Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

A MODULAR SIMULATION MODEL FOR MASS CASUALTY INCIDENTS

Kai Meisner

Medical Service Capability and
Force Development
Bundeswehr Medical Academy
Neuherbergstraße 11
München, 80937, GERMANY

Heiderose Stein
Nadiia Leopold
Tobias Uhlig
Oliver Rose

Department of Computer Science
University of the Bundeswehr Munich
Werner-Heisenberg-Weg 39
Neubiberg, 85577, GERMANY

ABSTRACT

During military conflicts, the number of casualties is likely to exceed medical capabilities. For best treatment results, the patients must be distributed according to their needs to the available resources such as medical facilities and means of transportation. Computer simulations are used to verify and optimize current medical planning. However, recent models lack the capability of testing a wide range of decision rules. In this paper, we address this issue and propose a modular simulation concept whose components can be adapted and exchanged independently. Using modular submodels to control the simulated objects, we enable the implementation of a wide range of object behavior. A prototype implementation of the proposed concept is presented, showing the effects of applying different dispatching rules in an evacuation scenario.

1 INTRODUCTION

A mass casualty incident (MCI) is an event in which the medical need exceeds the response capabilities in the affected area (Debacker et al. 2012). These events are often caused by extreme weather conditions, terrorism, or epidemics (Wallemacq and House 2018). Especially during military conflicts, MCIs are likely to occur due to the medical resources being very limited (Neitzel and Ladehof 2015). At the same time, the number of casualties and evacuation times are expected to be high (Neitzel and Ladehof 2015). This can currently be observed in Ukraine, where at least 200,000 soldiers died since the Russian invasion in 2022 (Cooper et al. 2023). Neitzel and Ladehof (2015) described that during combat, wounded soldiers receive treatment along a hierarchical medical evacuation chain, as visualized in Figure 1.

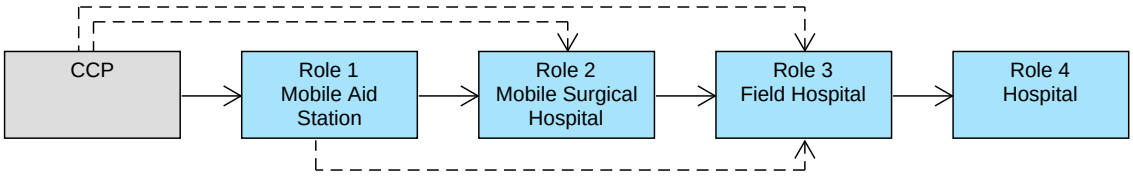


Figure 1: Casualties initially arrive at the casualty collection point (CCP) and subsequently receive treatment along the four roles (solid arrows) with increasing medical capabilities. Certain roles can be skipped to avoid bottlenecks (dashed arrows).

The military evacuation chain consists of a casualty collection point (CCP) and four different roles, namely Role 1 to Role 4. Medical treatment along this chain is to ensure that patients are evacuated from the battlefield while maintaining continuum of care. Each role contains a set of medical facilities with different resources and capabilities, such as operating rooms and intensive care units. Their resources and capabilities are based on the ones of the previous role and extend these. The path of a patient along this chain is visualized by arrows in Figure 1. Accordingly, casualties enter the medical evacuation chain at the CCP, where they are collected during combat. From there on, the patients get treated along the different roles of this chain in increasing order. Transportation between the roles, in the following denoted as "evacuation", can either be performed ground-based or airborne. Unlike the concept of a strict hierarchical chain, current considerations lean toward a more flexible approach, where roles can be skipped (Sell 2019). This may help to improve patient distribution among the available resources and therefore avoid bottlenecks in early roles (Neitzel and Ladehof 2015). The alternative paths are visualized by dashed arrows in Figure 1.

An even distribution of patients to the medical resources according to their needs in a timely manner is a complex logistic task and requires accurate planning. NATO's current medical plans are based on assumptions from which casualty rates and the most likely scenarios are derived (NATO 2019). Meisner et al. (2023) pointed out that constructive simulation is essential for verifying these plans and that current models lack the flexibility needed to represent the dynamics of combat. As discussed by Tippong et al. (2022), current research also does not cover the design of new network structures and the coordination of ambulance sharing. However, these two are essential for validating existing medical evacuation chain planning and developing new concepts. Especially for generating and testing new medical dispatching policies, a flexible simulation model is essential to allow testing a broad range of different concepts.

First considerations of an appropriate simulation model were presented in (Meisner et al. 2023). Based on the briefly described idea of a decoupled simulation model, we propose a modular simulation model, which offers the required flexibility by supporting a wide range of scenarios and decision policies. Our paper is structured as follows: In the next section, we briefly describe the medical evacuation chain and discuss previously introduced simulation models. This results in a description of current shortfalls regarding the models' flexibility in Section 3. Our conceptual approach, its prototype implementation, and the results of a high-level evacuation scenario are presented in Section 4. Finally, we conclude our work and give a short outlook.

2 RELATED WORK

There is a substantial body of literature covering the optimization of medical evacuation using operations research techniques (e.g., Frial 2022; Jenkins et al. 2023). Yue et al. (2012) pointed out, that these approaches often fail to fully encompass the dynamics of medical response. Lechtenberg et al. (2017) proposed simulation as a possible technique to cope with this disadvantage.

A simulation of the medical evacuation chain during active combat was presented by Kleint and Geck (2022). The authors implemented a model for planning a sustainable evacuation chain and reviewing existing concepts. Different scenarios can be investigated by adjusting the frequency of occurrence of certain injury patterns. Various parameters regarding the quantitative and spatial planning of Medical Service resources are modifiable. Accordingly, this model aims to investigate whether the capacities of the given resources can provide the required medical capabilities. Examining the effects of different dispatching policies, however, is outside of the model's scope (Kleint and Geck 2021).

Evacuation chains with similarities to the ones applied in the military are also used during civil MCIs, as shown by Debacker et al. (2016). The authors proposed a model for simulating the medical response in the context of an airplane crash. Four different service points are defined as visualized in Figure 2. Here, patients are generated at the MCI site from where they can take different routes. Each of these service points prioritizes, treats, and evacuates the arriving patients. The path of each patient along the evacuation chain is determined by the victim's type of injury and the selected operational policy. In contrast

to the earlier described military medical evacuation chain, the route is not determined dynamically concerning certain parameters such as resource utilization. There are currently two distinct policies to choose from. Furthermore, triage procedures that lead to different prioritization of patients are customizable. The quantity and quality of resources can be adjusted, too. De Rouck et al. (2018) reimplemented this model allowing easier parameter adjustment and therefore higher flexibility without changing its main characteristics.

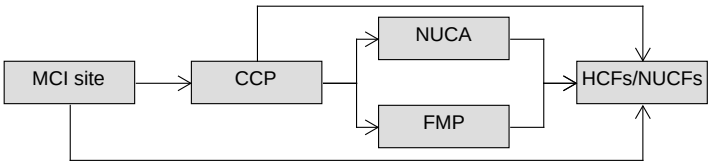


Figure 2: Debacker et al. (2016) defined the MCI site and the four service points. Patients can either directly get to the healthcare facilities (HCFs) and non-urgent care facilities (NUCFs) or move along a chain that contains the CCP and/or either the forward medical post (FMP) or the non-urgent care (NUCA).

3 PROBLEM STATEMENT

Planning the medical evacuation chain by focusing on the required resource capacities is not sufficient. Since MCIs are likely to occur, the number of casualties will exceed the medical capabilities by definition at one point in time and/or location. Thus, we especially need to investigate how the available resources can be utilized more efficiently. For this purpose, new network structures and flexible decision rules must be validated, as proposed by Tippong et al. (2022). Particularly with respect to dispatching, a wide range of aspects has to be examined in different scenarios, e.g.:

- How should the patients be prioritized regarding transportation and treatment?
- Which transporter should be used for transporting a specific patient?
- To which medical facility should this patient be brought?
- Should the transporter pick up additional patients from other locations?

The models described in Section 2 either only focus on spatial and quantitative planning (Kleint and Geck 2021) or offer a small range of predefined dispatching policies (Debacker et al. 2016). Both models lack flexibility in investigating a wide range of decision rules. Specifically, they do not allow testing new policies without implementing them in the existing model. This requires significant effort due to the need to understand the model's structure. Therefore, a flexible model should not only provide different strategies but also enable easy implementation of new ones. This could be achieved by providing generic interfaces to define the behavior of simulated objects instead of implementing concrete strategies. This approach enables later adjustments to meet specific requirements and enables the specification of a broader range of scenarios compared to mere parameterization. For instance, complex casualty arrival processes can be defined for an existing model if the appropriate interface is provided. To achieve this desired flexibility, we propose a modular simulation model that allows flexible adjustment of the behavior of simulated objects, enabling testing of a wide range of decision policies in various scenarios.

4 MODULAR SIMULATION MODEL

In this section, a modular and decoupled simulation modeling approach is presented, aiming to provide the described flexibility. Initially, an overview of the proposed method is given. Based on that, the conceptual design is described and a prototype implementation presented. This prototype is then used to investigate a high-level evacuation scenario. Finally, the results of this study are described.

4.1 Approach

Kleint and Geck (2021) implemented their model using the commercial simulation software AnyLogic. Utilizing the same software, Possik et al. (2021) showed how decoupling the visualization from the simulation model provides more flexibility. For this purpose, the High Level Architecture framework (HLA) (IEEE Computer Society 2010) was used, which defines major functional elements, interfaces, and design rules to allow re-usage and interoperation of simulation systems and assets. Via a publish/subscribe-concept, the simulation sends updates to the visualization, which processes the event accordingly. This allowed an independent development of both components while additionally using different technologies.

Inspired by this approach, we propose to further decouple simulation models as shown in Figure 3. Here, the red rectangles represent the different components of the simulation. An arbitrary number of submodels is visualized by a green rectangle. Note that the same colors are used for corresponding components in later figures in this paper, too. The purpose of each component and the submodels are the following:

- **Model**: Represents the simulated scenario containing all objects and their states.
- **Event engine**: Schedules timed events and triggers their execution at the model.
- **Visualization**: Visualizes the current model state.
- **Submodels**: Interact with the model via predefined interfaces controlling the simulated objects and triggering their state changes at the model.

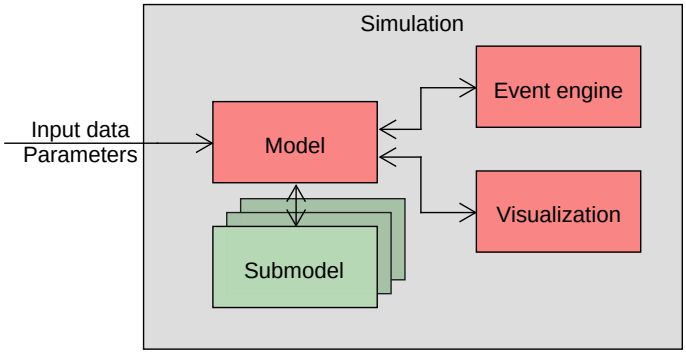


Figure 3: In the proposed structure, the simulation is divided into three components (red) and a set of submodels (green).

Dividing the simulation into the three components comes with two advantages. First, by defining their interfaces, the components can be implemented and tested independently, which allows early prototyping and uncomplicated maintenance. In early project phases, e.g., an existing event scheduler may be used as the event engine to save time. Later, this scheduler can easily be replaced by an optimized one, meeting the specific requirements of this simulation. Second, a simulation using this approach does not depend on software offered by a single provider. This may become relevant while dealing with military data.

While input data and parameters define the model, the submodels describe the scenario and object behavior. The model informs each of them about relevant state changes within the simulation. At the same time, it provides an interface for each submodel, which defines a set of basic actions that can be used to control the simulated objects. This allows combining these actions to model complex behavior. Thus, the submodels can react to state changes according to any implemented policy. This way, the actual object behavior is not determined by the model's developer but implemented later via submodels. At the same time, implementing a submodel can be agnostic to the model, only the interface must be met. Consequently, a wide range of object behavior can be modeled without having to change the simulation itself and existing

submodels can easily be adapted and exchanged. Compared to the recently proposed models described in Section 2, this allows significantly higher flexibility when it comes to testing different decision policies.

4.2 Conceptual Design

In this section, the conceptual design implementing the proposed approach is presented. An overview of the concept is shown in Figure 4. Here, the rectangles represent the different simulation components and the arrows the communication between them together with a method and/or the passed parameters. As described earlier, the approach is inspired by Possik et al. (2021) where the authors use HLA. However, it is important to highlight that while our concept and component names may bear similarities to HLA, we do not actually implement this framework. This decision is motivated by the considerable overhead associated with implementing HLA. Additionally, we do not intend to fully exploit the benefits of HLA, as described by Possik et al. (2021), such as running simulation models on separate computers with diverse operating systems implemented in different programming languages.

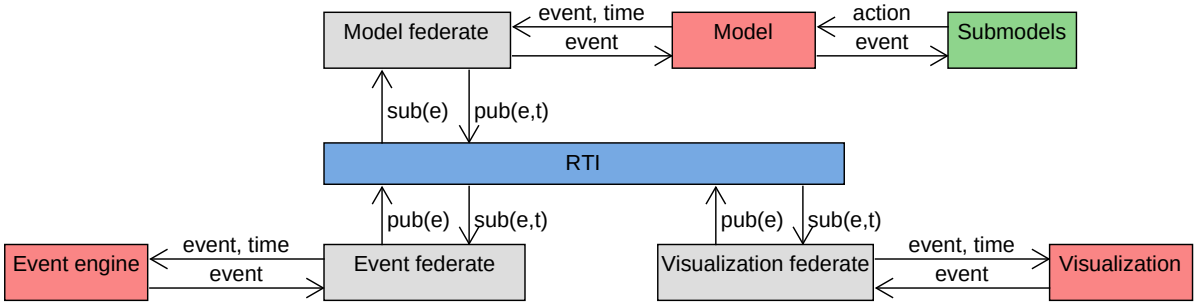


Figure 4: In the proposed conceptual design, the components communicate via a publish/subscribe-concept using the runtime infrastructure (RTI). With each publish (pub) or subscribe (sub), an event (e) and a duration (t) until the event's execution can be passed. The submodels trigger state changes via actions.

The three components model, visualization, and event engine as well as the submodels in Figure 4 correspond to the ones described in Section 4.1. Each component has a federate which is used to communicate with the Runtime Infrastructure (RTI). The RTI coordinates the data exchange between federates. Each federate provides the service of communicating with the RTI. As explained later in this section, the submodels only interact with the model directly, which is why they do not require federates. The communication between the RTI and the federates is implemented using a publish/subscribe-concept. Each federate can subscribe to a set of other federates and gets notified by the RTI if a subscribed federate publishes a message. There are four kinds of messages:

- *pub(e)*: Publishes an event *e*.
- *pub(e, t)*: Publishes an event *e* and the time *t* of its execution.
- *sub(e)*: Receives an event *e* from a subscribed federate.
- *sub(e, t)*: Receives an event *e* and the time *t* until its execution from a subscribed federate.

Via its federate, the model sends events for two purposes: It either informs the event engine to schedule a time-triggered event or it sends its state changes to the visualization. In the first case, the event to be executed is sent together with the duration until the execution. If this delay is linked to any kind of visualization, this event is also sent to the corresponding component. In the second case, the event is not time triggered and the model can execute it without the event engine. Therefore, the notification only needs to be sent to the visualization. Whenever the event engine executes a time-triggered event, the corresponding event is sent to the model for performing the associated state changes and to the visualization which may

show the event to the user. Events may be sent from the visualization to the other two components if a user interaction is required during the simulation execution.

In contrast to the communication described for the components, the submodels directly communicate with the model. For this purpose, the submodels use a set of defined actions to trigger state changes of simulated objects within the model. Thereby, the submodels' behavior can be controlled better. First, the space of possible actions is defined explicitly. Second, we can control which notification each submodel gets from the model to limit its information used for the decision policies according to the scenario. Third and most important, the model checks the feasibility of an action triggered since the submodels do not directly interact with the simulated objects. By only triggering state changes, the model can verify the viability of this action, simulate its effects and notify the other components as well as submodels accordingly.

4.3 Prototype Implementation

To demonstrate the feasibility of the proposed approach, we implemented a prototype considering a simple evacuation scenario shown in Figure 5. Here, casualties arrive at the CCP and need to be transported to one of the mobile aid stations (MAS), where they receive treatment. Each MAS and transporter has a fixed capacity defining the number of patients that can be treated or transported simultaneously. For reasons of simplification, the patients' priority is only defined by their place in the queue, patients cannot die, and only one transporter is considered. Patients that have finished treatment leave the system.

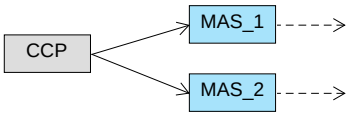


Figure 5: The prototype scenario consists of transporting patients from the casualty collection point (CCP) to one of the mobile aid stations (MAS).

In our prototype, we use the Java programming language and the AnyLogic simulation kernel as our event engine. Further, the built-in visualization of this simulation tool is utilized for reducing the prototyping effort. Thus, these two simulation components together with their federates are combined into an AnyLogic component and its federate, respectively, as illustrated in Figure 6.

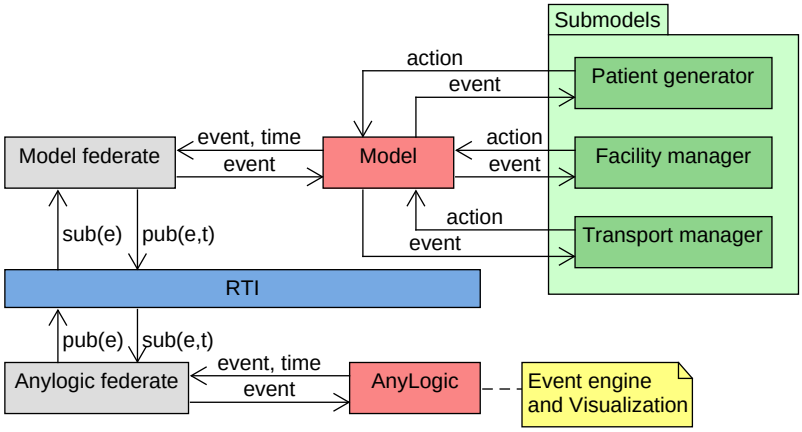


Figure 6: Three different submodels are considered while the visualization and event engine are combined and implemented using AnyLogic.

For controlling the simulation objects, three different submodels are implemented:

- **Patient generator**: Generates new patients at the CCP at a defined rate.
- **Facility manager** Decides which patient in the queues of the MASs gets treated next.
- **Transport manager** Controls the transporter, decides where to drive and which patients to pick up and drop off.

The UML diagram in Figure 7 shows the associated Java class for each submodel together with each interface implemented by the model. Each of the three classes *TransportManager*, *PatientGenerator* and *FacilityManager* has different *handle*-methods, which are used by the model to notify the submodels about state changes. The methods of the interfaces are the actions that can be used by the submodels to control the simulated objects. To clarify how the components work together, we focus on describing the *TransportManager*. The other two submodels work accordingly.

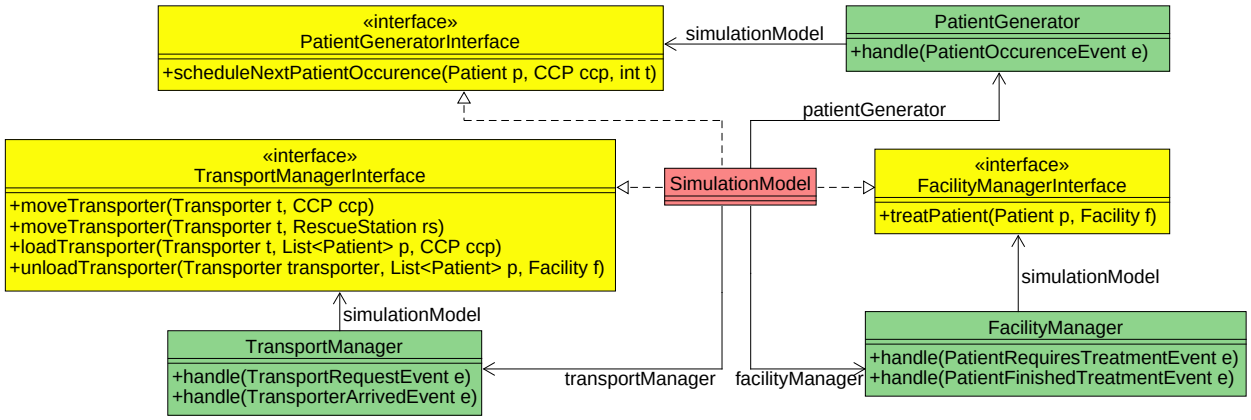


Figure 7: The simulation model (red) implements one interface (yellow) for each submodel (green) that provides the functionality which can be used for implementing various strategies.

The *SimulationModel*, which represents the model component, can communicate with the *TransportManager* via a reference. The notification about state changes is sent by using the *handle* method together with passing the appropriate event-object which contains all required information as attributes. Here, two events can occur:

- *TransportRequestEvent*: A patient needs to be transported from the CCP to one of the MASs.
- *TransporterArrivedEvent*: A transporter arrived at its destination.

The *TransportManager* can communicate with the *SimulationModel* via the *simulationModel* reference and the *TransportManagerInterface*. The methods provided by the interface are the actions the *TransportManager* can use:

- *moveTransporter*: Moves the transporter to the destination.
- *loadTransporter*: Loads the patients from a CCP to the transporter.
- *unloadTransporter*: Unloads patients from a transporter to a MAS.

Based on these events, an example of how the different components work together is visualized as a sequence diagram in Figure 8.

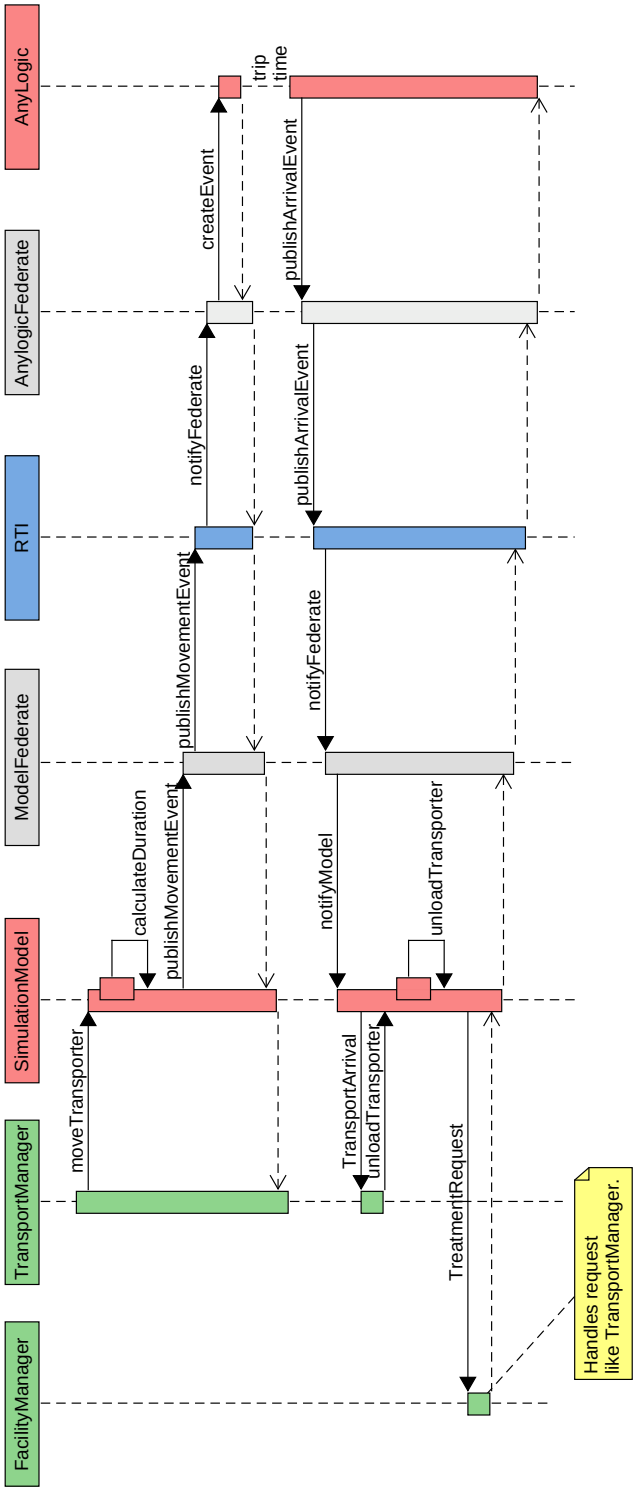
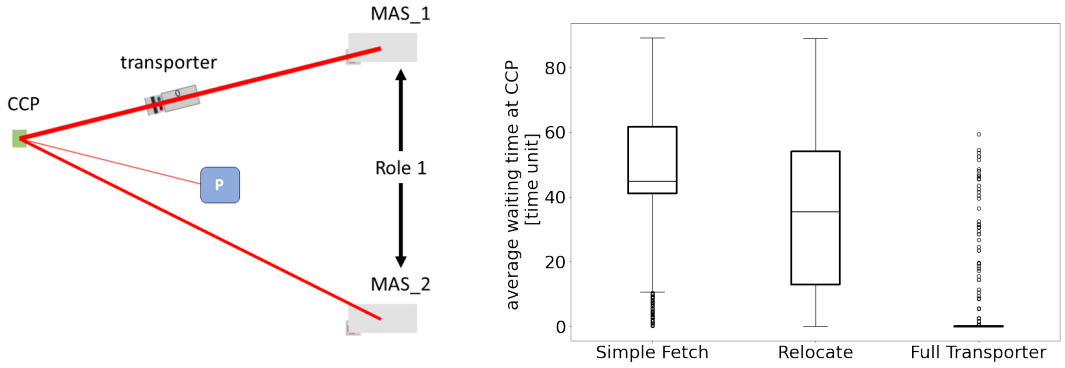


Figure 8: Sequence diagram of the *TransportManager* moving a loaded transporter to a MAS and unloading it.

Note that the method names used to label the transitions are adjusted for clarification purposes. In the presented use case, the transporter is assumed to just have picked up patients at the CCP. Next, the *TransportManager* tells the model to move the transporter via *moveTransporter*. As a result, the *SimulationModel* calculates the driving duration based on the average transporter speed and the distance to its destination. A *MovementEvent* containing the duration is created and sent to *AnyLogic* via the federates and the *RTI* where it is scheduled. Once triggered, an *ArrivalEvent* is created by *AnyLogic* and is sent back to the model which notifies the *TransportManager*. Next, the transporter gets unloaded via *unloadTransporter*. Note that, as described earlier, the *TransportManager* only triggers this action. The unloading process itself is done by the *SimulationModel*. For this purpose, the model checks if the respective patients are actually in the transporter and adds them to the queue of the facility accordingly. Next, the model notifies the *FacilityManager* about the patients' arrival using a *TreatmentRequest*.

4.4 Results

The approach proposed in the last section was implemented using Java 18 and AnyLogic 8. We tested the described scenario with different dispatching rules to verify that we gained the intended flexibility. The results from running the model are shown in Figure 9.



(a) Screenshot of the running simulation

(b) Waiting times of patients for different strategies

Figure 9: Results of testing three different dispatching strategies using the proposed prototype implementation.

A screenshot of the running model can be seen in Figure 9a. Here, the simulated objects from the scenario described in the previous section are visualized. The red lines show the routes taken by the transporter. Their width illustrates how often this route was used. Note that the middle route is used for the initial movement of the transporter to the CCP, starting from the blue square. For showing the model's flexibility regarding different dispatching policies, three different *TransportManager*s are created, which implement one of the following strategies each:

- **Simple Fetch**: Once a patient appears at the CCP, the transporter drives there and picks up the patient as well as the ones that appeared during the trip. Once loaded, the transporter drives to one of the MASs, depending on the length of their queue.
- **Relocate**: Same as *Simple Fetch* except that the transporter instantly drives back to the CCP once the patients are dropped at the MAS and waits for the next patient's arrival.
- **Full Transporter**: Same as *Relocate* except that the transporter waits at the CCP until its maximum transport capacity is reached.

To test these strategies, we consider a high-level scenario to help illustrate the effects. Accordingly, the results only serve for visualization purposes, not for medical insights. Here, the transporter has a capacity of five patients. The patients arrive at the CCP every 30 to 120 time units, equally distributed. Driving from the CCP to one of the MASs takes about 45 time units. For each strategy, the waiting times of the patients at the CCP are shown as boxplots in Figure 9b. The waiting time stops as soon as a patient is loaded into the transporter. We chose to investigate this time since it illustrates the results of applying different strategies clearly. Further, this time is essential since it is in the transporter that each patient is first seen by medical personnel and thus can be stabilized (Neitzel and Ladehof 2015). As shown, the patients have to wait the longest time if *Simple Fetch* is applied. Here, the transporter often leaves the CCP without being fully utilized. Patients appearing while the transporter is on its way to the MAS have to wait relatively long. Using *Relocate*, waiting times decreased by about 20%. This results from driving back to the CCP right after the patients were dropped off at the MAS, as the time it takes to drive from the MAS to the CCP once a patient appeared can be saved. In case of *Full Transporter*, the median waiting time is even significantly lower since the transporter waits at the CCP until its capacity is reached. As a result, many patients can be loaded into the transporter right after they appeared.

Again, these observations don not allow making recommendations about the dispatching in a real-world application. Also, the results don not yield any general insights regarding the different strategies since their efficiency highly depends on the setup of this scenario. However, they clearly shows that various dispatching rules with significantly different outcomes can be implemented using only three actions. At the same time, adjusting the strategy only meant changing a few lines of code and could therefore be implemented with little effort.

5 CONCLUSION AND OUTLOOK

We proposed a concept for modeling MCIs to overcome the lack of flexibility current models have. Our modular approach divides the simulation into different components communicating with each other via a publish/subscribe-concept. Further, we decoupled the simulation behavior from the model into different submodels. These are informed about state changes within the simulation and can control the simulated objects via predefined actions. We demonstrated the feasibility of this approach by implementing a prototype containing different dispatching strategies in an evacuation scenario.

The flexibility gained by our approach is twofold. First, each simulation component is exchangeable. Therefore, different visualizations, simulation kernels etc. can be used according to the user's needs without adapting the remaining components. Second, in contrast to offering parametrizable dispatching rules, our model enables us to flexibly define the simulated objects' behavior. This way, a wide range of decision policies can be implemented with low effort. Also, the simulation model verifies the feasibility of each object's action, supporting the process of implementing new strategies. Therefore, our approach offers the flexibility required for the simulation-based validation and optimization of medical plans in the future.

This paper showed the viability of our approach for a simple evacuation scenario. However, to support more realistic scenarios considering the whole medical evacuation chain, this approach should be developed further. First, a generic way of describing the required objects evacuation chain should be developed. This ensures the expandability of the model for different requirements in the future. Second, the set of provided interfaces as well as their supported actions and events must be considered in detail, since these are the key to the flexibility of our approach.

ACKNOWLEDGMENTS

This research is part of the project "LogSimSanDstBw – Simulationsbasierte Logistikanalysen" and is funded by dtec.bw - Digitalization and Technology Research Center of the Bundeswehr. dtec.bw is funded by the European Union - NextGenerationEU.

REFERENCES

Cooper, H., E. Schmitt, and T. Gibbons-Neff. 2023. "Soaring Death Toll Gives Grim Insight into Russian Tactics". *The New York Times*. https://www.nytimes.com/2023/02/02/us/politics/ukraine-russia-casualties.html, accessed 5th July 2023.

De Rouck, R., M. Debacker, I. Hubloue, S. Koghee, F. Van Utterbeeck, and E. Dhondt. 2018. "SIMEDIS 2.0: On the Road Toward a Comprehensive Mass Casualty Incident Medical Management Simulator". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and J. Björn, 2713–2724. Piscataway, New Jersey: Institute of Electrical and Electronics Engineering, Inc.

Debacker, M., I. Hubloue, E. Dhondt, G. Rockenschaub, A. Rüter, T. Codreanu, K. L. Koenig, C. Schultz, K. Peleg, P. Halpern, S. Stratton, F. Della Corte, H. Delooz, P. L. Ingrassia, D. Colombo, and M. Castrèn. 2012. "Utstein-Style Template for Uniform Data Reporting of Acute Medical Response in Disasters". *PLoS Currents* 4.

Debacker, M., F. Van Utterbeeck, C. Ullrich, E. Dhondt, and I. Hubloue. 2016. "SIMEDIS: A Discrete-Event Simulation Model for Testing Responses to Mass Casualty Incidents". *Journal of Medical Systems* 40(12).

Frial, V. B. 2022. "Evaluating the Military Medical Evacuation Dispatching and Delivery Problem via Simulation and Self-Exciting Hawkes Process". Master thesis, Department of Operational Sciences, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio. https://scholar.afit.edu/cgi/viewcontent.cgi?article=6342&context=etd, accessed 7th July 2023.

IEEE Computer Society 2010. "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Framework and Rules - Redline". *IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000) - Redline*:1–38.

Jenkins, P. R., M. J. Robbins, and B. J. Lunday. 2023. "Optimising Aerial Military Medical Evacuation Dispatching Decisions via Operations Research Techniques". *BMJ Military Health* 169(1):90–92.

Kleint, R., and A. Geck. 2021. "Simulation-Based Decision Support for the Logistic System of the German Armed Forces". https://www.sto.nato.int/publications/STO%20Meeting%20Proceedings/STO-MP-MSG-184/MP-MSG-184-13P.pdf, accessed 5th July 2023.

Kleint, R., and A. Geck. 2022. "Simulation-Based Decision Support for the Logistic System of the German Armed Forces". In *Towards Training and Decision Support for Complex Multi-Domain Operations*: NATO Science and Technology Organization.

Lechtenberg, S., A. Widera, and B. Hellingrath. 2017. "Research Directions on Decision Support in Disaster Relief Logistics". In *2017 4th International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, edited by M. Middelhoff and S. Yahiaoui, 1–8. Piscataway, New Jersey: Institute of Electrical and Electronics Engineering, Inc.

Meisner, K., T. Mayer, H. Stein, N. Leopold, T. Uhlig, and O. Rose. 2023. "Konzeptionierung eines Simulationsmodells der Rettungskette unter Gefechtsbedingungen". In *Simulation in Produktion und Logistik 2023*.

NATO 2019. "Allied Joint Doctrine for Medical Support". NATO Standardization Office.

Neitzel, C., and K. Ladehof. 2015. *Taktische Medizin*. 2nd ed. Berlin Heidelberg: Springer.

Possik, J., S. Gorecki, A. Asgary, A. O. Solis, G. Zacharewicz, M. Tofighi, M. A. Shafiee, A. A. Merchant, M. Aarabi, A. Guimaraes, and N. Nadri. 2021. "A Distributed Simulation Approach to Integrate AnyLogic and Unity for Virtual Reality Applications: Case of COVID-19 Modelling and Training in a Dialysis Unit". In *Proceedings of the 2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, edited by J. M. Cecilia and F. J. Martinez, 1–7. Piscataway, New Jersey: Institute of Electrical and Electronics Engineering, Inc.

Sell, L. 2019. "The NATO Future Medical Support Concept". Supreme Allied Commander, Transformation. https://www.innovationhub-act.org/sites/default/files/docs/FMSC%2020190814.pdf, accessed 5th July 2023.

Tippong, D., S. Petrovic, and V. Akbari. 2022. "A Review of Applications of Operational Research in Healthcare Coordination in Disaster Management". *European Journal of Operational Research* 301(1):1–17.

Wallemacq, P., and R. House. 2018. "Economic Losses, Poverty and Disasters 1998–2017". United Nations Office for Disaster Risk Reduction.

Yue, Y., L. Marla, and R. Krishnan. 2012. "An Efficient Simulation-Based Approach to Ambulance Fleet Allocation and Dynamic Redeployment". In *Proceedings of the AAAI Conference on Artificial Intelligence*, edited by J. Hoffmann and B. Selman, Volume 26, 398–405. Palo Alto, California: Association for the Advancement of Artificial Intelligence Press.

AUTHOR BIOGRAPHIES

KAI MEISNER is currently employed at the Bundeswehr Medical Academy. He is an external Ph.D. candidate at the University of the Bundeswehr Munich, where he also received his M.Sc. degree in Computer Science. His research focuses on the simulation based analysis and optimization of the medical evacuation chain. His e-mail address is kai.meisner@unibw.de.

HEIDEROSE STEIN is a research assistant in the multidisciplinary modeling and simulation team at the University of the Bundeswehr Munich. She holds a Master's Degree in Chemical Engineering of the Technical University of Munich. Her research interests include investigating how to use simulation to solve decision problems, techniques to efficiently search or reduce the solution space and how to speed up simulation runs or data farming experimentation. She is a member of ASIM. Her e-mail address is heiderose.stein@unibw.de.

NADIIA LEOPOLD is a doctoral student at the Department of Computer Science of the University of the Bundeswehr Munich. She received her degree in Computer Science from the National Aviation University, Kyiv. Her research interests include modeling and simulation, data analysis and machine learning. Her e-mail address is nadiia.leopold@unibw.de.

TOBIAS UHLIG is a postdoctoral researcher at the University of the Bundeswehr Munich. He holds a M.Sc. degree in Computer Science from Dresden University of Technology and a Ph.D. degree in Computer Science from the University of the Bundeswehr Munich. His research interests include operational modeling, natural computing, and heuristic optimization. He is a member of the ASIM and the IEEE RAS Technical Committee on Semiconductor Manufacturing Automation. He is one of the founding members of the ASIM SPL workgroup on the Investigation of Energy-related Influences in SPL. His e-mail address is tobias.uhlig@unibw.de.

OLIVER ROSE holds the Chair for Modeling and Simulation at the Department of Computer Science of the University of the Bundeswehr Munich. He received a M.Sc. degree in Applied Mathematics and a Ph.D. degree in Computer Science from Julius-Maximilians-University Würzburg. His research focuses on the operational modeling, analysis, and material flow control of complex manufacturing facilities and semiconductor factories. He is a member of INFORMS Simulation Society, ASIM, and GI. His email address is oliver.rose@unibw.de.