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

HYBRID MODELS WITH REAL-TIME DATA IN HEALTHCARE: A FOCUS ON DATA SYNCHRONIZATION AND EXPERIMENTATION

Navonil Mustafee
Alison Harper

Centre for Simulation, Analytics
and Modelling
University of Exeter Business School
Streatham Campus
Exeter, EX4 4ST, UK

Joe Viana

Department of Accounting
and Operations Management
BI Norwegian Business School
Nydalsveien 37
Oslo, 0484, NORWAY

ABSTRACT

Conventional simulation models used in Operations Research and Management Science (OR/MS) use historical data. With the increasing availability of real-time data, technologies commonly associated with applied computing, such as Data Acquisition Systems (DAS), may need to be integrated with conventional OR/MS models to develop Hybrid Models (HMs). We distinguish between HMs that use only real-time data – we refer to them as Digital Twins (DTs) – and those using a combination of historical and real-time data – called Real-time Simulation (RtS). Our previous contribution focused on the challenges of such integration, a concept referred to as information fusion, and presented a conceptualization of DT/RtS. This paper focuses on DT/RtS data synchronization and methods that could be employed from Parallel and Distributed Simulation (PADS). The conceptualizations and discussions reflect on the authors' experience implementing an RtS of a network of Emergency Departments and Urgent Care Centers in the UK.

1 INTRODUCTION

OR/MS models developed using simulation techniques such as agent-based and discrete-event simulation (DES) have conventionally used historical data and distributions to populate and drive the model. Indeed, most contemporary modeling and simulation (M&S) studies in OR/MS use this approach. Over the past decades, with the advent of *integrative* software (having a shared database) such as Enterprise Resource Planning (ERP) systems and technologies such as web services, Information Systems (IS) are being designed to make operational data readily available to external systems. Similarly, legacy systems are using defacto solutions to serve the same outcome. From our experience working with several publicly-funded National Health Service (NHS) Trusts operating in the South West of England, we note that most healthcare IS that capture patient flow data in urgent and emergency care settings are relatively older systems. We refer to them as legacy systems. Examples include *TrakCare* and *Adastra*. There are also examples of Trusts replacing legacy IS with systems such as *EPIC*.

Development of a computational OR/MS model requires data. For example, an emergency department (ED) model implemented in DES usually involves the modeler accessing data from a legacy patient flow system, then using a distribution fitting tool to identify probability distributions that best represents the data available, and, assigning the distributions to DES model elements such as model arrivals, service times etc. Observations, interviews, and the literature are additional sources of data.

New patient flow systems such as *EPIC* have an open architecture, making it possible to query data in real time (Warwick et al. 2023). Such systems enable real-time communication of events of interest, e.g., patient arrival, triage start and stop, allocation of major/minor cubicles to patients, as and when they occur in ED. Modelers can use real-time data to implement Digital Twins (DTs) that "mirror" the current state of urgent and emergency facilities. Further, using RFID data on the location of patients and mobile hospital equipment (Magliulo et al. 2012) could enhance the passive "monitoring" of the system of interest. However, as the current system evolves through wallclock time (real-time), and in cases where short-term forecasts signal the possibility of breaches in Key Performance Indicators (KPIs), there may exist a relatively small window of opportunity to affect a course correction (Harper and Mustafee 2019). This requires an additional computational element in the DT, wherein the latter triggers a faster than real-time experimentation using the computer model. We use the term **OR/MS DT** to refer to a system that includes both real-time monitoring of the physical system ("mirroring") *and* faster-than-real-time experimentation. The distinction between DT and OR/MS DT is important as this paper is for the M&S community, where the computational model is the focus of research and practice.

A modeler faces challenges in retrieving real-time data from legacy systems. For example, in our *NHSquicker* project (Mustafee and Powell 2021), the wait time data we receive is "*near*" real-time (rather than real-time). As the legacy systems use database triggers (PL/SQL) or application code, executed at pre-defined intervals (e.g., 5-15 minutes), to summarize a small sub-set of routinely collected operational data (e.g., maximum wait times, numbers of patients waiting) before sending it to our backend *NHSquicker* server. In this case, a computational model must include methods and approaches that fuse historical data with real-time data, the former used to plug information deficiency. This concept is called information fusion (Mustafee et al. 2023). In our work, we define a computational model that is data deficient as a **Real-time Simulation (RtS)**. Here, only a subset of the data required to initialize and drive the model is available real-time. Where a higher volume of real-time data is available, it is defined as a DT.

We have described the terminologies (computational model, RtS, OR/MS DT) and their contextualization with systems we have encountered in our ED overcrowding research. A shift from conventional model development to an OR/MS DT requires a modeling approach between the two extremes, called *Real-time Simulation* or *RtS*. RtS is pragmatic in recognizing that the modeler may not have access to updated real-time information on every data structure and variable implemented in a model, for example where multiple data source types are required (e.g., queues, servers, resources, entities). In contrast, OR/MS DTs operate on the implicit assumption of data sufficiency/ abundance. In our view, the shift from conventional modeling to OR/MS DT is hastened by recognizing the challenges of real-time data access and the limitations of automated data acquisition systems. This would acknowledge the need for research in RtS with a focus on deficient but readily-available real-time data. RtS may pave the way for OR/MS DT implementations when data availability is not a concern, representing a shift from the conventional to the OR/MS DT paradigm, achieved through the intermediary RtS element.

1.1 Contribution to the Hybrid Modeling Literature

Hybrid modeling in M&S, which Tolk et al. (2021) define as the combined application of simulation with knowledge artifacts from a broader array of disciplines, can leverage cross-disciplinary methods and techniques and apply them to various stages of an M&S study, such as conceptual modeling, input/output data analysis, verification and validation, and experimentation (Powell and Mustafee 2017). Mustafee et al. (2020) present examples of hybrid models that have combined simulation models with approaches from fields such as Soft and Hard OR, Computer Science, and Applied Computing.

OR/MS DT and RtS are hybrid models that use real-time data streams. Approaches from Applied Computing, such as integration with Data Acquisition Systems (DAS), distributed databases and real-time connectivity with front-end computational models, are enablers of the shift from conventional modeling to OR/MS DTs. Our previous contribution on hybrid models with real-time data focused on the Applied Computing dimension (Mustafee et al. 2023) and information fusion to initialize and execute RtS models (Figure 1).

This paper focuses on another challenge of real-time M&S: data synchronization in OR/MS DT and RtS during experimentation. In Computer Science, the field of Parallel and Distributed Simulation, or

PADS (Fujimoto 2001) has investigated novel algorithms (e.g., conservation and optimistic time management protocols), PADS standards (e.g., IEEE 1516 HLA) and middleware implementations (e.g., DMSO RTI 1.4NG) to speed up the execution of simulation and foster model reuse through use of interoperability standards. Concerning speed-up, PADS focuses on the distribution of the execution of a single run of a simulation over multiple processors (parallel simulation) or a network of computers (distributed simulation). The first author of this paper has previous work on PADS approaches to supply chain simulations (Mustafee et al. 2009; Mustafee et al. 2021); this motivated the exploration of the use of these techniques in our current work. The paper thus contributes to the hybrid modeling literature by anchoring Applied Computing and Computer Science methods with traditional DES modeling (Figure 1).

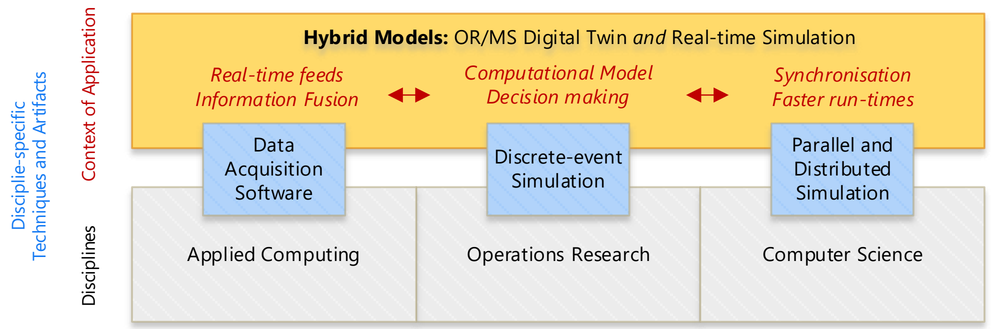
Disciplie-specific Techniques and Artifacts
Context of Application
Disciplines
Hybrid Models: OR/MS Digital Twin and Real-time Simulation
Real-time feeds Information Fusion
Computational Model Decision making
Synchronization Faster run-times
Data Acquisition Software
Discrete-event Simulation
Parallel and Distributed Simulation
Applied Computing
Operations Research
Computer Science

Figure 1: Hybrid OR/MS DT and RtS models use approaches from Applied Computing and Computer Science. DAS, DES and PADS are examples, and the mix of techniques and artifacts will be study specific.

The techniques and artifacts indicated in Figure 1 (DAS, DES, PADS) are specific to the authors' current work on implementing an RtS of a UK network of Emergency Departments and Urgent Care Centers; the current work extends our *NHSquicker* project that provides near real-time data on wait times from a network of EDs and UCCs in the South West of England (Mustafee and Powell 2021). Similarly, for other studies, the mix of hybrid techniques will depend upon the study objectives. Thus, authors may implement real-time modeling using OR techniques such as agent-based simulation or machine learning (we used DES); they may acquire real-time data from a network of sensors (our DAS were Healthcare Information Systems); they may consider new computer architectures such as GPGPU for faster execution (in the Computer Science intersect, our focus was on data synchronization using PADS).

The remainder of the paper is organized as follows. Section 2 reviews the literature and categorizes real-time models into DT, OR/MS DT and RtS. Section 3 presents the conceptualization of OR/MS DT and RtS, distinguishing them from the conventional OR/MS models. Section 4 discusses data synchronization in ORMS DT/RtS using PADS terminology. Section 5 concludes the paper and discusses future work.

2 LITERATURE REVIEW

Several applications of RtS in healthcare, mostly focused on EDs, have been proposed or conceptualized, and the progression of research has been toward proposing multiple DAS; platforms for integrating multiple data sources and types; and real-time system monitoring with prescriptive capability. This progress reflects manufacturing applications where sensors and well-demarcated processes simplify the development of digital replicas. RtS has been in use in manufacturing systems for decades (Liu et al. 2021), with Annan and Banks (1992) describing one of the earliest unifying frameworks for connecting the real-world system and the control system. Terms such as 'online simulation', 'data-driven simulation', 'digital twin', and 'symbiotic simulation' have all been used to describe a system of communication between a simulation model and a corresponding real-world system (e.g., Onggo et al. 2018; Onggo et al. 2021).

The concept of DT has seen a rapid increase in research interest in healthcare. A search on Scopus (*TITLE-ABS-KEY* (digital) *AND TITLE-ABS-KEY* (twin) *AND TITLE-ABS-KEY* (healthcare)) (all years, 27-02-2023) revealed an exponential rise from 4 publications in 2018 to 138 publications in 2022; of a total of 266 documents, 23 were review papers. Most studies proposed clinical and medical device applications, rather than operational support for service delivery (Sun et al. 2022; Armeni et al. 2022; Sun et al. 2023), and many focused on technical and methodological approaches, data collection, and integration challenges. In their review, VanDerHorn and Mahadevan (2021) noted that the literature has few examples of practical implementation of DTs that consider the implementation challenges and strategies. With increasing numbers of studies focusing on complex operational data collection, integration platforms, and measuring and monitoring algorithms, the gap between research and applied DTs in healthcare appears to be widening.

There is broad agreement that RtS and DTs involve both physical and virtual entities, and physical-to-virtual and virtual-to-physical twining processes. These elements can be characterized as a closed-loop system where information is exchanged and updated between the virtual and physical environments. This means that the physical-to-virtual connection closes the loop between virtual environment generated hypotheses (via RtS/DT core model simulation experiments) and the consequences realized in the physical environment following actions taken in response to results reported by the core model.

2.1 Three Forms of Real-time Models: DT, OR/MS DT, and RtS

DT: In its simplest form, a DT could be a link with a real system, with the physical-to-virtual link providing data to the virtual system via the updates in the physical system. This behavior is illustrated in Figure 2, where the real system evolves in wallclock time, and the digital entities representing the physical system are mirrored by the DT, which also executes in wall clock time. Several examples of city DTs primarily focus on city situational awareness by collecting, monitoring, and managing city data (Shahat et al. 2021).

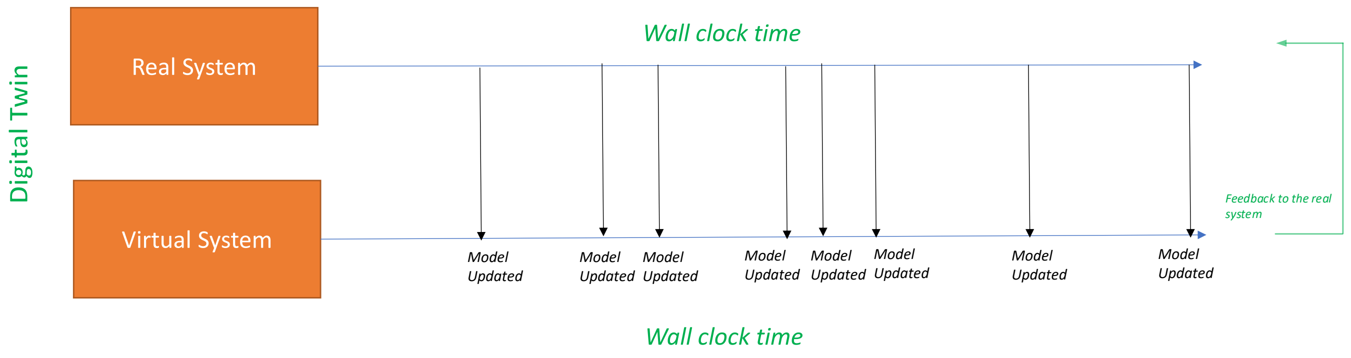

Figure 2: A DT with situational awareness and monitoring function will receive data from the physical system (black arrows) and update its virtual representation (green arrow represents feedback).

OR/MS DT: It includes a computational model to extend the traditional DT functionality, described above. It enables faster-than-real-time experimentation, providing feedback to the real system ('closing-the-loop'). An OR/MS DT is not information deficient. All data needed to initialize and execute the model is fulfilled through real-time feeds. In Figure 3, the DT model of Figure 2 is extended with the additional component of a computer model for experimentation (represented with the letter *'E'*; the red line which extends from *'E'* illustrates experimentation runtime). Note that the experiments only start if an information update from the physical system indicates a breach in KPI thresholds (e.g., the second information update in Figure 3). In cases where the physical system runs on accepted parameters (e.g., in Figure 3, see the first information update point), only the virtual system is updated.

RtS: It is a core computational model that receives real-time feeds from the physical system. Unlike DTs, RtS does not include situational awareness and monitoring functions. It is also data-deficient in terms of the real-time data that it receives from the physical system. Thus, RtS model initialization and experimentation adopts an *"information fusion"* approach, which combines historical data (in Figure 4, this is shown as a green line between the real and the physical system) with real-time data (black arrow).

Information fusion is represented as an ellipse. Like OR/MS DT, the execution of the RtS experiments is triggered by changes in the real-world system considered important to warrant a response. The model is first initialized through the fusion technique and then executed, with the results communicated back to the real system.

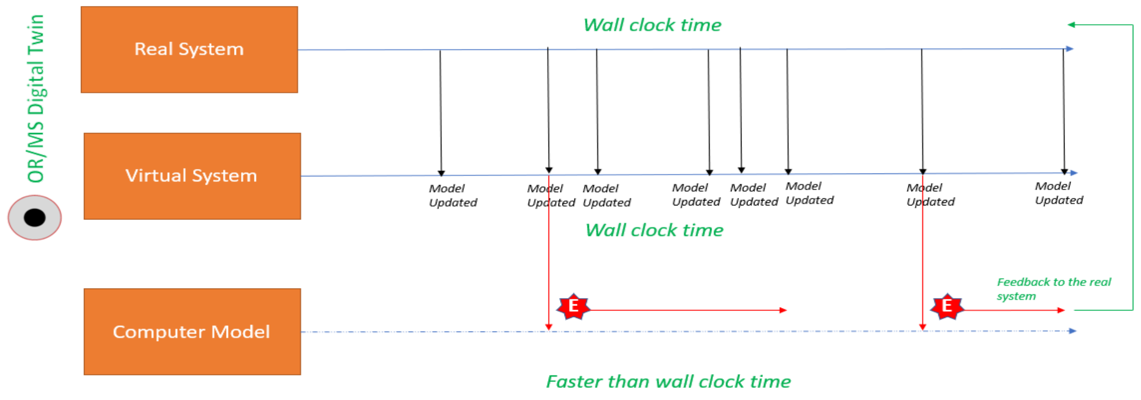

Figure 3: An OR/MS DT that updates the virtual representation of the real system as per wall clock time and executes experiments faster than wall clock time. The arrows represent real-time data; E signifies the start of experimentation.

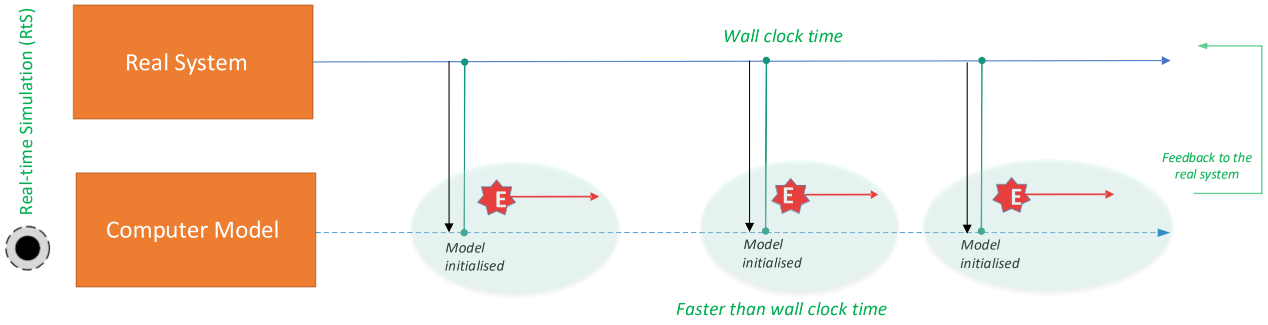

Figure 4: A RtS implements "information fusion" (illustrated as green ellipses) where real-time data is fused with historical data to compensate for the deficiency of the real-time feeds. The objective of RtS is to execute experiments and offer feasible solutions aimed at the physical system in the short term. In manufacturing, automation may implement the feedback (e.g., actuators); in healthcare, experimental results are communicated to the stakeholders for further action.

3 FRAMEWORK

Several efforts have been made to characterize DTs, and to distinguish them from other M&S approaches (e.g., Hassani et al. 2022; VanDerHorn and Mahadevan 2021; Jones et al. 2020; Barricelli et al. 2019). Our previous work presented the conceptualization of conventional OR/MS models, OR/MS DTs and RtS using the dimensions of modeling objective, data requirements, model implementation, and experimentation (Mustafee et al. 2023). In Figure 5, the black circle (bottom left) represents a conventional computational model reliant on historical data. The OR/MS DT (top right) is represented as a concentric circle with a red border; the grey area represents real-time data, the black circle is the core computational element. The RtS (middle) is illustratred as a concentric circle with a dashed border conveying that only a subset of the data required to initialize and drive the model is available real-time; again the black circle is the core computational element. The larger width of the grey area in the OR/MS DT represents the higher volume of real-time data available to it.

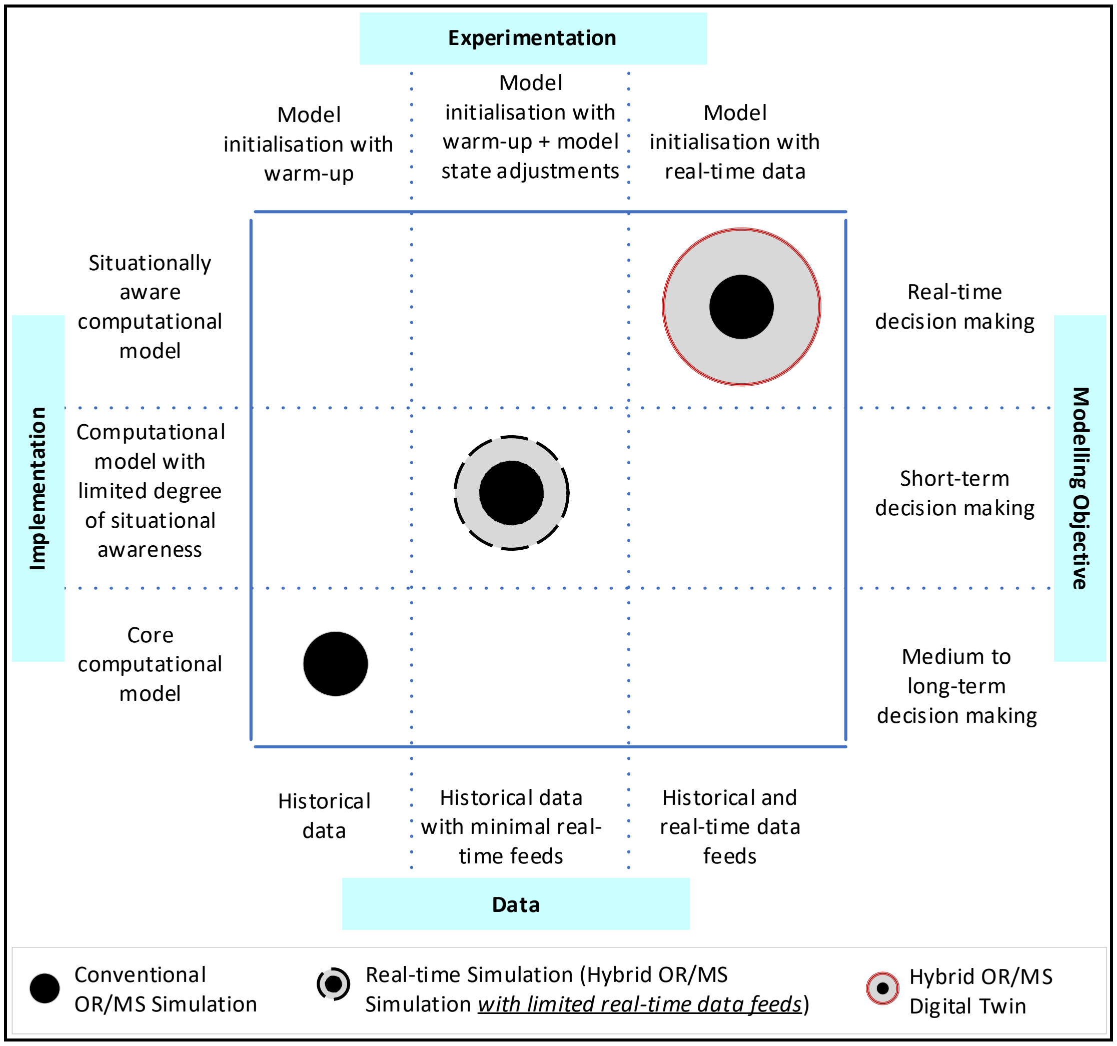
Experimentation
Model initialisation with warm-up
Model initialisation with warm-up + model state adjustments
Model initialisation with real-time data
Implementation
Situationally aware computational model
Computational model with limited degree of situational awareness
Core computational model
Modelling Objective
Real-time decision making
Short-term decision making
Medium to long-term decision making
Historical data
Historical data with minimal real-time feeds
Historical and real-time data feeds
Data
Conventional OR/MS Simulation
Real-time Simulation (Hybrid OR/MS Simulation *with limited real-time data feeds*)
Hybrid OR/MS Digital Twin

Figure 5: Conceptualisation of RtS and DT using the dimensions of modelling objective, data requirements, model implementation and experimentation (Mustafee et al. 2023).

According to this representation (Figure 5), the key difference between the three approaches can be summarized below:

Modeling Objective: DTs are primarily used for monitoring and prediction, but they can additionally use a computational model (OR/MS DTs), and the objective is to support real-time decision-making (e.g., re-location of ambulances based on their current location and availability). Conventional simulation models are primarily used for medium- to long-term decision-making (e.g., increasing bed capacity).

Data requirement: At its core, a RtS is a computational model with some of its historical distributions complemented with real-time data feeds. Therefore, it depends on historical data as only minimal real time variables are available, unlike an OR/MS DT, which may use multiple integrated data sources with high DAS integration and real-time data availability. A conventional simulation model uses only historical data for initialization and execution. RtS enables short-term decision-making, where a window of opportunity exists to tweak the physical system, potentially improving system performance, e.g., in response to an expected rise in ED wait times, increase staffing of ED by transferring clinicians from other departments.

Implementation: OR/MS DTs are situationally aware computational models that fulfil the objectives of the virtual representation of the physical elements of a system as it evolves through real-time and the faster-than-real-time simulation experimentation at specific pre-defined trigger points (see Figure 3).

Compared to OR/MS DTs, we argue that the primary objective of RtS is faster-than-real-time experimentation (Figure 4), with a limited degree of situational awareness (SA) demonstrated through these models; the degree of SA is dependent on the volume of real-time data received, viz-a-viz, the data required to initialize and drive the RtS.

Experimentation: Figure 5 illustrates that an OR/MS DT will be initialized using predominantly real-time data feeds, diminishing the need for warm-up time and model state adjustments. RtS, which is deficient in real-time data, will need to implement novel initialization strategies, which take into consideration the model state at the end of the warm-up time, and compares with real-time values received for a sub-set of variables (Mustafee et al. 2023).

4 SYNCHRONISATION OF REAL-TIME COMPUTATIONAL MODELS

The experimentation stage of the core OR/MS DT and RtS computational model is discussed further in this section. We focus on the issue of synchronization of the internal state of the model during its execution. We identify two cases: **(A)** The DT/RtS receives no new data during the execution of the computational model (i.e., the execution of the model is complete before the next tranche of data is received), and **(B)** The DT/RtS receives new data from the physical system before the end of the execution of the computational model. To further elaborate on the problem context, for (A), we use the example of an **RtS with fixed-time updates**, i.e., updates from the real system are sent at a pre-defined interval, e.g., 5 minutes; for (B), the example is that of an **RtS with variable time updates**, i.e., real-time data is sent to the RtS as and when the physical system changes. Although the examples and figures in this section only refer to RtS, as the synchronization aspect is related to the core computation model – a part of both OR/MS DT and RtS, most of the discussions are applicable to OR/MS DTs.

Case (A): RtS with fixed-time updates: We make the following assumptions (Figure 6).

(a) Data is received at constant intervals (one minute); thus, increments to wallclock time (WC_t) are equally spaced with WC_t, WC_{t+1} and WC_{t+2} representing the current time, one minute past the current time, and two minutes past the time, respectively.

(b) The simulation starts upon receipt of data at wallclock time *t* (WC_t). At this time, through information fusion, the computational model is initialized, and the simulation start time (Sim_t) will be the same as the current wall clock time, i.e., $Sim_t = WC_t$. The same approach is adopted for future experiments. For example, in Figure 6, we see that when a new tranche of data is received at WC_{t+1}, the simulation start time takes the value of the current wallclock time ($Sim_t = WC_{t+1}$).

(c) In this example, we assume that the computational model will execute for 100 units of simulated time Sim_{t+100} and which represents the future system state at WC_{t+100}. Simulation is faster than real-time; thus, although Sim_{t+100} represents the future state of the system at WC_{t+100}, in our example, its execution is complete even before the next real-time feed comes in at $WC_{t+1;}$ as per (a), we can thus conclude that the first experiments executed in less than a minute of wallclock time.

(d) In Figure 6, the second experiment starts at WC_{t+1}. Unlike the previous WC_t experiment, it concludes after it receives the new tranche of data at WC_{t+2}. Thus, as per (a), the second experiment took over one minute of wallclock time. Note: as this is the second experiment, the model was initialized with the time of the second update, i.e., Sim_t=$WC_{t+1.}$ Thus, there will be a difference of one unit of time between the simulation time (*Sim*) and the future state of the system that it represents (*WC)* – see (e).

(e) Figure 6 highlights a possible error (the purple circle) when we compare the real-world data received at WC_{t+2} with the internal state of the model at time Sim_{t+1} (which represents the simulated state of the real system at WC_{t+2}; note that Sim_{t+1} is not shown in the figure, but it could be visualized as incremental simulated time units on the red line representing the experimentation element). Comparing with the previous state may necessitate frequently saving the model's state (model dumps). The simulated variables may now differ from the values received from the real-time feed.

At the point the new data is received (*WC_{t+2}*), the experiment would likely be further in simulated time; for example, it may be at *Sim_{t+80}* which represents the simulated state of the system at *WC_{t+81}*.

We refer to the abovementioned error as a ***causality error*** **(CE) in real-time simulation**, a term borrowed from the PADS literature. The error exists due to data synchronization issues between real-time data and the internal state of the computation time, and since they execute at different speeds, viz., wallclock time versus simulated time. The first experiment concluded before new data was received, so synchronization was not an issue. In the second case, a decision may need to be taken whether to complete the execution of the current experiment (results may be erroneous), interrupt the current run and execute a new experiment at *WC_{t+1}*, or a third option could be to consider a conservative time management algorithm from the PADS literature. The definition of CE in PADS and an overview of the algorithms will be presented after the discussion on RtS with variable-time updates.

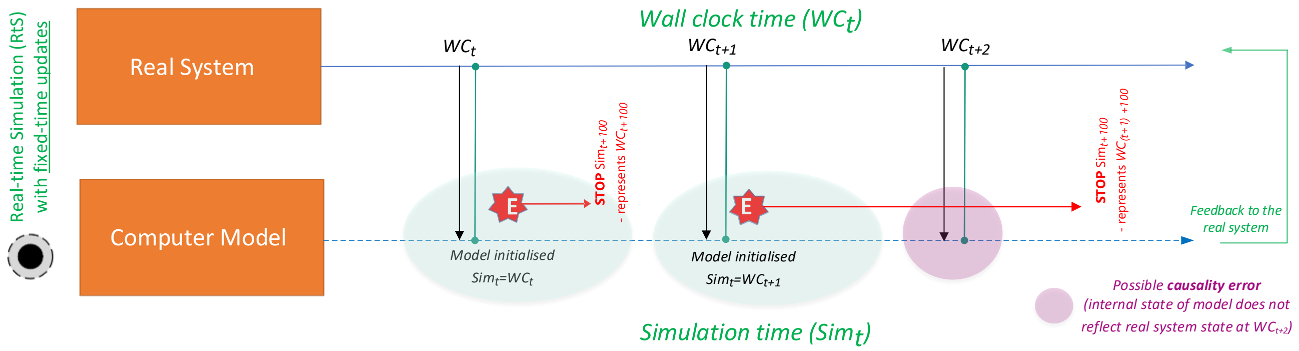
Real-time Simulation (RtS) with fixed-time updates
Wall clock time (WC_t)
Real System
Computer Model
WC_t
WC_{t+1}
WC_{t+2}
E
E
STOP Sim_{t+100} - represents WC_{t+100}
STOP Sim_{t+100} - represents $WC_{(t+1)+100}$
Model initialised Sim_t=WC_t
Model initialised Sim_t=WC_{t+1}
Feedback to the real system
Possible causality error (internal state of model does not reflect real system state at WC_{t+2})
Simulation time (Sim_t)

Figure 6: RtS with fixed-time updates. In the first experiment, the model completes execution before the next update; the second experiment spans across data updates and may introduce causality errors in RtS.

Case (B): RtS with variable-time updates:

(a) Data is received at variable intervals. This is illustrated in Figure 7, where the physical system sends data at *WC_t*, *WC_{t+2}*, *WC_{t+3}* and *WC_{t+6}*. Thus, increments to *WC_t* are not equally spaced.

(b) The simulation starts upon receipt of data at *WC_t*. At this time, through information fusion, the computational model is initialized, and the *$Sim_t = WC_t$*.

(c) Like case (A), we assume that the computational model will execute for 100 units of simulated time *Sim_{t+100}* which represents the future system state at *WC_{t+100}*. In our example, the model executes until after *WC_{t+6}*. From the start of the experiment to its completion, in addition to (b) – data for model initialization, it receives three further tranches of data at *WC_{t+2}*, *WC_{t+3}* and *WC_{t+6}*.

(d) Figure 7 highlights possible errors (illustrated as purple circles). These errors arise if the real-world data received at the three points are different from the computational model's internal state at *Sim_{t+2}*, *Sim_{t+3}* and *Sim_{t+6}*, which represent the simulated state of the real system at *WC_{t+2}*, *WC_{t+3}* and *WC_{t+6}*. Comparing the model's previous internal state with the new values may necessitate frequently saving the model's state as timestamped instances of model execution. At the point the new data is received (say, *WC_{t+2}*), the experiment would likely be further in simulated time; for example, it may be at *Sim_{t+20}* which represents the simulated state of the system at *WC_{t+20}*.

Thus, like case (A), RtS with variable-time updates may be challenged by the synchronization issue of real-time data with the internal model state and may lead to ***causality error*** **(CE) in real-time simulation.** A decision may be taken during the RtS design phase that since the updates from the real system cannot be predicted (unlike the fixed-time updates), it may be necessary to allow the experiments to complete execution (rather than an interrupt and restart the experiment strategy). Thus, some tolerance may need to be inbuilt to accommodate the occurrence of CEs in RtS. Another PADS approach could be considered, where the RtS may "rollback" computation to a previously valid state, rather than restarting the computations from the start. Rollback will necessitate frequently saving the model's state. For example, during execution, data received at WC_{t+2}, when compared to the model's prior internal state at Sim_{t+2}, may be considered valid and the simulation is allowed to progress; however, at WC_{t+3}, as new data is received and compared with Sim_{t+3}, it may indicate the need for the reversal of the computation. Since the model's internal state at Sim_{t+2} was considered valid, the simulation could be rolled back to Sim_{t+2} and continued from this point. This could lead to faster runtimes, depending on the number of rollbacks necessary.

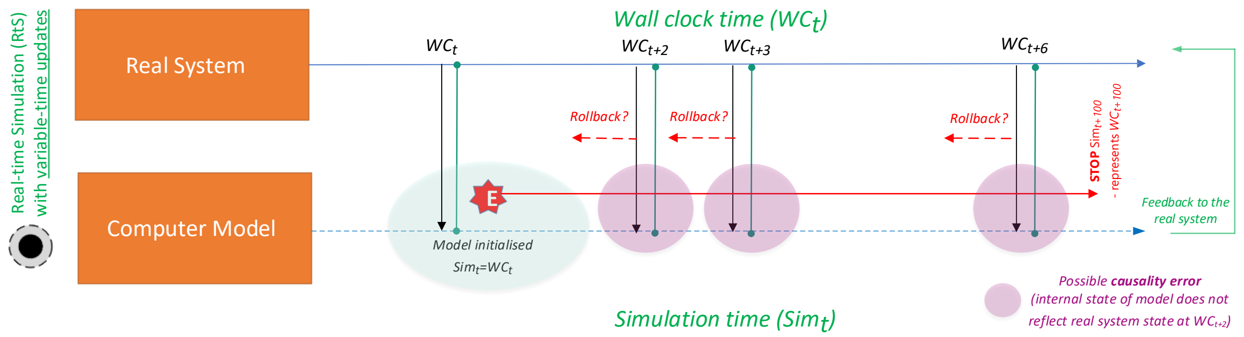
Real-time Simulation (RtS) with variable-time updates
Real System
Computer Model
Wall clock time (WC_t)
WC_t
WC_{t+2}
WC_{t+3}
WC_{t+6}
Rollback?
Rollback?
Rollback?
E
Model initialised $Sim_t = WC_t$
STOP Sim_{t+100} - represents WC_{t+100}
Feedback to the real system
Simulation time (Sim_t)
Possible **causality error** (internal state of model does not reflect real system state at WC_{t+2})

Figure 7: An RtS with variable-time updates can receive real-time data at any point in time. A PADS rollback strategy will reverse computation to a previously valid state and resume simulation from this point.

4.1 Parallel and Distributed Simulation (PADS): Causality Errors and Synchronization Protocols

PADS mainly focuses on the distribution of the execution of a single run of a simulation over multiple processors (parallel simulation) or a network of computers (distributed simulation). In case of RtS, and the example presented in this paper (which anchors to our work on developing a RtS in the healthcare domain), the computational model only runs on one computer. However, the DAS, which sends fixed-time/ variable-time updates as timestamped messages from the physical system to the RtS, runs on a different computer. In the PADS literature, distributed simulation can be defined as the distribution of the execution of a single run of a simulation program across multiple processors (Fujimoto 2000). Thus the execution of RtS should arguably consider concepts from distributed simulation since they both work as one unifying system (namely, integrated DAS-RtS execution and distributed simulation federation), with the sub-systems executed over different computers, and coordination between the subsystems achieved through the exchange of timestamped messages between the subsystems. To achieve this, PADS software (distributed simulation *middleware*) is often used. It implements well-known distributed simulation time management algorithms to achieve synchronization between individual running simulations (Fujimoto 1990). The focus of this paper is not on the software, rather the concept of data synchronization between real-time feeds and the RtS computational model during the experimentation stage.

In distributed simulation, time management algorithms are required for the prevention of causality errors. Causality errors happen because of a failure to process simulation events in increasing timestamp order. More specifically, a causality error occurs when a simulation has processed an event with timestamp *T1* and subsequently receives another event with timestamp *T2*, wherein *T1* > *T2*. Since the execution of the event with time stamp *T1* will have normally changed the state variables that will be used by the event with timestamp *T2*, this would amount to simulating a system in which the future could affect the past (Fujimoto 1990). For a serial simulator that has only one event list and one logical clock it is fairly easy to avoid causality errors. In the case of distributed simulation, the avoidance of causality is a lot more difficult because it must deal with multiple event lists and multiple logical clocks that are assigned to various processors. The reason for this is explained next with reference to Mustafee et al. (2014).

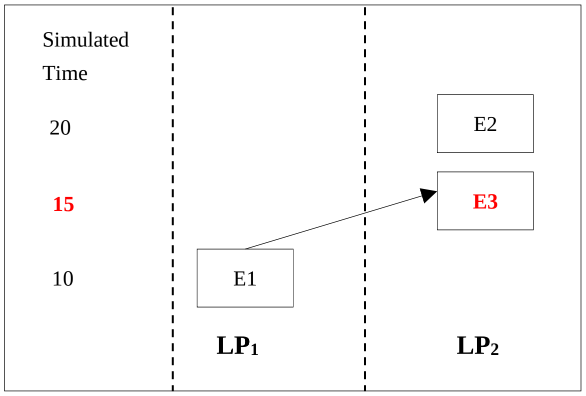

Figure 8: Execution of events in a distributed simulation (adapted from Fujimoto, 1990). The new event (E3) and the time of the new scheduled event (15) are indicated in red.

In Figure 8 above, the simulation represents a physical system that has two physical processes, say, PP_1 and PP_2. Logical simulation processes LP_1 and LP_2 model the two physical processes. Each of these logical processes have their own simulation engine, simulation clock and event list. During simulation initialization the event lists of both LP_1 and LP_2 are populated with the events *E1* and *E2* respectively. The timestamps for *E1* and *E2* are 10 and 20 respectively. It will be possible for LP_1 to process event *E1* without any causality error since the timestamp of *E1* < timestamp of *E2*. But LP_2 (running in parallel to LP_1) will not be able to execute event E2 at time 20 since the execution of *E1* schedules another event *E3* for LP_2 at time 15. In such a case, if LP_2 had been allowed to execute *E2* at simulated time 20 then it would have resulted in a causality error because the time stamp of *E3* < the time stamp of *E2*.

In our example, we can consider LP_1 as DAS and LP_2 as a RtS computational model with DAS sending timestamped data updates to RtS during the execution of experiment (Figure 6; Figure 7). Compared to the example presented in Figure 8, and where LP_1 and LP_2 were computer simulations, the key difference with DAS-RtS system is that, while DAS sends messages at wallclock time, the RtS computational model executes faster-than-real-time. Irrespective of this difference, it is a worthwhile exercise to consider the synchronization protocols for distributed simulation that prevent or correct such causality errors. In PADS, the conservative and optimistic algorithms that achieve synchronization between the individual running simulations are implemented using distributed simulation middleware software. Examples of such middleware include Distributed Interactive Simulation (Miller and Thorpe 1995), FAMAS (Boer 2005), and IEEE 1516 High Level Architecture – Run Time Infrastructure (HLA-RTI) (IEEE 2000). In the remainder of this section, we use HLA as an example to discuss the time synchronization protocols. In HLA terminology, a distributed simulation is called a federation, with each constituent simulation referred to as a federate.

In *conservative time synchronization*, the federate only advances simulation time when HLA-RTI guarantees that it will not receive any external events in its past. *Next Event Request (NER)* and *Time Advance Request (TAR)* are the two most commonly used variants of this mechanism that can be invoked by a federate to request time advances from RTI. RtS with fixed-time updates (Figure 6) could benefit from research into using a conservative approach to synchronizing the core model with DAS data updates.

In *optimistic time synchronization*, the federate advances time without any constraint but must be prepared to rollback to a previous state when it receives an event in its past. An optimistic synchronization protocol like Virtual Time, and its implementation, called the Time Warp mechanism, executes events without considering the event time ordering (Jefferson 1985). It has to save its state frequently so that a rollback to a previous state can occur when an event with a time stamp less than the current simulation time is received. Drawing parallels with the challenges to RtS experimentation, computation rollback applies to RtS with variable-time updates (Figure 7). Implementing this strategy would require comparing the RtS model's present internal state with previous state values. This may necessitate frequently saving the core computational model's state as timestamped dumps of RtS state. A direction of future work is to implement algorithms that implement optimistic behavior in RtS, whereby (1) the core computational model is permitted to execute without any constraint but saving the internal RtS state from time to time, (2) any new updates from DAS are communicated to the RtS core model, (3) an algorithm decides whether a rollback is necessary, and if so, the rollback point, and (4) based on the rollback point, initialize the internal state of the RtS core model with the instance of the saved system dump that is associated with the rollback point.

5 CONCLUSION

Our previous contribution to RtS in healthcare focused on increasing situational awareness (Harper et al. 2022), participatory design research for RtS development (Harper and Mustafee 2023), and methodological and technical insights into RtS implementation (Mustafee et al. 2023). One important challenge identified by Mustafee et al. (2023) was the concept of information fusion, i.e., combining real-time data with values generated from distributions to represent, at the start of the experimentation (and after model warmup), the best possible approximation of the physical system at the current wallclock time. This necessitated new approaches for model initialization and execution of RtS. In this paper we focus on data synchronization between the real-system and the DT/RtS computational model during the experimentation phase. While presented in the context of healthcare RtS, the approach is more widely applicable.

Real-time data is received as timestamped messages from Data Acquisition Systems (DAS). Due to interruptions in the communication channel, network latency, server outages etc., the communication between DAS and DT/RtS may be interrupted. This poses a problem for a DT that “mirrors” the current state of an operational system such as a factory or a hospital. Should the DT pause itself at the timestep of the last acknowledged message from the DAS? Or do algorithms that progress the DT based on best estimates need to be implemented? The synchronization problem becomes a more significant issue when the computational model executes faster-than-real-time experiments. Unlike the mirroring functionality, which, per our definition, applies only to DT, both DT and RtS include a core computational model. During model execution, if new data is received from DAS this may lead to a causality error in real-time simulation; how do DT/RtS respond? In this paper we discuss concepts from Parallel and Distributed Simulation (PADS) time management, causality error and synchronization protocols for distributed simulation that prevent or correct such causality errors.

REFERENCES

Annan, S. and J. Banks. 1992. “Design of a Knowledge-Based On-line Simulation System to Control a Manufacturing Shop Floor”. *IIE transactions*, 24(3), 72-83.

Armeni, P., I. Polat, L. M. De Rossi, L. Diaferia, S. Meregalli, and A. Gatti. 2022. “Digital Twins in Healthcare: Is It the Beginning of a New Era of Evidence-Based Medicine? A Critical Review”. *Journal of Personalized Medicine*, 12(8), 1255

Barricelli, B. R., E. Casiraghi, and D. Fogli. 2019. “A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications”. *IEEE Access*, 7, 167653-167671.

Boer, C. A. 2005. *Distributed Simulation in Industry.* Erasmus Research Institute of Management. ERIM PhD Series Research in Management. Erasmus University, Rotterdam.

Fujimoto, R.M. 1990. "Parallel Discrete Event Simulation." *Communications of the ACM*, 33(10):30-53.

Fujimoto, R.M. 2000. *Parallel and Distributed Simulation Systems*. New York, NY: John Wiley & Sons.

Fujimoto, R. M. 2001. “Parallel and Distributed Simulation Systems”. In *Proceeding of the 2001 Winter Simulation Conference*, edited by B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 147-157. Piscataway, New Jersey: IEEE.

Harper, A. and N. Mustafee. 2019. "A hybrid modelling approach using forecasting and real-time simulation to prevent emergency department overcrowding". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 1208-1219. Piscataway, New Jersey: IEEE.

Harper, A. and N. Mustafee. 2023. "Participatory Design Research for the Development of Real-time Simulation Models in Healthcare". *Health Systems*, 1-12. https://doi.org/10.1080/20476965.2023.2175730.

Harper, A., N. Mustafee, and M. Pitt. 2022. “Increasing Situation Awareness in Healthcare through Real-time Simulation”. *Journal of the Operational Research Society*, 1-11. https://doi.org/10.1080/01605682.2022.2147030.

Hassani, H., X. Huang, and S. MacFeely. 2022. “Impactful Digital Twin in the Healthcare Revolution”. *Big Data and Cognitive Computing*, 6(3), 83.

IEEE. 2000. *"IEEE Standard for Modelling and Simulation (M&S) High Level Architecture (HLA)*." New York: IEEE.

Jefferson, D. R. 1985. "Virtual Time." *ACM Transactions on Programming Languages and Systems*, 7(3): 404 – 425.

Jones, D., C. Snider, A. Nassehi, J. Yon, and B. Hicks. 2020. "Characterising the Digital Twin: A Systematic Literature Review". *CIRP Journal of Manufacturing Science and Technology*, 29, 36-52.

Liu, M., S. Fang, H. Dong, and C. Xu. 2021. "Review of Digital Twin About Concepts, Technologies, and Industrial applications". *Journal of Manufacturing Systems*, 58, 346-361.

Magliulo, M., L. Cella, and R. Pacelli. 2012. "Novel Technique Radio Frequency Identification (RFID) based to Manage Patient Flow in a Radiotherapy Department". In *Proceedings of the 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics*, pp. 972-975. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Miller, D.C. and J.A. Thorpe. 1995. "SIMNET: The Advent of Simulator Networking." *Proceedings of the IEEE*, 83(8):1114-1123.

Mustafee, N., A. Harper, and B.S. Onggo. 2020. "Hybrid Modelling and Simulation (M&S): Driving Innovation in the Theory and Practice of M&S". In *Proceedings of the 2020 Winter Simulation Conference*, edited by K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, 3140-3151. Piscataway, NJ: IEEE.

Mustafee, N., K. Katsaliaki, and S.J.E. Taylor, 2014. "A Review of Literature in Distributed Supply chain Simulation". In *Proceedings of the 2014 Winter Simulation Conference*, edited by A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, 2872-2883. Piscataway, NJ: IEEE.

Mustafee, N., K. Katsaliaki, and S.J.E. Taylor. 2021. "Distributed Approaches to Supply Chain Simulation: A Review". *ACM Transactions on Modeling and Computer Simulation*, 31(4), 1-31.

Mustafee, N., S.J.E. Taylor, K. Katsaliaki, and Brailsford, S. 2009. "Facilitating the Analysis of a UK NBS Chain Using the HLA." *Simulation: Transactions of the Society of Modelling and Simulation International*, 85(2):113-128.

Mustafee, N., A. Harper, and J. Viana. 2023. "Hybrid Models with Real-time Data: Characterising Real-time Simulation and Digital Twins". In *Proceedings of the OR Society Simulation Workshop 2023 (SW23)*, Southampton, pp. 261-271. https://doi.org/10.36819/sw23.031. UK Operational Research Society.

Mustafee, N. and J.H. Powell. 2021. "Providing Real-Time Information for Urgent Care". *Impact* 2021(1):25-29.

Onggo, B.S., N. Mustafee, A. Smart, A.A. Juan, and O. Molloy. 2018. "Symbiotic Simulation System: Hybrid Systems Model Meets Big Data Analytics". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 1358-1369. Piscataway, NJ: IEEE.

Onggo, B.S., C.G. Corlu, A.A. Juan, T. Monks, and R. de la Torre. 2021. "Combining Symbiotic Simulation Systems with Enterprise Data Storage Systems for Real-Time Decision-Making". *Enterprise Information Systems*, 15(2), 230-247.

Powell, J. H. and N. Mustafee. 2017. "Widening Requirements Capture with Soft Methods: An Investigation of Hybrid M&S Studies in Healthcare." *Journal of the Operational Research Society*, 68(10):1211-1222.

Sun, T., X. He, X. Song, L. Shu, and Z. Li. 2022. "The Digital Twin in Medicine: A Key to the Future of Healthcare?". *Frontiers in Medicine*, 9.

Sun, T., X. He, and Z. Li. 2023. "Digital Twin in Healthcare: Recent Updates And Challenges". *Digital Health*, 9, 20552076221149651.

Shahat, E., C.T.Hyun, and C.Yeom. 2021. "City Digital Twin Potentials: Review & Research Agenda". *Sustainability*, 13(6), 3386.

Tolk, A., A. Harper, and N. Mustafee. 2021. "Hybrid Models as Transdisciplinary Research Enablers." *European Journal of Operational Research*, 291(3): 1075-1090.

VanDerHorn, E. and S. Mahadevan. (2021). "Digital Twin: Generalisation, Characterisation and Implementation". *Decision Support Systems*, 145: 113524.

Warwick, B., J. DeLurgio, G. Fu, and J. Welsh. 2023. "Developing Healthcare Applications using Epic Electronic Health Record APIs. AWS Blog". https://aws.amazon.com/blogs/industries/connecting-aws-chatbot-to-epic-electronic-health-record-apis-using-amazon-lambda/. Last accessed May 2023.

AUTHOR BIOGRAPHIES

NAVONIL MUSTAFEE is Professor of Analytics and Operations Management at the University of Exeter Business School, UK. His research focuses on Modelling & Simulation (M&S) methodologies and Hybrid Modelling and their application in healthcare, supply chain management, circular economy and resilience and adaptation due to climate change. He is a Joint Editor-in-Chief of the Journal of Simulation (UK OR Society journal) and Vice-President of Publications at The Society of Modeling and Simulation International (SCS). His email address is n.mustafee@exeter.ac.uk.

ALISON HARPER is a Lecturer at the University of Exeter Business School, UK. She gained her PhD from the University of Exeter. Her research interests are applied health and social care research using data science and quantitative methods to model and improve services. Her email address is a.l.harper@exeter.ac.uk.

JOE VIANA is a Researcher on the Norwegian Research Council Measures for Improved Availability of medicines and Vaccines project (300867), at BI Norwegian Business School. He is a member of the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) InPlan-CARE network (443158418). He gained his PhD in from the University of Southampton. His interests are the application of simulation to improve the operation of health systems. His email address is joe.viana@bi.no.