WTA: Towards a Web-based Testbed Architecture

Valentin Siegert[0000−0001−5763−8265] and Martin Gaedke[0000−0002−6729−2912]

Distributed and Self-organizing Systems Group, Technische Universität Chemnitz,
Straße der Nationen 62, 09111 Chemnitz, Germany
{valentin.siegert,martin.gaedke}@informatik.tu-chemnitz.de

Abstract. Tests, evaluations, and solution comparisons in complex use cases are often realized by creating a testbed for a domain of use cases and solutions. Web-based testbeds add key advantages like results sharing, remote test execution, and collaboration. Focusing on their research objectives, creators see their testbeds as a means to that end. The resulting web-based testbeds are similar in structure and functionality, but there is no common architecture supporting their creation, introducing redundant design efforts. Therefore, we determine structural similarities based on insights into the architecture of current web-based testbeds, from which we derive a generic web-based testbed architecture. This framework of reference will help to develop future testbeds focusing on the testbed domain instead of reinventing general testbed functionality.

Keywords: Web · Testbeds · Software architecture

1 Introduction

Evaluations are an essential step for proving the capabilities of newly created solutions in developing software or publishing research findings. While limited feasibility and scalability can often be tested manually in the early stages with respective manual effort, comparing solutions in complex use cases is not possible. However, many conduct their evaluations in software development and research manually or within their environment to prove the initial step in the correct direction [15]. The need for evaluating solutions in complex use cases is emerging from the industry and the research community as first stage evaluations are limited by design. Proof of feasibility and scalability in bigger and more complex use cases as well as insights by comparing different solutions at such use cases establish the need for testbeds. These can support the evaluation and give those insights into complex constructs as they can be set up for different test runs in the same environmental conditions. Additionally, testbeds can also test a combination of known solutions on how they work together in exemplary use cases.

On the other hand, testbeds are often created out of the need for proof or insights. The testbeds' development process thus is not the main focus. Instead, the potentially obtained knowledge out of the testbed's results is the motivation. With such focus, researchers tend to develop their testbeds well suited for their

usage, which results in a limitation of testbeds and followingly more testbeds for slightly different motivations. For example, in the field of multi-agent trust management systems, Yu et al. [15] describe that researchers tend to design their evaluation environments, which are often also own testbeds.

In the past years, more and more web-based testbeds were presented and developed. The web-based architecture adds to testbeds key advantages like results sharing, remote test execution, and collaboration. They exist in different domains like Web Applications, Internet of Things (IoT), Semantic Web, Deep Web, and many more and are conceived from small-sized testbeds on one machine to globally distributed ones like PlanetLab [11], a testbed for network services. As web-based testbeds are not restricted to test things in a web-related domain, researchers of other domains make use of them as well. Some recent examples are microgrids [13], railways [10], or also underwater acoustic communication [16].

Even though the research domains are different, current web-based testbeds make use of similar concepts. Thus, most researchers seem to have a common understanding of how a web-based testbed needs to be implemented. Nevertheless, to the best of our knowledge, a general architecture for web-based testbeds does not exist. Cavalieri et al. [4] propose some principles, but they focused on industrial production systems and date back to the early 2000s. In other terms, researchers have to reinvent a web-based testbed for their own needs without being able to build upon a given architecture. Besides the manual reinvention effort, especially researchers from non-computer science domains might not exploit the full potential of a web-based testbed due to a lack of knowledge about the available functionalities and architectural choices. As testbeds are a means to achieve experimentation and evaluation objectives, their architecture and development are not of prime concern to most researchers and any reduction in the effort will allow them to focus more on their original research activities. A web-based testbed architecture can limit these several reinventions and in the best case also limit the reoccurring need for a new testbed due to better reusability.

With this work, we provide insights into relevant web-based testbeds and determine their structural similarities to create a web-based testbed architecture. Our approach can improve existing testbeds and give future needs of tests, evaluations, and comparisons of different solutions a chance to be done faster without having to reinvent what others already elaborated for their own needs. Our contributions are as follows: (1) We present a software architecture for web-based testbeds based on known principles and structural similarities in current testbeds. (2) The architecture integrates key advantages of the web like results sharing, remote test execution, and collaboration.

2 Recent Work: Web-based Testbeds' Similarities

The most common conceptual similarity to realize a web-based testbed architecture in recent work is a concept with three actors: (1) the central node within the testbed, (2) an experimenter as a user of the testbed, and (3) the laboratory itself with its testbed environment capability to manage evaluations. The central

node of a web-based testbed is often realized as a web server, which interacts as the interface between the experimenter and testbed. On the one side, it communicates with the experimenter via its web application. On the other side, the central node realizes the management of the testbed by preparation, execution, and clean-up of one evaluation [12]. The user interface is not necessarily a web UI [2], but in many cases it is. Such a UI supports thereby the users' work with the testbed by visualizations and maybe some wizard alike guides.
The Experimenter may only have central access via the web application on the testbed, but some indicate their testbed environment elements also as directly accessible. Such direct access can be distinguished into access to the environment elements with organizational relation [1] or to the ones required for the evaluation execution itself [14]. It is often realized via ssh and indicates the distribution of the testbed's actors on different machines.

The testbed's actual feasibility is delivered by the testbed environment and its elements, which represent the testbed's domain-specific motivation. In general it exists to set up the initial situation for each evaluation, to execute, and later collect all required results measured during execution. The environment elements interact therefore according to a description created by the experimenter. Some approaches highlight for this description also how to schedule it and call the elements of such procedures job, trace or observation [5, 12, 14].

Several approaches work on not only single but multi-tier architectures within all described technical actors. Multi-tier approaches appear e.g. at the web application which is developed in a multi-site fashion [6] or at the testbed environment, which can be organized in several tiers by domain [8], by evaluation need [2] or by testbed management need [1, 12]. Besides, some also present how to create the central node in a multi-tier fashion [6, 8, 9].

The literature supposes also access points for different testbeds, e.g. model-based testbed creation [3] or an EaaS architecture [9]. These approaches add a meta management layer, such that the original organization elements of the environment and central node of one testbed are also dynamically created by the experimenters' description.

3 Web-based Testbed Architecture (WTA)

To achieve a good web-based testbed architecture, several goals emerge from the identified conceptual similarities and the principles by Cavalieri et al. [4]. One architectural goal is to have a central node that serves as the interface between experimenters and testbed. It hereby should serve a web UI for the experimenters, independent of which device they use for access. We call this central node *Testbed Server*, which should contain besides the web application for the experimenter the laboratory management, which we call *Testbed Director*.

The web UI for the experimenters should be more than only the access point but deliver certain usability features to the experimenters. Thus, it should support all experimenters, also the rather inexperienced ones with a clear evaluation process and how to use the testbed. Wizard-like support with visual and textual

help would be one way to realize this. Further, the UI needs to use a standardized representation according to the testbed's domain in visual and textual descriptions of any testbed artifacts, like measurements, use cases, evaluation descriptions, or results. As a testbed serves the need for testing, the experimenter should be able to get creative with combinations of possible solutions or use cases. Therefore, the web UI requires a playground for experimenters to change preferences of an evaluation, a set of pre-created artifacts like measurements or use cases, and a possibility to upload own created elements, like features to test or own created artifacts. The web UI should also contain a visualization of used schedulers within the testbed, if its domain requires such as in [5, 12, 14].

To have the possibility of choosing pre-created artifacts for evaluation preferences, web-based testbeds require a place to store its artifacts in a central place. We call this place the Testbed Library, originating from the library component in Cavalieri et al. [4]. The provided use cases are better if they are more complex, meaning not only many actors or events, but also include unforeseen events.

As a web-based approach, key advantages of the web like sharing, remote execution, and collaborative work should be included in the architectural design. Hereby, especially the Testbed Library and the web UI require to enable the user to share stored artifacts, to work collaboratively on artifacts, and to start a remote evaluation. The testbed server thus requires to proxy any experimenter for the evaluation run and has to ensure that one evaluation finishes when it is started or gives feedback to the experimenter on why it stopped intermediately. Combining direct access of elements with the named web's key advantages, any element of a web-based testbed should be accessible via the web.

The Testbed Director should be the experimenter's proxy accordingly and do everything to manage one evaluation execution. Therefore, it needs to set up the initial evaluation situation in the Testbed Environment, start the execution, gather all required results, and insure following evaluations with a releasing of preserved testbed performance for an evaluation execution.

A *Testbed Environment* is required to serve the actual testbeds functionality of evaluations. Depending on the testbed's domain and motivation of creation its environment has to be designed. Mostly it is a distributed system of different actors simulating a situation for one use case. It can thereby involve devices and simulated or virtualized elements. A separation of organizational and executional elements in the Testbed Environment will improve the evaluations' execution [4]. Not only environment elements should be separated into these two categories, but also artifacts and communication channels.

A clear hierarchy supports the future adaptability of a testbed. The testbed server is hereby the root, follows with organizational elements first, and ends with executional elements. To set up the distributed testbed faster, a bottom-up registration supports the process of dynamic ordered evaluations. Any element besides artifacts and used communication channels can be organized in a multi-tier fashion if required for the testbed's domain. While the organizational elements of the testbed are mostly described by the testbed's creator, the executional ones can also be configured by the experimenters according to their

evaluations. EaaS infrastructures form the exceptions where also organizational elements are partially described by the experimenters.

A web-based testbed serves degrees of freedom in a 2-dimensional space. One is the freedom of experimenter interactions and one is the freedom of the testbed environment elements. In both dimensions, the creators require to identify the sweet spot and design the testbed accordingly. In terms of experimenter interactions it is a dimension with three possible values: (1) a testbed can be a strict demo without any possibility for an experimenter to choose anything, (2) a testbed with limited available possible use cases and solutions to choose of, and (3) a full creative playground where the experimenters can live out their creativity to create their preferred evaluation with many possibilities to choose and change individually. The dimension of the testbed environment elements' freedom is a scale between two extremes, where both are hindering a valuable evaluation execution. One is the full control of the testbed organizational elements over the executional ones, and the other is the opposite, so no control of the organizational elements over the executional ones.

Web-based testbeds should also be open for future changes in their architecture. Therefore a core of components should be given, but similar to an onion architecture [7], the creators should include interfaces for future changes according to newly discovered technology. A testbed can thereby serve more similar needs. Followingly the number of testbeds in one domain can decrease.

3.1 WTA Elements

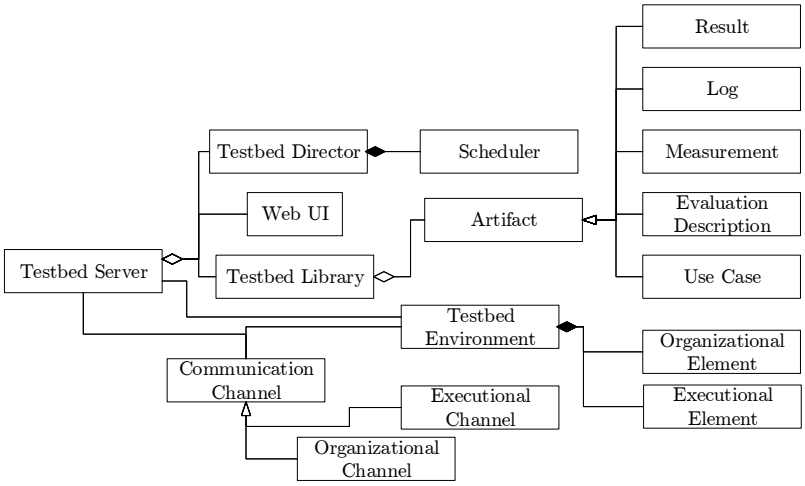

Fig. 1. Web-based testbed architecture elements.

The testbed server is the root of all involved elements and includes a Web UI, the Testbed Director, and the Testbed Library. Together with the testbed environ-

ment, they build the WTA components, which will be described in the following subsection with their interactions. Besides the WTA components, WTA includes elements like Scheduler, Artifact, and Communication Channel. Schedulers either order events into a sequence or schedule them on a timestamp if an evaluation description contains events or interactions to happen during the evaluation. Offering in a testbed several schedulers to choose from can be necessary because new insights into the testbed's domain could lead to new scheduling approaches in according evaluations. In WTA the schedulers relate to the events occurring in the testbed's domain and should be choosable by the experimenters for each evaluation if more than one is implemented.

Every web-based testbed has due to its distribution at least one communication channel to the experimenters' user agents. WTA-based testbeds need to communicate in the purpose of evaluation organization and maybe also within an executed use case between executional testbed environment elements. The organizational communication channels require thereby to be separated from the executional ones, such that they do not interfere with running evaluations. While the organizational channels mainly stick to web technology, executionals are influenced by the testbed's domain.

The artifacts of a WTA testbed are all the data within the testbed's process being stored, consumed, or both. Besides the evaluation results, which are the output artifacts, also input artifacts like standardized measurements, (real-world) use cases, or evaluation descriptions of experimenters exist in WTA. An instance of a WTA should also have measurements and use cases available to choose from in the Testbed Library. Additionally, some domains and their use cases might produce intermediate artifacts like logs, which need to be communicated to other environmental elements within the use case or serve an intermediate contemplation of an ongoing evaluation.

All WTA elements require to be accessible online. Thereby, experimenters can have potentially direct access to them and easily share especially the artifacts of a testbed. Maintaining a testbed and its executions is conducted in the distributed system of a WTA testbed with more efficiency due to such online accessibility. Therefore, all elements require a valid URL to be queried via the web.

3.2 WTA Components

WTA splits into three main components as in the conceptual similarities and includes the Web UI (1) such that the testbed server communicates with several user agents and serves as access points for the experimenters. We propose the implementation of the components in the component diagram in figure 2. The Web UI is then connected to two other components: the Testbed Director (2) and the Testbed Library (3). Therefore, the Testbed Director is the second half of the conceptual similarities' central node, and thus manages the Testbed Environment (4). It organizes the execution of a given evaluation description and later gathers all results from the environment.

The Testbed Library offers an interface for Web UI and Testbed Director and saves all central artifacts of the testbed. Hereby, it saves the results of

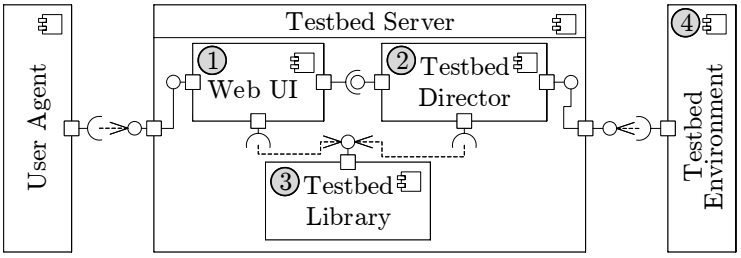

Fig. 2. Web-based testbed architecture components.

evaluations, real-world use cases, measurements, and other in advance created setup instructions for the testbed environment. All experimenters can access the library via the UI to either checkout results of passed evaluations or to create their next evaluation on the testbed. With such a library the ease of all experimenters is supported, such that they can focus on their main motivation of tests and evaluations and do not have to create everything from scratch.

The Testbed Environment is the most crucial aspect of the testbed as it realizes the required functionality. Thereby, its structure is highly dependent on the testbed's domain. It could be structured as in recent work from single-tier to multi-tier in as many layers as required for the testbed's domain. Also, additional layers to manage the correct testbed execution could be realized within.

The given architecture can by all components besides the User Agent be realized in a single- or multi-tier architecture. Figure 2 is hereby showing a single-tier version. Recent work proposes, that some domains or usages of a testbed might require such multi-tier testbed not only in executional elements but also in components like the Testbed Director or the Testbed Library.

4 Conclusion

In this work, we identified the recurrent innovation of web-based testbeds in different domains without a common architecture. To close the gap of especially testbed creators inexperienced in computer science towards full exploitation of the web-based testbed advantages, we identified conceptual similarities of recent web-based testbeds. Consequently, we presented a web-based testbed architecture (WTA) with emerging architectural goals out of the identified coneptual similarities and the principles initially proposed by Cavalieri et al. [4] which are still crucial. It conflates the common understanding and key advantages of the web. In the future, we need to support this first approach with a precise method supporting any testbed developer in his work, which will help especially researchers from non-computer science domains.

Acknowledgements We would like to thank Sebastian Heil for his valuable conceptual discussion and input. This work is funded by the Deutsche Forschungsgemeinschaft (German Research Foundation) - Project-ID 416228727 - SFB 1410.

References

1. Adjih, C., Baccelli, E., Fleury, E., et al.: FIT loT-LAB: A Large Scale Open Experimental loT Testbed. In: IEEE World Forum on Internet of Things, WF-IoT 2015 - Proceedings. pp. 459–464 (2015)
2. Akyildiz, I.F., Melodia, T., Chowdhury, K.R.: Wireless multimedia sensor networks: applications and testbeds. Proceedings of the IEEE **96**(10), 1588–1605 (2008)
3. Bertolino, A., De Angelis, G., Frantzen, L., Polini, A.: Model-based generation of testbeds for web services. Springer (2008)
4. Cavalieri, S., Macchi, M., Valckenaers, P.: Benchmarking the performance of manufacturing control systems: Design principles for a web-based simulated testbed. Journal of Intelligent Manufacturing **14**(1), 43–58 (2003)
5. Cecchet, E., Udayabhanu, V., Wood, T., Shenoy, P.: BenchLab: An Open Testbed for Realistic Benchmarking of Web Applications. In: The 2nd USENIX conference on Web application development. pp. 37–48 (2011)
6. Gao, Y., Zhang, J., Guan, G., Dong, W.: LinkLab: A Scalable and Heterogeneous Testbed for Remotely Developing and Experimenting IoT Applications. In: 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation. pp. 176–188 (2020)
7. Khalil, M.E., Ghani, K., Khalil, W.: Onion architecture: a new approach for xaas (every-thing-as-a service) based virtual collaborations. In: 2016 13th Learning and Technology Conference (L T). pp. 1–7 (2016)
8. Kouřil, D., Rebok, T., Jirsík, T., et al.: Cloud-based Testbed for Simulation of Cyber Attacks. In: 2014 IEEE Network Operations and Management Symposium (NOMS) (2014)
9. Lanza, J., Sánchez, L., Santana, J.R., et al.: Experimentation as a Service over Semantically Interoperable Internet of Things Testbeds. IEEE Access **6**, 51607–51725 (2018)
10. Neema, H., Koutsoukos, X., Potteiger, B., Tang, C.Y., Stouffer, K.: Simulation Testbed for Railway Infrastructure Security and Resilience Evaluation. In: 7th Symposium on Hot Topics in the Science of Security (2020)
11. Peterson, L., Roscoe, T.: The Design Principles of PlanetLab. ACM SIGOPS operating systems review **40**(1), 11–16 (2006)
12. Siegert, V., Noura, M., Gaedke, M.: aTLAS: a Testbed to Examine Trust for a Redecentralized Web. In: To be published in: Proceedings of The 2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (2020)
13. Vargas-Salgado, C., Aguila-Leon, J., Chiñas-Palacios, C., Hurtado-Perez, E.: Low-cost web-based Supervisory Control and Data Acquisition system for a microgrid testbed: A case study in design and implementation for academic and research applications. Heliyon **5**(9) (2019)
14. Werner-Allen, G., Swieskowski, P., Welsh, M.: MoteLab: A Wireless Sensor Network Testbed. In: 4th International Symposium on Information Processing in Sensor Networks. pp. 483–488. IEEE (2005)
15. Yu, H., Shen, Z., Leung, C., Miao, C., Lesser, V.R.: A Survey of Multi-Agent Trust Management Systems. IEEE Access **1**, 35–50 (2013)
16. Zia, M.Y.I., Otero, P., Siddiqui, A., Poncela, J.: Design of a Web Based Underwater Acoustic Communication Testbed and Simulation Platform. Wireless Personal Communications (2020)