50"/>
<processor id="CD-l" upperFail="0.025" lowerFail="0.01" internalBusDelay="0.15" internalBusBandwidth="2048" cost="140" clock="50"/>
<processor id="AB-h" upperFail="0.03" lowerFail="0.015" internalBusDelay="0.2" internalBusBandwidth="1024" cost="100" clock="100"/>
<processor id="AB-l" upperFail="0.025" lowerFail="0.01" internalBusDelay="0.2" internalBusBandwidth="1024" cost="140" clock="100"/>
<bus id="bus_connect" cost="100" delay="0.05" bandwidth="100"/>
<externalport id="export" upperFail="0.05" lowerFail="0.01"/>
<externalport id="new_ex_port" upperFail="0.05" lowerFail="0.01"/>
</repository>
<objectives>
<settings noSampling="50" noRun="3" noDuplicate="1" minCost="200" maxCost="10000">
<evaluations>ResponseTime</evaluations>
<evaluations>CPUUtilization</evaluations>
<evaluations>BusUtilization</evaluations>
<evaluations>Safety</evaluations>
<evaluations>Cost</evaluations>
</settings>
</objectives>
</aqosa.ir:AQOSAModel>