Correlating QoE and Technical Parameters of an SAP System in an Enterprise Environment

Kathrin Borchert*, Matthias Hirth*, Thomas Zinner*, Decebal Constantin Mocanu†
*Institute of Computer Science, University of Würzburg,
Am Hubland, 97074 Würzburg, Germany
Email: [kathrin.borchert|matthias.hirth|zinner]@informatik.uni-wuerzburg.de
†Department of Electrical Engineering, Eindhoven University of Technology, the Netherlands
Email: d.c.mocanu@tue.nl

***Abstract*—The impact of waiting times on the Quality of Experience (QoE) in enterprise and working environments has not been in the focus of current research. This mostly stems from two factors: i) the high complexity of enterprise systems exacerbates the exact monitoring of relevant application response times on user granularity and ii) disturbances of the day-to-day business by user studies resulting in additional costs due nonproductive times. This paper approaches these challenges by combining non-intrusive application monitoring of response times and subjective user ratings on the perceived application performance. We evaluate the possibility of predicting the QoE based on the objective measurements using different machine learning approaches. The results imply a high correlation for specific users, but do not allow to derive a generic model.**

I. Introduction

Distributed enterprise IT infrastructures mostly consist of thin client architectures. Users interact with universal and basic terminals, while applications requiring computations or data storage are carried out in central data centers. The applications running in the data center range from pure office applications to business operations and customer relations software. Business operations and customer relations software are typically an integrated solution covering accounting, controlling, sales and distribution, purchasing, manufacturing, stock keeping and human resource management. The corresponding software is complex, consists of a large number of use case specific modules, and is typically distributed across numerous physical and virtual machines. User transactions like the update or the selection of information have to be stored persistently in corresponding data bases. Depending on the size of a data base, the current system or network load, and the computational complexity, user triggered transactions may last up to tens of seconds. Due to the blocking nature of most of those transactions, the employee has to wait until corresponding information are retrieved or inserted before proceeding with the working task. The perceived waiting times can put a significant stress on the employees relying on the technical system, leading to a degradation of the users Quality of Experience (QoE). However, in contrast to end-consumer applications, there is little research on QoE models for enterprise applications.

Performing QoE studies in enterprise environments is challenging, mainly because of two reasons. First, the IT environment is highly complex and hard to analyze, due to the scale of the infrastructure, the diverse hard- and software components, and the proprietary components. Second, user studies in this environment are extremely costly, because the participants need to be familiar with the system to test, i.e., company employees are required and the user studies directly influence the daily business of the company. Additionally, it is often not possible to set up a dedicated test system for the user studies, thus the test parameters cannot be influenced directly.

This paper approaches these challenges by combining non-intrusive application monitoring of response times and subjective user ratings on the perceived application performance. Firstly, we monitor the system performance of specific transactions of an SAP module for a large user group. In parallel, we conducted a large scale user study. Both measurements are performed during live operations of the system. We then apply machine learning approaches to evaluate if it is possible to correlate the technical measurement with the perceived subjective quality.

The paper is structured as follows. Section II discusses related work on machine learning, the impact of waiting times on the QoE. The applied methodology and the datasets are described in Section III. Section IV presents the results of our evaluation, and Section V concludes the paper.

II. Background and Related Work

A. Impact of Waiting Times on QoE

The impact of delays on the QoE of interactive systems and especially web-based systems was already extensively studied. It has been shown that delay can—depending on its severity and the interaction type—impact the user interaction with a system, prolong task completion times, and ultimately lower the QoE. The assessed delay impacts largely focuses on web site loading delays (see e.g., [1]–[6]). The latter further quantified monetary effects of delay such as loss of revenue [6], [7] or lowered system interactions [5], [7], [8]. So far, most of the studies focus on web-based systems, because they are rather easy to access and can also easily be modified, e.g., to induce artificial delays for user studies. Even if these findings give a first impression on how delays are perceived by users of interactive systems, it remains unclear to which extend they can also be applied in an enterprise environment.

B. QoE and Machine Learning

Up to our knowledge, there isn't any research done on Machine Learning (ML) algorithms for estimating user QoE in the enterprise and working environments. However, in our previous work [9], we have found that ML methods can be a good solution to estimate the video quality perceived by users in video services streamed over impaired networks, while Bao et al. [10] showed that an ML approach may be a good solution to improve user experience in computer networks. Motivated by these results, in this paper, we perform a comparative study between various ML algorithms in order to estimate the user perceived quality of working experience in an enterprise environment. As in our setup, we have to estimate if the user is *happy* or *unhappy* with the quality of the business transactions, which took place in the last period of time. This means that we have to scrutinize ML algorithms suitable for binary classification problems. However, it is almost impossible to know a priori which class of ML algorithms would perform better in a specific domain, and for this reason we have selected to study algorithms which have completely different underlying mechanisms. Thus, we have considered for comparisons the following three classifiers: (1) a widely used ML algorithm, i.e. Support Vector Machine (SVM) [11]; (2) an ensemble method, i.e. Gradient Boosting (GB), due to its powerful discrimination capabilities [12]; and (3) a deep learning method, i.e. Deep Neural Networks (DNN), which constitute state-of-the-art in supervised learning nowadays [13]. A full review of these algorithms do not constitute a goal of this paper, but the interested reader is strongly advised to consult the corresponding references for more technical details.

III. Applied Methodology and Data Sources

The collected data contain technical measurements and user feedback which have been gathered during a working week in June 2015. In total 121 employees working at different branch offices at one specific SAP module were selected to participate in the study.

1) SAP Monitoring Data: The technical data from the SAP system is taken from the integrated performance monitoring of the system. Each business process performed, e.g., updating a customer's information, results in a certain number of dialog-based transactions, i.e., searching the specific customer, opening the customer's details, updating the information and closing the details. Each of this dialog based transaction consists of single or composed functions. The data of one transaction includes various information, e.g. the response time, the size of transferred data or the number of database interactions. The response time defines the time the system needs to complete the request of the user and send the response.

In this study we analyze a subset of 13 different transaction types which have been selected by experts. The subset contains the most important transactions of the business area in which the participants are working. Typically, the amount of transaction types varies between 40 and 75 per business area depending on application usage and the specific SAP module. During the monitoring period the response times of 745.000 transactions were measured.

2) User Feedback: The user feedback is collected using a newly developed application that aims at minimizing the effect of the user study on the daily business. Thus, it is not possible to use common rating scales widely used in dedicated subjective studies or gather user feedback at very high frequencies. To this end, the application shows a pop-up to the participants only once per hour, asking to rate the performance perceived during the last hour. The participants can rate by choosing between a happy smiley which defines a good or acceptable system performance and a sad smiley which represents a bad performance. If no response is given within a short time interval the pop-up is closed and the rating is marked as missing. The client side of the application is written in C# and supported by a server component written in PHP for storing the user ratings. During the test period, the pop-up was shown 4149 times, resulting in 2398 user ratings and 1751 responses marked missing.

IV. Initial Evaluation

To evaluate our proposed approach, we have devised two sets of experiments. Firstly, we have analyzed the performance of the ML algorithms while training the models under scrutiny on data coming from all users. In the second type of experiments, we have trained a separate model for each user to understand better the underlying user experience.

A. Data preprocessing

As each user has voted at the end of each hour the quality of its working experience during that hour, a data point is given by the quality of all the transactions performed by that user in that specific hour. For any transaction we have considered as being representative two main characteristics, the duration of the transaction, and when the transaction was finished (as transactions which end early in the hour may be perceived differently by the user than the most recently ended transactions). Since the number of transactions performed varies from one hour to another, we equalized the number of input features of any data point as follows. For all transactions of the same type performed during an hour, we have computed the mean, the standard deviation, and the L^2 norm of their characteristics. This leads in total to 78 input features (13 transaction type x 2 characteristics x 3 features/characteristic) for a data point. Finally, as usual in ML preprocessing, we have normalized the data to have zero mean and unit variance. For any data point the output is given by a binary value meaning that the user is *happy* or *unhappy* during that specific hour.

B. Settings, implementations, and evaluation metrics

We assessed SVM with radial basis function and GB with 100 estimators using their *scikit-learn* [14] implementations. For DNNs we used the Keras library [15], setting two hidden layers of 50 neurons each and dropout. Since the dataset has a different amount of data points in each class (i.e.

TABLE I
GENERALIZED MODELING CASE. CLASSIFICATION PERFORMANCE IN TERMS OF THE FOUR METRICS ANALYZED.

Model	Balanced Accuracy [%]	Accuracy [%]	Precision [%]	Recall [%]
SVM	58.2	57.6	90.8	57.4
GB	58.7	58.7	89.0	**60.3**
DNN	**58.8**	**59.5**	**90.9**	59.8

imbalanced dataset), to avoid biasing the ML algorithms, we have used for training 66% randomly chosen data points from the class with less data, while from the other class we have just picked randomly the same amount of samples. For each class, the remaining of the data points were used for testing. As this yields a different amount of testing data points in each class, we used four standard metrics to evaluate the models performance, i.e. Accuracy (the percentage of correctly classified data points from the total number of data points), Balanced Accuracy (a metric specially conceived to avoid inflated performance estimates by accuracy on imbalanced datasets), Precision (positive predictive value), and Recall (sensitivity).

C. Generalized modeling

In this set of experiment, we have built a generalized model for all users. The results depicted in Table I show that none of the models is capable to obtain a good performance, all of them performing just a bit better than random choice with a small advantage for DNN.

D. Personalized modeling

Intrigued by the above results and to understand better the users' choice, in the second set of experiments we have built a personalized ML model for each user. The results show that there is a very high variability in the working quality experienced by the users. More than that, by analyzing the balanced accuracy performance metric, depicted in Figure 1 we may observe that for approximately 10-20% of the users all models perform clearly worse than a random classifier, while for 5-10% of the users almost all models achieve a very good performance, i.e. over 80% balanced accuracy. It is interesting to see that in this set of experiments SVM outperforms DNN, this being a normal situation as DNNs needs in general a higher amount of data to achieve a good performance. Overall, these results suggest that it is very hard to build a general prediction model using just non-intrusive application monitoring, but it may be much easier to build good personalized prediction models for each user separately.

V. CONCLUSION

This paper provides a first step towards a QoE prediction based on application response times for business applications in an enterprise environment. For that, we evaluate the accuracy of different machine learning approaches on a dataset comprising response times of a productive SAP system and subjective ratings of users working with this system. The results suggest that it is very hard to build a general prediction model using just non-intrusive application monitoring, but it may be much easier to build good personalized prediction models for each user separately. Future work will mainly aim at further data acquisitions to achieve a better accuracy of the learned models.

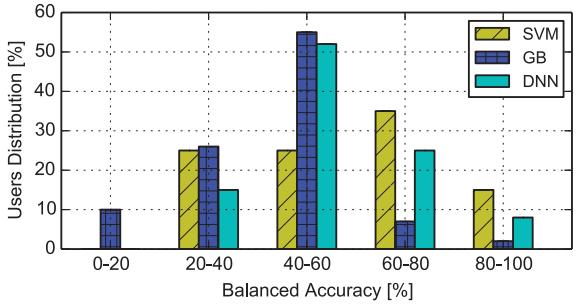

Fig. 1. Classification performance in terms of balanced accuracy for the personalized modeling case.

ACKNOWLEDGEMENT

This work is supported by the Deutsche Forschungsgemeinschaft under Grants HO TR 257/41-1 "Trade-offs between QoE and Energy Efficiency in Data Centers" and by kubus IT. The authors alone are responsible for the content.

REFERENCES

[1] S. Egger, T. Hoßfeld, R. Schatz, and M. Fiedler, "Tutorial: Waiting Times in Quality of Experience for Web based Services," in *IEEE QoMEX*, 2012.
[2] D. Strohmeier, M. Mikkola, and A. Raake, "The importance of task completion times for modeling web-QoE of consecutive web page requests," in *IEEE QoMEX*, 2013.
[3] I. Arapakis, X. Bai, and B. B. Cambazoglu, "Impact of response latency on user behavior in web search," in *ACM SIGIR Conference on Research & Development in Information Retrieval*, 2014.
[4] J. D. Brutlag, H. Hutchinson, and M. Stone, "User preference and search engine latency," in *JSM Proceedings, Qualtiy and Productivity Research Section*, 2008.
[5] S. Stefanov, "Yslow 2.0," in *China Software Developers Network*, 2008.
[6] G. Linden, "Make data useful," 2006.
[7] E. Schurman and J. Brutlag, "Performance related changes and their user impact," in *Velocity – Web Performance and Operations Conference*, 2009.
[8] J. Brutlag, "Speed matters for Google web search," 2009.
[9] D.C. Mocanu et al., "No-reference video quality measurement: added value of machine learning," *Journal of Electronic Imaging*, vol. 24, no. 6, 2015.
[10] Y. Bao, H. Wu, and X. Liu, "A closed-loop approach in data-driven resource allocation to improve network user experience," in *KDD*. ACM, 2016.
[11] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
[12] P. Bühlmann, "Bagging, boosting and ensemble methods," in *Handbook of Computational Statistics*. Springer, 2012, pp. 985–1022.
[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, 2015.
[14] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[15] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.