

xPACE and TASC Modeler: Tool Support for Data-Driven Context Modeling

Rodrigo Falcão¹, Rafael King¹ and Antônio Lázaro Carvalho²

¹Fraunhofer IESE, Fraunhofer-Platz 1, Kaiserslautern, 67663, Germany

²Computer Science Graduate Program, UFBA, Av. Milton Santos s/n, Salvador, 40170-110, Brazil

Abstract

From a requirements engineering point of view, the elicitation of context-aware functionalities calls for context modeling, an early step aimed at understanding the application contexts and how it may influence user tasks. In practice, however, context modeling activities have been overlooked by practitioners due to their high complexity. To improve this situation, we implemented xPACE and TASC Modeler, which are tools that support the automation of context modeling based on existing contextual data. In this demonstration paper, we present our implementation of a data-driven context modeling approach, which is composed of a contextual data processor (xPACE) and a context model generator (TASC Modeler). We successfully evaluated the results provided by the tools in a software development project.

Keywords

requirements elicitation, context modeling, automation

1. Introduction


Computers have become increasingly ubiquitous, and we are witnessing the rise of applications, sensors, and networks that together deliver smart behaviors to users. Context awareness plays a key role in this game as a core characteristic of ubiquitous computing [1] [2] and is frequently behind the perceived “intelligence” of modern software solutions[3].


Context-aware functionalities are functionalities that consider context to produce a certain system behavior, typically an adaptation or recommendation. From a requirements engineering point of view, the elicitation of context-aware functionalities requires context modeling, an early step involving identification of contextual elements, analysis of accessibility (i.e., which contextual elements have available sources from which their values can be read), analysis of the relevance of these contextual elements for user tasks of interests, and analysis of combinations of contextual elements for these user tasks [4]. The analysis of relevance and combinations can be challenging, though: In a scenario with dozens of contextual elements, how to figure out which contextual elements, particularly in combination with each other, may influence a given user task? As the number of contextual elements increases, the number of potential combinations

In: J. Fischbach, N. Condori-Fernández, J. Doerr, M. Ruiz, J.-P. Steghöfer, L. Pasquale, A. Zisman, R. Guizzardi, J. Horkoff, A. Perini, A. Susi, M. Daneva, A. Herrmann, K. Schneider, P. Mennig, F. Dalpiaz, D. Dell’Anna, S. Kopczyńska, L. Montgomery, A. G. Darby, and P. Sawyer (eds.): *Joint Proceedings of REFSQ-2022 Workshops, Doctoral Symposium, and Poster & Tools Track, Birmingham, UK, 21-03-2022*, published at <http://ceur-ws.org>

✉ rodrigo.falcao@iese.fraunhofer.de (R. Falcão)

ORCID 0000-0003-1222-0046 (R. Falcão); 0000-0001-7107-5390 (R. King); 0000-0002-0013-4565 (A. L. Carvalho)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

grows exponentially. In practice, practitioners have overlooked these context modeling activities due to their high complexity, regarding them as time-consuming, non-intuitive, and error-prone [5]. Hence, opportunities for discovering unexpected context-aware functionalities are missed.

We approached this problem by automating context modeling for the elicitation of context-aware functionalities. For this purpose, we designed a data-driven context modeling process (first introduced in [4]) that has three parts. First, the user task of interest is analyzed; then, contextual data that relates to the user task is collected; and finally, at the core of the data-driven context modeling process, the contextual data is processed to identify relevant combinations of contextual elements – and finally a context model is generated. Note that the model does tell how the context influences the task, but describing requirements based on it is a creative step.

To come alive, the proposed automation in the process requires the implementation of two software components. One is the *Contextual Data Processor*, which is responsible for analyzing a contextual dataset. Its output is used by the other component, the *Context Model Generator*, which is responsible for creating the concrete context model that is expected to support requirements engineers in the elicitation of context-aware functionalities. In this paper, we present our implementations of these software components: *xPACE* and *TASC Modeler*¹, and briefly discuss its application in a software development project at Fraunhofer IESE.

2. Solution overview

Once a user task of interest has been chosen and corresponding contextual data has been collected, a contextual dataset is available. Then the requirements engineer uses TASC Modeler to generate the desired context model, providing the contextual dataset as input, in addition to a metadata file that configures the behavior of both the context model generator – how the contexts will be presented in the model – and the contextual data processor – how the dataset should be analyzed. The detailed specification of the input files can be found with the source code¹. TASC Modeler forwards the dataset and the metadata to xPACE, which analyzes the data and returns the findings in a data structure named *standardized task-specific contexts*, which is created to decouple the data analysis from the context model generation. Then TASC Modeler translates the standardized task-specific contexts provided by xPACE into a context model representation. Figure 1a shows the component diagram of the solution.

In our case, the context model is represented as a directed acyclic graph with one root node. This representation supports data-driven context modeling approaches, expressing how the context influences individual user tasks. Each path from the root node towards a leaf describes how a context influences a user task of interest. Figure 2 shows an example. Consider the user task “Prepare a cup of coffee”. Each of the two paths contains a set of instantiated contextual elements that, together, were found to influence the task (e.g., “When location = WORK and time = AFTERNOON then user prepares a cup of coffee”, i.e., the context “location =WORK and time = AFTERNOON” influences the user task “Prepare a cup of coffee”, according to the model). Note in the example how the model structure differentiates intrinsic contextual elements (those that intrinsically characterize the entity “cup of coffee”, which is the object of the task), from extrinsic ones.

¹Source code available: <https://github.com/rmfalcao/tasc-modeler-xpace>

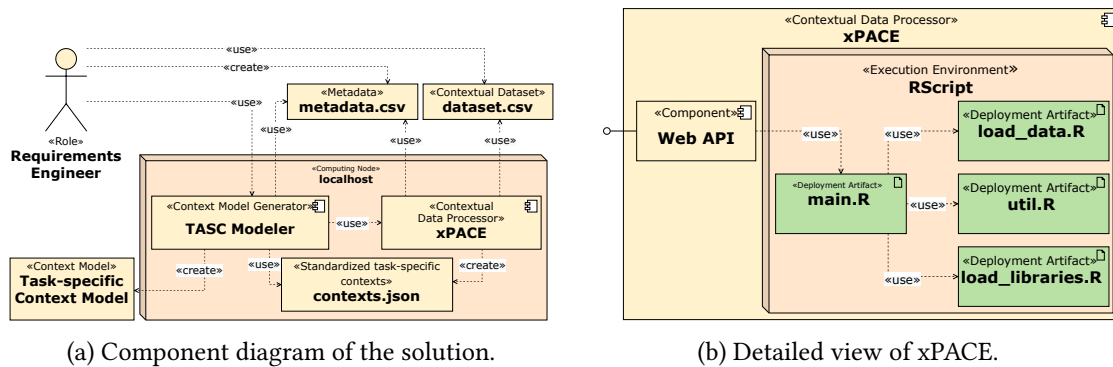


Figure 1: Architecture of the solutions (functional view at runtime).

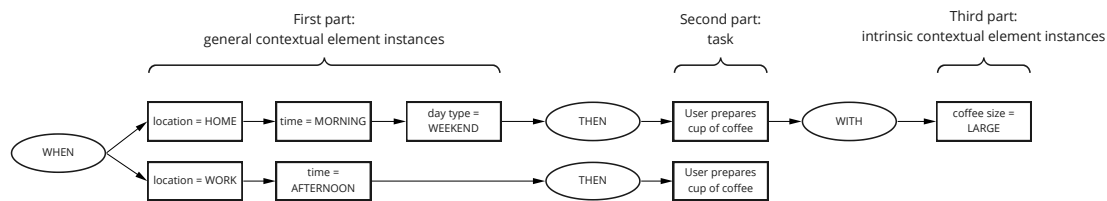


Figure 2: Example of a task-specific context model.

3. xPACE – eXtended Pairwise Analysis of Contextual Elements

The *eXtended Pairwise Analysis of Contextual Elements* (xPACE) was implemented using a strategy that can be divided into two parts. First, it uses statistical methods to search for correlations between pairs of contextual elements (CEs) in the contextual dataset. Whenever correlations are found, the algorithm identifies which CE instances (i.e., concrete values of the CEs) are correlated. This is necessary because it is not enough to know that two CEs relate to each other (e.g. “location” and “time”), but it also needs to be know which instances relate to each other (e.g., “at home” and “evening”). After that, the pair is ordered to express the direction of the relationship. After analyzing all pairs of CEs, we end up with a *list of pairwise relations*, which is a set of directed pairwise relations among CE instances. The second part of the strategy takes the list of pairwise relations and builds a graph G by treating each pair as an edge of the graph. When all pairs have been added to G , each path in the graph starting from a root node and ending on a leaf node will represent a relevant combination of CEs that were found by xPACE to influence the user task. Figure 3 contains an activity diagram that illustrates the data processor algorithm. The employed statistical methods are named in the corresponding steps.

The algorithm is implemented using the R language², and a Web API component implemented in Java provides handy access to it. The core takes as input the name of the user task and two files: a contextual dataset and a metadata file that describes the dataset; as output, it produces a standardized task-specific contexts file, which can be used by any context model generator

²<https://www.r-project.org/>

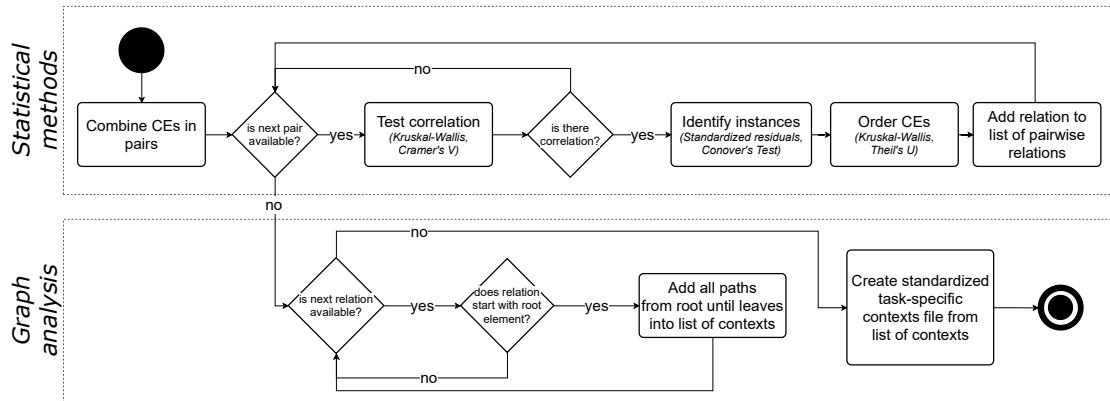


Figure 3: Activity diagram illustrating the algorithm of xPACE.

that complies with its layout. Figure 1b shows the internal structure of xPACE. The principal file in the core is `main.R`. It contains the algorithm that we implemented to analyze contextual data to identify potential relevant contexts for the user task in focus. The other R files are helpers (“`load_libraries.R`” loads the necessary packages into the memory, “`util.R`” defines some functions, and “`load_data.R`” reads the input files into the memory).

4. TASC Modeler

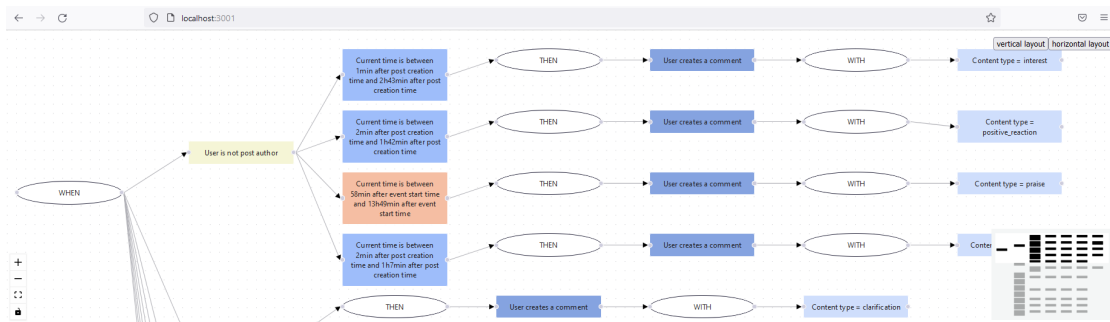


Figure 4: Screenshot of the TASC Modeler showing an excerpt of the task-specific context model.

The *Task-specific Context Modeler* (TASC Modeler) is responsible for creating the graphical representation of the context model. The requirements engineer interacts directly with TASC Modeler, providing the contextual dataset, the metadata, and the task name as input. TASC Modeler then interfaces with a contextual data processor to provide the input data and receive the standardized task-specific context files as response. In our case, TASC Modeler uses xPACE, but it could be any other contextual data processor implementation able to generate the standardized task-specific context file via a REST API. TASC Modeler reads from a configuration file the information about which contextual data processor it should use.

We implemented TASC Modeler as a single-page application written in Typescript using the React Library³. When the application is loaded, the user is presented a form where they can provide the dataset file, the metadata file, and the name of the user task in focus. When the button “Generate” is pressed, the application sends the input data to xPACE through its Web API. Then xPACE returns its implementation of the standardized task-specific contexts file, which is used by TASC Modeler to build the context model.

5. Application and evaluation

We used TASC Modeler and xPACE to support the elicitation of context-aware functionalities for Dorffunk⁴, a live communication app with approx. 25,000 active users developed and maintained by Fraunhofer IESE. We chose a user task (“create a comment”), collected contextual data (approx. 56,000 tuples from 15 contextual elements), and used the tools. Figure 4 shows a screenshot of TASC Modeler with an excerpt of the data-driven context model generated using Dorffunk data. The generated context model was evaluated in a controlled experiment with professional software engineers (because elicitation of context-aware functionalities is often not limited to requirements engineers [5]), where it showed its potential to support the identification of relevant contexts for given user tasks. All participants were asked to elaborate context-aware functionalities to improve the targeted user task. Participants of the treatment group received the data-driven context model, whereas participants of the control group received the list of contextual elements that were available to describe context-aware functionalities. On average, participants of the treatment group were able to elaborate context-aware functionalities that combine more contextual elements. They also stated that they found the context model valuable for supporting the elicitation of context-aware functionalities.

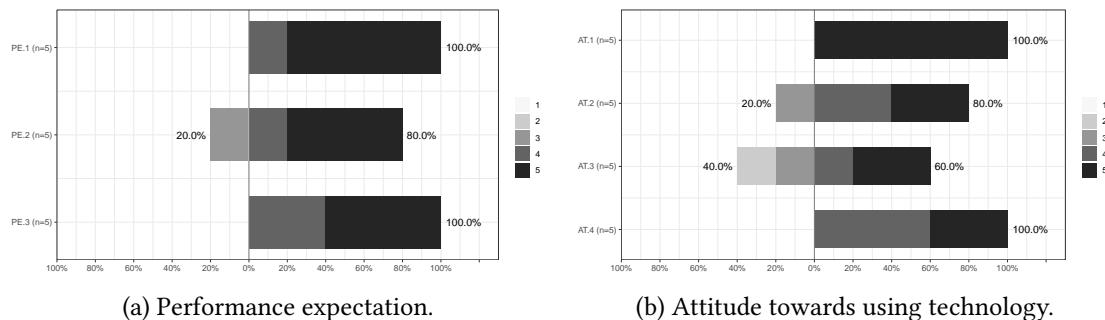


Figure 5: Participants’ assessment of the model’s usefulness.

While the details of the experiment can be found in our previous paper [6], we here reproduce some parts that concern the usage of the context model. Our hypothesis was: “The data-driven context model is perceived by individuals as a useful instrument to support the elicitation of context-aware functionalities”. In order to test this hypothesis, we employed the UTAUT

³<https://reactjs.org/>

⁴<https://www.digitale-doerfer.de/unsere-loesungen/dorffunk/>

(Unified Theory of Acceptance and Use of Technology [7]). In Figure 5, the positive trend towards the acceptance of the context model can be noted.

6. Conclusion and future work

Context modeling to support the elicitation of context-aware functionalities has been disregarded by practitioners due to its high complexity. To improve this scenario, we designed and implemented a data-driven context modeling process that automates the analysis of combinations of contextual elements that influence user tasks and the generation of the context model. The automation of this process is supported by two tools: the Contextual Data Processor and the Context Model Generator. In this demo paper, we presented our implementation of these tools: xPACE and TASC Modeler.

As future work, we plan to apply the tools in different projects and evaluate their quality attributes, in particular time behavior, capacity, and scalability, when they have to deal with much bigger contextual datasets. So far, our evaluation has focused on the results produced by the tools and not on their usage, so we will now also evaluate their usability, which is an essential step towards enabling adoption by practitioners. Moreover, since data processing and context model representation are decoupled, we also want to check to which extent other representations (such as CGM [8]) may be suitable for this particular purpose.

Acknowledgements

This work was partially supported by CNPq, Brazil.

References

- [1] S. Poslad, *Ubiquitous computing: smart devices, environments and interactions*, John Wiley & Sons, 2011.
- [2] R. O. Spínola, G. H. Travassos, Towards a framework to characterize ubiquitous software projects, *Information and Software Technology* 54 (2012) 759–785.
- [3] M. K. Pinheiro, C. Souveyet, Supporting context on software applications: a survey on context engineering, *Modélisation et utilisation du contexte* 2 (2018).
- [4] R. Falcão, Improving the elicitation of delightful context-aware features: A data-based approach, in: *RE17, 2017*, pp. 562–567. doi:10.1109/RE.2017.42.
- [5] R. Falcão, K. Villela, V. Vieira, M. Trapp, I. L. de Faria, The practical role of context modeling in the elicitation of context-aware functionalities: a survey, in: *RE21, IEEE, 2021*, pp. 35–45.
- [6] R. Falcão, M. Trapp, V. Vieira, A. Vianna Dias da Silva, Using a data-driven context model to support the elicitation of context-aware functionalities—a controlled experiment, in: *PROFES 2021, Springer, 2021*, pp. 119–135.
- [7] V. Venkatesh, M. G. Morris, G. B. Davis, F. D. Davis, User acceptance of information technology: Toward a unified view, *MIS quarterly* (2003) 425–478.
- [8] R. Ali, F. Dalpiaz, P. Giorgini, A goal-based framework for contextual requirements modeling and analysis, *Requirements Engineering* 15 (2010) 439–458.