

Clounaq - Cloud-native architectural quality

(Tool Demonstration)

Robin Lichtenthäler

Distributed Systems Group, University of Bamberg, An der Weberei 5, 96047 Bamberg, Germany

Abstract

Implementing cloud-native applications can be complex and challenging due to the breadth of the topic. A possibility to evaluate potential impacts from architectural characteristics on quality attributes could support the development and evolution of cloud-native applications. This work describes the Clounaq approach which aims to provide such a possibility. With the Clounaq tool software architectures can be modeled and evaluated based on a quality model which describes relationships between architectural characteristics and multiple quality aspects.

Keywords

cloud computing, cloud-native, quality evaluation, architectural model

1. Introduction

Cloud-native as a term has established itself in academia [1] and industry¹ for describing applications and tooling that take advantage of the characteristics of modern cloud environments [2, 3]. However, cloud-native covers a broad range of aspects impeding a clear and commonly accepted understanding of the term. Nevertheless, many benefits are associated with cloud-native. Thus, our motivation is how developers can be supported in developing software in a cloud-native way. Regarding the type of applications, our focus is on *enterprise applications* [4], that means web applications implemented by organizations to serve their customers or to satisfy internal need. Furthermore, our approach is focused on the design time. Although we acknowledge that an unambiguous measurement of the quality of an application is only possible at runtime, we argue that evaluations should be possible at design time already. When potential impacts of certain architectural characteristics on quality aspects can be evaluated and investigated at design time, problems can be avoided early on and an application can be designed according to cloud-native concepts from the beginning.


Our support for designing and evaluating applications is provided in the form of 1) A quality model that structures design-time architectural characteristics of cloud-native applications according to higher level quality aspects, and 2) a web-based tool which enables the modeling and evaluation of software architectures based on the quality model. We named our approach


16th Central European Workshop on Services and their Composition (ZEUS), February 29 – March 1, 2024, Ulm, Germany

✉ robin.lichtenthaeler@uni-bamberg.de (R. Lichtenthäler)

🌐 <https://www.uni-bamberg.de/pi/team/lichtenthaeler-robin/> (R. Lichtenthäler)

🆔 0000-0002-9608-619X (R. Lichtenthäler)

 © 2024 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)

¹<https://github.com/cncf/toc/blob/main/DEFINITION.md>

S. Böhm and D. Lübke (Eds.): 16th ZEUS Workshop, ZEUS 2024, Ulm, Germany, 29 February–1 March 2024, published at <http://ceur-ws.org>

Clounaq which is short for *Cloud-native architectural quality*. After a short overview of the quality model, we specifically present the implementation in this work. While working on it, several challenges had to be considered which we list here in a structured way:

- C1 Choosing a modeling approach and a suitable level of abstraction
- C2 Including support for the integration with other approaches and tools
- C3 Presenting information on modeling options and evaluation results in a meaningful way
- C4 Handling the complexity of both the quality model and the architectural models
- C5 Ensuring modifiability and extensibility of the tool

These challenges shaped the implementation of the approach and are the reasons for certain design decisions. Throughout this work we refer back to these challenges to highlight their causes or how they influenced the implementation. In section 2 we present the idea behind and the current state of the quality model. This serves as a foundation for the presentation of our tool in section 3, before we conclude our work in section 4.

2. Cloud-native Quality Model

The quality model for cloud-native software architectures² provides the conceptual foundation for our approach. Its initial formulation is based on literature [5] and parts have been validated empirically based on a survey [3]. In essence, the quality model includes *quality aspects* which represent higher level quality attributes, *product factors* which represent architectural characteristics whose prevalence in an architecture can be measured, and *impacts* which describe how the presence of a product factor impacts different quality aspects. An exemplary factor is *Service replication* which captures the characteristic of replicating service instances across infrastructure entities. A replication of service instances is considered to have a positive impact on *Time-behaviour*, because requests can be served from replicas closest to the respective client. And it is considered to have a positive impact on *Availability*, because even if one instance becomes unavailable, another replica can still serve requests.

To now evaluate an application based on this product factor, the extent of replication needs to be measurable. Therefore *measures* are used which can quantify characteristics of an application at a specified level of abstraction (C1). The values of measures are interpreted through *evaluations*. For the example of service replication *service replication level* (based on [6]) is a measure that quantifies the average level of service replication. A possible evaluation for this could be that a value of 1 means no replication and therefore no impact on availability and time-behaviour. And a value between 1 and 3 could mean a low replication and thus a slightly positive impact on availability and time-behavior.

This is merely one example from the quality model. Because it tries to cover the breath of the topic of cloud-native applications and multiple quality aspects in combination, the quality model itself has a certain complexity (C4). Furthermore, the quality model is not completely specified yet regarding the quantitative measures and evaluations. The aim is to develop the model further together with the tool based on the application of our approach to different use cases.

²<https://r0light.github.io/cna-quality-model/>

3. Clounaq Tool

The Clounaq tool³ is the implementation of our quality evaluation approach. An overview of its main parts is provided in fig. 1 which is discussed more in detail in the following subsections.

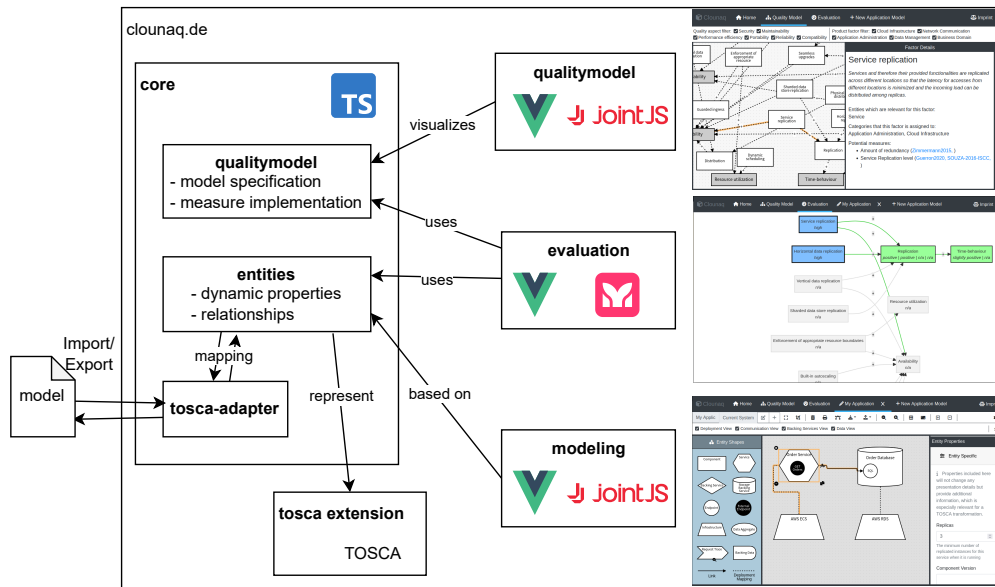


Figure 1: The internal design of the clounaq tool and corresponding views.

3.1. Functionality

The three main parts of the tool are accessible through different tabs (see right side of fig. 1):

Quality Model Tab This tab shows the quality model as described in section 2 with all product factors and quality aspects. Because of the complexity of the model (C4), it is possible to filter the shown quality aspects by their common high-level aspects or by different implementation aspects. Furthermore, to tackle C3, when a factor is selected, additional information is presented in a details section and related literature is linked.

Modeling Tab(s) For each created or imported software architecture model a new modeling tab is added. Therefore it is possible to work on multiple models at the same time. The initial modeling prototype which is now integrated in this tab was presented by Durr and Lichtenthaler [7]. In that work, we also considered different modeling options (C1) and decided to build on the TOSCA standard [8] because of its extensibility. Therefore we created a TOSCA profile that defines the different entity types required to model architectures of cloud-native applications and to evaluate them. With the decision for TOSCA it is in general possible to import models created with our tool also in other tools (C2) which support the TOSCA standard (e.g. Winery [9]). Currently, only an import and export for our TOSCA extension and for a custom JSON format is supported, but a mapping to formats of Infrastructure as Code (IaC)

³<https://clounaq.de>

tools could also be implemented. The modeling itself uses a graphical notation taking into account principles [10] for clarity and understandability (tackling C3). Additionally, also for the architectural models filters are available for temporarily focusing only on certain entities (C4).

Evaluation Tab In this tab, an architectural model can be selected for evaluation. For the product factors the values of specified measures are calculated and interpreted based on specified evaluations. With these evaluations, in turn, the quality aspects are evaluated and the results are presented with details so the user can comprehend their calculation (C3). Furthermore, either the perspective of product factors or of quality aspects can be selected (C4).

3.2. Implementation

The tool is implemented as a Single Page Application (SPA) using Typescript and Vue.js. It runs entirely in the browser of the user and does not have an additional backend. This design is intentional to simplify hosting (it is currently hosted using Github Pages) and support the reproducibility of our approach. Accordingly, it is open source and available at Github⁴, meaning that it can also be modified according to own needs (C5). Internally, the application is structured in four main parts represented by folders underneath `src` (see also fig. 1): In `core`, the entities are implemented as plain classes, but with a flexible properties attribute that is filled based on the underlying TOSCA extension. Therefore, properties can be specified only in TOSCA and the tool automatically adopts them. Furthermore, the quality model is specified in plain JSON and can be modified as such (C5) with changes being reflected in the tool. In `qualitymodel`, the quality model tab is implemented, solely depending on the specification of the model in `core`. Similarly, `evaluation` includes the code for the evaluation tab, depending solely on the code in `core` to evaluate a `System` entity based on the specified quality model and implemented measures and evaluations. Finally, `modeling` provides the modeling tab implementation based on JointJS. For each entity, a diagramming shape is specified and the properties editor dynamically includes the properties of each entity type.

3.3. Development Roadmap

The tool is in active development. While the modeling functionality and the visualization of the quality model are considered done, the evaluation functionality is being worked on: Specifically, additional measure calculations need to be defined and implemented. Where applicable, this also means additional properties need to be added to entities so that measures can be calculated. Finally, the evaluation tab should display all such relevant information in a structured way (C3).

4. Conclusion

This work presents the current state of the tool. However, it is refined in parallel with the overall approach. The tool enables the practical application of the approach, so that it can be tested and improved. Through quality evaluation results, useful features or required changes for the tool are uncovered. We therefore expect an ongoing development of both the approach and the tool when applying them to use cases, as we plan to do in future work.

⁴<https://github.com/r0light/cna-quality-tool>

References

- [1] N. Kratzke, P.-C. Quint, Understanding Cloud-native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study, *Journal of Systems and Software* 126 (2017) 1–16. doi:10.1016/j.jss.2017.01.001.
- [2] D. Gannon, R. Barga, N. Sundaresan, Cloud-Native Applications, *IEEE Cloud Computing* 4 (2017) 16–21. doi:10.1109/mcc.2017.4250939.
- [3] R. Lichtenthaler, J. Fritzscht, G. Wirtz, Cloud-Native Architectural Characteristics and their Impacts on Software Quality: A Validation Survey, in: *2023 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, IEEE Computer Society, Los Alamitos, CA, USA, 2023. doi:10.1109/SOSE58276.2023.00008.
- [4] T. Cerny, J. Svacina, D. Das, V. Bushong, M. Bures, P. Tisnovsky, K. Frajtak, D. Shin, J. Huang, On Code Analysis Opportunities and Challenges for Enterprise Systems and Microservices, *IEEE Access* 8 (2020) 159449–159470. doi:10.1109/access.2020.3019985.
- [5] R. Lichtenthaler, G. Wirtz, Towards a Quality Model for Cloud-native Applications, in: *Service-Oriented and Cloud Computing*, Springer, 2022, pp. 109–117. doi:10.1007/978-3-031-04718-3_7.
- [6] R. H. de Souza, P. A. Flores, M. A. R. Dantas, F. Siqueira, Architectural recovering model for distributed databases: A reliability, availability and serviceability approach, in: *Symposium on Computers and Communication (ISCC)*, IEEE, 2016. doi:10.1109/iscc.2016.7543799.
- [7] K. Durr, R. Lichtenthaler, An evaluation of modeling options for cloud-native application architectures to enable quality investigations, in: *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*, IEEE, 2022. doi:10.1109/ucc56403.2022.00053.
- [8] OASIS, TOSCA Simple Profile in YAML Version 1.3, 2020. URL: <https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/>, oASIS Standard.
- [9] O. Kopp, T. Binz, U. Breitenbucher, F. Leymann, Winery – A Modeling Tool for TOSCA-Based Cloud Applications, Springer Berlin Heidelberg, 2013, pp. 700–704. doi:10.1007/978-3-642-45005-1_64.
- [10] D. Moody, The “physics” of notations: Toward a scientific basis for constructing visual notations in software engineering, *IEEE Transactions on Software Engineering* 35 (2009) 756–779. doi:10.1109/tse.2009.67.