Exploring the Impact of Inaccuracy and Imprecision of QoS Assumptions on Proactive Constraint-Based QoS Prediction for Service Orchestrations

Dragan Ivanović
idragan@clip.dia.fi.upm.es
Universidad Politcnica de Madrid (UPM)

Manuel Carro
mcarro@fi.upm.es

Manuel Hermenegildo
herme@fi.upm.es

UPM and IMDEA Software Institute

***Abstract*—Constraint-based Quality of Service (QoS) prediction is a method for predicting violations of Service Level Agreements (SLAs) in an executing instance of a service orchestration. It uses assumptions about the ranges of QoS values for component services in the orchestration. Experiments suggest that the method, when given correct component QoS assumptions, produces highly accurate predictions according to a series of quality-of-prediction metrics, and that it does so well ahead of the time when the prediction is to happen. We study the behavior of this method when the component QoS assumptions become incorrect or too vague. We conclude that the effect is a graceful deterioration in prediction quality, unless gross (order-of-magnitude) imprecisions are introduced. However, the method is very sensitive to the loss of information on the lower bounds for component QoS values, since the knowledge of the upper bounds is not sufficient for failure prediction.**

Keywords: Service orchestration; Quality of service; Prediction; Constraints; Quality of prediction; Experimental validation.

I. Introduction

In service-oriented applications, service compositions are used to perform complex, higher level, or cross-organizational tasks in a platform-independent manner, using loosely coupled component services. An important aspect of composition usability are its Quality of Service (QoS) properties, such as execution time or availability, and their permissible value ranges which, when relevant, are defined in Service Level Agreements (SLAs) between service users and providers.

Predicting an SLA violation at run time, for a given executing instance of a service orchestration, allows service providers to act proactively, plan ahead, and initiate instance-level adaptation that may try to avoid the SLA violation or to mitigate its undesirable effects. Constraint-based QoS prediction for service orchestrations [1] is a method that can be used for that purpose. As shown in Figure 1, it can be instrumented as a two-stage process which repeats itself until the orchestration finishes. In the first step, a constraint satisfaction problem (CSP) [2] is constructed. It models the QoS, such as the execution time, for the entire orchestration, based on two key pieces of information. The first one is the continuation, which describes what remains to be done until the end of the executing instance, as well as the current data items. The CSP is derived from the structure of the continuation, by establishing logically sound relationships (constraints) between the numeric QoS values of complex constructs (sequences, branches, loops, parallel flows) and those of their components.

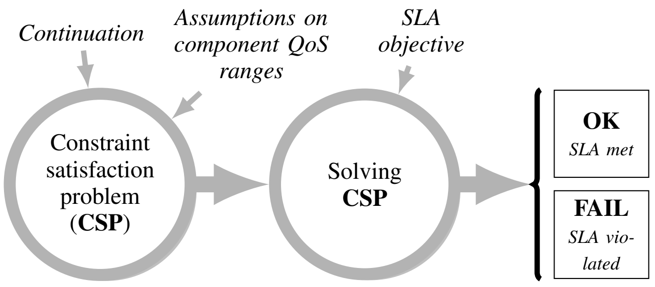
Continuation
Assumptions on component QoS ranges
SLA objective
Constraint satisfaction problem (CSP)
Solving CSP
OK
SLA met
FAIL
SLA violated

Figure 1. The steps in the constraint-based QoS prediction approach.

The second important ingredient of the CSP are the assumptions on the upper and lower bounds of the component service QoS value ranges. Since we build constraints based on crisp logical reasoning with equalities and inequalities, and not on statistical reasoning about the expected or most probable values, we require only the bounds and not the information regarding the statistical distribution of the component QoS values between them. The accuracy of the assumed bounds tells us the probability that the empirically measured component QoS falls inside these bounds: fully accurate assumed bounds contain all possible component QoS values, while less accurate ones leave out some of them. For accurate bounds, one set of assumed bounds is more precise than another one if it is contained in the latter, i.e., it is tighter. Ideally, the most precise and accurate assumed bounds define the tightest interval containing the possible QoS values.

In the second step of our prediction method, the CSP is solved against an SLA objective, which is typically a limit imposed on the QoS of the remainder of the executing orchestration. We consider two cases: when the SLA objective is met (*OK*), and when the SLA objective is violated (*FAIL*). When one of the cases is proven unsatisfiable, the other one is necessarily satisfied (resp., FAIL or OK) and is used as the prediction.

A number of factors may affect the quality of the constraint-based QoS prediction. In this paper, we explore the impact of inaccuracy and imprecision of the assumed component QoS bounds, in an attempt to find out how these factors can be expected to reflect on the quality of prediction, i.e., how robust the constraint-based approach is with respect to suboptimal assumptions about the outside world.

Other approaches for predicting SLA violations in service compositions have been proposed. Some of them [3], [4] rely on statistical reasoning, and use data mining to build models of the QoS of orchestrations from the historically recorded execution data. Other approaches [5] rely on monitoring and online testing [6], and yet others use run-time verification based on model checking [7], [8]. While these proposals differ from the constraint-based approach in a number of respects, we tried to follow some common methodologies for evaluating quality of prediction.

We proceed by first motivating the experimental approach used, then presenting the evaluation framework and the experimental results, and, finally, providing some conclusions.

II. Motivation

To assess the impact of inaccurate and imprecise assumptions on the QoS of the components on the quality of the constraint-based prediction, it is useful to establish a baseline case against which the comparison is made. In the baseline case, we use the most accurate and precise component QoS assumptions, and measure the quality of prediction obtained under such ideal conditions. Still, the results in the baseline case will depend on a number of factors, the most important ones being the structure and logic of the orchestration itself, and the particular constraint solving domain or technique used for the prediction. The latter is briefly commented upon in Section III-C.

In this paper, we base our evaluation on an industrial service workflow, proposed by Leitner et al. [9] in the context of fault prediction based on data-mining and cost-optimization of service composition adaptations.[1] This service workflow uses 18 component services to handle order processing, manufacturing, billing, quality control, and shipment tasks in an on-demand production line scenario. The control structure of the orchestration contains some of the typical control constructs that can be found in real-world orchestrations: branching, looping, parallel flow control constructs, and sequential chains of activities. We look at the execution time as the QoS attribute of interest.

The first question is how to measure the quality of prediction in the baseline case. We need to establish a set of indicators (also known as *prediction quality metrics*) such as precision, accuracy, and recall, and, besides, to look at how long before the predicted event the prediction was made, which is clearly very important for adaptation. The accuracy and precision of prediction is of course different from the

[1]More details are available at
`http://www.infosys.tuwien.ac.at/prototypes/VRESCo/experimentation`

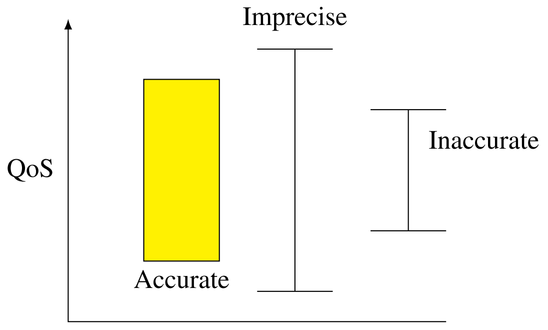

Figure 2. Lack of accuracy and precision in the estimation of QoS bounds.

accuracy and precision of the assumptions on which the prediction is based.

A second issue is how to define the situations where the assumptions on the QoS of some component differ from the real bounds. The bounds we work with can be tighter than the actual ones, and thus they may not contain some of the values that the actual QoS can take, but all points in the range are valid. A representation of the situation is labeled as *inaccurate* in Figure 2. It is also possible that all valid values lie inside the range, but some points in the range are not valid. This is labeled *imprecise* in Figure 2. Estimated ranges can also mix approximations in different directions in both extremes.

III. Evaluation Framework

A. Experimental Setup

The knowledge of precise and accurate bounds of the QoS of a component is generally only possible *a posteriori* or *post mortem*. For that reason, and also to ensure that all other factors remain equal when introducing inaccuracies and imprecisions in the assumptions, we based our experiments on execution logs from 100 runs of the sample industrial orchestration, which had approximately 160 events recorded per instance.

The prediction was simulated by feeding the time-stamped events from the log for each instance to the constraint-based predictor in the same manner as it would be done at run-time. At each point of prediction, the continuation was constructed from the known original structure of the orchestration and the events observed in the log.

For the baseline experiment under precise and accurate assumptions, we have used the minima and maxima of the component running times recorded in the logs.

Experiments were conducted against a number of execution time limits (SLA objectives), which were chosen based on the recorded running times in the logs to ensure that a certain percentage of the 100 instances violated the limit. These percentages, the failure rates, ranged from small and moderate ones (1%, 5%, 10%) to relatively high ones (20%, 33%) and were, for completeness and observation of trends, extended to unrealistically high ones (failure rates $\geq 50\%$).

The average time spent to make about 160 predictions per instance was between 283 ms to 491 ms, which corresponds to between 1% and 2% of the average instance running time.[2] This suggests that the prototype implementation of the predictor does not impose a significant prediction overhead. The *first* OK / FAIL prediction for every instance was taken as the definitive one.

B. Prediction Quality Metrics

Different measurements can be used to assess the quality of prediction of SLA violations. In our approach we use the evaluation metrics proposed by Salfner et al. [10] in a recent study of online failure prediction methods. All of these metrics are based on counting *true positives* (*TP*, failure predicted and occurred), *false positives* (*FP*, failure predicted, but did not occur), *true negatives* (*TN*, non-failure predicted and no failure occurred), and *false negatives* (*FN*, non-failure predicted, but failure occurred). To these, we add the cases labeled *UP* (number of successes when the method could not predict either OK or FAIL) and *UN* (number of failures in a similar case) All of these metrics are real numbers between 0 (the worst case) and 1 (the best case). We proceed by explaining each one of them in turn.

- Precision $p = TP/(TP+FP)$: when we have a decision on a failure, how precise can we expect it to be?
- Negative predictive value $v = TN/(TN+FN)$: how many success predictions turned out correct.
- Recall $r = TP/(TP+FN+UP)$: how many failures were correctly predicted.
- Specificity $s = TN/(TN+FP+UN)$: how many successes were correctly predicted (a counterpart to recall).
- Accuracy $a = (TP+TN)/(TP+FP+UP+TN+FN+UN)$: how many instances were correctly predicted.

Since in most failure prediction techniques there is a trade-off between precision and recall, as pointed out by Salfner et al. [10], a harmonic mean between these two metrics, known as $F = 2pr/(p+r)$, can be introduced. In our case, it also tends to have a value close to 1. (We defer discussion of some additional adaptation-specific metrics to Section IV-D.)

C. Choice of Constraint Solver

In our experiment, we used the Eclipse CLP system [11] with its interval constraint (*ic*) solver, which is licensed as open source and freely available. The solver supports linear and non-linear arithmetic constraints (equalities and inequalities) over real and integer variables that range over intervals of finite or infinite size (with the finite domain as the special case). These capabilities fit well with the shape of the CSP generated from the continuation [1], which may contain non-linear (min/max, multiplicative) and disjunctive constraints, and where ranges of the component QoS values are represented as intervals. Other, publicly and commercially available constraint solvers can be used as well, as long as they have can express the same kind of constraints, or in special cases where the constraints can be simplified, e.g., made linear.

Some constraint solvers are better suited for some constraint domains and classes of constraints than the others. The results obtained from constraint solvers are sets of values for the constrained variables that are always complete (no solution is left out), but maybe not correct (they may contain values that are not part of any solution [2]). More precise constraint solvers would be able to narrow down the value sets closer to the actual answers, and consequently can detect inconsistent constraints stores, which translates in more precise failure or success predictions.

For our evaluation, the prediction of SLA violation or success depends on the ability of the constraint solver to deduce unsatisfiability in either the OK or FAIL case of the CSP from Figure 1 as early as possible in the lifetime of the executing instance for which the prediction is made, as discussed in the Section IV-C. The results in this paper illustrate what can be done with a state-of-the-art, open source constraint solver, but do not rule out a better quality of prediction with more advanced solvers or in special (e.g., linear) cases.

IV. Experimental Results

A. Baseline Case

The quality prediction metrics for the baseline case (i.e., using the most precise accurate assumptions about the bounds of the component QoS value ranges) are shown in Table I. Columns *u* and *m* are treated in Section IV-D.

Precision *p* tends to increase with the fault rate, because at small fault rates the SLA failures are so rare that even a single false positive diagnosis constitutes a significant proportion of the predicted failures. Recall *r* also tends to be a value close to 1, generally higher than *p*, except for the fault rate 10%, where unpredicted positives (*UP*) constitute a significant fraction of true positives (*TP*). The *F*-measure ranges between *p* and *r* and it amplifies the effects of the worse among them. Specificity *s* and negative predictive value *v* maintain values close or equal to 1 across all failure rates. The accuracy *a* is consistently high across failure rates, and for the low to medium failure rates (between 0% to 10%) it ranges between 97% and 99%.

B. Simulating Inaccurate and Imprecise Assumptions

To compare with the baseline case, we run a number of experiments with inaccurate and imprecise assumptions (Figure 2). For clarity of data presentation, the results presented in the text that follows do not mix over-approximation of one bound with under-approximation of the other bound.

[2]The experiments were performed on a low-end Intel x64 machine with solid-state disk and 4GB RAM, running Mac OS X 10.7.3.

Failure	Results						Basic Metrics					Aggregate Metrics		
rate (%)	*TP*	*FP*	*TN*	*FN*	*UP*	*UN*	*p*	*r*	*s*	*v*	*a*	*F*	*u*	*m*
0	0	0	99	0	0	1	–	–	0.9900	0.9900	0.9900	–	–	–
1	1	1	98	0	0	0	0.5000	1.0000	0.9899	1.0000	0.9900	0.6667	0.6644	1.0000
5	5	1	94	0	0	0	0.8333	1.0000	0.9895	1.0000	0.9900	0.9091	0.9047	1.0000
10	7	0	90	0	3	0	1.0000	0.7000	1.0000	1.0000	0.9700	0.8235	1.0000	0.8235
20	17	2	78	0	3	0	0.8947	0.8500	0.9750	1.0000	0.9500	0.8718	0.9331	0.9189
25	24	1	74	0	1	0	0.9600	0.9600	0.9867	1.0000	0.9800	0.9600	0.9732	0.9796
34	33	2	64	0	1	0	0.9429	0.9706	0.9697	1.0000	0.9700	0.9565	0.9561	0.9851
50	49	2	48	0	1	0	0.9608	0.9800	0.9600	1.0000	0.9700	0.9703	0.9604	0.9899
67	66	2	30	0	1	1	0.9706	0.9851	0.9091	0.9677	0.9600	0.9778	0.9388	0.9763
75	71	2	23	0	4	0	0.9726	0.9467	0.9200	1.0000	0.9400	0.9595	0.9456	0.9726
100	100	0	0	0	0	0	1.0000	1.0000	–	–	1.0000	1.0000	–	–

Table I
COMPARATIVE INDICATORS FOR QUALITY OF PREDICTION IN THE CASE OF THE MOST PRECISE AND ACCURATE COMPONENT QoS ASSUMPTIONS.

Figure 3 shows the comparison of precision, accuracy, specificity, and recall for three groups of inaccurate and imprecise assumptions. The first group of tightly clustered thick lines in the top part of each graph, marked with empty and full circles, pluses, and asterisks, correspond to predictions that were based on *inaccurate* assumptions. The baseline component ranges $[a,b]$ are here replaced with tighter ranges $[a',b']$, $a < a' < b' < b$, that cover 98%, 90%, 80%, 50% and 30% of the original range $[a,b]$.

The great degree of overlap between the lines in this group can be explained, in our experiment, by looking into the distributions of the execution times for individual service components. It turns out that the spread of these time ranges is measured in tens or hundreds of milliseconds, i.e., it is about two orders of magnitude smaller when compared to the total orchestration execution time. Therefore, under-approximating such already narrow ranges does not have a big impact on the generated constraints. In other situations, where components have larger QoS value spreads, deviations from the baseline due to under-approximation should be expected to be greater than in our experiment.

The second group of lines are dashed and correspond to the cases of accurate, but imprecise assumptions where the baseline component ranges $[a,b]$ are replaced with wider ranges $[a/k, b\times k]$,where the lower and the upper bound are under- and over-approximated, respectively, by factor k that is 1.5 (□), 2 (◊) and 10 (■). While the first two over-approximation factors exhibit gradual, but not radical, deterioration of prediction quality, the latter is an order-of-magnitude over-approximation which leads to significantly worse prediction quality, which is hardly unexpected.

The third group are the thin solid lines on the graphs that also correspond to the cases of accurate, but imprecise assumptions, but differ from the second group of lines in that the baseline component ranges $[a,b]$ are here replaced with even wider ranges $[0, b\times k]$, where the lower bound is lost, and the upper bound is over-approximated by a factor k that is 1 (△), 1.5 (▽) and 2 (▲). (Due to overlapping, these appear as 6-spike stars on some graphs.)

It turns out that loosening the lower bound, however small, hurts

prediction quality even more than an order-of-magnitude over-approximation of the upper bound. The reason for this is that an assumed lower bound of 0 prevents the constraint-based predictor from discarding the case when there is definitely no failure, and therefore the predictor is unable to predict failure in a large number of cases. The importance of the lower bounds in constraint-based prediction is therefore at odds with the usual perception that upper bounds are key predictors of failures.

C. Prediction Timing

Figure 4 shows the distribution of the time at which a (true or false) negative prediction was made, measuring from the orchestration start across different SLA failure rates for the baseline case. The red line above represents the time limit for each fault rate. For small and medium fault rates, the picture shows that on average the predictor was able to issue a negative prediction relatively early, approximately within one fifth of the instance execution time. Only for very large failure rates (50% and over) the prediction was made much later, closer to the execution time limit. The reason for that is that, other things being equal, a higher execution time limit makes it easier for the constraint solver to discard the possibility of an SLA violation.

The left graph of Figure 5 compares the average true negative prediction times between the three group of inaccurate assumption cases, and the two groups of accurate, but imprecise ones (using the same markings as before). Again, the inaccurate cases behave closely to the baseline case, while a later prediction due to the loss of precision is visible in the cases of imprecision (with and without the lower bound). The worst case here is the tenfold over-approximation, which produces the most delayed true negative predictions.

Figure 6 shows the distribution of lead times between the (true or false) positive predictions and the SLA running time limit, for the baseline case. The results suggest a relatively stable average lead time of about 9 sec. across the failure

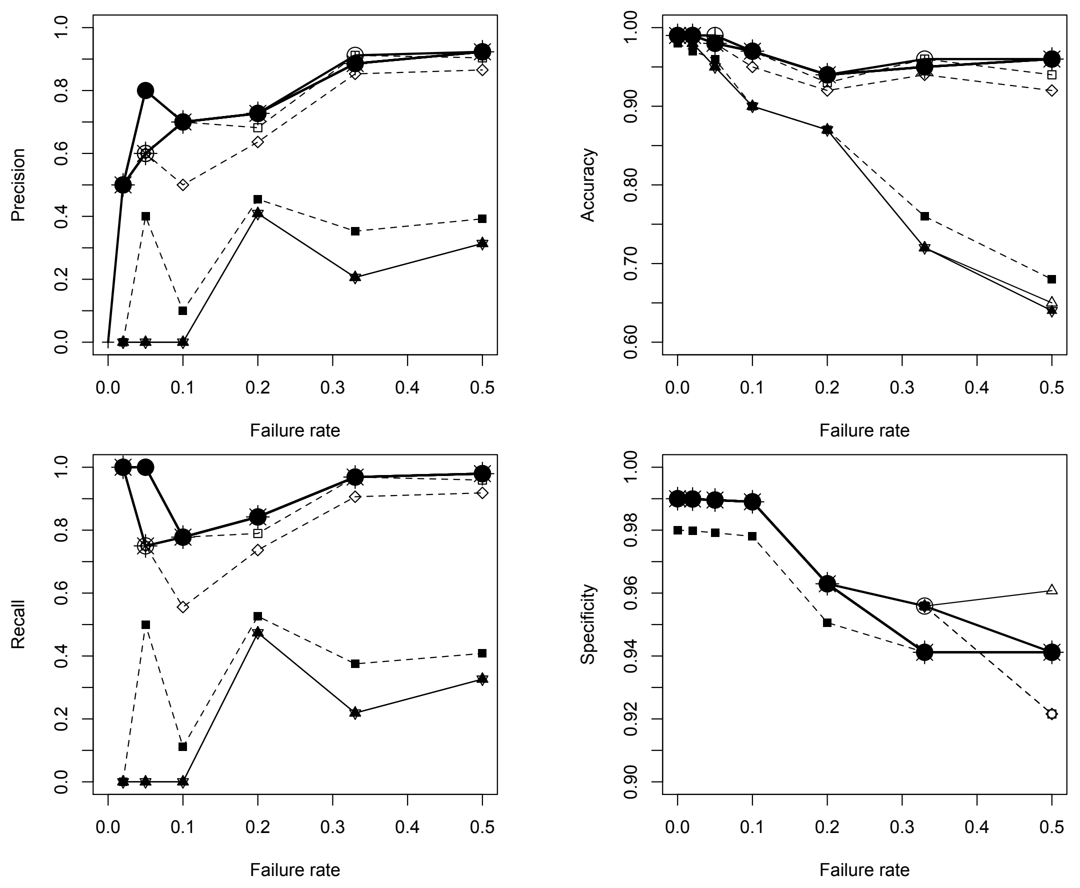

Figure 3. Comparative measures of prediction quality for inaccurate and imprecise assumptions.

rates, which corresponds to about one fourth of the average instance execution time. Of course, whether such lead time is sufficient for adaptation purposes depends on the type of adaptation to be employed. Figure 5, right, shows the comparison between the average true positive prediction time leads under inaccurate and imprecise assumptions, which qualitatively follows the precision and accuracy pattern. The earliest true positive predictions are obtained under the accurate assumptions (with varying precision), while the imprecise assumptions that ignore the lower bounds are practically unable to predict true positive before reaching the time limit corresponding to each of the fault rates.

D. Suitability for Predictive Adaptation

The usability of a SLA failure prediction method in the context of predictive adaptation (i.e., adaptation triggered ahead of the actual failure with the goal to avoid it or to mitigate its effects) depends not only on the type of adaptation, but also on some general characteristics of the prediction methods such as examined by Sammodi et al. [6], who have identified several helpful metrics for that purpose.

The measure $u = 2ps/(p+s)$ harmonizes prediction precision and specificity, and quantifies the extent to which the prediction method manages to avoid unnecessary adaptations. The values for u in Table I tend to be high and to move in harmony with precision p, because specificity is (in the baseline case) always equal or very close to 1. The degradation of the u metrics under various types of inaccurate or imprecise assumptions, shown on the left side of Figure 7, follows a similar pattern to that of precision in Figure 3.

The measure $m = 2rv/(r+v)$, on the other hand, harmonizes recall and negative predictive value, and quantifies the extent to which the prediction method manages to avoid missed adaptations. Again, since v is always very close to 1, the values for m in Table I follow those of recall r. Figure 7, right, shows that m degrades under inaccurate and imprecise assumptions following a pattern similar to that for recall in Figure 3.

V. Conclusions

Experiments using the prototype predictor based on a realistic experimental service orchestration case suggest that constraint-based SLA failure prediction, under the precise and accurate component QoS assumptions, is a very reliable prediction method that offers a good combination of precision, specificity, recall, and other quality of prediction

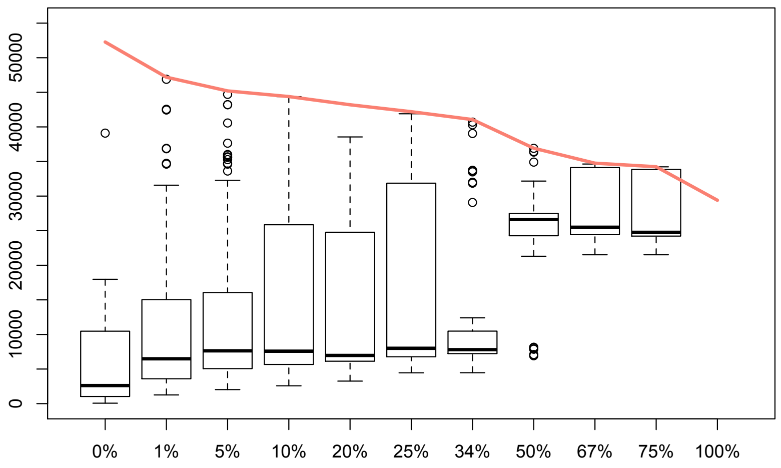

Figure 4. Distribution of negative prediction times (ms).

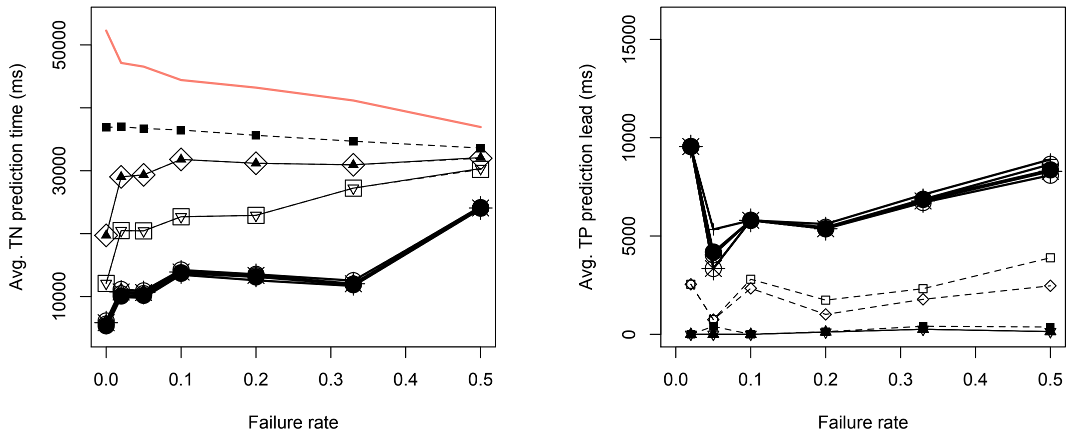

Figure 5. Comparative true negative/positive prediction times for inaccurate and imprecise QoS assumptions.

metrics. For reasonable failure rates, the method is able to make an early prediction of the non-failure of an SLA according to execution time, as well as to predict failures significantly ahead of actual failures.

When the accuracy and precision of the component QoS assumptions deteriorates, the method's quality of prediction generally tends to deteriorate gracefully, unless gross (order-of-magnitude) imprecisions are introduced. However, the method is very sensitive to the loss of information on the lower bounds for component QoS ranges, which tends to decrease the quality of prediction more than a significant over-approximation of the upper bounds. This requires special care when collecting the component QoS information using external sources and/or collecting events from logs and monitoring.

The prediction quality attributes, such as precision, recall, negative predictive value, and specificity, suggest that this prediction method can be expected to provide a good basis for predictive run-time adaptation with the aim of avoiding detected failures or compensating their effects.

Our future work will aim at validating the approach in live, production deployments, and experimentally evaluating other possible uses for the constraint predictor, such as reasoning on possible QoS ranges in (non-)failure cases and detecting events in the orchestration that lead to SLA failure or compliance.

VI. Acknowledgments

The authors are thankful to Phillip Leitner of the Technical University of Vienna for providing the experimental workflow and the execution logs, as well as to Andreas Metzger, Eric Schmieders, and Osama Sammodi of the Ruhr Institute for Software Technology (Paluno) for suggesting the metrics used for evaluating the quality of the constraint-based prediction.

The research leading to these results has received funding from the European Community's Seventh Framework Programme under the Network of Excellence S-Cube (Grant

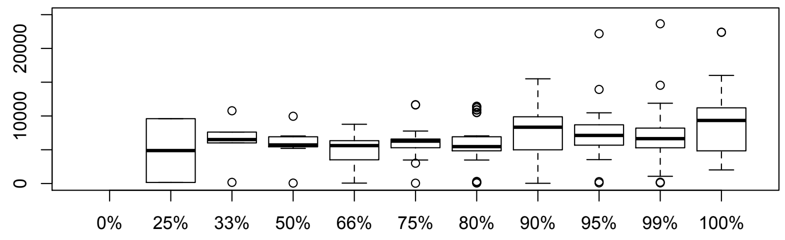

Figure 6. Distribution of positive prediction time leads ahead of failure (ms).

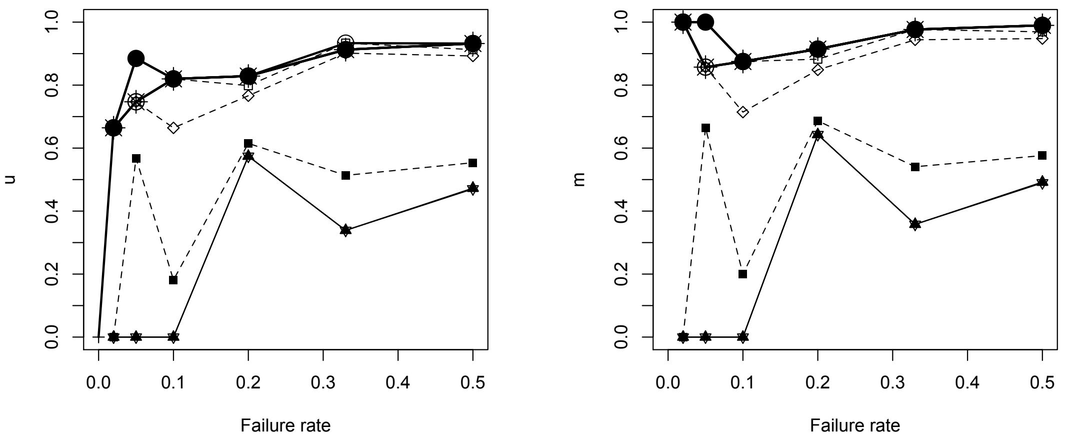

Figure 7. Adaptation-related metrics under inaccurate or imprecise QoS assumptions.

Agreement n° 215483). The authors were also partially supported by Spanish MINECO project 2008-05624/TIN *DOVES* and CM project P2009/TIC/1465 *PROMETIDOS*.

References

[1] D. Ivanović, M. Carro, and M. Hermenegildo, "Constraint-Based Runtime Prediction of SLA Violations in Service Orchestrations," in *Service-Oriented Computing – ICSOC 2011*, ser. LNCS, G. Kappel, H. Motahari, and Z. Maamar, Eds., no. 7084. Springer Verlag, December 2011, pp. 62–76, best paper award.

[2] R. Dechter, *Constraint Processing*. Morgan Kauffman Publishers, 2003.

[3] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar, "Monitoring, prediction and prevention of sla violations in composite services," in *ICWS*. IEEE Computer Society, 2010, pp. 369–376.

[4] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, and F. Leymann, "Runtime prediction of service level agreement violations for composite services," in *IC-SOC/ServiceWave Workshops*, ser. Lecture Notes in Computer Science, A. Dan, F. Gittler, and F. Toumani, Eds., vol. 6275, 2009, pp. 176–186.

[5] A. Metzger, O. Sammodi, K. Pohl, and M. Rzepka, "Towards pro-active adaptation with confidence: Augmenting service monitoring with online testing," in *Proceedings of the ICSE 2010 Workshop on Software Engineering for Adaptive and Self-managing Systems (SEAMS 10)*, Cape Town, South Africa, 2010.

[6] O. Sammodi, A. Metzger, X. Franch, M. Oriol, J. Marco, and K. Pohl, "Usage-based online testing for proactive adaptation of service-based applications," in *COMPSAC 2011 – The Computed World: Software Beyond the Digital Society*. IEEE Computer Society, 2011.

[7] E. Schmieders and A. Metzger, "Preventing performance violations of service compositions using assumption-based runtime verification," in *ServiceWave 2011*, ser. LNCS, A. Zisman, I. Llorente, S. M., A. W., and V. J., Eds. Springer, 2011.

[8] E. Schmieders, A. Micsik, M. Oriol, K. Mahbub, and R. Kazhamiakin, "Combining SLA prediction and cross layer adaptation for preventing SLA violations," in *2nd Workshop on Software Services: Cloud Computing and Applications based on Software Services*, 2011.

[9] P. Leitner, W. Hummer, and S. Dustdar, "Cost-based optimization of service compositions," *IEEE Transactions on Services Computing*, vol. 99, no. PrePrints, 2011.

[10] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Comput. Surv.*, vol. 42, no. 3, 2010.

[11] K. R. Apt and M. G. Wallace, *Constraint Logic Programming Using ECLIPSE*. Cambridge University Press, 2007.