Health Economics Unit

NHS

Modelling Urgent and Emergency Flow

Authors: Joseph Lillington (HEU), Javier Igea (previously HEU), Santosh Kumar (HEU), Sophie Hodges (HEU), David Sgorbati (HEU), Department of Health and Social Care (Various).

Introduction

We further developed a pre-existing urgent and emergency care (UEC) flow model to increase its complexity and make it more representative of real-world care settings. We intend to further broaden the remit of the model in future work, allowing decision-makers to analyse the consequences to UEC flow of operational changes.

The aims of the programme were to:

- Adapt an existing UEC flow model to the pathway of a particular hospital.
- Adjust the model parameters to reflect reality.
- Design a model with the functionality to assess the impact of operational changes on flow performance.

The project involved further developing a type of model known as a discrete-event simulation. This models the events that happen to patients as they pass through A&E, for example, triage, assessment, diagnostics, treatment, discharge and admission. Other urgent care departments are modelled, such as same day emergency care (SDEC) and urgent treatment centres (UTCs), which allows for the diversion of patients away from the emergency department.

The model code was developed in Python/JavaScript/R and parameters/pathway were informed by data and stakeholders.

Why is the project important?

This project was commissioned for many reasons:

- Nationally, emergency departments are under pressure facing extreme demand.
- There is general consensus that the faster patients are seen, the better the outcomes.
- UEC systems are relatively expensive for the NHS and are areas of high-potential for cost-savings.
- Simulations are an inexpensive way of testing out the impact of operational changes before they are made.
- Discrete-event simulations are well suited for this type of modelling as they mimic the impact of resource capacity limitations known to exist in UEC systems e.g. lack of space, lack of spaces, prioritisation of clinical care, etc.

Methodology

This project methodology involved many different stages:

- **Existing code base** – We analysed the existing codebase to understand its structure and the generic modelled pathway.
- **Code restructure** – We refactored the existing code base to increase its flexibility, reduce code volume, and better handle errors.
- **UEC pathway adaptation** – We reconfigured the existing model to mimic an existing hospital pathway. We incorporated features (such as introduction of an SDEC) which were deemed important for operational planning.
- **Reparameterisation** – We adjusted the parameters of the model to be more suited to the target hospital, based on data analysis and expected bounds.
- **Stakeholder opinion** – We engaged with clinical experts to better understand their UEC pathway.
- **Quality assurance** – Multiple data scientists worked on the model using Git version control.
- **Baseline modelling** – We analysed the flow performance of the 'default' model. An example visualisation is shown in Figure 1.
- **Scenario modelling** – We performed scenario modelling/ sensitivity analysis to compare results against our baseline model. This was to explore potential scenarios that might occur in the future.
- **Process modelling** – We used a technique called process mapping to validate our simulated pathway. An example part of a process map is shown in Figure 2.
- **Recommendations** – We reviewed limitations in the modelling to make targeted recommendations for future iterative development. All code was provided upon project completion. The code was routed in object-oriented programming. Figure 2 shows an example piece of code.

Figure 1 Example outputs from the model

Duration - Day 1 time = 11:20

Waiting Room - Occupancy: (23%)

WR #1	WR #8	WR #15	WR #22	WR #29	WR #36
WR #2	WR #9	WR #16	WR #23	WR #30	WR #37
WR #3	WR #10	WR #17	WR #24	WR #31	WR #38
WR #4	WR #11	WR #18	WR #25	WR #32	WR #39
WR #5	WR #12	WR #19	WR #26	WR #33	WR #40
WR #6	WR #13	WR #20	WR #27	WR #34	
WR #7	WR #14	WR #21	WR #28	WR #35	

Resus Room - Occupancy: (20%)

RR #1

Sdec Room - Occupancy: (0%)

SR #1	SR #6

Majors Room - Occupancy: (27%)

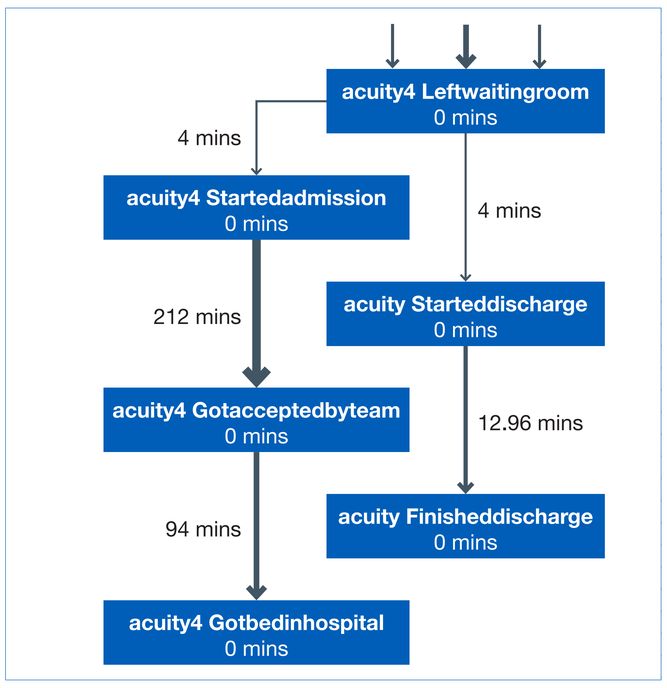
MR #1	MR #6	MR #11	MR #16	MR #21	MR #26
MR #2	MR #7	MR #12	MR #17	MR #22	
MR #3	MR #8	MR #13	MR #18	MR #23	
MR #4	MR #9	MR #14	MR #19	MR #24	
MR #5	MR #10	MR #15	MR #20	MR #25	

Minors Room - Occupancy: (27%)

TR #1	TR #6	TR #11	TR #16	TR #21
TR #2	TR #7	TR #12	TR #17	TR #22
TR #3	TR #8	TR #13	TR #18	TR #23
TR #4	TR #9	TR #14	TR #19	TR #24
TR #5	TR #10	TR #15	TR #20	TR #25

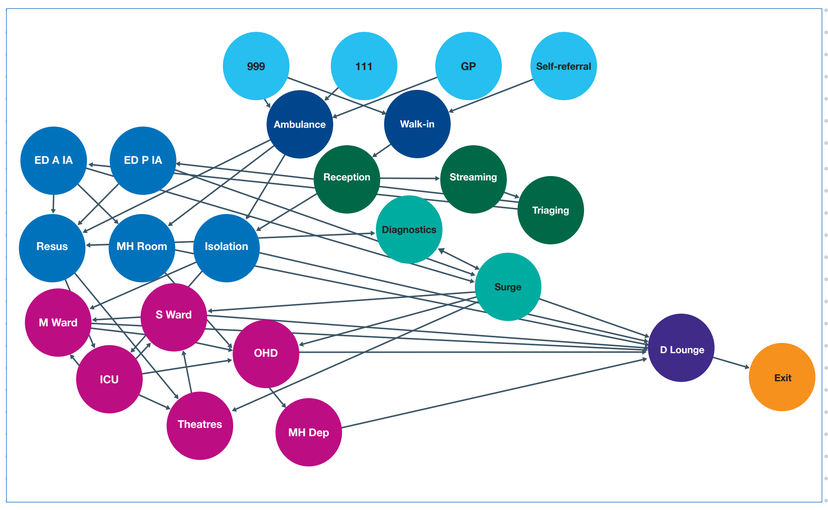
Activity
1
2
3
4
5

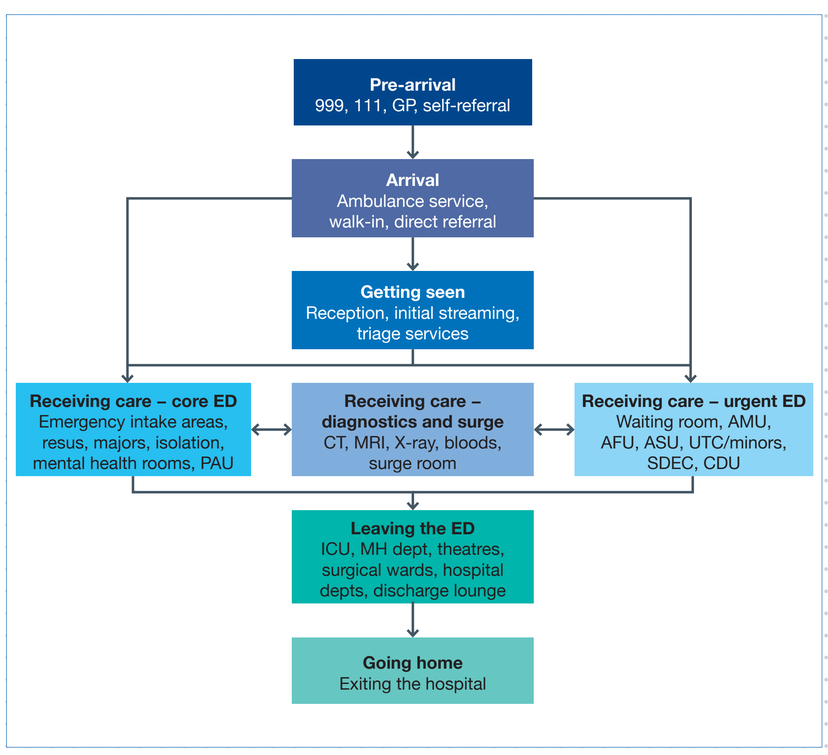
Figure 2 A partial view of the modelled process map of UEC flow. The code is routed in object-oriented programming, as exemplified in the right figure.




```
# Imports
import simpy
import random
import json


# Create class ED
class ED(object):

    # Initialise
    def __init__(self, env, num_doctors, num_sdec_doctors, num_rat_doctors, num_nt_doctors,
                 max_resus, max_sdec, max_majors, max_minors, max_waiting_room):
        self.env = env
        self.doctors = simpy.PreemptiveResource(env, num_doctors)
        self.sdec_doctors = simpy.PreemptiveResource(env, num_sdec_doctors)
        self.rat_doctors = simpy.PreemptiveResource(env, num_rat_doctors)
        self.nt_doctors = simpy.PreemptiveResource(env, num_nt_doctors)
        self.resus = simpy.Container(env, max_resus, init=max_resus)
        self.sdec = simpy.Container(env, max_sdec, init=max_sdec)
        self.majors = simpy.Container(env, max_majors, init=max_majors)
        self.minors = simpy.Container(env, max_minors, init=max_minors)
        self.waiting_room = simpy.Container(env, max_waiting_room, init=max_waiting_room)

    # Method for performing assessment
    def doctor_performs_assessment(self, time_category):
        yield self.env.timeout(time_category)

    # Method for performing CT
    def patient_performs_CT(self, time_CT):
        yield self.env.timeout(time_CT)

    # Method for performing x-ray
    def patient_performs_XRay(self, patient, time_XRay):
        yield self.env.timeout(time_XRay)

    # Method for performing blood test
    def patient_performs_blood(self, time_blood):
        yield self.env.timeout(time_blood)
```

Figure 3 The future modelled pathway, both simplified [Top] and complex [Acuity 1-2 patients only shown, Bottom].





Recommendations

This is a powerful model and has significant potential for helping decision-makers understand the impact of operational changes on UEC flow.

We make the following recommendations for future development of the tool:

- The model needs to be further developed to reflect system pathways. The wider pathway that we intend to analyse is shown in Figure 3. Example components to be added include:
  - Frailty assessment units (FAU), and other key urgent areas.
  - Wider population information (type of referral, chief complaint, etc.)
  - Downstream effects (department occupancy, etc.)
- The model needs to be better parameterised considering uncertainty in patient processes. This will in part involve analysing the Emergency Care Dataset (ECDS) and discussing process variation with ED stakeholders.

Part of ML