Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, eds.

FROM LOGISTICS PROCESS MODELS TO AUTOMATED INTEGRATION TESTING: PROOF-OF-CONCEPT USING OPEN-SOURCE SIMULATION SOFTWARE

Paul Reichardt
Wladimir Hofmann
Tobias Reggelin

Faculty of Mechanical Engineering,
Otto-von-Guericke-Universität
Universitätspl. 2
39106 Magdeburg, GERMANY

Sebastian Lang

Fraunhofer Institute for Factory Operation and Automation IFF
Sandtorstraße 22
39106 Magdeburg, GERMANY

ABSTRACT

This paper explores the practical integration of simulation methods into software development processes. An automated integration testing approach is presented, which enables continuous virtual commissioning. For this purpose, an analysis of the current state of knowledge and the standards of software development is presented. This is followed by a case study on logistics order management, referring to a typical B2B application in the retail logistics sector. The proof-of-concept shows how the usage of a simulation model for automated integration testing and its inclusion into continuous-integration can help to ensure software quality, particularly for process-centered logistics applications. The implemented setup proves the feasibility of the approach, using standard open-source development tools, and a Python-based open-source simulation library.

1 INTRODUCTION

The planning and design of supply chains and intralogistics systems are frequently supported by simulation studies, used for comparing design alternatives, assessing their feasibility, as well as estimating KPIs like lead-time or throughput. When it comes to the realization phase of logistics systems, major challenges relate to the development of controls and operational IT systems. Given the fact that testing, integration, and commissioning of these systems consume a significant chunk of the realization phase, it becomes clear that it is beneficial to test a developed system as early as possible. Using simulation models to emulate the controlled parts of a software to be able to conduct early testing is also referred to as virtual commissioning, typically used in the context of automated material flow systems.

This paper shows how simulation-based testing can be integrated into agile software development tool-chains often found in practice. It aims to give a practical example of how simulation models for this use-case could be implemented, using accessible open-source tools and libraries. For this purpose, a case study on an application for order management and delivery optimization is presented, serving as an example for a typical web-based business application supporting main logistics processes. Following an explanation of the context in which the case-study takes place, the use of a virtual commissioning approach is motivated. Subsequently, the associated processes and requirements are introduced and explained in detail, followed by a literature overview. Furthermore, the development of the specific concept for virtual commissioning is presented, in combination with the application design and its overall structure. The last section contains summary and discussion, which provide an evaluation of the contribution to research as well as the deduction of future research activity approaches.

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

2 SCENARIO, SCOPE AND MOTIVATION

The general trend towards e-commerce in combination with home delivery affects the shopping of groceries. An increasing amount of food is sold online, mostly with a delivery to the front door. The decisive factors for the strong growth of the market are the increasing affinity of consumers for online food shopping as well as the entry of further competitors and that's why there is an increasing market within this segment (Etezadzadeh 2016; Straube et al. 2016). The growing importance of this type of shopping requires significant logistical effort, including an increasing effort required for the planning and organization of inner-city distribution processes with the special challenges of the last mile and prompt delivery (Gerdes and Heinemann 2019; Etezadzadeh 2016; Straube et al. 2016).

As a practical reference, the use-case of a food shop is chosen, which can be described to be a typical small and medium enterprise (SME). The company experienced a massive shift of demand towards at-home delivery due to changing consumer habits and Covid19-related restrictions. Due to this shift, the growing urban delivery network becomes more complex to manage, which in turn calls for a software-based support of the company's daily logistics operations. A dedicated software application needs to deal with the management of the data of customers, orders, depots, tours, as well as available delivery vehicles. Additionally, an optimized planning of tours is needed. The corresponding application needs to be developed and integrated into daily operations. A particular challenge relates to the commissioning and the transition from the old to the new system. The running business processes must not be interrupted and the functional reliability must be guaranteed at all times. The testing and commissioning process, therefore, has an outstanding importance for the total project success.

In the field of application development, there is a general tendency for functional processes to become more complex and increasingly require the coordinated interaction of a multitude of interfaces and influencing variables. Commissioning and the elimination of possible functional and control errors take up a significant share of the total project duration (Wünsch 2008). From an organizational point of view, commissioning involves different components which need to be functionally integrated. Delays and rework can generate immense costs in this phase. Thus, the functionality of a software solution needs to be tested and validated as well as possible before it is put into operative action.

3 PROCESSES AND REQUIREMENTS

Regarding the use case, the core processes to be supported by the application are:

- Registration of customers and tracking their orders
- Managing location data of the depots to plan the best-possible deliveries
- Keeping track of the trucks, delivering goods according to the planned tours

Figure 1 shows the business process model and notation (BPMN) diagram and contains the major processes found when interacting with the software.

4 RELATED LITERATURE

From an academic point of view, the planning of transport operations and logistics processes in a broader sense are well explored. Classical approaches use historical data, heuristics, and current process data (Tempelmeier 2018). But, if process parameters change fundamentally or if incremental capacity limits are reached, they are often no longer able to derive accurate decisions. With the use of simulation-based systems, even scenarios that have not occurred so far can be processed and appropriate measures anticipated (González-Reséndiz et al. 2018; Munoz-Villamizar et al. 2013; Sayyadi and Awasthi 2018; Günthner et al. 2013).

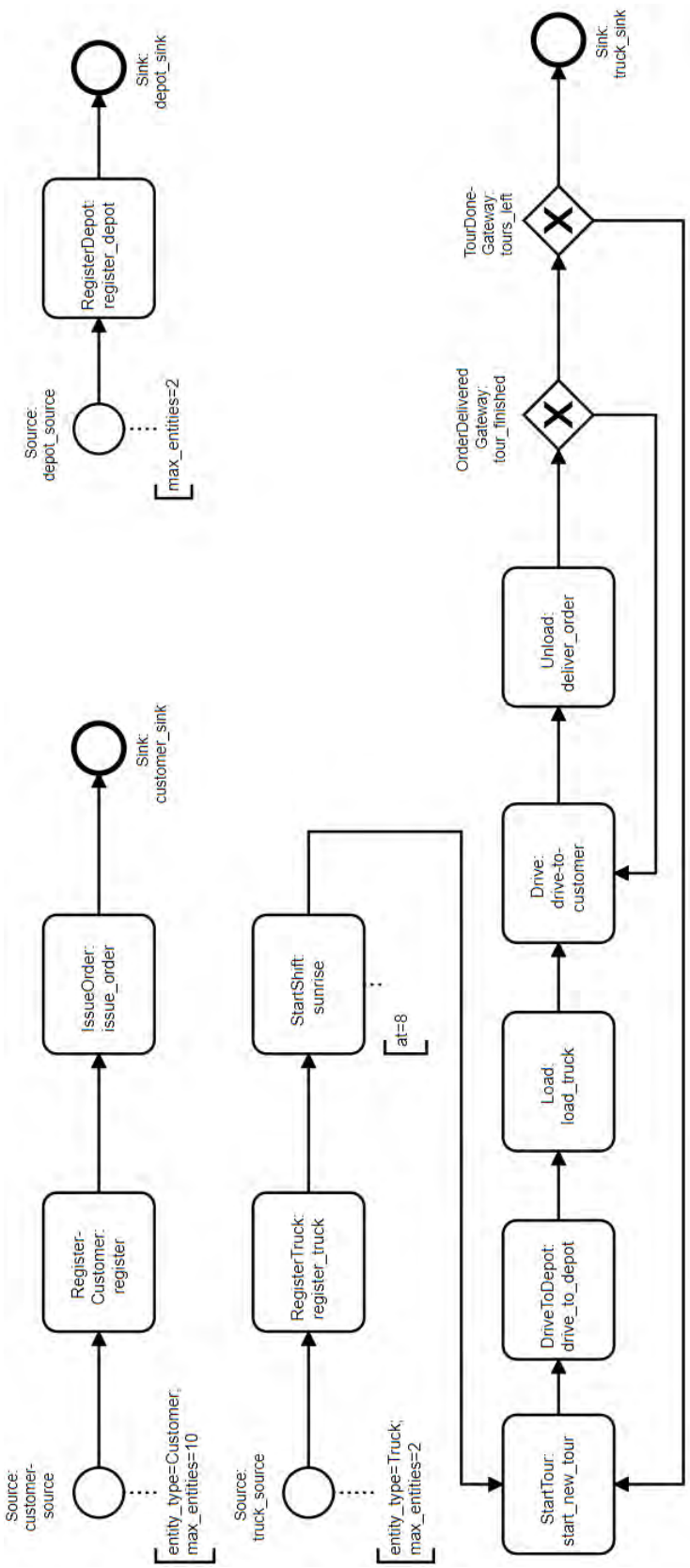
Source: depot_source
max_entities=2
RegisterDepot: register_depot
Sink: depot_sink
Source: customer-source
entity_type=Customer; max_entities=10
Register-Customer: register
IssueOrder: issue_order
Sink: customer_sink
Source: truck_source
entity_type=Truck; max_entities=2
RegisterTruck: register_truck
StartShift: sunrise
at=8
StartTour: start_new_tour
DriveToDepot: drive_to_depot
Load: load_truck
Drive: drive-to-customer
Unload: deliver_order
OrderDelivered Gateway: tour_finished
TourDone-Gateway: tours_left
Sink: truck_sink

Figure 1: BPMN diagram of the operational logistics processes.

Virtual commissioning can be described as the testing of software against the digital counterpart of a real system, using a simulation model to emulate interactions with the real world (Lee and Park 2014). Virtual commissioning can be used to solve various problems before they occur during actual commissioning in the real environment. During commissioning, especially during the integration of software applications, unforeseen delays often happen. In particular, problems arise due to errors from previous development phases as well as from complications within the integration process. A high percentage of the time of commissioning is spend for the elimination of these software errors. Furthermore, plausibility, capacity and thresholds of the operational system shall be tested to validate the function of the application (Kampker et al. 2020; Lee and Park 2014). Virtual commissioning can be used to solve various problems before they occur during actual commissioning in the real environment. During commissioning, especially during the integration of software applications, unforeseen delays often happen. In particular, problems arise due to errors from previous development phases as well as from complications within the integration process. A high percentage of the time of commissioning is spent for the elimination of these software errors.

In order to reduce the occurrence of software errors, validation has to be started early on. The aim is to eliminate the errors as soon as possible to increase the software quality. This reduces uncertainty and, thus, cuts time effort during later commissioning. In current process models of agile software development, this is frequently done by the use of iterative testing (Wolf and Bleek 2011; Yu and Petter 2014). In the present case, in particular the so-called continuous integration procedure is used. Continuous integration represents an often-used approach to increase and verify software quality. Typically, it includes automatic builds, tests, and (static) analyses of source code, performed upon check-in into the version management system (Meyer 2014; Russo et al. 2008; Wolf and Bleek 2011; Yu and Petter 2014).

Classical software development concepts are strongly pre-planning related and often highly complex. Framework parameters are defined at the beginning of a project. In the planning and development phase, an attempt is made to take all relevant possible scenarios into account as best as possible and to avoid potential faults before they occur (Hegmanns et al. 2014). As a result, the planning process is usually very extensive. In addition, framework parameters that change during the project can usually only be insufficiently taken into account (Hegmanns et al. 2014). As a result, the planning process is usually very extensive. In addition, framework parameters that change during the project can usually only be insufficiently taken into account. In the course of the project, the application is accordingly created, introduced, and then made functional according to the “trial and error” principle and improved until one is satisfied with the results. This costs time, effort, and performance.

Agile approaches on the other hand are based on the insight of changing requirements as well as technical and project-related uncertainties, which necessitate an iterative development with incremental addition of new functionalities, frequent code changes, as well as frequent releases and deployments of newer software versions in order to quickly obtain working software and user feedback. This approach is technically supported by tools and architectures that enable the simple and automated build, testing, and deployment of software units. One of the enabling technologies is, e.g., containerization, which simplifies the fast provision of lightweight, uniform execution environments for software including its dependencies. With a higher frequency of code changes and deployments, the importance of automated testing also increases, in order to ensure stable and working functionality for every software version without proportionally increasing manual testing effort.

From a practical point of view, virtual commissioning – i.e., the use of simulation models for realizing complex test scenarios with a high degree of automation – enables leaner project structures, more agile development processes, and close-to-operational commissioning. This saves costs, time, and maintenance effort, resulting in increased customer satisfaction (Günthner et al. 2013; Kullmann 2019).

5 CONTINUOUS INTEGRATION

To make sure that all required processes are adequately supported by a developed software, *continuous integration* (CI) represents an industry best-practice, which makes use of automated builds and test suites to continuously check software during development, typically supported by a "build-pipeline" which integrates and checks all code checked into the version control system.

Figure 2 shows a schematic process model for continuous integration. The open-source code hosting solution Gitea and the continuous integration platform Drone represent examples from the class of tools for creating a continuous integration setup as it is found in practice.

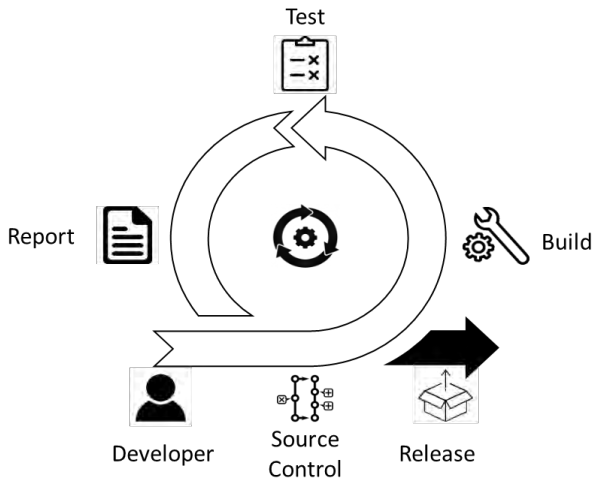

Figure 2: Scheme of continuous integration process (cf. Rogner 2019)

In order to create realistic test cases that are executed during the testing step, a simulation model shall be used to exercise complex processes, which the developed application is supposed to support.

A corresponding simulation tool needs to meet multiple related requirements. First, it needs to support the specification of business processes to be executed. Second, it needs to provide interfaces suitable to connect with the application under test. Third, it needs to be run as part of an automated test environment.

A common feature of different CI tools is the support for running containerized versions of the application under test as well as supporting services. Figure 3 shows steps of a simple continuous integration pipeline. When code changes are pushed, the application and its dependencies are bundled ("image"), and a corresponding container is started. Additional supporting services are run as needed by the application (e.g., databases, or for providing geo-information and routing). Furthermore, a service containing the simulation model is created, and the model is run, interacting with the application and verifying the tested behavior.

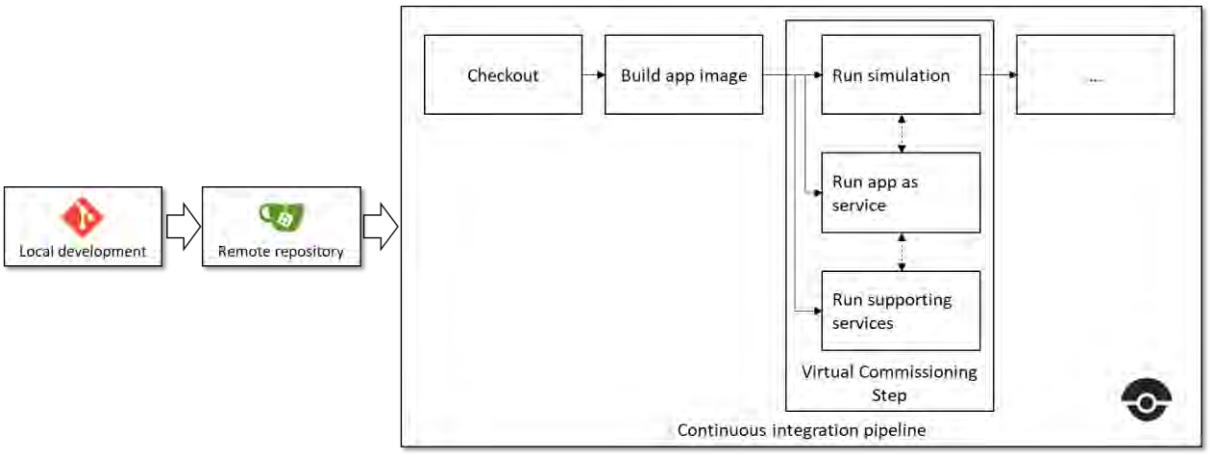

Figure 3: Continuous integration pipeline containing a step with simulation-based integration tests.

6 DESIGN OF THE SAMPLE APPLICATION

The way how simulation-based integration tests can be implemented heavily depends on the structure of the tested software. Motivated by the scenario described above, the next section presents a corresponding application implemented using popular open-source standards and technologies. While there exist numerous alternatives for all used technologies, the basic structure is typically found in practice and was, therefore, chosen for the example.

The sample application is dealing with management of the data of customers, orders, depots, tours, and delivery vehicles, as well as with calculating efficient tours and assigning them to available vehicles. The app adopts a basic 3-layer structure consisting of a browser-based user interface, a backend containing the business logic and optimization algorithms, and a persisting database.

Figure 4 shows a sample screen of the user interface, containing an overview of calculated tours. Following the notion of agile development, aspects of the scenario are only implemented in a simplified manner (e.g., assuming only one dimension when optimizing tours for capacity-constrained vehicles).

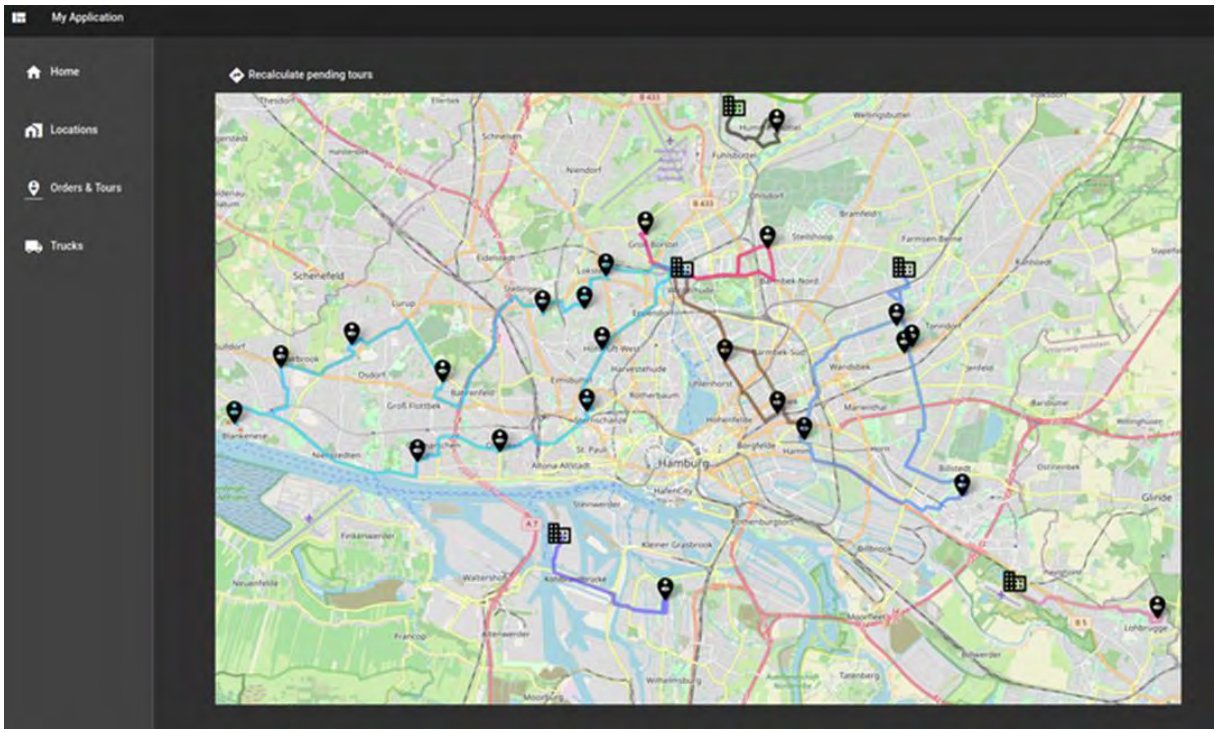

Figure 4: User interface of the sample application (tour-planning screen).

The backend for the proof-of-concept is implemented using Django, Django-Rest-Framework, and is relying on Google OR Tools for optimization tasks. Tour planning is modelled as a capacitated vehicle routing problem with multiple depots. For an optimal assignment of pending tours to available trucks, OR tools offers a minimum-cost-flow solver, which is used on a corresponding bi-partite graph (Christie 2020; Encode OSS 2020; GitHub 2020; Google Developers 2020) To create the required distance matrices, the Open Source Routing Machine (OSRM) is used, which is provided as a ready-to-use Docker image. OSRM offers a convenient Application Programming Interface (API) which is synchronously consumed upon creation of every new customer or depot. Open Street Map data are publicly available. SQLite provides a simple database solution, which can be exchanged for a client/server Relational Database Management System (RDBMS) like Postgres or MariaDB. The frontend is built with Angular, Material, and Leaflet.js. Figure 5 summarizes the setup.

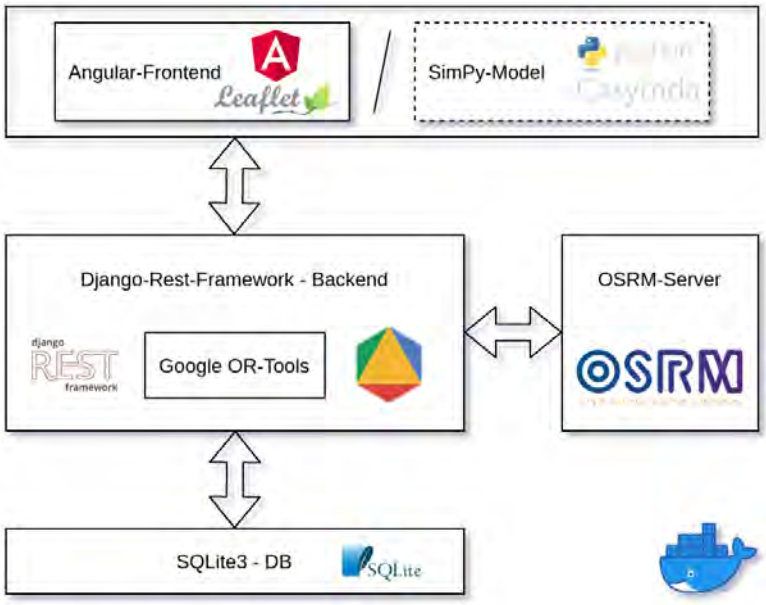

Figure 5: Sample application structure and used technologies.

7 TEST SCENARIO AND SIMULATION MODEL IMPLEMENTATION

The described application offers different possibilities for automated integration testing. One approach could include end-to-end testing using scripted UI interactions. While this is desirable in order to verify that all parts work together as expected, a possible disadvantage could arise from the fact that an application's presentation can often be considered less stable and subject to more frequent changes, compared to the back-end part of a system containing defined business logic. To create tests that ensure main functionality of the system while decreasing the risk of frequently adapting the tests to account for cosmetic changes, the integration tests implemented as part of this paper are run directly against the application back-end, effectively replacing human interaction via a web browser. Corresponding interactions are, therefore, implemented as HTTP requests.

The simulation tool used for creating the model and specifying the integration tests needs to allow for scripted API interactions as well as for simple automated, preferably containerized execution inside the continuous integration infrastructure. To meet these requirements, SimPy, a popular, open-source, Python-based discrete event simulation library, was chosen as the basis for the simulation model implementation for this proof-of-concept. Additional modeling functionality has been developed to generate SimPy model Python code from the XML representation of BPMN process diagrams. This facilitates the creation of larger, process-oriented simulation models based on graphical modeling. The developed functionality is available as part of the Python package "Casymda" (GitHub 2020). Further features include a browser-based real-time token animation for executed process models for debugging and presentation.

The behavior of different types of steps of the simulated process can be scripted using common Python code. For API interactions, the Python HTTP client library *requests* are used, assertions are part of the language itself. The simulation model code is versioned as part of the application source code repository. Whenever business processes to be supported by the application change, a new version of the model can be generated and the software as well as the simulation-based integration tests can evolve in an agile way. When changes are checked into the central repository, the CI pipeline automatically starts the current version of the application and runs the simulation model with defined scenarios (Fig. **7** shows sample output from a successful simulation run). In case the integration test encounters unexpected behavior (e.g. errors or inconsistent responses), the version is reported as failing the tests, which can then trigger additional

notifications or prevent source code updates. For examination, the simulation can be run locally in interactive mode for identification of error causes and debugging of the application.

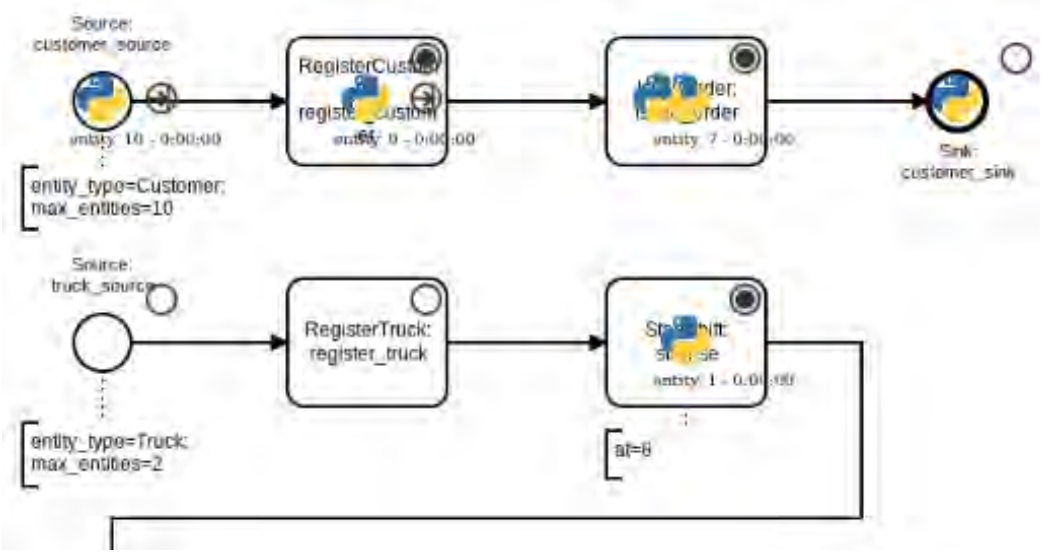

Figure 6: Real-time token-animation of simulated processes (exemplary extract, cf. figure 1).

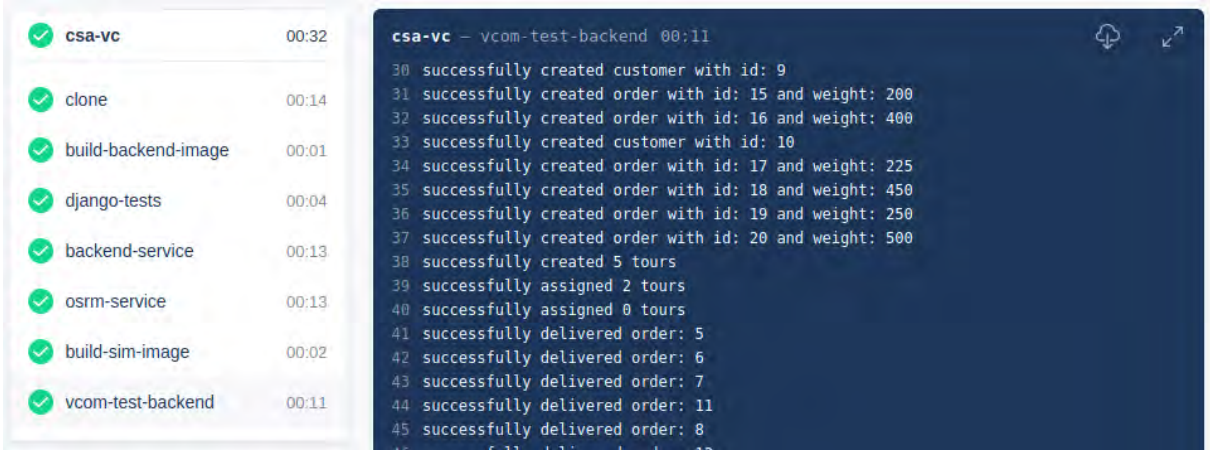

Figure 7: Standard output of simulation-based integration tests as part of a build pipeline (Drone).

For the sake of simplicity of this proof of concept, a basic simulation scenario was created, consisting of the following, statically defined parameters. Deterministic values ensure the exact reproducibility of the tests:

- Creation of two separate depots at defined positions
- Registration of two trucks (capacity 1,000 units each)
- Registration of 10 customers at predefined locations (of which placing 3 orders with different weight)
- Subsequent tour planning, capacity check, and simulated delivery

When all tours of all trucks are finished (no discrete events left to be processed), the completion of all orders of all customers is asserted. This scenario covers essential process paths and does not exercise further supported process variations, such as customers cancelling orders, or re-calculating tours after new orders have been added. Further possibly useful, highly context-dependent extensions could include assertions on solution performance, e.g., using KPIs that were forecasted by models from system's planning phases.

8 SUMMARY AND DISCUSSION

The environment in which software components are used requires a fast and reliable development and commissioning process. At the same time, the complexity of the software as well as the requirements regarding real-time capability for feedback to the real-world environment are increasing.

Extensive and automated integration testing with simulation models helps to enable and sustain software quality, particularly in the context of process-oriented logistics applications. As shown in the proof-of-concept, today's widespread software development tools and standards allow for an efficient integration of simulation techniques and virtual commissioning approaches into the development process.

Nevertheless, the individual creation of an emulating simulation model can be a resource-consuming task as well. Future approaches to improve the benefits of virtual application commissioning are to change the cost-benefit ratio of creating simulation models. On the one hand, this can be done by reducing the effort in model creation, e.g., by reusing existing models or modules. On the other hand, appropriate approaches can be used for the early integration of virtual commissioning during the application development process:

- Definition of the procedure model for the integration of virtual commissioning already during the conception of the application development process
- Use of uniform methods and tools within the applications
- Development of a specification techniques to be able to adapt the virtual commissioning to specific applications and, if necessary, to extend it by necessary components

Strategically, the use of simulation models for automated integration testing bears high potential for time- and cost-effective launches of new processes and software applications. Numerous applications are conceivable, ranging from large systems to the digitalization of the processes of small and medium size enterprises.

REFERENCES

Christie, T. 2020. Home – Django REST framework. https://www.django-rest-framework.org/, accessed 16th November 2020.

Encode OSS Ltd. 2020. The Web framework for perfectionists with deadlines | Django. https://www.djangoproject.com/, accessed 16th November 2020.

Etezadzadeh, C. 2016. "Urban Product Requirements". In *Smart City – Future City?,* edited by C. Etezadzadeh, 17–25. Wiesbaden: Springer Fachmedien Wiesbaden.

Gerdes, J., and G. Heinemann. 2019. "Urbane Logistik der Zukunft – ganzheitlich, nachhaltig und effizient". In *Handel mit Mehrwert*: *Digitaler Wandel in Märkten, Geschäftsmodellen und Geschäftssystemen,* edited by G. Heinemann, 397–420. Wiesbaden, Germany: Springer Gabler.

GitHub. 2020. bpmn-io/bpmn-js. https://github.com/bpmn-io/bpmn-js, accessed 30th November 2020.

GitHub. 2020. encode/django-rest-framework. https://github.com/encode/django-rest-framework, accessed 16th November 2020.

González-Reséndiz, J., K. C. Arredondo-Soto, A. Realyvásquez-Vargas, H. Híjar-Rivera, and T. Carrillo-Gutiérrez. 2018. "Integrating Simulation-Based Optimization for Lean Logistics: A Case Study". *Applied Sciences* 8(12):2448.

Google Developers. 2020. Assignment | OR-Tools | Google Developers. https://developers.google.com/optimization/assignment/overview, accessed 16th November 2020.

Günthner, W. A., J. Durchholz, E. Klenk, J. Boppert, T. Knössl, and M. Klevers. 2013. *Schlanke Logistikprozesse*: *Handbuch für den Planer*. Dordrecht: Springer. http://gbv.eblib.com/patron/FullRecord.aspx?p=1317750.

Hegmanns, T., A. Kuhn, M. Roidl, S. Schieweck, M. ten Hompel, M. Güller, M. Austerjost, J. Roßmann, and K. Eilers. 2014. "Planung und Berechnung der systemischen Leistungsverfügbarkeit komplexer Logistiksysteme". In *Logistics Journal: Proceedings, Vol. 2014*. Rostock, Wissenschaftliche Gesellschaft für Technische Logistik e. V.

Kampker, A., S. Wessel, N. Lutz, M. Reibetanz, and M. Hehl. 2020. "Virtual Commissioning for Scalable Production Systems in the Automotive Industry: Model for evaluating benefit and effort of virtual commissioning". In *2020 9th International Conference on Industrial Technology and Management (ICITM),* 107–111. IEEE.

Kullmann, T. 2019. "Schneller zum Ziel mit höherer Qualität". *CITplus* 22(9):23–25.

Lee, C. G., and S. C. Park. 2014. "Survey on the virtual commissioning of manufacturing systems". *Journal of Computational Design and Engineering* 1(3):213–222.

Meyer, M. 2014. "Continuous Integration and Its Tools". *IEEE Software* 31(3):14–16.

Munoz-Villamizar, A., J. R. Montoya-Torres, A. A. Juan, and J. Caceres-Cruz. 2013. "A Simulation-based Algorithm for the Integrated Location and Routing Problem in Urban Logistics". In Proceedings of the 2013 Winter Simulation Conference, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, 2032–2041. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rogner, J. 2019. Continuous Integration: Die kontinuierliche Qualitätssteigerung. https://codefluegel.com/blog/die-kontinuierliche-qualitaetssteigerung-continuous-integration/, accessed 25th November 2020.

Russo, B., E. Damiani, S. Hissam, B. Lundell, and G. Succi, eds. 2008. *Open Source Development, Communities and Quality*. IFIP 20th World Computer Congress, Working Group 2.3 on Open Source Software, September 7-10, 2008, Milano, Italy. New York, NY: Springer.

Sayyadi, R., and A. Awasthi. 2018. "A simulation-based optimisation approach for identifying key determinants for sustainable transportation planning". *International Journal of Systems Science: Operations & Logistics* 5(2):161–174.

Straube, F., B. Nitsche, and A. Figiel. 2016. "Konsumententrends und ihr Einfluss auf die Lebensmittellogistik". In *Zukunftstrends in der Lebensmittellogistik*: *Herausforderungen und Lösungsimpulse*, edited by F. Straube, 10-25. Berlin: Universitätsverlag der TU Berlin.

Tempelmeier, H., ed. 2018. *Planung logistischer Systeme*. Berlin, Germany: Springer Vieweg. http://www.springer.com/.

Wolf, H., and W.-G. Bleek. 2011. *Agile Softwareentwicklung*: *Werte, Konzepte und Methoden*. Heidelberg: dpunkt.verlag. http://gbv.eblib.com/patron/FullRecord.aspx?p=952271.

Wünsch, G. 2008. *Methoden für die virtuelle Inbetriebnahme automatisierter Produktionssysteme*. München: Utz.

Yu, X., and S. Petter. 2014. "Understanding agile software development practices using shared mental models theory". *Information and Software Technology* 56(8):911–921.

AUTHOR BIOGRAPHIES

PAUL REICHARDT is a research fellow at the Otto von Guericke University Magdeburg. He earned his master's degree in Industrial Engineering for Logistics at the Otto von Guericke University Magdeburg. He worked in national and international positions in the automotive industry as well as in systems and mechanical engineering. He manages industrial and research projects. His research interests include structure and process planning for logistics and factory systems, industrial projects, and practical application of machine learning and artificial intelligence in production planning and control. His email address is paul.reichardt@ovgu.de. His ResearchGate profile is https://www.researchgate.net/profile/Paul_Reichardt2.

WLADIMIR HOFMANN is a software developer from Hamburg, Germany. He works on application back-end systems, e.g., for automotive and logistics companies. Research interests involve the integration of simulation technologies with enterprise applications to facilitate efficient commissioning and operations. His email address is: wladimir.hofmann@outlook.com

TOBIAS REGGELIN is a project manager, researcher, and lecturer at Otto von Guericke University Magdeburg and Fraunhofer Institute for Factory Operation and Automation IFF Magdeburg. His main research and work interests include modeling and simulation of production and logistics systems and developing and applying logistics management games. Tobias Reggelin received a doctoral degree in engineering from the Otto von Guericke University Magdeburg. Furthermore, he holds a master's degree in Engineering Management from the Rose-Hulman Institute of Technology in Terre Haute, IN and a diploma degree in Industrial Engineering in Logistics from the Otto von Guericke University Magdeburg. His email address is tobias.reggelin@ovgu.de.

SEBASTIAN LANG is research fellow at the Fraunhofer Institute for Factory Operation and Automation IFF. He holds a master's degree in mechanical engineering with focus on production technologies and a master's and bachelor's degree in industrial engineering and logistics. His research interests include studying and applying methods of machine learning and artificial intelligence, simulation modeling, and mathematical optimization for problems in production and logistics. His e-mail address is sebastian.lang@iff.fraunhofer.de. His ResearchGate profile is https://www.researchgate.net/profile/Sebastian_Lang5.