Predicting Software Service Availability: Towards a Runtime Monitoring Approach

Davide Lorenzoli
Department of Computing
City University London
London, UK
Davide.Lorenzoli.1@soi.city.ac.uk

George Spanoudakis
Department of Computing
City University London
London, UK
G.Spanoudakis@soi.city.ac.uk

***Abstract*— This paper presents a prediction model for software services availability measured by the mean-time-to-repair (MTTR) and mean-time-to-failure (MTTF) of a service. The prediction model is based on the experimental identification of probabilistic prediction for variables that affect MTTR/MTTF, based on monitoring service data collected at runtime.**

Keywords - Run-time QoS Prediction, Software Services.

I. INTRODUCTION

Monitoring quality-of-service (QoS) properties of software services at runtime is necessary for verifying whether services deliver the levels promised to their consumers. Monitoring has, thus, been supported by several approaches, which, however, can only detect violations of QoS properties after they have occurred without being able to predict them. This is a significant limitation as that capability to predict violations of QoS service properties at runtime is important for the dynamic and proactive adaptation of service-based systems.

In this paper, we present a runtime prediction model for a key QoS property of software services, namely software service availability. The model is based on the prediction of two measures of service availability: the *mean-time-to-failure* (MTTF) and *mean-time-to-repair* (MTTR) of a software service, defined as the average up and the average down time in the operational life of a service, respectively.

The rest of this paper is structured as follows. In Section II, we give an overview of the prediction model for software service MTTR and MTTF and in Section III, we present the results of an initial experimental evaluation of it. Subsequently, in Section IV and V, we discuss related work and provide conclusions and directions for future work, respectively.

II. MTTR AND MTTF PREDICTION MODELS

In our model, the MTTR of a software service is defined as the average time from a failure of a service to respond to an operation call until it restarts responding to operation calls again. MTTR needs to be bounded to ensure the timely reactivation of a service after periods of unavailability. Hence in a service level agreement (SLA), this would be typically specified as a constraint MTTR ≤ K where K is a constant time measure.

The estimation of the probability of violating the constraint MTTR ≤ K at a future time point t_e is based on identifying the probability distribution functions of two variables: (1) the MTTR of the service, and (2) the time between non-served calls of service operations that occur in a period during which a service has been available (referred to as *time-to-failure* or "TTF"). MTTR and TTF values correspond to the periods shown in Figure 1. More specifically, MTTR is computed as the average of TTR values, i.e., the time difference between the first served call of a service following a period of unavailability and the initial non served call (NS Call) of the service that initiated this period. TTF is the difference between the timestamps of two NS calls of the service that initiate two distinct and successive periods of unavailability.

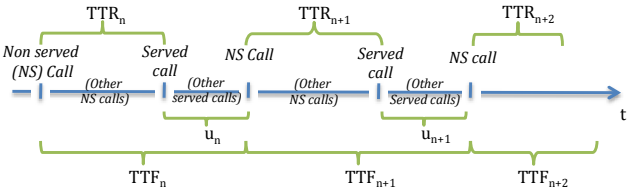

Figure 1. TTR and TTF values

The probability to violate the constraint MTTR ≤ K at the end of the time period *p* can be estimated approximately by the following formula:

$$\Pr\left(\bigwedge_{y=1}^{M} E_y\right) = \begin{cases} 1 - \sum_{y=1}^{M} \Pr(y) \times \Pr\left(MTTR_y \le MTTR_{crit}\right), & MTTR_c > K \\ \sum_{y=1}^{M} \Pr(y) \times \Pr\left(MTTR_y > MTTR_{crit}\right), & MTTR_c \le K \end{cases} \quad (1)$$

The above formula distinguishes two cases: (a) the case where the last recorded MTTR value at the time when the prediction is requested violates the constraint (i.e., the case where $MTTR_c > K$), and (b) the case where the last recorded MTTR value at the time when the prediction is requested does not violate the constraint (i.e., the case when $MTTR_c \le K$). In the former case, the probability of violation is computed as the probability of not seeing a value of MTTR in period *p* (i.e., $MTTR_y$) that is sufficiently small to restore the current violation. In the second case, the probability of the violation is computed as the probability of seeing a large enough $MTTR_y$ value (i.e., a value greater than $MTTR_{crit}$) that would violate the constraint.

In the case of MTTF, the SLA constraint to be monitored and forecasted is MTTF ≥ K (the largest the MTTF the less frequent the failures of the given services)

and the probability of violating the constraint can be estimated approximately by the formula:

$$\Pr\left(\bigwedge_{y=1}^{M} E'_y\right) = \begin{cases} 1 - \sum_{y=1}^{M} \Pr(y) \times \Pr\left(MTTF_y \geq MTTF_{crit}\right), & MTTF_c < K \\ \sum_{y=1}^{M} \Pr(y) \times \Pr\left(MTTF_y < MTTF_{crit}\right), & MTTF_c \geq K \end{cases} \quad (2)$$

Details about the derivation of formulas (1) and (2) are omitted due to space restrictions but can be found in 0.

III. Experimental Results

To evaluate the precision and recall of the MTTR and MTTF prediction models, we used monitoring data generated from the invocation of a web service of Yahoo allowing programmatic search of Internet pages (*WebSearchService)*. Through this process, we collected a total of 5500 invocation and 5500 response events

To evaluate our prediction model, we divided the total time range of the 5500 invocations in 9 equal sub-ranges and for each of them we computed the $MTTR_c$ and $MTTF_c$ values for the end of each sub-range. We also used five different QoS constraints for MTTR and MTTF, based on different K values. The *K* values were determined by the MTTR and MTTF values at the end time point t_c of each of the nine sub-ranges as: $0.75\times MTT*_c$, $MTT*_c-1$, $MTT*_c$, $MTT*_c+1$, $1.25\times MTT*_c$. For each *K,* we generated predictions using combinations of prediction windows of 1, 10, 60 and 600 seconds, and history sizes of 100, 300 and 500 data points. Hence, we performed 540 predictions in total for each of the MTTR and MTTF.

The precision and recall of these predictions were measured using the following formulas:

$$\text{Precision} = (\text{TP} + \text{FN})/(\text{TP} + \text{FP} + \text{TN} + \text{FN})$$

$$\text{Recall} = \text{TP}/(\text{TP} + \text{TN})$$

In these formulas, *TP* is the number of correct positive predictions of QoS constraint violations; *FP* is the number of incorrect predictions of QoS constraint violations; *TN* is the number of correct predictions of QoS constraint satisfaction, and *FN* is the number of incorrect predictions of QoS constraint satisfaction. We also investigated the effect of the size of the historic event set (HS) used to generate the QoS prediction model, and the prediction window (PW) on precision and recall. Table 4 shows the recall and precision measures for MTTR and MTTF.

TABLE 1. MTTR AND MTTF PRECISION AND RECALL

		MTTR		MTTF	
		Precision	**Recall**	**Precision**	**Recall**
Prediction window (secs)	**1**	0.96	0.94	0.90	0.93
	10	0.81	0.71	0.79	0.78
	60	0.77	0.61	0.60	0.63
	600	0.47	0.39	0.56	0.67
History size (events)	**100**	0.74	0.67	0.65	0.61
	300	0.75	0.60	0.72	0.83
	500	0.76	0.58	0.77	0.83
Overall		0.75	0.64	0.71	0.750

As shown in the table, the overall precision and recall of predictions for all different combinations of HS and PW were 0.75, 0.63 for MTTR and 0.65, 0.7 for MTTF, respectively. As expected, precision and recall improved significantly for both models when considering shorter prediction periods, rising to 0.96 and 0.94 for MTTF and 0.9, 0.94 for MTTR when the prediction window was set to 1sec. The results also indicated that the history size had no consistent effect in the recall and precision of the two models.

IV. Related Work

Existing techniques for predicting software system properties may be classified with respect to different criteria, including the property of the system that a technique aims to predict and its algorithmic approach 0.

With respect to the former criterion, there have been techniques focusing on prediction of software systems failures 0, trends in different system parameters such as server workloads, or CPU loads and network throughput 0.

With respect to the algorithmic approach, there are techniques using regression models 0, various mean time prediction models 0; and FSA based prediction 0.

Our prediction approach for software services MTTR and MTTF is, to the best of our knowledge, novel both in terms of its algorithmic basis and its focus on prediction of threshold constraints for these availability properties.

V. Conclusions

In this paper, we have presented a black-box approach for predicting software service availability based on forecasts of the MTTR and MTTF of a service. In this approach, MTTR/MTTF measures computed from captured service invocations and responses at runtime are used for the generation of a probabilistic model for MTTR and MTTF.

Currently, we are working on developing prediction models for other aggregate QoS properties of software services (e.g., service throughput), without relying on behavioural, compositional or usage service models since such models are not widely available.

VI. Acknowledgments

The work presented in this paper has been supported by the EU Commission; F7 Project SLA@SOI (No. 216556).

References

W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 1979.

B.D. Lee and J.M. Schopf. Run-time prediction of parallel applications on shared environments. *IEEE Cluster Computing,* 0:487, 2003.

D. Lorenzoli, L. Mariani, and M. Pezze. Towards self-protecting enterprise applications. In 15th *ISSRE, pp 39--48*, 2007.

F. Salfner, M. Lenk and M. Malek. A survey of online failure prediction models. *ACM Comput. Surv.* 42(3), Article 10, 2010.

D. Lorenzoli and G. Spanoudakis. Runtime prediction of MTTR and MTTF violations. Technical report. City University London, 2011

R. Vilalta et al. Predictive algorithms in the management of computer systems. *IBM Systems Journal*, 41(3):461–474, 2002