SimulationOps - Towards a Simulation-as-a-Service Platform for Resilient Societies Using a Cross-domain Data Mesh

Hannes Restel
Fraunhofer FOKUS
hannes.restel@fokus.fraunhofer.de

ABSTRACT

Cross-domain simulations can be a feasible approach for enhancing disaster resilience as well as promoting resilient societies. This work-in-progress proposes a data-centric process model and software platform architecture called "SimulationOps" aimed at improving cross-domain collaboration between researchers (simulation analysts, simulation modelers) and stakeholders (disaster responders, decision makers) throughout the simulation life cycle for combined simulation artifacts. This way, stakeholders are supported in mitigating disasters, improving overall resilience by gained insights, and improvements in quality and velocity. Applying a four-cycle Design Science Research model to the simulation lifecycle, it combines ideas from modern and agile software engineering practices, simulation-as-a-service approach, and the Data Mesh approach. It combines the technical IT level with the organizational process level to smoothen the workflow for creating, running, and improving cross-domain computer simulation components for both producers as well as consumers of the simulation life cycle.

Keywords

Disaster Resilience, SimulationOps, Simulation Life Cycle, Design Science Research, Data Mesh

INTRODUCTION AND MOTIVATION

In order to strengthen resilience, (computer) simulations can be performed to support the four overarching goals of resiliency, namely *anticipate, withstand, recover,* and *evolve* (Paton and Johnston 2006; Ron Ross et al. 2021).

Computer simulations can be used, both preventively for disaster risk reduction (UNDRR 2016) as well as in an ongoing crisis/emergency situation. By simulating possible scenarios in the real world, potentially threatening situations from small-scale to large-scale disasters can be mitigated/countered beforehand, e.g., by analyzing the readiness of emergency response resources in an affected area/location (Osaragi and Hirokawa 2019), performing risk assessments (Ramírez et al. 2015), improving training for incidence response personnel (Zobel 2020), performing what-if analysis for population evacuation and community resilience planning (Daudé et al. 2019; Ganji et al. 2019), or by creating ever-updating forecasts for an ongoing crisis (updating estimated development of situation for 10 minutes - 1h – 24 h in the future).

By combining different input data and different categories of simulation models/components from multiple domains, even more complex situations/scenarios can be simulated, e.g., how a gradual loss of electricity in the grid (network model) triggered by a severe weather event may affect actions of the inhabitants (agent-based model). This allows for identifying potential cascading effects, and even for discovering previously unknown and surprising causalities.

In the two basic scenarios a) *sudden-onset disaster situations* (natural disasters like earthquakes, floods) as well as b) *slow-onset disaster situations* potentially affecting whole parts of society (Covid-19, climate change), simulation systems may offer a higher-order value. A higher-order value is realized especially, when simulations are combined to simulate novel and complex situations and produce results quickly (time criticality, e.g., "where and when is the levee most likely to break during a flood?").

Simulations serve a variety of objectives towards resiliency, the most common being *validation, what-if analysis, enhancing, forecasting,* and *training*. To achieve those objectives using combined simulations, the cooperation of experts and their knowledge across multiple domains is often required. This can be a complex and time-consuming process, as we can see from the example of the COVID-19 pandemic, which required combining particle simulations with medical expertise to model the spread of aerosols for deciding if and to what degree wearing face masks is useful. Bringing together experts from different domains and integrating simulation components from different sources requires interoperability across technical, semantic, organizational, and political dimensions.

For the development and integration of combined simulators/simulations, several approaches and models already exist (e.g., "High Level Architecture", HLA, (IEEE Computer Society 2010)) as well as technical characteristics and deployment mechanisms (e.g., "Modeling and Simulation-as-a-Service" (Siegfried et al. 2018) or "Cloud Based Distributed Simulation" (Chaudhry et al. 2022)). These existing solution approaches already integrate important technical-methodological issues, e.g., solutions for time synchronization of simulators along the simulation steps.

Complex combined simulations require cross-domain and interdisciplinary collaboration, not only on the technical interoperability levels (syntactic, semantic), but especially on the organizational process level. This addresses a gap in existing models, as criteria such as cross-domain collaboration, high-velocity, and short feedback cycles are barely prioritized. Input data and output data are also not explicitly considered as "first-level citizens", and certainly not as a value-creating product. This leads to the following research question: How can collaboration at the process level be promoted, maintaining a high velocity for the creation and execution of (novel/innovative) cross-domain simulations in a multi-domain environment while maintaining high quality standards?

The solution approach is to first formally classify both simulation components and their results (i.e., data) into the *Four-Cycle Design Science Research* (DSR, (Drechsler and Hevner 2016)) and identify those forces that affect the quality and speed of simulation artifacts to understand how they can be improved. Based on these findings, a data-centric process model and a system architecture are derived, which enable the creation and operation of complex cross-domain simulation systems with short creation and development cycles along the *simulation development life cycle* (analogous to the software development life cycle). The resulting approach is called "*SimulationOps*" (a combination of "simulation" and "operations").

The resulting *SimulationOps* approach is aligned with the following hypotheses:

- Short iterations in the design cycle (including fast *evaluation steps*) enable better quality of the resulting artifacts for cross-domain simulations.
- Cooperation between producers (experts, researchers) of the simulation components and consumers (researchers, disaster management) should be as close as possible, i.e., a close interlocking of *design cycle* and *relevance cycle* is emphasized.
- A data-centric approach promoting the concept of *data products* allows for a fitting level of abstraction to connects producers and consumers interested in simulation components as well as generated simulation result data, alike.
- For the necessary *data platform architecture*, the *data mesh* approach is a suitable model of data storage (persistence) for cross-domain simulations and is preferable to the "single unified data model" in the sense of a data warehouse, or "raw data" in the sense of a data lake.

The *SimulationOps* approach can help to gain a better understanding of complex systems and develop more robust solutions for managing crises and disasters. This work-in-progress paper outlines the principal ideas of the process model and the platform from a 30,000-foot perspective and does not dive into technical details of how the platform can be implemented.

This work is structured as follows: The upcoming chapter "*Related Work*" discusses the concepts of Design Science Research and combines it with the various simulation artifacts along the simulation development life cycle. It then displays existing approaches for scaling simulations and concludes the section by introducing the ideas of the data mesh. The chapter "*Process Model and Platform Architecture*" introduces the proposed "SimulationOps" and how it will work on an organizational process level as well as the software architecture level. The final chapter "Conclusion and Future Work" summarizes the outlined approach and gives a glimpse into the open issues and future work.

RELATED WORK

In the landscape of research methods, simulations are located in the quadrant of the 'quantitative-constructivist' (Wilde and Hess 2006). Unlike observing behaviors in the real world (e.g., case studies, grounded theory), simulations are a creative approach (in the sense of creating and using) to actively support finding answers to research hypotheses in a measurable/quantifiable way. As such, they align with other research methods such as formal-deductive analysis and reference modelling (Wilde and Hess 2006).

Thus, simulations can be understood as *artifacts* in the sense of the *Design Science Research* (DSR) approach, as they are based on a 'construction science paradigm', i.e. they "strive to gain knowledge by creating and evaluating IT solutions in the form of models, methods or systems" (Simon 2008).

DSR is particularly fitting for the presented process model outlined in the chapters below, as it focuses on the interfaces between "people, organizational systems, and technical systems within a particular application domain" (Drechsler and Hevner 2016), which in this case is public disaster resilience. In this work, the extended version of DSR by Drechsler and Hevner is used that builds upon four cycles instead of three cycles: *change & impact cycle*, *relevance cycle*, *design cycle*, and *rigor cycle* (see Figure 1).

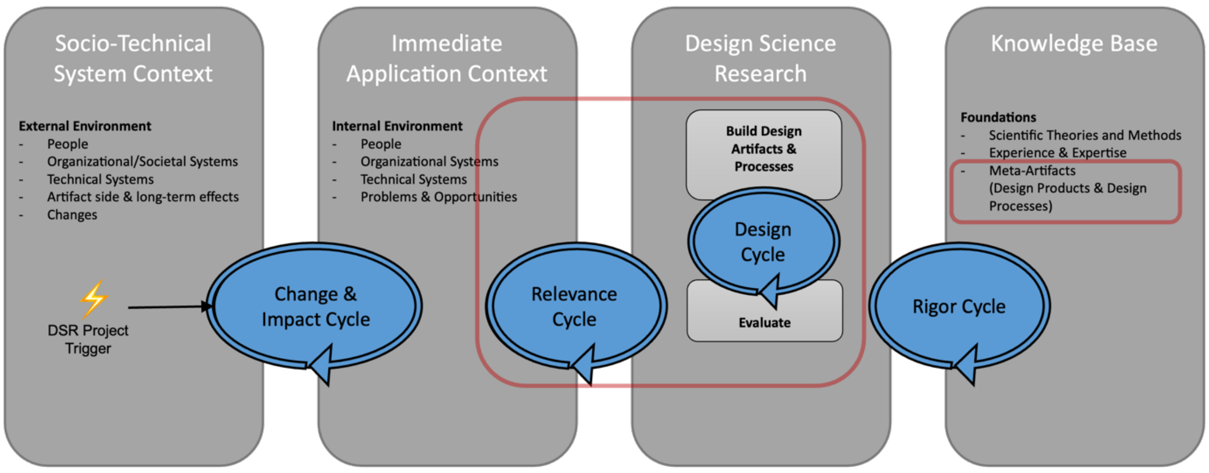

Figure 1. Four Cycle View of Design Science Research (simplified from (Drechsler and Hevner 2016))

The *design cycle* of DSR focuses on the artifact (re)design and artifact evaluation. 'Artifacts' in the context of this work are simulation components represented as source code during the development phase, as well as executable artifacts at the runtime level when simulation components are used to simulate a defined scenario.

As per the definition of DSR, the simulation artifacts will be iterated (redesigned, refined) multiple times across the simulation development life cycle. As external environments may change drastically fast in case of disasters (Earthquakes in Turkey and Syria in 2023, Ukrainian Crisis since 2022, Ahrtal floodings in Germany in 2021, Covid 19 since 2020), this also triggers changes in the *change & impact cycle* as well as the *relevance cycle,* which directly affects the *design cycle*, i.e., how and what to simulate. To improve resilience, simulations need to be adopted as fast as possible to identify the matching resilience measures to be taken.

Formally speaking, this work-in-progress paper addresses the *process model* within (and between) the *relevance cycle* and the *design cycle*. The proposed process model is positioned within the *rigor cycle* as a meta-artifact of the knowledge base (marked in red in Figure 1).

A variety of simulation artifacts "processed" in the *relevance cycle* and the *design cycle* exist, that are defined as followed:

- *Model*: A mathematical or logical representation of a system of entities, phenomena, or processes (from (Open Simulation Platform 2022)).
- *Sub-model*: A model which is part of a larger model (from (Open Simulation Platform 2022)).
- *Simulation*: The computation of the behavior of a model under specified conditions (from (Open Simulation Platform 2022)).
- *Co-simulation*: A simulation technique in which the computations associated with different sub-models are performed mostly independently from each other, the only interaction being the exchange of output values at

certain discrete points in time (communication points) (from (Open Simulation Platform 2022)).
- *Coupled simulation*: A simulation technique in which the computations associated with different sub-models are intertwined.
- *Combined simulation*: A logical generalization of co-simulations and coupled simulations.
- *Distributed simulation*: The distribution of the execution of a simulation program across multiple processors (Fujimoto 2000).
- *Simulator* (or *simulation program/software/tool*): Software for the purpose of carrying out computer simulations (from (Open Simulation Platform 2022)).
- *Sub-simulator*: A simulation program that performs the computations associated with a sub-model in a co-simulation (from (Open Simulation Platform 2022)).
- *Composite simulator*: A simulator that is composed of two or more simulators.
- *Cooperating simulators*: A composite simulator pattern where two or more simulators use the results of one or more simulators as input.
- *Module*: A software component using one or more simulation artifacts to carry out a simulation for a specific simulation objective being *validation, what-if analysis, enhance, forecast,* and *training*.

Combined simulations may use one or more underlying simulators based on different technologies and products, e.g., Tensor Flow for machine learning algorithms, flow networks for traffic and communication networks, physics simulations, agent-based systems. Simulations may be run on-demand (even thousands of times for Monte Carlo simulations for optimization problems) or continuously (for creating continuous forecasts of developing situations of real-world events to support event management as well as emergency response teams.

Over the past decades, various theoretical and practical models have been developed on how to develop and run simulations and combined situations. Most famously, the *High Level Architecture (HLA)* publicly available as a set of standards (IEEE Computer Society 2010) influenced a lot of other work and is used up to this point. The CSIAC currently creates a comprehensive approach towards “Modelling & Simulation as a Service” since 2018 (Siegfried et al. 2018) which covers plenty of aspects and is built upon modern technologies such as messaging and containerization, which is unfortunately not open to the public. Further approaches compatible with modern software architecture targeting *simulation-as-a-service* principles are proposed by (Gütlein et al. 2021; Gütlein and Djanatliev 2020). A workflow architecture for cloud-based distributed simulations is proposed by (Chaudhry et al. 2022). All those approaches seem to miss explicit tasks (process steps) for interlocking the *relevance cycle* with the *design cycle* to speed up the development and deployment of the simulation artifacts, as well as to elevate the artifact *evaluate task* in the *design cycle*. Furthermore, they do focus on the simulation artifacts, but do not emphasize the relevance of the simulation results, i.e., the generated data.

In the case of combined simulations, where multiple simulation artifacts are combined from different domains, the number of changes/iterations multiplies even further. To reduce effort/friction when executing/evaluating the iteratively refined artifacts ('simulation runs' in our case), a matching collaborative process model is required to ease the creation, deployment, and execution/evaluation of the simulations. As interpreted in an agile mindset, the goal is to reduce ‘*waste’* and increase the ‘*velocity’* in the *simulation development life cycle* (SDLC*)*.

That *simulation development life cycle* shows parallels to the creation of software artifacts (software development life cycle) in at least two aspects: (1) Creating simulation components corresponds to the software development phase, and (2) "running" one or more simulation components within a scenario corresponds to the operations phase of software development, on the logical level representing the "runtime infrastructure" using a term coined by the HLA, cf. (IEEE Computer Society 2010).

In a "modern" software development process, software artifacts are "owned" (created and managed) by cross-functional teams with a high degree of specific domain knowledge and autonomy. An update of the source code usually results in the (automatic) execution of a build pipeline (Continuous Integration, Continuous Deployment, CI/CD), that performs automated steps for testing, compiling, building, and deploying the software artifacts.

Data Mesh

Data in enterprise IT landscapes are usually divided into an *operational plane* and an *analytical plane*, each having distinct usages for the data: The operational plane manages data (create, process, store) relevant for the day-to-day business of the organization/enterprise and therefore focuses on the present, whereas the analytical plane provides analytical and historical data derived from the operational plane in order to analyze those afterwards (Dehghani 2022).

Simulators and running simulations can be understood to be part of the operational plane having internal transactions, data processing, and internal data storage. The resulting data sets of a simulation (and even the intermediate steps) logically belong to the analytical plane. Result data can be used later on for multiple use cases (e.g., aggregating historical data of multiple simulation runs for optimization problems for decision support systems). Data on the analytical plane as generated by simulation runs therefore is of further business value to drive the *relevance cycle*. To match the combined simulations and simulation artifacts, a *data platform architecture* that represents the most suitable way of data management of the cross-domain input data and output data must be identified.

The *data mesh* is a recent approach initially introduced in 2019 by Dehghani and has been better worked out in 2022 (Dehghani 2019, 2020, 2022). The "data mesh objective is to create a foundation for getting value from analytical data and historical facts at scale" (Dehghani 2020), and therefore treats data as "first-class citizens" and even products within an IT landscape. Unlike previous approaches of a "data warehouse" and a "data lake" it solves issues when dealing with data ownership and bounded contexts (see Figure 2).

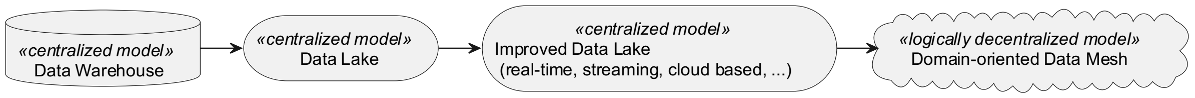

Figure 2. Evolution from Data Warehouse to Data Mesh

Four "collectively necessary and sufficient" principles are proposed in order to make (analytical) data usable through a data mesh (Dehghani 2020):

1. domain-oriented decentralized data ownership and architecture,
2. data as a product,
3. self-serve data infrastructure as a platform, and
4. federated computational governance.

Implementing those principles gives structure to how the various simulation components (simulators, modules, management of raw data) are to be arranged in the overall system architecture, both paying respects to the technology dimension as well as the organizational dimension of collaboration between simulation teams and customers. This "inverted model and topology based on domains and not technology stack" (Dehghani 2020) seems fitting, because this value-driven approach prioritizes the importance of the *relevance cycl*e by means of asking questions "why" first (use cases, business cases), instead of technical details ("how") from the *design cycle*. This supports a higher velocity in creating time-critical simulations as required in crisis situations by focusing on relevant design artifacts and therefore reduces possible "waste", i.e., is less likely to "produce no value for the customer or user" (Sedano et al. 2017).

Understanding data as products and keeping them in the decentralized bounded-context of the originating domain resembles the *microservices approach* (Newman 2021), which is fitting and goes hand in hand with the "as-a-service" approach for using the simulation artifacts of the platform.

PROCESS MODEL AND PLATFORM ARCHITECTURE

To achieve efficient higher-order value for complex simulations, it is important to establish strong collaboration and compatible communication channels between the various domains and teams involved. Drawing upon the strengths of each individual domain, it must be ensured, that:

1) a common mindset is established between the producers (experts, developers) of the simulation components as well as the consumers (fellow researchers, disaster responders, decision-makers, emergency managers). Consumers configure and execute simulations that rely on those simulation components. Having a common understanding eases collaboration which is required; especially when developers are users as well, and when services shall be provided as a business case.
2) simulation components are as loosely coupled as possible, to not restrict and to not slow down the producers. Producers must be free to use the most fitting technology regarding a problem.

Consumers trigger simulations on the operational level. Depending on the implementation of the simulation and the used simulators, the results (and even intermediary results) are processed by the matching *simulator domain* and are published/persisted on the analytical plane.

From the consumer's point of view, data generated and processed in/by the operational plane is not relevant. Yet, simulation components may depend on both, operational data (and operational capabilities) and analytical data as input. Both kinds of data shall be accessible via well-defined and documented data access interfaces logically managed/provided by the source domain. On the technical level, the data access may be provided ("served") using a multitude of interface styles in a polyglot form, such as declarative APIs (REST endpoints, GraphQL, gRPC), files on a file server, event streaming, or a database access (relational SQL, NoSQL).

Data Platform Architecture

As identified in the preceding sections, both the simulation artifacts as well as the data artifacts used in the *relevance cycle* and the *design cycle* can be understood as products. Those products shall be "self-serve" and offered as services that are used ("consumed") by the consumer configuring and running simulations to meet their goals (an abstract view seen in Figure 3).

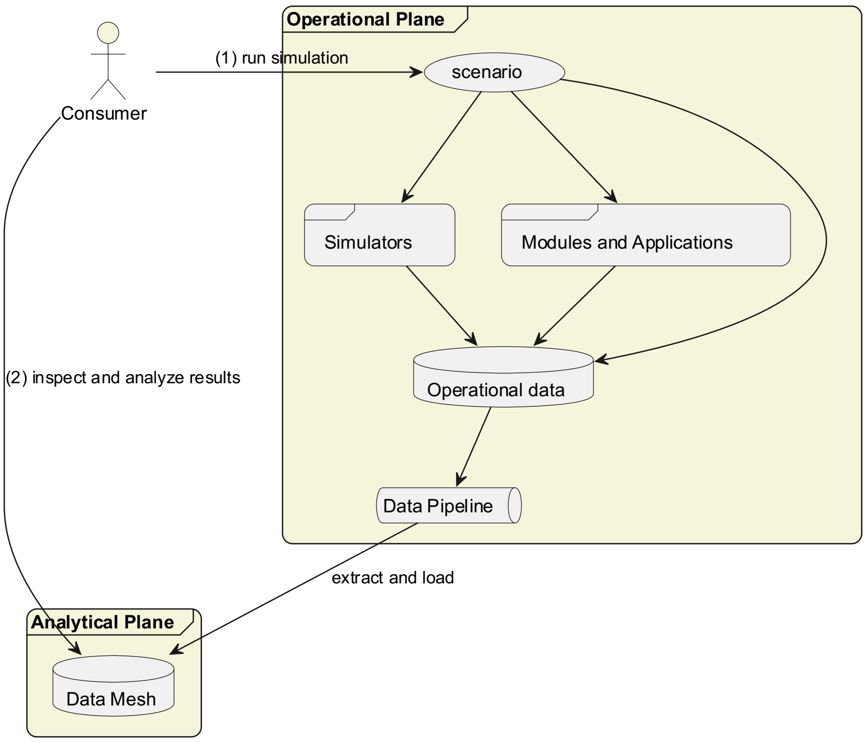

Figure 3. Running a simulation on the platform

From a business perspective of the *relevance cycle*, the platform is suitable, if it is "discoverable, addressable, trustworthy, self-describing, interoperable, and secure"[1] (Dehghani 2022). The architecture shall be able to easily scale required infrastructure components when new simulators/simulations are on-boarded, e.g., when adding new computing resources, or new databases in data mesh). For a conceivable technical implementation, it is currently envisaged to heavily rely upon containerization for realizing all simulation and data services running on a public/private/hybrid cloud environment.

As shown in Figure 4, the platform is logically comprised of four building blocks:

- *Artifact Management*: Simulation components in their "raw" form (as source code or as deployable artifacts) are stored in the repositories. Using build pipelines (continuous integration, continuous deployment), the raw artifacts are prepared, and they are deployed onto the platform itself as running artifacts. The pipelines may be triggered automatically (e.g., when the source code has been updated), or actively by a user of the platform.
- *Simulators-as-a-Service*: Once created by the *artifact management*, simulator components (not actual simulations!) are deployed to the platform as actual instances and can be used for the simulations. Simulators may be basic (atomic) simulators or composed simulators. Each (composed) simulator is made available "as-a-service", i.e., it has a distinct interface. Each simulator component may be instantiated multiple times at once on the platform to support scalability, or to support running multiple simulations at once each using a dedicated instance if a simulator component.

[1] Those objectives can be realized providing 15 capabilities on a more detailed level named in the chapter "Process and Role Integration".

- *Modules and Applications*: Simulations may be run having different objectives which, in turn, require different usages of the deployed simulation components. Those objectives are bundled as modules which in turn may contain or realize different applications (having an actual graphical user interface for the users).
- *Data Mesh*: This is the data platform layer of the simulation platform. This logical building block contains capabilities of data management as realized by the data products. The data products use various persistence and communication/messaging technologies, such as relational databases or a message broker.

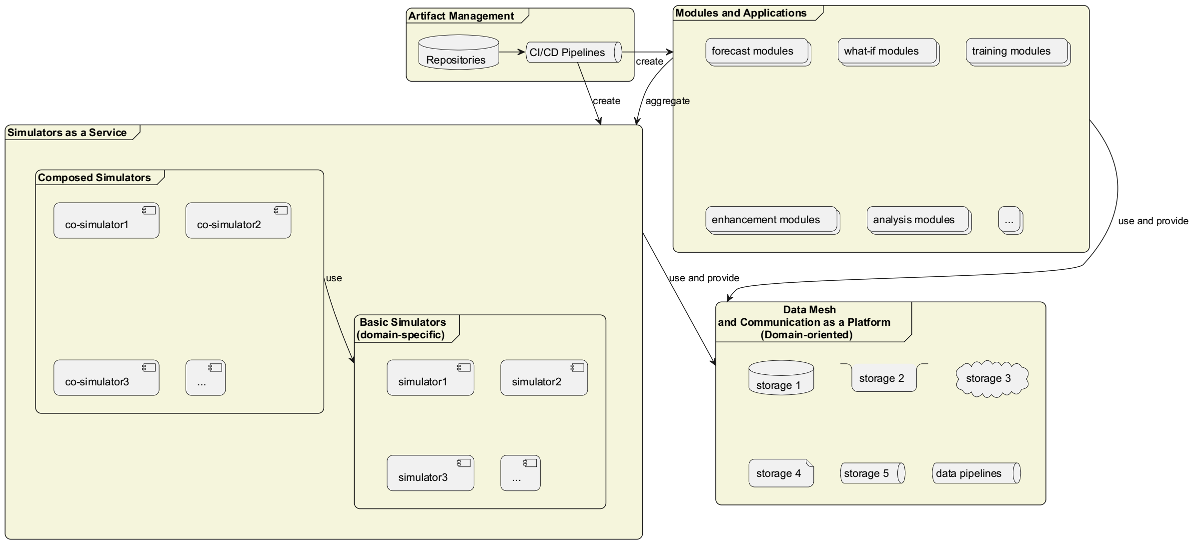

Figure 4. Platform Architecture

Process and Role Integration

This section outlines the identified roles (and their responsibilities) that use the platform to develop and (re-)use the simulation artifacts and data artifacts. Using the building blocks of the platform presented above, the usage of the building block from the individual perspective of each role is presented.

Producer Role

The *producer role* creates products within clearly defined bounded contexts. Products are simulation components (simulators, co-simulators, modules) as well as data products to be used by the consumer role (see Figure 5).

As a cross-functional team, members of the producer role are responsible for the code, artifacts, data, and parts of the infrastructure within their assigned domain(s) to provide value to the consumer. This includes development, release, and deployment of simulation artifacts and data product artifacts. As part of the data product artifacts, the producer role also manages “external data sets” (e.g., geographic data, structural designs, infrastructure networks, sensor data, social media data), which is domain-specific data from external sources required to run the simulations that is preprocessed to match the data platform. Maintaining and serving the source-domain data products includes cleansing, deduplicating, preparing, and aggregating which is supported by as much automation as feasible using data pipelines of the producers' domain(s). The producer role is not only a technical role: Established *service level objectives* to assure relevance and quality of the provided artifacts to the consumers is vital. A deep understanding of how consumers will use the (data) products and support them in being "comfortable" how to use the products is realized through openness and intensive communication with the customers. Customers may be producers of other simulation/data products, or they are “end users” of those products located in the *relevance cycle*.

Finally, as a positive side effect, this "cross-skill pollination" between simulation experts, software engineering experts, data engineering experts across team members, and their customers promotes a beneficial exchange of knowledge (Dehghani 2019).

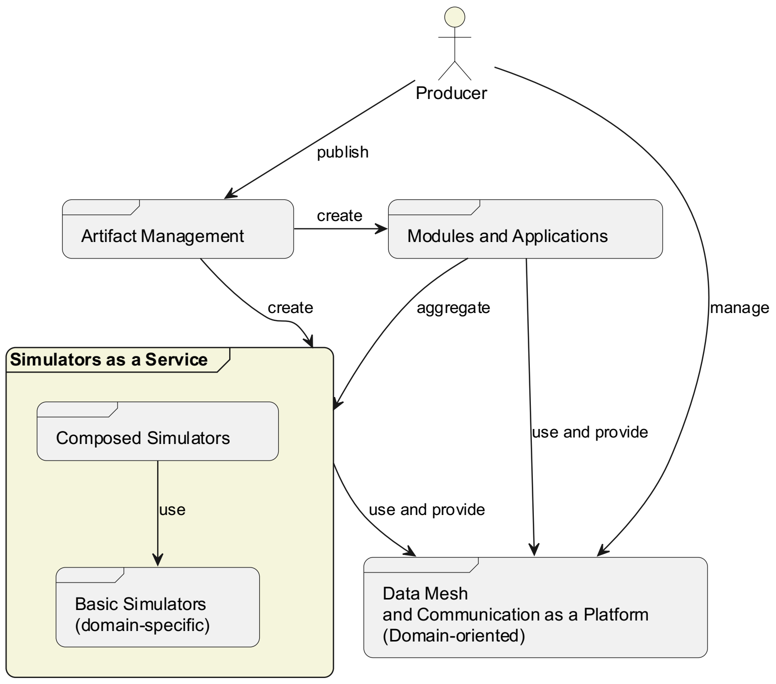

Figure 5. View of the Producer role

Consumer Role

The *consumer role* uses simulation products and data products to run simulations and therefore gains knowledge and support in finding answers to their hypotheses, and business goals (see Figure 6).

A consumer defines a scenario for a simulation with a specific goal in mind (forecast, enhancement, what-if analysis, validation, training) using the offered (cross-domain) simulation products and data products from the platform. Depending on the goal, the simulation runs interactively or as a (non-visible) job in the background.

On a technical level, the consumer defines which simulation services and data products to consume within a simulation scenario. The resulting data from the simulation is provided as browsable and/or downloadable data artifacts.

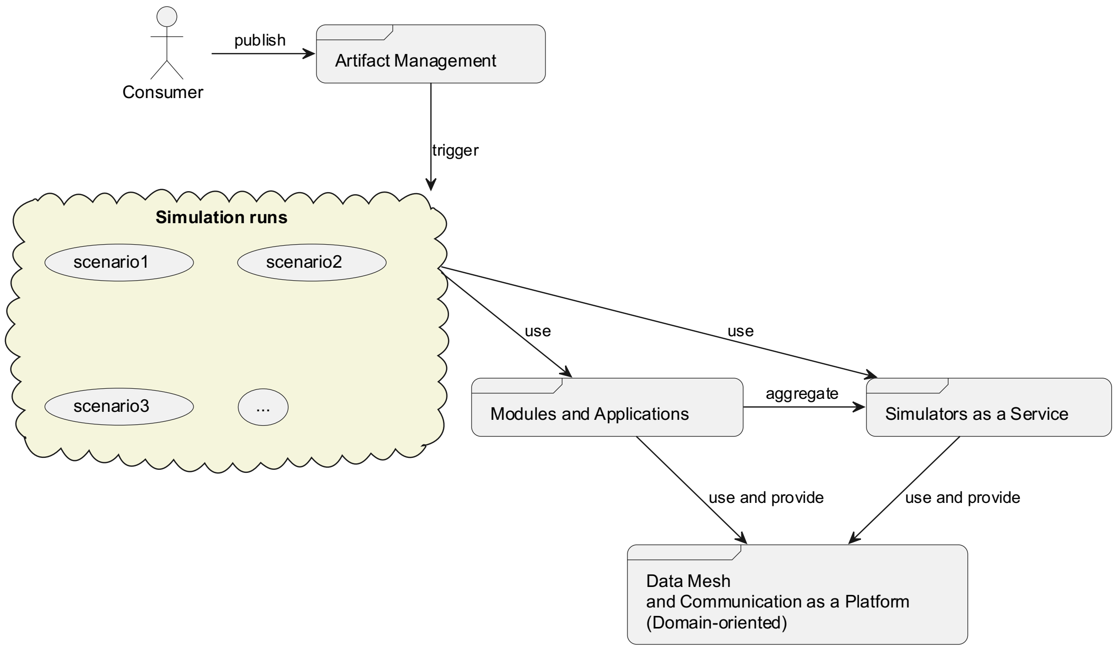

Figure 6. View of the Consumer role

Supporting Roles: Architectural Role and Operations Role

The producer role and the consumer role are assisted by the *architectural role* and the *operations role*. Despite not being actively involved in running the simulations – because the platform is self-serve – they address the fitness between the different cycles of DSR by supporting the “adaption to change” and the “goodness of fit”.

The team members of the architectural role support producers and consumers to use the platform as efficiently as possible by reducing the complexity of the designed artifacts (simulation products and data products) by offering adequate abstractions. This includes supporting producers in identifying and implementing the “architectural quantum” (Richards and Ford 2020) needed for a product to generate value, i.e. the combination of the operational system, CI/CD pipelines, simulation component, data product, and interfaces.

The team members of the operations role live outside the domains. They provision, run, and manage the platform and underlying infrastructure components. Therefore, the supporting roles provide one, more, or all of the capabilities as suggested by Dehghani: *(1) Scalable polyglot big data storage, (2) Encryption for data at rest and in motion, (3) Data product versioning, (4) Data product schema, (5) Data product de-identification, (6) Unified data access control and logging, (7), Data pipeline implementation and orchestration, (8) Data product discovery, catalog registration and publishing, (9) Data governance and standardization, (10) Data product lineage, (11) Data product monitoring/alerting/log, (12) Data product quality metrics (collection and sharing), (13) In-memory data caching, (14) Federated identity management, (15) Compute and data locality* (Dehghani 2020).

Benefits and Limitations for Usage Scenarios

The software architecture and process integration of the *simulation-as-a-service* platform were outlined in the previous sections at a conceptual/logical level. However, no information has yet been provided on how the platform could be utilized in a real-world scenario. Once implemented, to satisfy the relevance cycle, the platform should be usable both outside of a disaster situation as well as during an ongoing (sudden-onset or slow-onset) disaster. As no restrictions can be assumed on which simulation objectives (e.g., *training*, *validation*, *what-if analysis, enhancing, forecast*) will be needed during a situation, the platform is intended to support all objectives[2] during a disaster/emergency as depicted in Table 1.

Table 1. Intended applications for the simulation platform

Disaster Situation	**Simulation objective**				
	Training	Validation	What-if	Enhancement	Forecast
None	x	x	x	x	x
Slow-onset	x	x	x	x	x
Sudden-onset	(x)	x	x	x	x

Different deployment strategies and operating environments may be required depending on the usage scenario. Simulations (e.g., *training*, *validation*, *what-if analysis*) performed outside of a disaster situation considering parameters such as geography and community-specifics, do not have restrictions on the IT environment. Therefore, a full-featured (private) cloud environment can be used. Running simulations (on-site) during an ongoing disaster/crisis/emergency can be achieved two-fold: 1) using a running simulation platform at another location unaffected by the disaster, or 2) using an on-site standalone deployment of the simulation platform (*self-reliant, mobile, off-grid, off-line*), e.g., powered by a battery/generator and without the need for external online/cloud-based services. Trade-offs must be considered in both options. The former requires a reliant communications infrastructure, which might be unavailable in the disaster area. The latter is likely to be restricted in computational power and access to simulation experts. Due to the limitations in the latter option, the simulation platform will likely be used mainly from a consumer role; although simulations can be configured, run, and combined, the creation of new simulation components (simulators, applications) will be reduced.

[2] Possibly except for ‘*training*’, as this seems not to be feasible in sudden-onset situations, due to time-criticality.

CONCLUSION AND FUTURE WORK

This work-in-progress outlines a process model as well as a platform architecture to strengthen and speed up the development and usage of cross-domain simulations to improve resilience. It emphasizes the collaboration between experts, researchers, and disaster management. This "*SimulationOps*" knits together aspects of the Four-Cycle Model for Design Science Research, domain-awareness and bounded context, Data Meshes, and an "as-a-Service" platform architecture driven by the following core principles:

- *Value and Velocity*: Cross-domain simulations can be created with high quality, high velocity, and low waste. Closely linking the *design cycle* and the *relevance cycle* together allows for fast iterative improvements that positively impact the *change and impact cycle* and improve the fitness function of "adaptation to change". This ultimately leads to a quicker and better answer to the original questions/hypotheses that led to the creation of the simulations, particularly the expected insights gained for decision-making in crisis and disaster situations.
- *Simulation-as-a-Service*: Through appropriate abstraction layers, not only domain experts and simulation experts but also external customers can be enabled to perform simulations.
- *Data-as-a-Product*: Different simulation objectives are enabled (validation, what-if, enhance, forecast, training) by not only running new simulations but also gaining new insights based on data from existing simulation runs (based on analytical data).

This paper outlines just a first draft, and much further work is still required, and open issues must be answered. To this end, the following items are suggested as starting points:

- Solve how to handle bounded contexts for analytical data that is based on *coupled simulations* and therefore shared data ownership.
- Evaluate existing solutions to realize the data mesh.
- Identify requirements across all envisioned simulation components and derive matching data products.
- Identify a matching *federated computational governance* to give orientation for decision-makers and other stakeholders.
- Follow an iterative approach to turn the process model and architecture platform into reality by implementing a minimum viable product, especially taking into account and prioritizing the 15 capabilities defined by (Dehghani 2020) (see section "*Supporting Roles*").
- Design a resilient deployment and operations model of the *simulation platform*, as a standalone (self-reliant and mobile) solution with no need for external online/cloud-based services.

Finally, following the DSR approach, the initially created minimal viable platform artifact itself needs to be evaluated in the design cycle, and relevance and value must be evaluated in the relevance cycle with the help of fellow researchers and other stakeholders.

REFERENCES

Chaudhry, N. R., A. Anagnostou, and S. J. E. Taylor. 2022. "A Workflow Architecture for Cloud-based Distributed Simulation." *ACM Trans. Model. Comput. Simul.*, 32 (2): 1–26. https://doi.org/10.1145/3503510.

Daudé, É., K. Chapuis, C. Caron, A. Drogoul, B. Gaudou, S. Rey-Coyrehourcq, A. Saval, P. Taillandier, and P. Tranouez. 2019. "ESCAPE: Exploring by Simulation Cities Awareness on Population Evacuation."

Dehghani, Z. 2019. "How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh." *martinfowler.com*. Accessed February 16, 2023. https://martinfowler.com/articles/data-monolith-to-mesh.html.

Dehghani, Z. 2020. "Data Mesh Principles and Logical Architecture." *martinfowler.com*. Accessed February 16, 2023. https://martinfowler.com/articles/data-mesh-principles.html.

Dehghani, Z. 2022. *Data mesh: delivering data-driven value at scale*. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly.

Drechsler, A., and A. Hevner. 2016. "A Four-Cycle Model of IS Design Science Research: Capturing the Dynamic Nature of IS Artifact Design." *Breakthr. Emerg. Insights Ongoing Des. Sci. Proj. Res.--Prog. Pap. Poster Present. 11th Int. Conf. Des. Sci. Res. Inf. Syst. Technol. DESRIST*. St. John, Canada.

Fujimoto, R. M. 2000. *Parallel and Distribution Simulation Systems*. New York: Wiley.

Ganji, A., N. Alimohammadi, and S. Miles. 2019. "Challenges in Community Resilience Planning and Opportunities with Simulation Modeling." *ArXiv*.

Gütlein, M., and A. Djanatliev. 2020. "Modeling and Simulation as a Service using Apache Kafka." *Proc. 10th Int. Conf. Simul. Model. Methodol. Technol. Appl.*, 171–180. Lieusaint - Paris, France: SCITEPRESS - Science and Technology Publications.

Gütlein, M., R. German, and A. Djanatliev. 2021. "Hide Your Model! Layer Abstractions for Data-Driven Co-Simulations." *2021 Winter Simul. Conf. WSC*, 1–12. Phoenix, AZ, USA: IEEE.

IEEE Computer Society. 2010. "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Framework and Rules." *IEEE Std 1516-2010 Revis. IEEE Std 1516-2000*, 1–38. https://doi.org/10.1109/IEEESTD.2010.5553440.

Newman, S. 2021. *Building microservices: designing fine-grained systems*. Beijing Boston Farnham Sebastopol Tokyo.

Open Simulation Platform. 2022. "OSP Interface Specification (OSP-IS)." Open Simulation Platform.

Osaragi, T., and N. Hirokawa. 2019. "Simulation Analysis of Fire Hydrant Usability Levels after Large Earthquake."

Paton, D., and D. M. Johnston (Eds.). 2006. *Disaster resilience: an integrated approach*. Springfield, Ill: Charles C Thomas.

Ramírez, J., M. Mendes, and S. Monedero. 2015. "Enhanced forest fire risk assessment through the use of fire simulation models."

Richards, M., and N. Ford. 2020. *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol, CA: O'Reilly Media, Inc.

Ron Ross, Victoria Pillitteri, Richard Graubart, Deborah Bodeau, and Rosalie McQuaid. 2021. *Developing Cyber-Resilient Systems: A Systems Security Engineering Approach*. NIST SP 800-160v2r1. Gaithersburg, MD: National Institute of Standards and Technology.

Sedano, T., P. Ralph, and C. Peraire. 2017. "Software Development Waste." *2017 IEEEACM 39th Int. Conf. Softw. Eng. ICSE*, 130–140. Buenos Aires: IEEE.

Siegfried, R., J. Lloyd, and T. van den Berg. 2018. "A New Reality: Modelling & Simulation as a Service." Accessed November 29, 2022. https://csiac.org/articles/a-new-reality-modelling-simulation-as-a-service/.

Simon, H. A. 2008. *The sciences of the artificial*. Cambridge, Mass.: MIT Press.

UNDRR (Ed.). 2016. "Report of the Open-ended Intergovernmental Expert Working Group on Indicators and Terminology relating to Disaster Risk Reduction." *A_71_644-EN*. United Nations.

Wilde, T., and T. Hess. 2006. *Methodenspektrum der Wirtschaftsinformatik: Überblick und Portfoliobildung*. Arbeitsbericht. München: Ludwig-Maximilians-Universität München, Institut für Wirtschaftsinformatik und Neue Medien (WIM).

Zobel, C. 2020. "Virtual and Live Simulation-Based Training for Incident Commanders."