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

SUPPORTING EMERGENCY DEPARTMENT RISK MITIGATION WITH A MODULAR AND REUSABLE AGENT-BASED SIMULATION INFRASTRUCTURE

Thomas Godfrey
Steffen Zschaler

Department of Informatics, King's College London
Bush House, Strand Campus
30 Aldwych, London, WC2B 4BG, UK

Rahul Batra
Sam Douthwaite
Jonathan Edgeworth

Guy's and St Thomas' NHS Foundation Trust
Westminster Bridge Rd
London SE1 7EH, UK

Matthew Edwards

King's College Hospital NHS Foundation Trust
Denmark Hill
London SE5 9RS, UK

Simon Miles

Aerogility Ltd
Smithfield Business Centre
5 St John's Ln
London EC1M 4BH, UK

ABSTRACT

For emergency departments (EDs) to maintain sustainable care of patients, hospital management must continually explore potential interventions to clinical practice. Agent-based modelling (ABM) can be a valuable tool to support this planning in a controlled environment. Existing approaches to ABM development are best suited for one-off models. However, conditions in EDs can change frequently, making the use of one-off models infeasible. Decision-makers must be able to trust simulations appropriately for them to be effective in intervention exploration. Domain-specific modelling languages (DSMLs) can address these challenges by offering a reusable library of appropriately abstract, domain-familiar, modelling concepts across case studies and automatic translation of these concepts into executable models. In this paper, we present a DSML to support repeated modelling exercises in the ED domain and illustrate the use and reuse of this DSML across two concrete case studies in London-based NHS emergency departments.

1 INTRODUCTION

EDs must run 24/7 and cater for a wide array of emergency, unscheduled, patient visits. In the UK, the NHS continually monitor the resilience of Trust EDs to adverse factors including infectious diseases, patient attendance rises, and resource availability (England 2022). EDs must constantly evolve to keep up with demand, however, evaluating potential interventions to clinical practice can be complex due to the safety-critical nature of the ED. Particularly if there is limited evidence to support an intervention, it may be difficult to justify the risks of conducting real-world trials.

ABMs can help support the evaluation of an intervention without impacting the real system. Interventions can be replicated within a model and experimented with in order to evaluate their potential beyond what may be possible, or safe, in the real world (Peck 2004). By exploring models under varying conditions, users can determine the resiliency of potential interventions before enacting any real-world changes. For simulation to be effective in the healthcare domain, however, it is important that models can be developed rapidly, and be seen with credibility.

DSMLs can address these challenges by implementing models using high-abstraction statements that are familiar to members of the target domain rather than with generic statements such as with general-purpose programming languages (Fowler 2010). The abstract syntax of a DSML defines the meta-model of language concepts that can be instantiated to capture domain processes and the concrete syntax defines how these language concepts are presented to the user. With a DSML, domain stakeholders can directly collaborate on model implementations which can be automatically generated into computer-readable code. In this paper, we present a healthcare-based DSML developed in collaboration with healthcare professionals from NHS Hospitals.

In this paper, we make the following contributions:

- We present a novel family of DSMLs capturing relevant aspects of work in the ED domain.
- We show how models in these DSMLs can be automatically translated into agent-based simulations.
- We demonstrate the usefulness and reusability of the DSMLs in two case studies.

In section 3 we present the design of our healthcare DSML, using a real-world case study model as a running example. In section 4 we then discuss two case studies to illustrate the use of our DSML across different modelling instances. In section 5 we call upon our experience from the case studies and refer back to the requirements discussed in section 3 to evaluate our DSML.

2 RELATED WORK

Abar et al. provide an excellent review of the state-of-the-art of agent-based modelling languages and tools, mapping eighty-five ABM toolkits according to their modelling capabilities and their required modelling effort (Abar et al. 2017). The reviewed tools are designed for a wide scope of target domains from generic social systems to engineering, economics, biology, etc. and are written in a range of implementation languages from general-purpose programming languages such as Java and C to custom languages such as FlexScript for FlexSim (Nordgren 2003). Across the current ABM toolkits, however, models are typically written from an agent perspective. Model developers are required to define models according to the properties and interactions of individual agents, mapping their knowledge of high-level domain processes to low-level, granular, process descriptions. This abstraction process can be complex, time-consuming, alienating to domain experts, and therefore prone to error.

While some modelling tools, such as AnyLogic (Borshchev 2014), offer domain-appropriate, sometimes graphical, notations which are often more accessible than general-purpose programming code, these languages are not suitable for our target domain as they do not appropriately hide the technical complexity of model implementations and/or use healthcare-specific syntax. For example, FlexSim does include a healthcare module with domain-appropriate process templates, however, the model definitions still use terms such as ‘token’, ‘state’, ‘event trigger’ and ‘process flow’ inherited from event-driven modelling terminologies. The low abstraction level of languages such as these can make the reuse and maintenance of models challenging (Polack and Alden 2020). Instead, implementing models with modular, high-level statements (such as in a DSML) can make maintenance easier— allowing the modification of individual components without affecting the overall structure of the model.

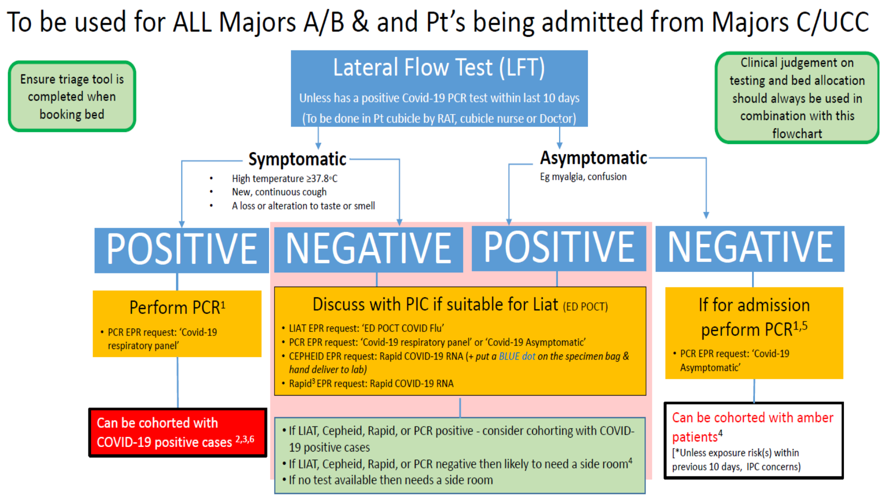
The work of Parunak (Parunak 2021) closely matches our vision for domain-appropriate modelling languages. Their DSML for defence modelling uses a variety of technologies including spreadsheets to capture problem descriptions, from which they can automatically generate executable ABMs. The high-abstraction DSML requires less of a mental shift for domain experts to understand and manipulate models themselves. The current work applies a similar approach to the healthcare domain and employs explicit model-driven engineering processes in our DSML development.

3 A FAMILY OF HEALTHCARE MODELLING LANGUAGES

We began a collaboration with St Thomas Hospital ED in October 2020 with the aim of introducing ABM to support decision-making on infection control practices. The fast-paced nature of healthcare interventions meant that our models needed to be developed within short time frames, and because ABM was new to the department, we needed to establish a suitable degree of credibility for our simulations. To facilitate these requirements our DSML needed to meet the following criteria:

1. The DSML must have sufficient scope so as to capture emergency department healthcare processes.
2. The DSML must have sufficient generalisability to be reused across different modelling case studies in the domain.
3. The DSML must facilitate the timely production, and refactoring, of models
4. The DSML must expose an accessible and usable language syntax for our domain experts such that they could read and understand models without intervention.

We based our language on a pre-existing information format used in the hospital, called action cards. Action cards describe the processes staff should complete during patient treatment via flowcharts. Senior clinicians will draft action cards and publish them for staff to access on the Trust Intranet. We analysed some existing action cards to understand the domain concepts being captured and how these concepts were presented. Action cards explicitly capture information such as the order of treatments and decision points for staff. An example action card is shown in Figure 1.


To be used for ALL Majors A/B & and Pt's being admitted from Majors C/UCC

Ensure triage tool is completed when booking bed

Lateral Flow Test (LFT)
Unless has a positive Covid-19 PCR test within last 10 days
(To be done in Pt cubicle by RAT, cubicle nurse or Doctor)

Clinical judgement on testing and bed allocation should always be used in combination with this flowchart

Symptomatic
- High temperature ≥37.8◦C
- New, continuous cough
- A loss or alteration to taste or smell

Asymptomatic
Eg myalgia, confusion

POSITIVE

NEGATIVE

POSITIVE

NEGATIVE

Perform PCR[1]
- PCR EPR request: 'Covid-19 respiratory panel'

Discuss with PIC if suitable for Liat (ED POCT)
- LIAT EPR request: 'ED POCT COVID Flu'
- PCR EPR request: 'Covid-19 respiratory panel' or 'Covid-19 Asymptomatic'
- CEPHEID EPR request: Rapid COVID-19 RNA (+ *put a BLUE dot on the specimen bag & hand deliver to lab)*
- Rapid[3] EPR request: Rapid COVID-19 RNA

If for admission perform PCR[1,5]
- PCR EPR request: 'Covid-19 Asymptomatic'

Can be cohorted with COVID-19 positive cases [2,3,6]

- If LIAT, Cepheid, Rapid, or PCR positive - consider cohorting with COVID-19 positive cases
- If LIAT, Cepheid, Rapid, or PCR negative then likely to need a side room[4]
- If no test available then needs a side room

Can be cohorted with amber patients[4]
[*Unless exposure risk(s) within previous 10 days, IPC concerns)

Figure 1: A real-world action card example.

For the purposes of implementing a model of domain processes, it is necessary for us to explicitly encode properties in our DSML that are left implicit in action cards. For example: required staff types, action duration, and the ED layout. The diverse range of domain information needed to model ED processes led us to divide our DSML definition into modular packages as shown in Figure 2. Each package is responsible for orthogonal domain processes. The action card module is responsible for defining high-level global processes. The people module is responsible for capturing the properties of staff and patients. The disease module is responsible for defining the properties of diseases, their prevalence, spread properties, and the tests used to detect diseases. The built environment language is responsible for defining the physical layout of the hospital, including the definition of rooms and resources. Finally, the agent language is an intermediary, agent-oriented, DSML designed to subdivide the model generation process.

In the following section, we will discuss the meta-model of our DSML alongside a case study: The investigation of alternative testing pathways for COVID-19 in St. Thomas' Hospital ED.

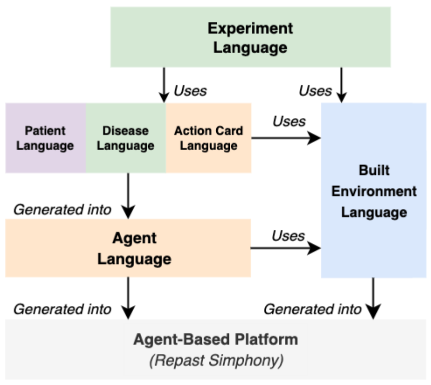


Figure 2: The DSML structure.

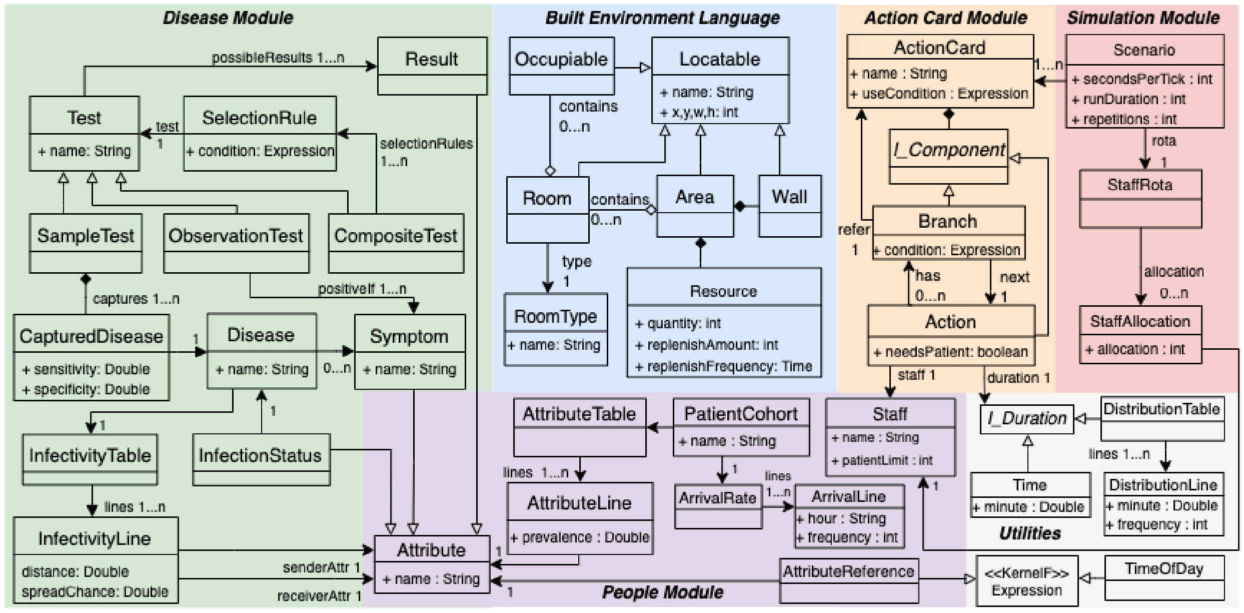


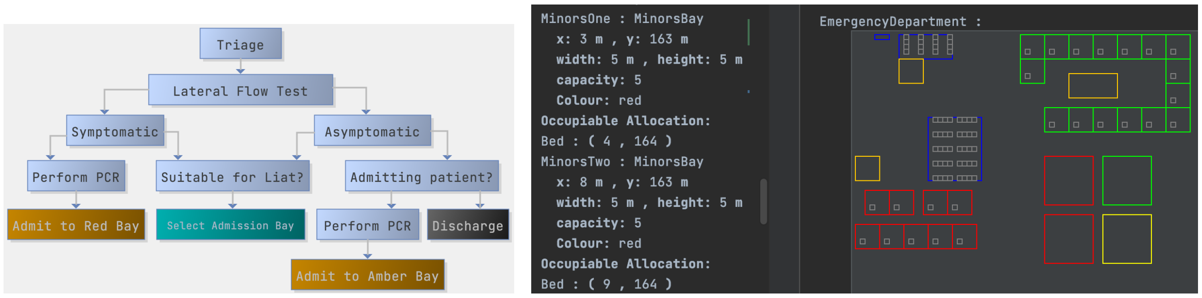
Figure 3: The healthcare DSML modules.

3.1 COVID-19 Case Study

The management of COVID-19 in St. Thomas' ED has been an ongoing concern. It is vital that infectious patients are identified early to reduce the risk of disease spread to others. If the patient is to be admitted to the wider hospital wards, it is especially important that patients have been accurately diagnosed to ensure they are cohorted appropriately with other patients. Outbreaks of COVID-19 can result in costly and potentially dangerous closures of entire wards in the hospital.

Three types of tests were used in 2020 for detection of COVID-19. Lateral flow tests (LFT) have a lower accuracy but can be completed at the patient bedside and can produce results within 20 minutes. Rapid-PCR tests have high accuracy but need to be processed in analyser machines, in batches, for 30 to 45 minutes. Lab-based PCR tests offer highly-accurate testing but have significantly slower response times due to them being processed in a separate laboratory in the hospital. Lab-based PCR tests are often referred to as 'confirmatory tests' due to their higher accuracy.

Hospital admission wards were divided into bays according to COVID-19 infectious status. *Red bays* were for patients who had received confirmatory positive test results for COVID-19 and *Green bays* were



```
MinorsOne : MinorsBay
  x: 3 m , y: 163 m
  width: 5 m , height: 5 m
  capacity: 5
  Colour: red
Occupiable Allocation:
Bed : ( 4 , 164 )
MinorsTwo : MinorsBay
  x: 8 m , y: 163 m
  width: 5 m , height: 5 m
  capacity: 5
  Colour: red
Occupiable Allocation:
Bed : ( 9 , 164 )
```

```
EmergencyDepartment :
```

(a) Action card DSML concrete syntax. (b) Built environment DSML concrete syntax.

Figure 4: Example models.

for patients who had received confirmatory negative test results. *Amber bays* were for patients who had received a negative COVID-19 result but were still awaiting a confirmatory test. A very limited number of isolated *Side Rooms* were available for undetermined cases or for patients with more severe health complications.

In early 2021, as COVID-19 prevalence was predicted to decrease, the ED were interested in exploring potential action cards with the aim of reducing the use of different test types while minimising potential risks to patient safety. The aim of our model was to compare two prospective action cards to the current pathway. In the current pathway, LFTs, rapid-PCR, and lab-based PCR tests were all used. In the first prospective action card, LFTs are removed, and in the second prospective action card, rapid-PCR testing is removed. We wished to compare the directional changes in key performance indicators (KPIs) across the different action cards: *1. The percentage chance of a COVID-19-positive patient being admitted to an amber bay*, 2. *The number of LFT, rapid-PCR, and Lab-based PCR tests used per day*, and *3. The average patient length of stay (LoS)*.

3.2 Action Card Module

Shown in the top-right of Figure 3: The `ActionCard` concept contains a set of `I_Components` for capturing action cards. `Actions` represent individual steps of global processes in the hospital (such as administering a test, taking history, and admitting/discharging the patient). These `Actions` include properties such as their `Duration` (as either a single value `Time` or a `DistributionTable`), the required `Staff`, and whether the patient is required to be present during the action. `Branches` between actions specify the order of operation for the `Actions` and can optionally have associated 'conditions' implemented as KernelF `Expressions` (Völter 2018). These conditions are evaluated during simulation to determine whether the next `Action` can be started or not.

3.2.1 Defining the COVID-19 Action Cards

First, we implemented the baseline action card in our DSML. This required some clarification from our domain experts on more fine-grained details such as the staff involved in each action, the location, and the duration. The representation of the action card in our DSML is shown in Figure 4a.

We specified the series of actions staff must take during the patient pathway from initial triage and LFT testing to admission. Each step is defined as its own `Action` with properties such as the duration, required `Staff`, and whether the patient needs to present or not. The sub-types of the `I_Duration` definition meant that we could capture some actions (such as *Triage*) using a static `Time` value and other actions (such as the *Symptomatic* check) using a `DistributionTable` on a case-per-case basis, and swap between these as necessary without impacting the rest of the model.

3.3 People Module

Shown in the bottom of Figure 3: The people module is used to capture patients and staff. The `PatientCohort` concept captures the properties of patients via `AttributeTables` which define the prevalence distribution of different `Attributes` in patients. For example, the `Severity Attribute` represents the seriousness of the patient's condition and hence what area of the ED they should be streamed to. `Attributes` can be referenced via an `AttributeReference` which is a sub-type of `Expressions`, and so can be included in `Branch` conditional statements.

`Staff` are defined by their name and patient allocation limit (i.e. the maximum number of patients the staff can be assigned to at one time). The default location for `Staff` is defined by a reference to the `Room` concept in the built environment language. If defined, `Staff` will start the simulation in this location and will return there when idle. The module generator includes default behaviours for patient selection in which staff prioritise seeing patients to which they are currently assigned. If there are no such patients, and the staff member has not yet reached their patient allocation limit, then they will allocate themselves to a new waiting patient. These behaviours are written using sub-types of the `MessageSelectionRule` concept in the agent language discussed in section 3.7. The different concept sub-types will allow us to implement specific selection strategies in future work without impacting the rest of the model definition.

3.4 Built Environment Language

Shown in the middle of Figure 3: The built environment language contains concepts for capturing physical space as sub-types of `Locatables` with an x,y location and width and height properties. `Areas` represent collections of `Rooms`, which can contain `Occupiables` such as chairs, desks, etc. In the future, we can add new types of furniture, etc. by creating new sub-types of the `Occupiable` concept. Areas can be subdivided by adding `Walls` which cannot be traversed by agents in the model. `Rooms` have an associated `RoomType` to group collections of rooms such as cubicles, receptions, etc. `Resources` can be used to define items such as test cartridges, etc. and have properties to capture the starting quantity of the `Resource` as well as the replenishment frequency and quantity.

3.4.1 Defining the St. Thomas' Emergency Department Environment.

A representation of the St. Thomas' ED implemented using our built environment language is shown in Figure 4b. Each `Room` is represented by a rectangle and is coloured according to the `RoomType`. The grey squares represent `Occupiables` which include chairs in waiting rooms and beds in cubicles. The built environment language uses a dual-projection concrete syntax. `Locatables` are defined textually and a graphical representation of the built environment is displayed simultaneously to assist with model readability. In the future, we can implement more detailed architectural models by extending the existing DSML concepts with additional properties and alternative abstract syntax.

3.5 Disease Module

Shown in the left of Figure 3: The disease module captures the properties of infectious diseases. The `Disease` concept can have related `Symptoms` and `Tests` used to detect the disease. We extend the `Attribute` concept in the people module to add an `InfectionStatus` property for patients. This property defines, for each `Disease`, whether the patient is *Infected* (including *Symptomatic* or *Asymptomatic*) or if they are *Susceptible* as inspired by compartment models in epidemiology (Brauer 2008).

The module includes two sub-types of `Test`: `SampleTest` and `ObservationTest`. `SampleTests` represent black-box tests with sensitivity and specificity values. `ObservationTests` replicate medical assessments mechanistically by checking the presence of `Symptoms` in patients. `SampleTests` are a sub-type of the `Resource` concept and so can be supplied a starting quantity, replenishment frequency, and

Area allocation. The Result concept represents the range of outcomes a Test can produce. Results are sub-types of Attributes and so can be evaluated as part of Branch conditions. Each Test can reference an ActionCard to specify the series of actions that staff must complete whenever the test is administered. CompositeTests represent collections of Tests that can be referenced as a single object by actions. SelectionRules can be used to define which specific test within this group should be used in different scenarios. Expression statements within SelectionRules allow us to reference Attributes of patients etc. for decision-making.

The InfectivityTable is used to define how Diseases spread between patients and is inspired by (Laskowski et al. 2011). For each Disease, the user can specify its *spreadChance* per second based on Attributes of the spreader and/or receiver, as well as the *distance* between the two people.

3.5.1 Defining the COVID-19 Disease and Related Tests

We implemented a *COVID-19* Disease and an AttributeTable for the distribution of patients' InfectionStatus. We then defined two types of SampleTest: *LFT* and *LabPCR*, which included COVID-19 as their CapturedDisease and were given appropriate sensitivity and specificity values based on data collected from clinical trials (Merrick et al. 2021). A *RapidPCR* CompositeTest was implemented to capture two sub-types of the rapid-PCR test in use in the ED —each with their own distinct properties such as duration, sensitivity, and specificity. A SelectionRule was defined to state that one type of test should be used during daylight hours and the other test should be used during the night. We implemented an ObservationTest to represent the check staff perform to detect if a patient is symptomatic for COVID-19. We defined a set of Symptoms and associated them with the COVID-19 Disease via the 'presentsWith' reference. As a sub-type of the Attribute concept in the patient language, these Symptoms were given associated prevalence rates in their AttributeTables. We referenced these Symptoms in our ObservationTest via the 'positiveIf' reference.

3.6 Simulation Module

Shown in the right of Figure 3: The simulation language exposes concepts for configuring simulation experiments as Scenarios. In a Scenario, the user can define the type and quantity of Staff to include via a StaffMap, the length of simulation runs, the tick rate, and the number of run repetitions. Changing the tick rate of the simulation will automatically adjust all timing-based properties of the model definition during generation.

3.7 Agent Language

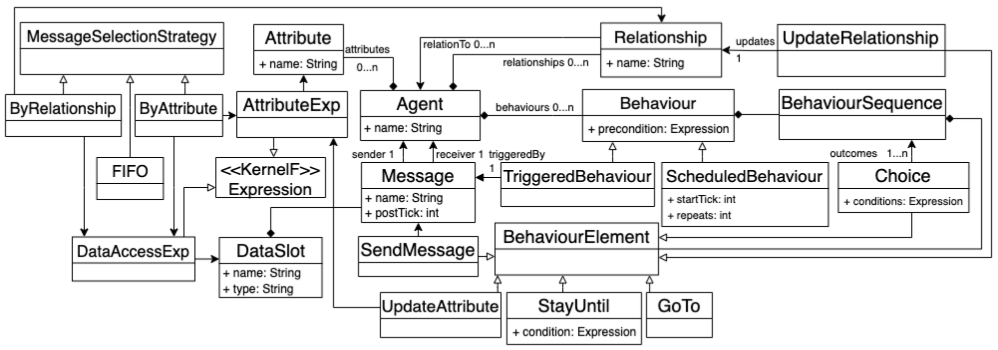


Figure 5: The agent language meta-model.

Rather than translating models directly into general-purpose programming code, we implemented an intermediary agent-based language inspired by actor languages (Agha 1986). Every model implemented in the healthcare DSML modules is automatically generated into this agent language, which itself can be automatically generated into Repast Simphony code (North et al. 2013). The language (shown in Figure 5) expresses models according to `Agents` with associated `Attributes`, `Behaviours` and `Relationships` (i.e. networks of associations between different agents that can be created and removed during runtime).

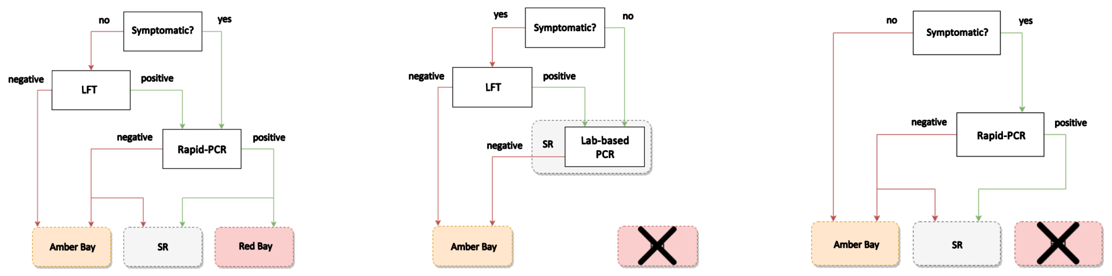
`Behaviours` contain a `BehaviourSequence` of `BehaviourElements`. Different sub-types of `BehaviourElements` represent basic actions for agents to complete. This includes: Updating the value of the agent's attribute via an `AttributeExp` (`UpdateAttribute`), moving to a location (`GoTo`), staying at the current location until a condition is met (`StayUntil`), updating a `Relationship` (`UpdateRelationship`), and sending a `Message` (`SendMessage`). The `Message` concept contains a set of `DataSlots` which are *name, type* pairs allowing `Agents` to share information regarding instances of other `Agents`, etc. via a `DataAccessExp`. `Choices` add flow operations, defining which `BehaviourSequence` should be followed according to the outcome of an `Expression` conditional statement.

`Behaviours` are triggered either by time (`ScheduledBehaviour`) or by the receipt of a `Message` (`TriggeredBehaviour`). Scheduled behaviours have a *starting tick* and an optional *repeats* tick frequency. Triggered behaviours are executed when the associated `Message` is received by the owning agent. `Behaviours` can have `Preconditions` in the form of `Expressions` that will be evaluated to determine if the agent will start the behaviour or not. For example, by default, all `Behaviours` that involve moving to a `RoomType` will automatically have a pre-condition to check that there exists a `Room` of the given type that is not currently occupied.

The `MessageSelectionStrategy` concept is used to define how agents select `Messages`. The strategy sub-types define the selection semantics: `ByAttribute` involves evaluating the value of the sending `Agent`'s `Attribute`, `ByRelationship` involves checking if the sender is in the set of `Relationships` of the receiving `Agent`, and `FIFO` prioritises `Messages` that were sent less recently. `MessageSelectionStrategies` are used to implement `PatientSelectionStrategies` in the people module. The 'patient allocation limit' of `Staff` is translated into a `Relationship`, and the status of this `Relationship` is evaluated by each `Agent` of the correct type. Every time an `Action` is completed, a `Message` is sent which is accessible to all `Agents` of the `StaffType` who can perform the `Patient`'s next `Action`. The `DataSlot` for that `Message` contains the `Patient` instance. The first `Agent` that meets all conditions of their `MessageSelectionStrategy` will remove that `Message` and select the contained `Patient` instance for their next `Action`.

3.8 Model Execution

Models expressed in the DSML are generated into Repast Simphony code (North et al. 2013). Repast Simphony includes functions for agent scheduling, parameter sweeps, visual interfaces, and model data collection. We use RRepast (García and Rodríguez-Patón 2016) (An R package that interfaces with Repast Simphony) to assist with sensitivity analysis of our models. All model parameters are automatically generated into Repast parameter XML files which can be read by RRepast scripts. During generation, `Agents` are converted into Repast agent classes. `Attributes` are translated into variables within those classes and Repast parameters so that they can be referenced and manipulated during batch runs. `Messages` are converted into classes which are referenced by generated `TriggeredBehaviour` methods. `ScheduledBehaviours` are translated into methods with the 'Schedule' annotation so that they can be recognised by the Repast scheduler. `BehaviourSteps` are converted into inner classes within the agent classes and are instantiated as needed at run-time.



(a) Using both LFT and rapid-PCR testing.

(b) Using only LFTs.

(c) Using only rapid-PCRs.

Figure 6: The COVID-19 testing action cards.

4 CASE STUDIES

In this section, we discuss two case studies conducted in collaboration with London-based NHS EDs. In each study, we used our DSML to develop models to explore potential interventions to the ED with the goal of satisfying the KPIs discussed in section 1: Resource usage, patient flow, and patient safety.

4.1 COVID-19 Case Study Results and Discussion

In section 3, we discussed how our DSML was used to capture domain processes for a COVID-19 case study. We modelled a baseline action card and two prospective action cards and ran simulation experiments to compare the relative directional changes to the three KPIs discussed in section 3.1. For each `ActionCard`, we ran the model for 7 days simulation time with a tick rate of 60 seconds and a warm-in period of 1 day. We ran repeats of each model across COVID-19 prevalence rates between 0.25% to 1% at 0.25% intervals. Stability analysis showed that we needed to execute 50 repetitions of our model in order for stochastic effects to normalise. To calibrate the model, we compared our model outputs to real-world aggregated data collected from the ED.

During calibration, we found that the model outputs included a much lower use of rapid-PCR tests than observed in real-world data. This test type was used in instances where there was a discrepant result between the triage `ObservationTest` and the LFT result. If there were fewer discrepancies, then the use of the rapid-PCR would be lower, leading us to understand that our `ObservationTest` had been modelled as more accurate than observed in the real world. Subsequently, we refactored our model to capture the triage checks as a `SampleTest` instead. We calibrated the sensitivity and specificity values of this test with a parameter sweep using data collected from the ED. As both of these test types are sub-types of the `Test` concept, their respective references in the action card language were unaffected, and all other aspects of the model did not need to be refactored.

We conducted face validity of our model with a small group of domain experts. The healthcare-focused language design allowed us to show action cards directly within our DSML and discuss any ambiguities. Clinicians were able to understand the DSML without previous training and were able to identify potential issues with a proposed action card, sparking a discussion on its real-world implications on clinical practice even before executing the model.

Our model results suggested that the baseline action card using both rapid-PCR and LFT (action card 6a) was the most effective: there was the lowest risk of a COVID-19-positive patient being admitted to an amber bay. Replacing rapid-PCR testing with Lab-based PCR testing (action card 6b) did not significantly affect risk for the tested COVID-19 prevalence rates, but patient waiting times were increased due to the longer processing times of Lab-based PCR tests. Removing LFT testing and relying on symptomatic checks at triage (action card 6c) resulted in a significant increase in risk. The probability of COVID-19-positive patients being admitted to an amber bay was 3 times higher than the baseline at 1% community prevalence for COVID-19, suggesting that this action card was unlikely to perform adequately in the real world with respect to the tested KPIs.

4.2 Group A Streptococcus

In December 2022, the NHS received a significant increase in paediatric attendances due to a rise in GAS cases in the UK (Singh 2022). This put increased strain on the Kings' College Hospital (KCH) paediatric department. It became crucial that staff manage the flow of patients such that serious illnesses could be detected and isolated early. Existing guidelines at KCH did not include an early-detection protocol for GAS. Mild and serious cases were mixed, both with each other and with patients presenting with non-GAS illnesses, contributing to a significant rise in patient waiting times. We wished to explore the potential for a new triage system that would isolate GAS cases early in the patient journey. The new process would require a small set of staff to be exclusively dedicated to GAS triage, moving them away from their typical roles in the department. The question was whether a dedicated GAS-triage action card would impact the following KPIs when compared to the baseline existing process: *1. The early detection rate of GAS cases as measured by patient time-until-seen* and *2. The average patient length of stay*

We developed our model in collaboration with a senior clinician at KCH. While formalising the details of `Actions` in our DSML, we simultaneously updated a real-world action card to help facilitate discussion between domain experts before running model experiments. Our DSML concepts could be reused for this study. The DSML structure provided the groundwork of domain analysis (Godfrey et al. 2022) and provided a framework for which we could rapidly instantiate concepts to capture new domain processes. We implemented two `PatientCohorts` with respective `ArrivalRates` and `AttributeTables`, including prevalence for a new Group A Strep `Disease`. One cohort included GAS patients and the other included non-GAS patients. 40% of the GAS cohort had a symptomatic `InfectionStatus` for GAS (0% in the non-GAS cohort) and their `ArrivalRate` had a higher concentration of patients during daylight hours to capture the domain property that GAS attendances were more likely during the day than during the night at KCH. The baseline model and the GAS triage model include the same pool of `Staff`: 3 majors doctors, 3 majors nurses, 2 minors doctors, 2 minors nurses, and 2 senior doctors. However, in the GAS triage process, 1 senior doctor and 1 majors nurse are instead converted into GAS staff.

Each model run represented a 7-day period with a tick rate of 60 seconds and a warm-in period of 1 day. During calibration, we found that patient LoS was significantly higher than observed in the real world. When observing model behaviours during simulation runs, we found that patients were crowding in waiting rooms after triage. The nurses were overloaded with patients resulting in long waiting times. Through discussion with our domain expert, we established that the likely cause of this issue was that our model explicitly encoded patient waiting times within the duration of some actions. For example, our action cards stated that patients must wait for a set period of time before discharge therefore artificially inflating the patient LoS in an attempt to abstract treatment processes. This led to a deeper discussion of domain processes, benefiting not just the development of our model, but also facilitating further analysis of real-world processes. While our preliminary results indicate that the GAS triage process does not significantly increase patient LoS, and does reduce time-until-seen in comparison to the baseline, future work will focus on the further calibration of our model to real-world data once it is available for access.

5 CONCLUSION AND FUTURE WORK

In section 1, we listed the requirements for our DSML. Requirement 1 concerned the scope of our DSML for capturing ED processes and requirement 2 concerned the generalisability of our language for different case studies. Through the case studies discussed in section 4, we have demonstrated that our DSML was capable of capturing the relevant domain processes for two different case studies, demonstrating that our language had sufficient scope and generalisability for our current applications. However, the case studies we have completed so far have all been conducted with London-based NHS EDs with a focus on infection control interventions. Further work will focus on testing our DSML generalisability on a wider set of case studies with different domain problems and environmental contexts.

Requirement 3 concerns the timely production and refactoring of models. Our DSML helps facilitate this through the reuse of language concepts across different case studies. Because our DSML uses high-abstraction concepts, model construction and refactoring involves a significantly lower number of statements compared to using a general-purpose programming language or technical agent-oriented language. Language concepts such as action cards, diseases, and tests, abstract away from technical implementation details. Users can generate and manipulate large blocks of logic without concerning themselves with the underlying implementation. For example, if an `Action` is defined that requires a majors cubicle `Room`, then the user does not need to specify how the staff agents will move to that room, or how the room will be perceived as ‘occupied’ and thus unavailable to other agents. A potential concern is that the high abstraction level of our DSML concepts could reduce coverage such that certain domain processes could not be captured in the language as-is (Völter et al. 2013). The modular nature of our DSML design supports language refactoring and extension (Godfrey et al. 2022). However, future case studies will expose the complexity of this process, and thus the suitability of our DSML for the healthcare domain where model development must be fast-paced.

Requirement 4 concerns the accessibility and usability of our DSML. During our case studies, we were able to demonstrate and discuss our models intermittently during their development. The domain experts were able to identify real-world processes in the model definitions and identify inaccuracies where they occurred. For example, in the GAS case study in section 4.2, we were able to collaboratively develop an action card in our DSML, engaging domain experts in model implementation. An important aspect of language accessibility is the model complexity exposed to the user. With the modular design of our DSML, we can manipulate the scope and level-of-detail of domain concepts included in each model, and the level of detail at which we choose to model processes. Only the domain details relevant to the problem, the simulation experiments, and available data in which to evaluate those experiments need to be explicitly encoded and extraneous detail can be ignored by the model developer. For example, in the GAS case study, we could have modelled the triage process as an `ObservationTest` such that the symptom `Attributes` of patients that match those of GAS are checked in order to provide a positive or negative outcome. However, the resolution of patient symptom data available at the time of the study was not sufficient and it was instead more appropriate to explicitly encode triage results as probability functions in our action card `Branches`. Should we have access to sufficient data on GAS symptom prevalence in paediatric attendances at KCH, we could swap these probability functions for an `ObservationTest` conditional statement in the `Branches`, without impacting the structure of the rest of the model.

While the DSML is not yet suitable for domain experts to manipulate models independently, the languages are accessible enough for domain experts to read, understand, and identify errors in models. Future work will involve conducting formal usability studies such as those discussed in Alaca et al. (2021) and exploring the generalisability of our DSML to further healthcare case studies in new contexts such as non-UK hospitals.

REFERENCES

Abar, S., G. K. Theodoropoulos, P. Lemarinier, and G. M. O’Hare. 2017. “Agent Based Modelling and Simulation Tools: A Review of the State-of-Art Software”. *Computer Science Review* 24:13–33.

Agha, G. 1986. “An Overview of Actor Languages”. *ACM Sigplan Notices* 21(10):58–67.

Alaca, O. F., B. T. Tezel, M. Challenger, M. Goulão, V. Amaral, and G. Kardas. 2021. “AgentDSM-Eval: A Framework for the Evaluation of Domain-Specific Modeling Languages for Multi-Agent Systems”. *Computer Standards & Interfaces* 76:103513.

Borshchev, A. 2014. *Multi-Method Modelling: AnyLogic*, Chapter 12, 248–279. John Wiley & Sons, Ltd.

Brauer, F. 2008. *Compartmental Models in Epidemiology*, 19–79. Berlin, Heidelberg: Springer Berlin Heidelberg.

NHS England 2022, Aug. “Next Steps in Increasing Capacity and Operational Resilience in Urgent and Emergency Care Ahead of Winter”. https://www.england.nhs.uk/long-read/next-steps-in-increasing-capacity-and-operational-resilience-in-urgent-and-emergency-care-ahead-of-winter/#core-objectives-and-key-actions-for-operational-resilience. Accessed: 2023-04-13.

Fowler, M. 2010. *Domain Specific Languages*. 1st ed. Addison-Wesley Professional.

García, A. P., and A. Rodríguez-Patón. 2016. "Analyzing Repast Symphony Models in R with RRepast Package". *bioRxiv*.

Godfrey, T., R. Batra, S. Douthwaite, J. Edgeworth, S. Miles, and S. Zschaler. 2022. "A Methodology for DSML-Assisted Participatory Agent-Based Enterprise Modelling". In *The Practice of Enterprise Modeling*, edited by B. S. Barn and K. Sandkuhl, 201–215. Cham: Springer International Publishing.

Laskowski, M., B. C. P. Demianyk, J. Witt, S. Mukhi, M. R. Friesen, and R. D. McLeod. 2011. "Agent-Based Modeling of the Spread of Influenza-Like Illness in an Emergency Department: A Simulation Study". *IEEE Transactions on Information Technology in Biomedicine* 15:877–889.

Merrick, B., M. Noronha, R. Batra, S. Douthwaite, G. Nebbia, L. Snell, S. Pickering, R. Galao, J. Whitfield, A. Jahangeer, R. Gunawardena, T. Godfrey, R. Laifa, K. Webber, P. Cliff, E. Cunningham, S. Neil, H. Gettings, J. Edgeworth, and H. Harrison. 2021. "Real-world Deployment of Lateral Flow SARS-CoV-2 Antigen Detection in the Emergency Department to Provide Rapid, Accurate and Safe Diagnosis of COVID-19". *Infection Prevention in Practice* 3(4):100186.

Nordgren, W. B. 2003. "Flexible Simulation (Flexsim) Software: Flexsim Simulation Environment". In *Proceedings of the 2003 Winter Simulation Conference*, edited by S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 197–200. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

North, M. J., N. T. Collier, J. Ozik, E. R. Tatara, C. M. Macal, M. Bragen, and P. Sydelko. 2013. "Complex Adaptive Systems Modeling with Repast Simphony". *Complex Adaptive Systems Modeling* 1:1–26.

Parunak, H. V. D. 2021. "Social Simulation for Non-Hackers". In *Multi-Agent-Based Simulation XXII: 22nd International Workshop, MABS 2021, Virtual Event, May 3-7, 2021, Revised Selected Papers*, 1–14. Berlin, Heidelberg: Springer-Verlag.

Peck, S. L. 2004. "Simulation as Experiment: a Philosophical Reassessment for Biological Modeling". *Trends in Ecology & Evolution* 19(10):530–534.

Polack, F., and K. Alden. 2020. "On Developing and Validating Dynamic Systems: Simulation Engineering.". *The Journal of Object Technology* 19(3):1–13.

Singh, Namita 2022. "Emergency Department Flooding with Children as Strep A Death Toll Climbs to 15 - Live". https://www.independent.co.uk/news/health/strep-a-deaths-2022-symptoms-antibiotics-vaccine-b2241913.html. accessed 9th July.

Völter, M. 2018. "The Design, Evolution, and Use of KernelF: An Extensible and Embeddable Functional Language". In *Theory and Practice of Model Transformation: 11th International Conference, ICMT 2018, Held as Part of STAF 2018, Toulouse, France, June 25–26, 2018, Proceedings 11*, 3–55. Springer.

Völter, M., S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. C. L. Kats, E. Visser, and G. Wachsmuth. 2013. *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. dslbook.org.

AUTHOR BIOGRAPHIES

Thomas Godfrey is a PhD student at King's College London, studying domain-specific modelling Languages for participatory modelling of healthcare systems. His email address is thomas.1.godfrey@kcl.ac.uk.

Rahul Batra is the Deputy Director of the Centre for Clinical Infection and Diagnostics Research (CIDR) at Guy's and St Thomas' Hospital. His research interests include point-of-care patient testing. His email address is rahul.batra1@gstt.nhs.uk.

Sam Douthwaite is a Consultant in Infectious Diseases in the Virology Department of Guy's and St Thomas' Hospital in London. His research interests include HIV, viral hepatitis and Tropical infections. His email address is Sam.Douthwaite@gstt.nhs.uk.

Jonathan Edgeworth is a Consultant Microbiologist and Director of CIDR at Guy's and St Thomas' Hospital. His research interests include microbial infection spread, screening, and antibiotics. His email address is Jonathan.Edgeworth@gstt.nhs.uk.

Matt Edwards is a Paediatric and Adult Emergency Medicine Consultant at Kings College Hospital. His research interests include translational learning from healthcare incidents and participatory simulation of EDs. His email address is medwards9@nhs.net.

Simon Miles is the Head of AI at Aerogility. His research interests include agent-based systems, research, mentoring and educating on AI topics. His email address is simon.miles@aerogility.com and his homepage is https://nms.kcl.ac.uk/simon.miles/.

Steffen Zschaler is a Reader in Computer Science at King's College London. His research interests include software engineering, model-driven engineering, non-functional properties, model optimisation, and component-based software engineering. His email address is steffen.zschaler@kcl.ac.uk and his homepage is http://www.steffen-zschaler.de/.