Predicting Performance of Multi-Agent Systems During Feasibility Study

S.Ajitha,
M.S.R.I.T, Bangalore,
India

T.V.Suresh Kumar,
M.S.R.I.T, Bangalore
India

D.Evangelin Geetha
M.S.R.I.T, Bangalore
India

K.Rajanikanth
M.S.R.I.T, Bangalore
India

***Abstract*—Agent Oriented software engineering (AOSE) is a software paradigm that has grasped the attention of researchers/developers for the last few years. As a result, many different methods have been introduced to enable researchers/developers to develop multi agent systems. However Performance, a non- functional attribute have not been given that much importance for producing quality software. Performance issues must be considered throughout software project development. Predicting performance early in the life cycle during feasibility study is not considered for predicting performance. In this paper, we consider the data collected (technical and environmental factors) during feasibility study of Multi-Agent software development to predict performance. We derive an algorithm to predict the performance metrics and simulate the results using a case study on scheduling the use of runways on an airport.**

Keywords-Multi Agents, Agent Oriented Software Engineering, Software Performance Engineering, Feasibility Study, Use case point

I. INTRODUCTION

Building high-quality, industrial strength software is difficult. Developing software in domains like tele-communication, industrial control and business process management represents one of the most complex construction tasks humans undertake. Against this background a wide range of software engineering paradigms have been devised. For designing and implementing complex software systems as a collection of interacting, autonomous agents, affords software engineers significant advantages over contemporary methods [13]. An agent is an encapsulated computer system situated in some environment and capable of flexible, autonomous action in that environment. In other words agents provide high level communication and interaction which can perceive the situation of the environment and respond appropriately. Agents are different from objects which are static and cannot change with the environment. When adopting an agent-oriented view most problems require multiple agents, to represent the decentralized nature of the problem, the multiple loci of control, multiple perspectives or the competing interests.

Performance is an important issue for a Multi-Agent system. Even several development methodologies for MAS have been evolved; performance issues are addressed very rarely. For software developers, if performance evaluation is intrinsic throughout Software Development Life Cycle (SDLC), it is easier for them to produce quality software. Several industries canopy their performance issues and addressing performance issues after coding phase. Coding phase has severe consequences for software's, whose performance is critical. It gives impetus thinking in looking into issues while developing software systems. The plausible solutions may be easy for team members to construct quality software systems. Current situation in building performance intensive software system is cumbersome. This led to several researchers to work in this important performance engineering area. Basic idea of works including ours is to make team members to understand easily the performance issues and consequences of Multi-Agent systems. In [14] the author discusses the importance of performance engineering in Agent systems and suggested to define benchmarks and metrics that help to compare and contrast different Agent systems to support software engineering themes within Agent systems. In [11, 12] estimating costs for agent oriented software is discussed. Few authors Addressed the performance issues early in SDLC [5], [6], [7], [8].

In general, performance evaluation system must address performance questions from varied stakeholders. The questions range from user's point of view to tester point of view. From user point of view, the questions may be related to response time of the software solutions. For remaining it may be from analysis phase to testing phase. Whatever may be the likely questions to encounter it is important to develop performance evaluation systems with easy to use by team members. Selecting appropriate evaluation techniques and corresponding performance metrics required. The prediction process using these techniques and metrics do help team members of software systems to evaluate the performance at various phases of SDLC. Several authors worked towards this direction to

978-1-4244-4711-4/09/$25.00 ©2009 IEEE

increase the acceptance of performance evaluation of software systems at all phases of SDLC [10], [16], [17].

Software Performance Engineering (SPE) is a method to predict the performance of any software systems early (analysis phase) in the life cycle [4].We use the same technique for Multi-Agent Systems. SPE continues through the detailed design, coding and testing stages to predict and manage the performance of the evolving software and to monitor, report actual performance against specifications and predictions. Modeling software systems to predict the performance using Unified Modeling Language (UML) is a better choice since UML is seamless from requirements to deployment. The Unified Modeling Language (UML) is a graphical modeling language that is used for visualizing, specifying constructing and documenting software systems [18]. UML is a large and varied modeling language intended to model most application domains, to use in many styles of programming, and at many stages of the development lifecycle. Performance models can be generated from use cases during requirements phase [8], sequence diagrams during analysis and design phase [8] and from state chart and activity diagrams during design phase [5], [10].

In all the approaches, the researchers do not consider the data gathered during feasibility study of the project. During feasibility study, project managers, CEOs, and concerned will participate in discussions with stakeholders to assess the project feasibility. In all these feasibility studies several issues would be discussed: Identification of document, description of current situation, problem description, business and financial aspects, technical aspects and organizational aspects of proposed development, development costs and operational costs, envisaged benefits and recommendation. These technical factors gathered during feasibility study are not accounted for estimating performance in early SDLC. These technical factors can be realized only during detailed design or deployment of software systems. In all the works done in early phases, the authors have not considered the technical factors that are important for early prediction. Keeping these in view, in this paper, we propose an algorithm to predict performance of Multi-Agent Systems with technical factors and environmental factors gathered during feasibility studies. We use in this paper technical and environmental factors gathered for estimating size using use case point approach [2].

II. Proposed Algorithm for Multi-Agent Systems

Performance Engineering is important for software engineering and in particular for software quality. Smith describes the SPE process to assess the performance of software systems early in the life cycle by using two models, software execution model and system execution model [3]. The software resource requirements required for software execution model of SPE process can be obtained by use case point method by considering appropriate technical and environmental factors. The guidance provided by the use case point method [2] is used to calculate the effort based on the nature of the actors and use cases available in the use case model developed during requirement analysis. The procedure for estimating effort using use case point approach is well defined in [2].

Each technical and environmental factor is assigned a value between 0 and 5 depending on its assumed influence on the project. The adjusted use case points obtained from use case model are converted into equivalent Lines of Code (LOC) by considering the programming language to be used for implementation. The number of LOCs in turn converted into number of executable statements (in assembler program) [15]. The amount of data in terms of kilobytes is obtained from number of executable statements. Then the software execution model solved with the amount of data obtained and the overhead specifications (computer resource requirement for each of the software request) [3]. The total response time and overhead specifications (computer resource requirement for each of the software request) [3]. The total response time and for the scenario obtained from software execution model [3]. Total hardware device requirement provided as input to the system execution model and the performance metrics such as throughput, response time, residence time, and device utilization, queue length for each device and also for the system obtained. Based on these and with reference to [9] we have developed the following algorithm for Multi-Agent Systems.

Step1: Study the Feasibility issues in Multi-Agent Systems.

Step2: Compare it with earlier Projects.

Step3: Develop Use Case Model for the entire application.

Step4: Develop Use Case model for the Agent Interactions.

Step5: Classify the Agent as Simple, Average, Complex using probability distribution.

Step6: Calculate Unadjusted Agent weight, Unadjusted Use Case Point, Assign values for environmental and technical factors depending on the scenario , Calculate Technical Complexity Factors (TCF), Environmental Factors (EF)

Step9:- Calculate Use Case Point

$$UCP = UUCW * TCF * EF$$

Step10:- Calculate Lines of Code (LOC) for a specific programming language, LOC for assembler Program, equivalent number of kilobytes

Step11:- Calculate performance with SPE Approach.

Feasibility study of any project tells the logical step whether to proceed with the project or not. A well written feasibility study will support the detailed planning and entering into new business. The feasibility study process involves making rational decisions about a number of enduring characteristics of a project including economic feasibility, technical feasibility, schedule feasibility and operational feasibility. We have drawn a Use Case diagram to represent the interaction between the Agents. Similar to the classification of actors in use case point approach we have classified the agents in our example as Simple, Average, Complex using triangular distribution by considering the interaction between the agents. The interactions are in the form of messages .If the number of messages is less we considered it as simple agent .Similarly we define average and complex categories based on number of messages among agents. Since the data collected during feasibility study is

highly valuable to development team, the performance assessment at this level will help the team to focus on the performance issues early.

Case Study

We consider scheduling the use of runways on an airport application to discuss the validation of our algorithm [1].

A. Description of the Case Study

The application we have considered is a Multi-Agent System. There are three agents in the system. The architecture of the agent we have considered is Belief Desire Intention (BDI). They are Aircraft agent, Air traffic control agent, Runway agent. These three agents can reside in three different locations and can interact with each other to get the requirements satisfied. The most significant performance scenarios we have considered in our application are arrival, departure, delay, weather, runway usage, slot for new aircraft and parking.

B. UML Models for the System

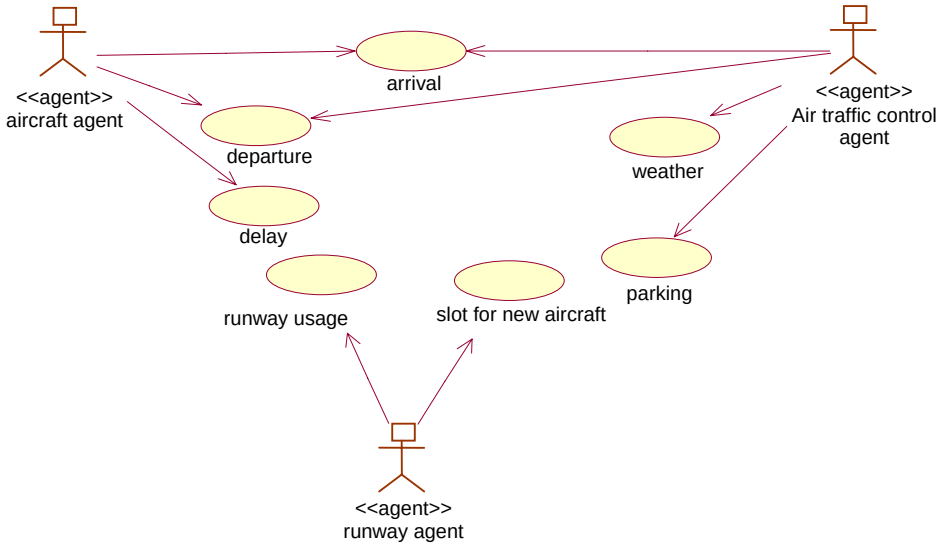

Figure 1. Use case diagram for the case study

The functional requirements for the interaction between the agents are modeled using use case diagram shown in Figure 1. The software module for the specified scenarios is shown in Figure 2. With these models and considering technical and environmental factors of use case point approach, the calculation proceeds as follows: The use case model given in Figure 1 consists of seven use cases namely, arrival, departure, delay, weather, runway usage, slot for new aircraft and parking and three agent namely, runway agent, aircraft agent and air traffic control agent. As discussed in [2], the use cases arrival, departure, delay, weather, runway usage, slot for new aircraft and parking belong to the category average, since the number of transactions for these use cases is between four and seven. The agents described in the model are categorized simple. Using the prototype tool Unadjusted Actor Weight (UAW)is calculated for the agents runway agent, aircraft agent and air traffic control agent, and Unadjusted Use Case Weight(UACW) is calculated for the use cases arrival, departure, delay, weather, runway usage, slot for new aircraft and parking and UAW and UUCW are summed up to get unadjusted use case points.

The application we have considered is highly distributed in nature, the values for the technical factors [2] are considered as follows; T1, T4, T8, T10, T11 the value is 5, for T2 the value is 4 and for T5, T9 the value is 3.We assume that the system is modeled using UML and the programming language to be used for implementation is Java. By considering these assumptions, the values for environmental factors are assigned as follows; E2 the value is 5, for E1,E3,E4 the value is 4 , for E5 the value is 1and for E8 the value is 2 [2].

By considering the values for technical and environmental factors and following the algorithm TFactor and EFactor are calculated. Adjusted use case point value is obtained as 78. The LOC is calculated by multiplying UCP by 53 considering Java as the programming language; the number of executable statements (6 statements per line of code) and the amount of data in kilobytes are calculated [15]. The amount of data obtained is used as input (software requirement) for software execution model to obtain the response time.

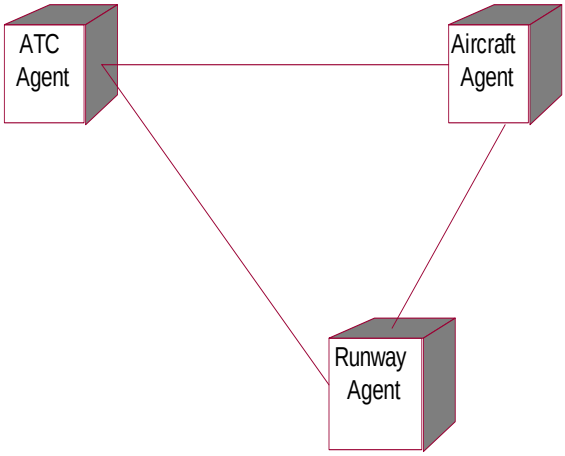

Figure 2. Deployment Diagram for the Case Study

C. *Software Execution Model*

Software execution Model is an early model to ensure that the software architecture will make it possible to meet system performance objectives. To obtain the complete performance the amount of data obtained from use case point estimation is integrated with the executing platform configuration, the architecture of software modules and hardware devices with user profile and software work load. The software resources we examine for this case study are: data size, file access. Hardware resources are CPU1, CPU2, CPU3, Delay and Satellite. By assuming approximate values for amount of hardware resource required for each of the software resource, we have obtained the response time for the scenario mentioned in section IV A as 6.04 seconds.

D. *System Execution Model*

The System execution model solution is obtained by solving queuing network model for the above software execution model. Different performance parameters such as throughput, response time, residence time, device utilization, queue length for each hardware resource and the system are obtained. The different performances metrics are obtained in the form of graphs as given in Figure 3

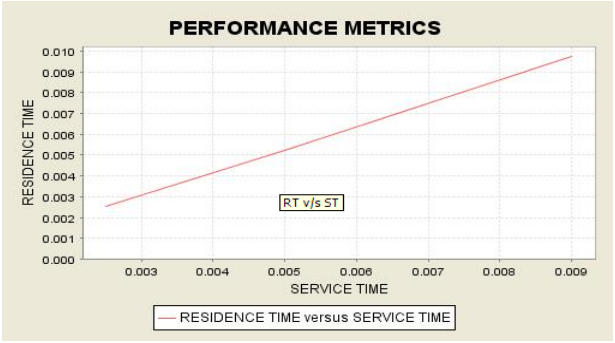

Figure 3. Performance Metrics

E. *Discussion of Case Study*

We have obtained simulated results for system execution model by assigning different values for technical and environmental factors depending on degree of influence (the range for technical and environmental factor from 0 to 5). 1 means weak influence, 3 means average, and 5 means strong influence throughout. The different categories of values and the corresponding response time obtained by software execution model are as follows:

- Strong influence of technical and environmental factors – 5.13 seconds

- Average influence of technical and environmental factors –5.8549 seconds
- Weak influence of technical and environmental factors –5.265 seconds
- Strong influence of technical and weak influence of environmental factors –9.495 seconds
- Weak influence of technical and strong influence of environmental factors –2.72 seconds

From the simulated results, it is observed that the strong, weak and average influences are almost same. The response time is higher when strong influence values for technical factors and weak influence values for environmental factors are considered compared to weak influence values for technical factors and weak influence values for environmental factors are considered.

III. CONCLUSION

In this paper, we proposed an algorithm to predict performance of Multi-Agent systems during feasibility study. Early performance prediction process requires several technical and environmental factors for predicting performance. We considered the data collected during cost estimation using use case point approach. We presented a case study by considering these factors. Future work includes predicting performance by considering agent characteristics such as cooperation, negotiation, mobility etc.

REFERENCES

[1] Ajitha S, Suresh Kumar T V, Evangelin Geetha D, Rajani Kanth K: "BDI Software Process for Agent Oriented Software using UML", Proceedings of International conference ICDM,09, Gaziabad

[2] Bente Anda, Hege Dreiem, dag I. K Sjobergand Magne Jorgensen, "Estimating Software development Effort based on Use Cases – Experiences from Industry",

www.idi.ntnu.no/emner/tdt4290/docs/faglig/uml2001-anda.pdf.

[3] Connie U. Smith, Performance Engineering of Software Systems, Reading, MA, Addison-Wesley,1990.

[4] Connie U, Smith and Lloyd G. Williams, Performance Solutions, 2000.

[5] D.C. Petriu, H Shen, "Applying the UML Performance Profile Graph Grammar-based derivation of LQN models from UML specifications", in Computer performance Evaluation-Modeling techniques and Tools, (T.Fields, P.Harrison, J.Bradley, U. Harder, Eds.) LNCS 2324, pp.158-177, Springer, 2002.

[6] Dorin Petriu, Murray Woodside: "Analysing Software Requirements Specifications for Performance", Proceedings of the 3rd international workshop on Software and performance 2002, Rome, Italy July 24-226, 2002, pp.1-9.

[7] Evangelin Geetha D, Suresh Kumar T V and Rajani Kanth K: "Early Performance Modeling for Web Based Applications", LNCS, Springer Verlag, December 2004, pp.400-409.

[8] Evangelin Geetha D, Ram Mohan Reddy, Suresh Kumar T V and Rajani Kanth K: "JAPET: A Java Based Tool for Performance Evaluation of Software Systems", Proceedings of SKIMA 2006, International Conference on Software Knowledge Information Management and Applications, December 2006, Chinag Mai, Thailand, pp. 64-69.

[9] Evangelin Geetha D, Suresh Kumar T V and Rajani Kanth K: "Predicting Performance of Software Systems during Feasibility Study of Software Project Management", 1-4244-0983-7/07, 2007 IEEE.

[10] Hoda Amer: "Automatic Transformation of UML Software Specification into LQN performance Models using Graph Grammar Techniques", Carleton University, 2001.

[11] Jorge J.Gomez-Sanz, Juan Pavon, Francisco Garito: "Estimating Costs for Agent Oriented Software", TIC2002-04516-C03-03

[12] Jose Manuel Fonseca, Eugenio de Oliveira, Adolfo Steiger-Garcao: "Multi-agent Negotiation Algorithms for Resources Cost Estimation: A Case Study

[13] Nicholoas .R. Jennings, " An Agent-Based approach for Building Complex Software Systems ",April 2001/vol 44 No 4 communications of the ACM.

[14] Omer Rana, Chris Preist, Michael Luck: "Progress in Multi-Agent Systems Research, March 2000

[15] Robert T. Futrell, Donald F. Shafer, and Linda I. Shafer: Quality Software Project Management", Pearson Education, 2006.

[16] Simonetta Balsamo Roberto Mamprin Moreno Marzolla: Performance Evaluation of Software Architectures With Queuing Network Models", 2004.

[17] Simonetta Balsamo Moreno Marzolla, "Performance Evaluation of UML Software Architectures with Multiclass Queueing Network Models", Proceedings of the 5th international workshop on Software and performance, 2005, Palma, Illes Balears, Spain July 12-14, 2005, pp. 37-42.

[18] www.omg.org