Continuous Certification of Non-Functional Properties Across System Changes: Appendix

Anonymous for double-blind review

A Experimental Process

Experiments have been run on an Apple MacBook Pro with 10 CPUs Apple M1 Pro, 32 GBs of RAM, operating system macOS Ventura 13.1, using Python 3.10.9 with libraries *numpy* v1.24.1 [2], *pandas* v1.5.1 [5,7] and *scikit-learn* v1.2.0 [6].

Our experimental process consists of: *i)* adaptive system model creation, *ii)* dataset creation, *iii)* execution of the certification scheme in this paper, *iv)* execution of SOTA, *v)* evaluation.

Adaptive model creation consists of a set of isolation forests [3,4], each forest trained on a specific component. Isolation forest is a ML solution having the same structure of a random forest, with decision trees split at random during training. The number of splits necessary to classify a data point is equivalent to the path from the root to the corresponding leaf of a tree. At inference time, the decision function is the average path length necessary to isolate a data point. The shorter this measure is, the more the data point is likely to be an anomaly, as a consequence of random trees splitting. We created then a training dataset for each system in Section 7.1 by extracting normal and anomalous data points from the available data with ratio 0.9–0.1 and trained each forest on a different component; dataset cardinality is $1,000$ and train–test split is 0.75–0.25.

Dataset creation generates for each experimental setting in Table 2 in the main paper (see Section 7.1) 10 datasets as follows. First, for each component, we randomly extracted if it is critical or not according to probability *non critical*. Then, for each data point (up to $1,000$), we randomly extracted if the data point correspond to a normal trace or a change according to probability *change*. If normal, the data point value is randomly extracted from the dataset of normal traces. If not, we first extracted the cause of the change in environmental changes, code changes with impact on the behavior, or code changes with no impact on the behavior according to probabilities Δ_b, Δ_c with cascading, and Δ_c. In the first case, we randomly extracted the number of affected components according to probability $n(comp)_b$ and the specific components. In particular, $n(comp)_b$ refers to the probability that one component is involved, and the probability that i components out of N are affected is retrieved as $(N+1-i)\cdot(1-n(comp)_b)/\sum_{j=1}^{N}$ (i.e., the probability that i components are affected decreases linearly as i increases). Fixed i, the specific components are chosen at random according to uniform probability. Data point value is randomly extracted from the dataset of anomalous traces. In the second case (code with impact on behavior), we randomly extracted the extent of the change (minor or major) according to probability *minor* We then extracted the affected components as outlined above according to probabilities $n(comp)_{min}$ and $n(comp)_{maj}$ depending on the text. We then extracted the components whose behavior changed as a result of code change among the remaining components according to random uniform probability. Data point value is randomly extracted from the dataset of anomalous traces. In the third case (code with no impact on the behavior), the data point value is randomly extracted from the dataset of normal traces. The outcome of each extraction is annotated in the dataset and available at `https://anonymous.4open.science/r/certification-across-system-changes-7B5D`.

Execution of our scheme. We applied our scheme on the generated dataset. Adaptive system model detects behavior changes: a data point corresponds to a behavioral change if at least one isolation forest detects an anomaly; affected components $\{c_i\}$ correspond to the set of isolation forests $\{\mathrm{ifo}_i\}$ having detected an anomaly. For each detected behavioral change, we annotate it as code change with impact on behavior if it is marked as such in the dataset, behavioral change otherwise. The remaining data points are annotated as normal or code changes according to corresponding annotation. Finally, for each data point annotated as code change regardless its impact on the behavior, we annotated its extent and the presence of critical components according to corresponding annotations.

Execution of state of the art scheme. We applied SOTA on the generated dataset. Each data point is annotated as normal if the corresponding annotation is normal or behavioral change. The remaining data points are annotated as code change, with code change extent copied appropriately. Involved components in each code change correspond to components directly affected by the change (i.e., components whose code changed as a direct consequence of code changes, no cascading effects). The presence of critical components is finally annotated accordingly.

Evaluation. We generated 10 datasets for each experimental setting in Table 2 in the main paper and target system. We then executed our scheme and SOTA according to the procedure above, comparing the results retrieved by the schemes with the ground truth in the generated dataset. Concerning the evaluation of the adaptive system model, we applied our isolation forest-based classifier on the generated dataset for each execution of each experimental setting and averaged the results.

B Additional Experimental Results

Table 1 shows the accuracy (ACC), precision ($PREC$), and recall (REC) of our adaptive system model $\mathcal{B}$ based on isolation forest in the identification of behavioral changes in the three datasets D_{MS}, D_{SN}, and D_{TT}. We note that, $PREC$ is $\geq$97.9% in all systems, meaning that the risk of false positives is negligible. ACC and REC of *MS* and *TT* also show very high performance ($\geq$83%). On the contrary, *SN* shows low ACC=61% and REC=47%, meaning that 40% and

Table 1. Results of experimental evaluation of adaptive service model

Dataset	$\boldsymbol{ACC}$	$\boldsymbol{PREC}$	$\boldsymbol{REC}$
D_{MS} [1]	0.8735	0.9971	0.8314
D_{SN} [1]	0.6089	0.9794	0.4747
D_{TT} [8]	0.9577	0.9969	0.9451
AVG	0.8133	0.9911	0.7504

53% of the changes are not correctly identified or detected. The reason is that normal and anomalous response times in SN are often compatible.

References

1. Gan, Y., Zhang, Y., Cheng, D., Shetty, A., Rathi, P., Katarki, N., Bruno, A., Hu, J., Ritchken, B., Jackson, B., Hu, K., Pancholi, M., He, Y., Clancy, B., Colen, C., Wen, F., Leung, C., Wang, S., Zaruvinsky, L., Espinosa, M., Lin, R., Liu, Z., Padilla, J., Delimitrou, C.: An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. In: Proc. of ASPLOS 2019. Providence, RI, USA (April 2019)
2. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825) (Sep 2020)
3. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation Forest. In: Proc. of IEEE ICDM 2008. Pisa, Italy (December 2008)
4. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation-Based Anomaly Detection. ACM Trans. Knowl. Discov. Data **6**(1) (2012)
5. Wes McKinney: Data Structures for Statistical Computing in Python. In: Proc. of SciPy 2010. Austin, TX, USA (June – July 2010)
6. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011)
7. The pandas development team: pandas-dev/pandas: Pandas (Feb 2020). `https://doi.org/10.5281/zenodo.3509134`
8. Zhou, X., Peng, X., Xie, T., Sun, J., Xu, C., Ji, C., Zhao, W.: Benchmarking microservice systems for software engineering research. In: Proc. of ICSE 2018. Gothenburg, Sweden (May, June 2018)