International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.
© 2022 Scholar AI LLC.
Vol. 34 No. 2 September 2022, pp. 410-416

Early Guessing of Performance Using Simulation as Part of Service Development

Pélagie HOUNGUE, Romaric SAGBO

Information and Communication Technology department,
Institut de Mathématiques et de Sciences Physiques (IMSP/UAC)
Dangbo, Benin
pelagie.houngue@imsp-uac.org

(cc) BY

Abstract— In Service-Oriented Architectures (SOA), the key problem is the quick and accurate evaluation of web service performance. Despite the fact that the integration of the simulation step into the development cycle of softwares/web services can allow to learn early the behavior of the performance of software/web service, it is still a challenge to use simulation as part of service development. This integration can be used to assess the performance of a family of web services by developing one of them. In this paper, we propose a methodology that shows how the simulation step can be integrated to the development cycle of a family of services using a model-based approach to describe the services and by choosing a reference web service to be developed and used to guess the performance of the remaining services in the family.

Keywords— Model-based testing; Simulation; Performance; Web services; Zero-knownledge

I. INTRODUCTION

The development based on model allows to integrate early a performance analysis at the software development stage [1], [2], [3]. The evaluation of service performance is important [4], but in a Service-Oriented architecture there is some difficulties to assess the non-functional properties of the web service without any knowledge on the service's behavior. Nowadays, early performance evaluation is a critical step in web service development [5] and it is important to have a prediction model for performance evaluation at this stage of the development cycle when no historical data are not yet available [6].

In [5], we propose a model-based approach that uses a Symbolic Transition System (STS) to describe the web services as finite state automata and evaluate their performance. This model was extended for simulation purposes by adding state transition probabilities and the delay distributions on the transitions. We evaluated the accuracy of this approach using two different scenarios, a full-knowledge and partial-knowledge scenarios which provide good results. Furthermore, in [6], we define a zero-knowledge scenario which defines a model that allows to estimate the performance of the service when no a priori knowledge is not available on its performance. This model uses the description of the service in the WSDL file, the SOAP message that encodes the input parameters of the service and the STS-based model to guess the behavior of the service. Moreover, in [7] we extends our previous works by using data obtained from our zero-knowledge approach, where neither the service code nor test-based the information on service execution times are available, to generate the SLA template and use it to negotiate the preliminary SLA and monitor it.

In order to show the application of our methodology to estimate early the behavior of the service, we propose in this paper the different steps that need to be follow in order to use the simulation as part of the development [8]. This methodology will be applied to a family of services and will show how the performance of the entire family can be guessed by choosing one of the services as reference, by developing it and use it to perform the estimation of the performance for the others services members of the family. This work shows the integration of the performance analysis in the development process.

The remaining of this paper is organized as follows. Section II presents the overview of our zero-knowledge scenario defined in our previous works. Section III gives the description of our concept and Section IV presents the integration of the simulation phase in the development cycle and describes the different steps to be followed to estimate the behavior of a family of services. Subsequently, Section V gives an overview of the related work and Section VI gives the conclusion. Finally, Section VII points out some future works.

II. Overview of our zero-knowledge scenario

Our zero-knowledge approach presented in [6] allows to integrate early the web service performance analysis to the development stage in order to have an estimate of the behavior of a given web service using only the information provided by the WSDL file, the SOAP message used to query the service and the STS-based model [9]. For this purpose, an approach is defined to assess the complexity of the web service in order to evaluate some factors used to evaluate the interval of service times.

The first step of our approach built a profile table that associates parsing and construction times to the depth and the cardinality of the XML tree that encodes the input parameters of the web service (SOAP message).

The second step is to determine the class of complexity of the service from the WSDL by defining an approach that makes easy the complexity evaluation. At the end of this step, a complexity factor called γ is computed for the web service to be simulated and a class of complexity is associated to it.

In the last step the parsing and construction times (PT + CT) read from the profile table using the characteristics of the input SOAP message are used to estimate the interval in which the service times vary by applying the complexity factor γ and a factor α $(0 < \alpha \leq 1)$ defining from a reference web service chosen for its class of complexity. In order to generate the bounds of the interval, α has two values denoted αmin and αmax to compute respectively the lower bound and the upper bound of the service times interval. After this estimation, a correction factor denoted FDI (Data Intensive Factor) is applied in order to take into account how the use of different types of resources can impact the service performance.

With the interval of service times, a Monte Carlo [10] based approach is used to generate different random extractions of the partition for the different transitions of the STS-based model extended for simulation. One of the combinations is chosen randomly to annotate the STS-based model and the simulation script is generated to guess the behavior of the service.

The results obtained demonstrate that this approach can provide an approximation of service behavior that will be similar to the behavior obtained by running a real service after the development [6], [7].

The following equations are used to compute the bounds of the service times interval and the steps to define them are detailed in [6]. Equations (1) and (2) allows to compute respectively the lower bound and the upper bound of the service times interval. γ is defined from the complexity and the values of αREF are defined from the reference web service using (3) and (4).

$$ST_{min} = \frac{\text{PT+CT}}{[\alpha_{RE\ \ min}]^{\gamma}} * F_{DI} \quad (1)$$

$$ST_{max} = \frac{\text{PT+C}}{[\alpha_{REF_{max}}]^{\gamma}} * F_{DI} \quad (2)$$

$$\alpha_{REF_{min}} = \frac{[\text{PT+CT}]_{REF}}{[ST]_{REF_{min}}} \quad (3)$$

$$\alpha_{REF_{max}} = \frac{[\text{PT+C}\ \ ]_{REF}}{[ST]_{REF_{max}}} \quad (4)$$

III. The Concept Or Scenario

The assessment of the service performance early during the development process can give good indicators to analyze the quality of the service under development. This approach can also help the developer when he has to develop a family of services. In this section, we present the story that justifies our concept and how a solution can be found using our approach and frameworks presented in our previous works [5], [6] and [7].

3.1. Running Example or problem

Jack is a developer and have to participate to a call that consists to develop a family of web services in the transportation domain. The winner will be the one who can make a proposition that can show a reliable behavior of the web services from the service time point of view. It can be noticed that the developers did not have enough to develop all the services in the set. Also, if they have time, they cannot develop all of the service without be sure to win the call, then the idea to develop all of the service for the call has to be avoided. Jack asks us if there are some strategies that can help him to win this call, since he has just much time to develop correctly one service. We say to him that there is a solution. He can develop one service as reference for the family and guess the behavior of the others from the performance measured for this service.

In the following, we will give more detailed about the solution proposed to our friend Jack.

3.2. Solution

In order to estimate the behavior of a family of the web service, we propose to Jack a methodology that is based on the description of the web service as automaton using STS[11]. An STS-based model can be extended for simulation in order to assess the behavior of the service as shown in [5], [6], [7]. In this case, the developer has to:

1) Choose a reference web service for the family

2) Define the STS-based model of the reference service and describe the service using WSDL

3) Develop the reference web service

4) Measure the performance of the service developed

5) Save the performance in a database

6) Take the remaining web services and define for each of them the STS-based model, the WSDL file and the SOAP message that encodes the input parameters of the service

7) Apply our zero-knowledge scenario defined in [6] and presented in Section II using the first web service as reference to generate a simulation script for each service remaining in the family

8) Run the simulation script to guess the behavior of the remaining services and save the outputs.

This solution allows Jack to win the competition by providing a reliable estimation of the different services in the family.

The goal of this paper is to show how the simulation can be used as part of the development process and we present in Section IV the detailed methodology that makes this integration possible and solve this kind of situation with accurate precision.

IV. Development Cycle Using Simulation – Our Approach

This section gives a better definition of our methodology to simulation of the service performance as part of the development cycle. In our approach presented in [6], in order to guess the behavior of a service using simulation we need a reference web service choosing according to the class of the service to be simulate. In this case where we apply our approach on a family of the services, we choose a reference web service in the family and its performance will be used to guess the behavior of the service.

Fig. 1 gives a complete view of the development of a class of service. In the following we describe the different steps shown on this figure.

In the first part of the process, the developer chooses the reference web service in the family (1). This service needs to be chosen as representative of the family according to the experience of the developer as example. After that the WSDL file describing the service (2) and the STS-based model of the service (3) are defined and used (4 and 5) at the design stage to develop the service (6). The service developed is tested and the its performance is monitored (7) and save in a database prepared for a reference web service. These results will be used later in the performance estimation of the remaining services in the family.

The second part of the process consists to estimate the behavior of remaining services using the performance measured for the reference web service. The remaining family is taken (9) and the WSDL file, the STS-based model of the service and the input SOAP message (10, 11 and 12) are defined for each service. They will be used (13, 14 and 15) at the design stage to produce the STS-based model for simulation (16). This model is annotated with the transition probabilities and default distributions of delay needed to execute the task associated to the transition of the model. At the simulation stage (17), the delay distributions on the transition of the STS-model for simulation are updated. This information is computed using the profile table (8) and the performance measured for the reference service (19). The WSDL file of each service is used to compute the complexity of the service and the characteristics of the input SOAP message are used to read in the profile table the parsing and construction times, and use them to evaluate the bounds of the service times interval for each service. The interval of service times is divided randomly to annotate the STS-based models with the delay distributions (20). In our case, by default a uniform distribution between two values is selected. After that the simulation script is produced for each service and the service can be simulated. The outputs are saved in a database for future use (21). The updated STS-based model obtained (20) is send back to the design step and make available for the development stage. At this step the behavior of the candidate web services can be guessed early using the simulation scripts produced. In order to verify the accuracy of our approach, the development can continue (23) and the services developed will be tested and monitored (24) and the results saved in a database (25). After the development, the simulation and the testing results are sent for comparison (26-27).

The comparison will show the accuracy of the simulation results to guess the behavior of a family of services early using one of them as reference to simulate the others.

We can note that the step (22) in the figure can loop in order to update the model to take into account new information from the reference service test.

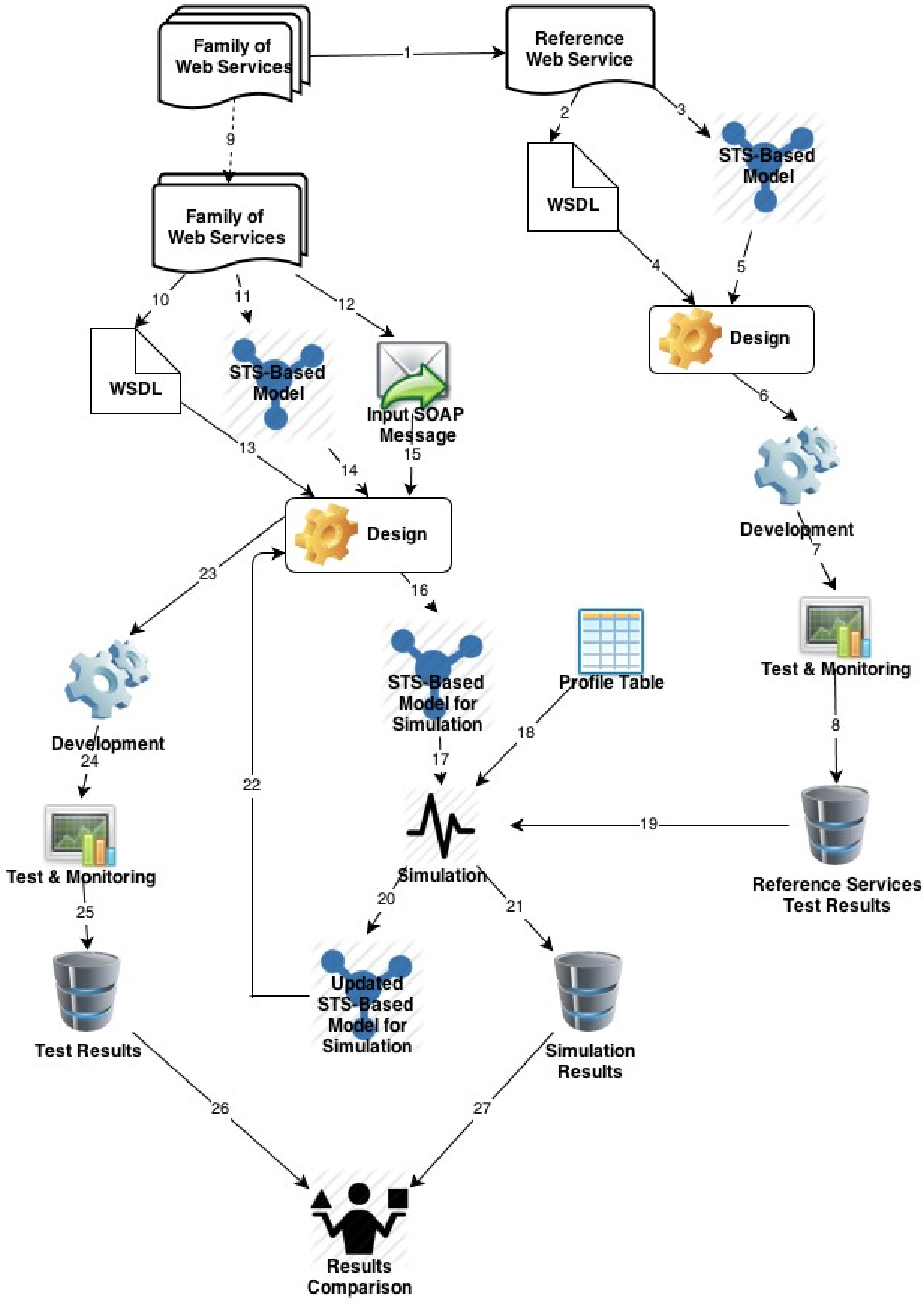

Fig. 1. Simulation in the development cycle

V. Related Works

The author of [8] gives an introduction of Software Development Life Cycle (SDLC), followed by the comprehensive comparison among the various SDLC models. The authors of [12] show that testing is an important part of software development, while [13] define a new software development model, which prioritizes security aspects at the different phases of the software life cycle and takes advantage of the benefits of the agile models. The authors of [14], [15] propose simulation modeling of UML software architectures by annotated UML diagrams with performance results.

The authors of [16], [17] propose to use simulation to evaluate the performance of the web services in order to have enough information on the behavior of the services and compose efficient web processes. The authors of [1] propose to extend the model-driven engineering with performance engineering in order to perform a performance evaluation process during the different development phases and. They extended a UML activity diagram with performance information and transform it in a simulation model for early performance prediction. They proposed to find the performance information used for the prediction from the developers experience and/or the collected performance data on existing systems or similar service. The authors of [18] show how to Increase the ROI of a Software Development Lifecycle by Managing the Risk using Monte Carlo and Discrete Event Simulation.

As in [19] where challenges and opportunities of simulation-driven functional product development and operation are shown, and [20] show the way to assess the risk of software development with Agile methodologies using simulation, in this paper as in the previous [5], [6], [7], we try to show that we can guess the behavior of service at early stage of the development by proposed a methodology to integrate simulation in development cycle based on the performance of reference service.

VI. Conclusion

Simulation driven software development means making use of simulation at each stage of the service development process. In this work, we show how simulation can fit into software delivery life cycles and how it can help businesses reach their goals.

Simulation most certainly has immense utility, but it should be used carefully following the situation when no better alternatives exist, and more accurate reference services can be provided to our model. The model proposed shows that there is a good perspective to have more information about performance of the future service in development.

VII. Future works

In our future work, we will provide enough benchmarks on our approach to show the application of our methodology which integrate the simulation as part of development cycle on a real family of service development. We can improve our approach by making random selection of the reference web service to verify if the results will be better. With the evolution of machine learning and Artificial intelligence, it could be better to investigate new opportunities to combine simulations and machine learning to accelerate development processes and guess the behavior of the service at early stage.

VIII. Acknowledgment

We thank the African Center of Excellence in Mathematical Sciences and Applications (CEA-SMIA) of University of Abomey-Calavi for funding this work.

References

[1] M. Fritzsche and J. Johannes, “Putting performance engineering into model-driven engineering: Model-driven performance engineering,” Model. Softw. Eng., pp. 164–175, 2008.

[2] L. Cheung, L. Golubchik, and F. Shai, “A Study of Web Services Performance Prediction: A Client’s Perspective,” 2011.

[3] D. C. Petriu, “Software model-based performance analysis,” John Wiley Sons, 2010.

[4] M. P. Papazoglou, V. Andrikopoulos, and S. Benbernou, “Managing Evolving Services,” IEEE Softw., vol. 28, no. 3, pp. 49–55, 2011.

[5] C. A. Ardagna, E. Damiani, and K. A. R. Sagbo, “Early Assessment of Service Performance Based on Simulation,” 2013.

[6] C. A. Ardagna, E. Damiani, K. A. R. Sagbo, and F. Frati, “Zero-knowledge evaluation of service performance based on simulation,” 2014. doi: 10.1109/HASE.2014.47.

[7] K. A. R. Sagbo, Y. P. E. Houngue, and E. Damiani, “SLA Negotiation and Monitoring from Simulation Data,” 2017. doi: 10.1109/SITIS.2016.126.

[8] V. Rastogi, “Software Development Life Cycle Models- Comparison , Consequences,” Int. J. Comput. Sci. Inf. Technol., 2015.

[9] L. Frantzen, J. Tretmans, and T. A. C. Willemse, “A Symbolic Framework for Model-Based Testing,” 2006.

[10] K. R. Koch, “Monte Carlo methods,” GEM - Int. J. Geomathematics, 2018, doi: 10.1007/s13137-017-0101-z.

[11] L. Frantzen, “Modeling Symbolic Transition Systems in XML.”

[12] M. E. Khan and F. Khan, “Importance of Software Testing in Software Development Life Cycle,” Int. J. Comput. Sci., 2014.

[13] J. de V. Mohino, J. B. Higuera, J. R. B. Higuera, and J. A. S. Montalvo, “The application of a new secure software development life cycle (S-SDLC) with agile methodologies,” Electron., 2019, doi: 10.3390/electronics8111218.

[14] S. Balsamo and M. Marzolla, “A simulation-based approach to software performance modeling,” ACM SIGSOFT Softw. Eng. Notes, 2003, doi: 10.1145/949952.940122.

[15] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni, “Model-based performance prediction in software development: A survey,” Softw. Eng. IEEE Trans., vol. 30, no. 5, pp. 295–310, 2004.

[16] S. Chandrasekaran, G. Silver, J. A. Miller, J. Cardoso, and A. P. Sheth, “XML-based modeling and simulation: web service technologies and their synergy with simulation,” in Proceedings of the 34th conference on Winter simulation: exploring new frontiers, 2002, pp. 606–615.

[17] S. Chandrasekaran, J. A. Miller, G. S. Silver, B. Arpinar, and A. P. Sheth, “Performance Analysis and Simulation of Composite Web Services,” Electron. Mark., vol. 13, no. 2, pp. 120–132, 2003.

[18] W. F. Kramer et al., “INCREASE RETURN on Investment of Software Development Life Cycle by Managing trie Risk--A Case Study.,” Def. Acquis. Res. J. A Publ. Def. Acquis. Univ., 2015.

[19] J. Pavasson, A. L. Ljung, M. Karlberg, I. A. S. Larsson, S. Johansson, and T. S. Lundström, “Challenges and opportunities within simulation-driven functional product development and operation,” 2014. doi: 10.1016/j.procir.2014.06.149.

[20] M. I. Lunesu, R. Tonelli, L. Marchesi, and M. Marchesi, “Assessing the risk of software development in agile methodologies using simulation,” IEEE Access, 2021, doi: 10.1109/ACCESS.2021.3115941.