Evaluating the UX of a Microservice Web App: CyberRanges in the µDevOps Project

Luis de-Marcos, José-María Gutiérrez-Martínez, Adrián Domínguez-Díaz, Sergio Caro-Álvaro, Daniel Rodríguez

Universidad de Alcalá. Dpto. de Ciencias de la Computación.

Edificio Politécnico. Campus Universitario. Alcalá de Henares. 28805. Madrid. Spain

{luis.demarcos, josem.gutierrez, adrian.dominguez, sergio.caro, daniel.rodriguezg}@uah.es

Abstract. *Microservice architectures are becoming increasingly important since they facilitate agile and modular production cycles to deliver applications using collections of loosely coupled and fine-grained services. The µDevOps is a research project formed by an international network of organizations including industry and academia that aims to tackle current challenges of microservice development operations. This paper presents the µDevOps project and the initial research carried out to evaluate the user experience of a microservice web application that delivers cybersecurity learning. Results point to critical elements of three main functionalities: library of scenarios, scenario information and entering scenario. Since there are several currently available solutions that offer similar services, the user experience of the microservice web app may play a critical role in determining which application will get a dominant role in the market.*

Keywords. User experience, microservice, usability, web application, cyberrange.

1 Introduction

Software industry has quickly moved toward ultra-agile and modular production cycles to deliver flexible, scalable, and user-centric applications in a timely way. A DevOps (Development Operations) process is a recent paradigm that puts together important advancements at architectural design, development, and deployment level. At architectural level, Microservices are an important evolution of service-oriented architectures aiming at structuring applications as a collection of loosely coupled and fine-grained services: it fosters a high modularity to make the application easier to understand, develop and test, and to facilitate the deployment, execution, and maintenance. At development and deployment level, DevOps stresses the agile code production by engaging the operational phase earlier in the development cycle: it fosters communication, collaboration and integration between development and IT operation teams, to favour rapid and continuous delivery cycles. The whole process is supported by new technologies (e.g., containers for deployment and cloud platforms at infrastructural level), which greatly alleviate several manual tasks, saving technical stakeholders time to deliver greater value. Even though Microservices and DevOps originate independently, they share the same set of principles and cultural background, stressing concepts like agility, flexibility, scalability, automation, user-oriented development, and cloud-based provisioning. Today they are viewed as a complement to each other, and likely, this production paradigm will underlie many software applications in the next years.

While researchers and practitioners have well caught the advantages of the interplay between the development and IT operation teams, this is by far not true for the third pillar of µDevOps, namely the quality assurance team. Software quality (SQ) is a pivotal asset and a key business driver for today's IT industry (Jones & Bonsignour, 2011). Innovation in this field has significant market opportunities for non-academic participants in the project. SQ embodies a wide set of non-functional requirements (in this context identified as Quality-of-Service (QoS) attributes) deemed crucial by end users, such as: security, reliability, performance, availability, usability. "Failures" in satisfying these requirements entail a large part of the business risk associated with a software product, as they can determine completely its success or failure. The continuous awareness, during development or operation, of user-perceived quality in the context where the system operates is paramount for decision-makers.

The µDevOps project aims to address all these challenges by forming an international and inter-sectoral network of organisations working on a joint research programme in the field of Software Quality Assurance (SQA) for Microservice Development Operations (we coin it as µDevOps) engineering processes. This paper presents µDevOps project and the initial findings of the work done to evaluate the

UX of a microservices web application. Since the UX plays a significant role in the success of an application, it is deemed critical to assess and create an infrastructure to continuously evaluate user experience. The rest of the paper is structured as follows. Section 2 introduces the µDevOps project. Section 3 presents the state of the art. Section 4 briefly outlines the approach. Section 5 reports initial findings. The paper concludes with final remarks and future research.

2 The µDevOps Project

Boundaries between development and operations are blurred and a highly dynamic environment demands for a new view of the role of testing for SQA. The µDevOps project grounds on the idea that the operational-time contextual information is destined to be predominant, and any realistic quality practice will not work if disregards it. The µDevOps research and innovation action intends to tackle the described challenges by promoting a cultural shift that places testing as a key cornerstone between the development and operation phases

This would amplify the intertwining and synergy between development and operations in the µDevOps paradigm, but requires a mind-set change that this project intends to promote. Toward this long-term goal, the proposers will implement a joint research program based on knowledge exchange between academic and industrial partners. The knowledge exchange program will revolve around the study, definition, implementation, and validation of a context-driven risk-based SQA testing process in µDevOps, including four scientific objectives:

- Context Learning: the objective is to study and develop new modelling and learning strategies to dynamically characterize the operational context of the microservice-based software under test at runtime
- Continuous in vivo testing: the objective is to define and develop techniques for in vivo tests generation and execution for µDevOps environment
- Risk Assessment to risk-based SQA in µDevOps: the objective is to study and develop metrics and methods to provide quantitative measures of the business risk of using a given functionality.
- µDevOps Development & Testing as a Service (D&TaaS): the objective is to make everything easily accessible and usable by a cloud infrastructure where both µDevOps-based development and testing will be offered as a service.

We structured the research programme in Work Packages. Figure 1 shows the work packages and the workflow of the project, providing an overview of the inter-relationship between WPs.

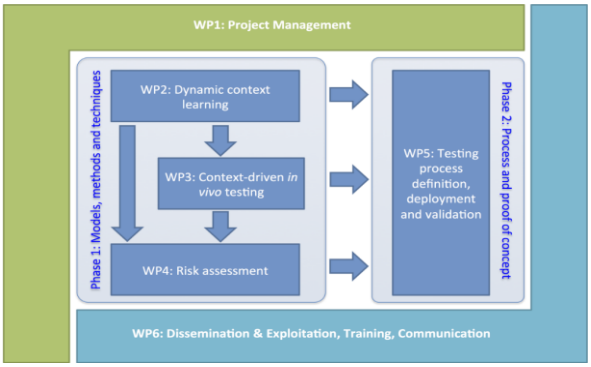

Figure 1. µDevOps project. Workflow of work packages

3 Testing and SQA in Microservices

Testing has been extensively investigated in the software architecture area, but it is still an open problem when dealing with microservice-based systems (Jones & Bonsignour, 2011). Indeed, testing microservice-based systems is extremely challenging, mainly because tools must be agnostic to the programming language and runtime environment of each microservice and because of the extremely rapid evolution of the business logic of each microservice within the system (Di Francesco et al., 2019). A limited number of testing approaches have been proposed. Schermann et al. (2016) propose “Bifrost”, a formal model for defining and automatically enacting live testing strategies for supporting multi-phase release strategies of microservice-based systems, with acceptable performance overhead under many release strategies in parallel. Release strategies are defined using a YAML-based Domain-Specific Language (DSL), thus allowing strategies to be transparently shared, reused, and versioned across projects and organisations. Heorhiadi et al. (2016) presented “Gremlin”, a purely network-oriented, systematic resiliency-testing framework inspired by software-defined networks. In Gremlin, a centralized control plane allows operators to provide high-level outage scenarios and assertions on how microservices should react during outages. Then, Gremlin automatically translates this information into a fixed set of fault injection rules applied to the network messages exchanged between microservices via network proxies. Experiments show that Gremlin can provide low-latency feedback to operators, and that the learning curve for creating scenarios and assertions is minimal. An architecture for automating acceptance testing in the context of microservice-based systems is proposed in (Schermann et al., 2016). The architecture heavily relies on Behaviour-Driven Development (BDD), where acceptance tests are defined as scenarios conforming to a simple syntax, even accessible by business stakeholders. The approach ensures that all microservices meet

requirements related to reusability, auditability, and maintainability. For what concerns SQA of microservices, performance (e.g., (Mazedur & Gao, 2015; Holger, 2016)) and maintainability (e.g., (Düllmann et al., 2017a; Hasselbring, 2017)) are the most investigated quality attributes in the state of the art. Many methods and techniques focussing on system performance have a special focus on scalability (Jones & Bonsignour, 2011), suggesting that researchers consider scalability as a sub-problem of performance when architecting with microservices. For example, O'Connor et al., 2017 presented a polyglot (i.e., supporting microservices developed in different languages, e.g., Java, JavaScript) auto scaling technique for the IBM Bluemix platform. The technique enables engineers to describe policies and set thresholds for scaling microservice-based applications based on their CPU, memory, and heap usage, and supports a shared multi-tenancy deployment model and it is delivered as managed cloud services. For what concerns maintainability, the relatively high scientific interest it is attracting might be related to the highly distributed nature of microservice-based systems and their inherent focus on services independence. As an example, Nuha et al. (2016) presented "MicroART", a technique for semi-automatically reconstructing the architecture of microservice-based systems to ease their maintenance and comprehension. MicroART is based on the Model-Driven Engineering paradigm (Seelam et al., 2015) and it reconstructs the software architecture of the system by mining the GitHub repositories with microservices source code and reusing existing monitoring platforms (e.g., New Relic). Other qualities are less explored with respect to performance and maintainability, including services compatibility (e.g., (Granchelli., 2017, Schmidt & Douglas, 2006)), security (e.g., (Sara et al., 2017, Viennot et al., 2015)), portability (e.g., (Yuqiong et al., 2015, Yahia et al., 2016)), or organisational alignment (Linthicum, 2016).

Relevant DevOps practices that relate to testing and SQA are continuous delivery (Justus & Zimmermann, 2016) and continuous monitoring (Jambunathan & Kalpana, 2016). In continuous delivery pipelines, testing activities are automated, using technologies as Cucumber or Selenium (Farley & Humble, 2010). However, it is an open challenge how to select the right load tests in such pipelines: with frequent releases, the capacity for executing load tests become critical and the need to select the right tests for maximum insight becomes crucial. One solution adopted in practice is canary testing, in which a new release is delivered to a few users for live testing. Ongoing efforts aim at providing more systematic support, such as the architecture-based performance engineering platform CASPA (van Hoorn et al., 2009). Currently, the main aim is to identify regressions over time (Aderaldo et al., 2017). No solutions have been proposed yet to systematically test systems for unexpected contexts. Overall, the activities are still usually separated between development and operation: while models and predictions may be used at development time to support design decisions, measurements-based approaches such as application performance management and monitoring are used at operation time. Initial approaches to reconcile both have been suggested, such as iObserve (Düllmann et al., 2017b) and CIPM (Mazkatli & Koziolek, 2018). Security testing in DevOps is also a relevant concern. There are tools helping to extract and automate tests to check for security issues, e.g., BDD-Security, Mittn by F-Secure, Contrast Security and Gauntlt. These are usually coupled with other means, like code analysis (e.g., tools like Veracode to scan the code to find vulnerabilities), runtime checks, cloud infrastructure best practices to check for configurations security best practices (e.g., Microsoft Azure Advisor, evident.io). DevSecOps refers to the practices to build security testing into DevOps. Reliability testing is much less investigated. An approach indirectly useful for reliability is the WESSBAS approach for the usage profile modelling (Vögele et al., 2016), which extracts probabilistic workload from measurements at operation time to inform models and decision-making at development-time.

4 Case Study: The CyberRanges Web Application

A cyber range is a virtual environment commonly used to provide a secure, legal environment for cybersecurity education, practice, and training. Isolation from threats is ensured by providing trainees the ability to recognize and respond to real-world challenges in a controlled environment. This approach guarantees that client infrastructure and data is never at risk because of the cybersecurity training (Aries Security, 2020).

CyberRanges[1] is a web application (Figure 2) developed by Silensec that delivers cyber security training and capability development exercises using technology and services for the design, delivery, and management of simulation-based, deep-dive experiences. It can be used to learn, train, test, measure, and improve the digital dexterity and cyber resilience of professionals, teams, and organizations by using a platform and technology that resembles a real-world scenario. CyberRanges uses a microservice architecture that strongly focuses on the functionality of the platform.

Since there are several competitors offering cyber range solutions, to get a dominant position is not only required to provide the better offering, in terms of functionality, library of scenarios and user base. But it

[1] https://www.cyberranges.com/

is also necessary to offer a seamless user experience that provides a meaningful and relevant interaction. Competitors to consider and compare include openly accessible solutions like TryHackMe[2] and HackTheBox[3], and commercial solutions that require a paid account like RangeForce[4] or ImmersiveLabs[5]. Similarly, online teaching and learning marketplaces like Udemy provide compelling and intuitive user experiences but lack of simulation required for cyber ranges.

The development team of the company assessed the UX strengths and weaknesses of the CyberRanges application based on the skills and competences of their staff. They identified a set of needs that could be addressed by the researchers. These needs can be summarized as follows: (1) Evaluate and redesign the UX for a more user-friendly experience, particularly simplifying the interface for the construction of scenarios and the UX inside them by suggesting changes to the actionable UI elements. For instance, changes are required in the interface to enter a scenario, in the library of scenarios (e.g., provide a more organic filter, better scrolling, and more on-screen information) as well as a possible redesign the statistics web page. (2) Get a better understanding of the process to build improve UX including the stages and steps (e.g., analysis, validation, etc.) to make it agile and incorporate feedback effectively. (3) Define the data to bring out and use for continuous improvement of UX (e.g., logs). Secondary less urgent needs include improving the analytics by suggesting new charts or changes to existing ones and analyzing the gamified elements and gameful design of the platform.

The case study presented here offers significant research possibilities in terms of the experimental testing and evaluation of the platform. Since cyberranges.com is going to publicly available soon, this opens the possibility for massive data collection. It would be possible soon to design and run experiments related to UX and user interaction.

[2] https://tryhackme.com
[3] https://www.hackthebox.com
[4] https://www.rangeforce.com
[5] https://www.immersivelabs.com

5 Approach for User Experience Evaluation

The sheer number of existing usability and UX evaluation methods includes, among others, observation, self-reporting, visual designs, idea descriptions, interaction analysis, lab tests, field studies and market feedback. The selection of one or other usually relies on the phase of product development, period of experience to evaluate and desired feedback. For instance, a visual design can be useful to assess the emotional expressions and reactions of a non-functional prototype or concept, while a field study (e.g., experience sampling or day reconstruction method) can be used to evaluate the long-term experience of a functional prototype.

Among the myriad of approaches, the μDevOps project opted for expert evaluation using sources like Yablonski's (2020) laws of UX and usability engineering. Given current state of the application and deployment plans for the subsequent months, we chose a method that can expedite the feedback required by the developers to improve the UX before launch. The usability evaluation was run by a group of three experts. Evaluation was performed during one month during Spring 2022. Experts meet initially to determine the goals, parameters, and steps for the study. They followed the template suggested by the Nielsen Norman Group[6]. The usability evaluation focused on the following functionalities: library of scenarios, scenario information, and entering scenario. These core functionalities to be subject of evaluation were pointed by the company development team based on their perceptions of what are more critical for their users. Each expert evaluated the user interface separately and they all joined to write the final report. The remaining of the paper summarizes the findings on which all evaluators agreed in their final report.

To further improve the evaluation process in the future, we can also explore the existing logs of the web application and implement the necessary changes in those logs to gather meaningful information about the user interaction that can be used to further improve the user experience.

[6] NN/g Checklist for Planning Usability Studies. https://www.nngroup.com/articles/usability-test-checklist/

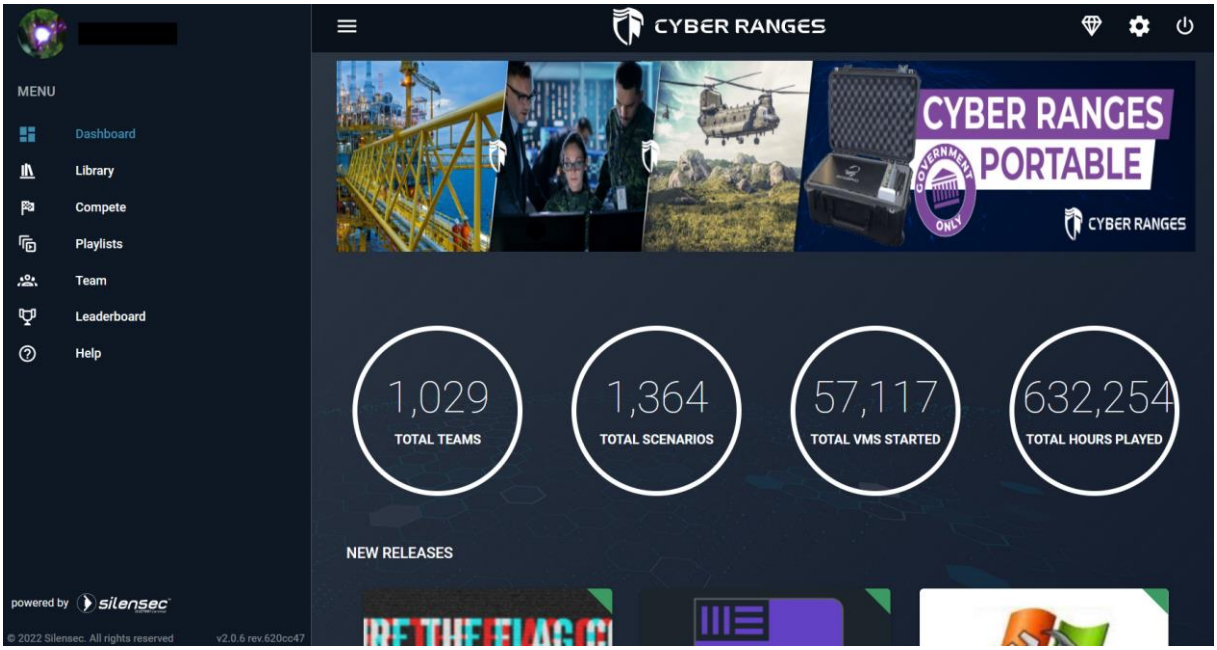

Figure 2. Cyberranges web app dashboard

6 Initial Findings

This section provides examples of the results of the evaluation of the user experience for three functionalities of the CyberRanges application: library of scenarios (Figure 3), scenario information (Figure 4) and entering scenario (Figure 5). We report here the functionalities that returned more issues according to the experts' evaluation.

The library of scenarios implements horizontal browsing for the elements which presents two usability issues. First, common usability heuristics suggests that horizontal scrolling is not recommended to present information. Second, the interface elements (buttons) that enable the horizontal browsing are only visible upon rollover of the mouse cursor. Also, in the library of scenarios assessment it was noticed that, upon application of a filter, the presentation of the list of scenarios changes substantially. The suggestion is to present the filtered output using a similar layout.

In the interface that shows information about scenarios, assessment found that the categories shown are different to the ones showed in the library. Apparently, the library contains tags for the whole collection while the scenario information screen shows the keywords. This is confusing. Suggestions include: (1) include the tags of the library also in the scenario information screen, (2) make clear that they represent different sets, and (3) make them clickable so that the user can access all the scenarios with same tags and keywords with just one action. In the scenario information interface, experts also suggest including an element (e.g., a button) to return to the library. In the current implementation the user needs to use the main menu on the right.

In the scenario loading user interface, it was suggested to improve the visibility of the loading bar. Recommendations are particularly focus on the fact that loading a scenario usually take a long time. The current bar's size and location on the top of the screen may prevent the user to see it initially, thus missing the important information it shows at a critical moment. The bar contrast with big 'Loading…' label at the bottom right that captures initial user attention. A wider suggestion is to rearrange the scenario loading UI. The screen presents parts that are mostly empty. Although the interface has a good approach, using tabs to present the mission, rules and leaderboards, designers may consider including further information and fill the empty spaces if needed, since the user may spend a substantial amount of time in this screen while the scenario loads. This may probably require motivating scenario creators to include all the additional information, which may be done by informing them of the space and time available. Similarly, the scenario loading screen could be used to provide required technical information that the user needs to complete it. For instance, the configuration of the VPN is shown on the main screen, but it would also be a good idea to include other reminders or links while loading.

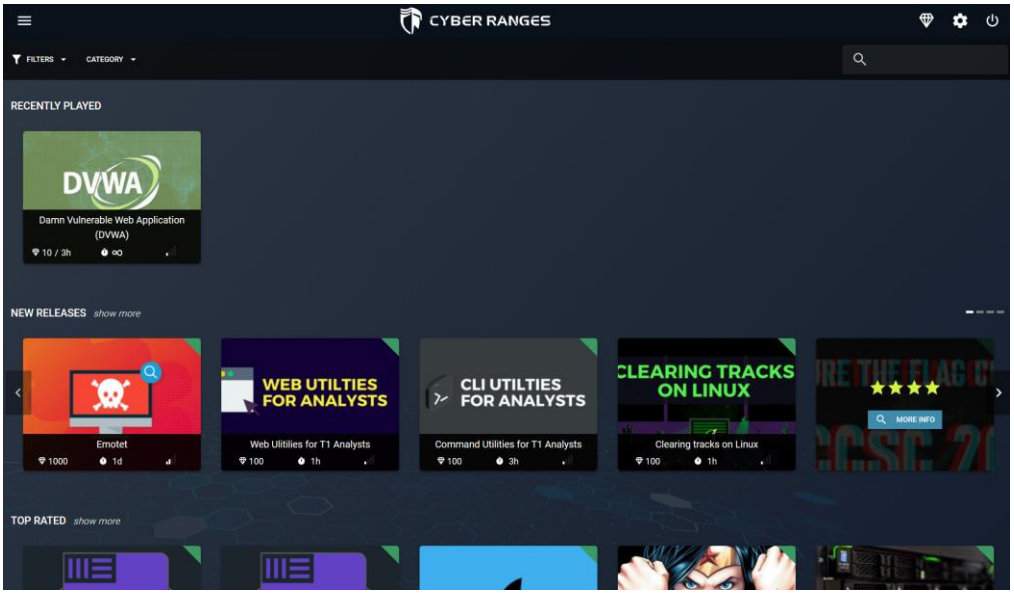

Figure 3. Scenario information

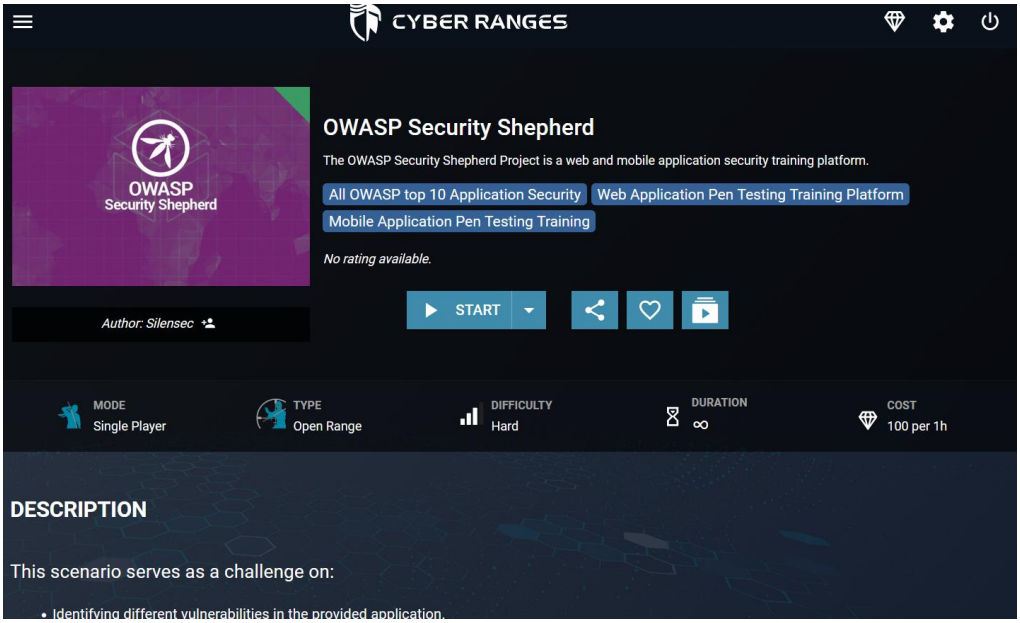

Figure 4. Scenario information

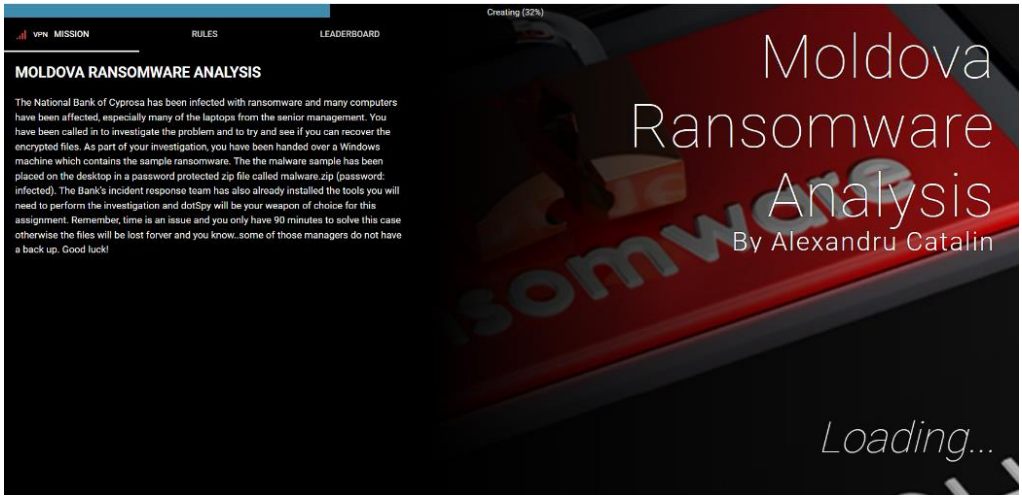

Figure 5. Scenario loading

7 Conclusion & Future Work

This paper described the initial approach and findings for the evaluation of the UX for experiential learning in the context of the CyberRanges web application for the µDevOps project. We initially presented the challenges of the microservices architectures that the µDevOps aims to address. We then introduced the case study of CyberRanges web application and the need for user experience evaluation in the context of microservices architectures. Initial findings suggest several improvements for critical functionalities of the application including the library of scenarios, the scenario information screen, and the scenario loading interface. Suggestions aim at dealing with important usability issues and improving the user experience. Since there are several commercial applications that offer cyber ranges and similar cybersecurity trainings, the user experience may play key role in the competition for determining which groups of solutions takes a significant part of the market share.

Future research can further dive into other approaches to UX evaluation that could be especially suitable for scientific objectives of the µDevOps project. Focusing on the first scientific objective, collecting real-time UX metrics could help to dynamically characterize the operational context of the microservice-based application, in terms of usage profile. This information could be used to improve the SQA processes by focusing testing efforts on those specific features in which more user experience issues are found. The integration of this data with artificial intelligence and machine learning techniques could help to create user models that predict UX problems, as well as to generate in-vivo test based on UX collected data and found issues. Another interesting problem in terms of UI and UX is to automatically create and manage replicas of corporate apps for CyberRanges exercises and other cybersecurity training.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 871342

References

Aderaldo, C. M., Mendonça, N. C., Pahl, C., & Jamshidi, P. (2017, May). Benchmark requirements for microservices architecture research. In Proceedings of the 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (pp. 8-13). IEEE Press.

Aries Security. (2020). What is a Cyber Range? A Definitive Guide and Definition. https://www.ariessecurity.com/what-is-a-cyber-range-a-definitive-guide-and-definition/ [Accessed on 24th May 2022]

Di Francesco, P., Malavolta, I., Lago, P. Architecting with Microservices: a Systematic Mapping Study, Journal of Systemsand Software, 2019.

Düllmann, Thomas F., and André van Hoorn. Model-driven Generation of Microservice Architectures for Benchmarking Performance and Resilience Engineering Approaches. Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion. ACM, 2017.

Thomas F. Düllmann, Robert Heinrich, Andre van Hoorn, Teerat Pitakrat, Jürgen Walter, and Felix Willnecker. Caspa: A platform for comparability of architecture-based software performance engineering approaches. In Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA 2017), 2017. IEEE Internation Conference on Software Architecture Workshops. 2017.

D. Farley and J. Humble, Continuous Delivery: Reliable software releases through Build, Test and Deployment Automation . Addison-Wesley Professional, 2010.

Granchelli, G., Cardarelli, M., Di Francesco, P., Malavolta, I., Iovino, L., Di Salle, A. MicroART: A Software Architecture Recovery Tool forMaintaining Microservice-based Systems. In Software Architecture Workshops (ICSAW), 2017 IEEE International Conference on (pp. 298-302). IEEE.

Hasselbring, Wilhelm, and Steinacker. Microservice architectures for scalability, agility and reliability in E-commerce. Software Architecture Workshops (ICSAW), 2017 IEEE International Conference on. IEEE, 2017.ç

Heorhiadi, V., Rajagopalan, S., Jamjoom, H., Reiter, M. K., & Sekar, V. (2016, June). Gremlin: systematic resilience testing of microservices. IEEE 36th International Conference on Distributed Computing Systems (ICDCS), 2016 (pp. 57-66).

K. Holger. Sustaining runtime performance while incrementally modernizing transactional monolithic software towards microservices. Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering. ACM, 2016.

Baskaran Jambunathan and Y Kalpana. Multi cloud deployment with containers. International Journal of Engineering and Technology (IJET), 8(1):421–428, 2016.

C. Jones, O. Bonsignour, The Economics of Software Quality, FINANCIAL TIMES/PRENTICE HALL, 2011.

Bogner Justus, and Alfred Zimmermann. "Towards integrating microservices with adaptable enterprise architecture." Enterprise Distributed Object Computing Workshop (EDOCW), 2016 IEEE 20th International. IEEE, 2016.

Linthicum, David S. "Practical use of microservices in moving workloads to the cloud." IEEE Cloud Computing 3.5 (2016): 6-9.

Rahman Mazedur, Jerry Gao. A reusable automated acceptance testing architecture for microservices in behavior-driven development. IEEE Symposium on Service-Oriented System Engineering (SOSE), IEEE, 2015.

Manar Mazkatli and Anne Koziolek. Continuous integration of performance model. In The 9th ACM/SPEC on International Conference on Performance Engineering Companion, Berlin, Germany, 2018, ICPE '18 Companion. ACM, New York, NY, USA. 2018.

Alshuqayran Nuha, Nour Ali, and Roger Evans. A systematic mapping study in microservice architecture. 9th International Conference on Service-Oriented Computing and Applications (SOCA). IEEE, 2016.

O'Connor Rory V., Peter Elger, and Paul M. Clarke. Continuous software engineering —A microservices architecture perspective. Journal of Software: Evolution and Process 29.11 (2017).

Hassan Sara, Nour Ali, and Rami Bahsoon. "Microservice Ambients: An Architectural Meta-modelling Approach for Microservice Granularity." Software Architecture (ICSA), 2017 IEEE International Conference on. IEEE, 2017.

Gerald Schermann, Dominik Schöni, Philipp Leitner, and Harald C. Gall. Bifrost: Supporting Continuous Deployment with Automated Enactment of Multi-Phase Live Testing Strategies. In Proceedings of the 17th International Middleware Conference (Middleware). 2016. ACM, New York, NY, USA.

Schmidt, Douglas C. "Model-driven engineering." COMPUTER-IEEE COMPUTER SOCIETY-39.2 (2006): 25.

Seelam, S. R., Dettori, P., Westerink, P., & Yang, B. B. (2015, March). Polyglot Application Auto Scaling Service for Platform as a Service Cloud. In Cloud Engineering (IC2E), 2015 IEEE International Conference on (pp. 84-91). IEEE.

A. van Hoorn, M. Rohr, W. Hasselbring, J. Waller, J. Ehlers, S. Frey, and D. Kieselhorst, “Continuous monitoring of software services: Design and application of the kieker framework,” 2009.

Viennot, N., Lécuyer, M., Bell, J., Geambasu, R., & Nieh, J. (2015, April). Synapse: a microservices architecture for heterogeneous-database web applications. In Proceedings of the Tenth European Conference on Computer Systems (p. 21). ACM.

Vögele, C., van Hoorn, A., Schulz, E., Hasselbring, W., & Krcmar, H. (2016). WESSBAS: Extraction of probabilistic workload specifications for load testing and performance prediction—A model-driven approach for session-based application systems. Software & Systems Modeling, 1-35

Yablonski, J. (2020). Laws of UX. O'Reilly Media.

Yahia, E. B. H., Réveillère, L., Bromberg, Y. D., Chevalier, R., & Cadot, A. Medley: An event-driven lightweight platform for service composition. In International Conference on Web Engineering (pp. 3-20). Springer, 2016.

Sun Yuqiong, Susanta Nanda, and Trent Jaeger. "Security-as-a-service for microservices-based cloud applications." Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on. IEEE, 2015.