Harris MRSA Model - Comprehensive Description

Overview

This is an Agent-Based Model (ABM) simulating MRSA transmission dynamics in a hospital setting. The model tracks patient admissions, discharges, transfers between ICU and ward, healthcare worker (HCW) visits to patients.

Next Steps

- finish visit process including transmission to patient, from patient.
- finish disease model especially transitions from C to I, I to R
- address death vs. discharge

Model Entry Point and Initialization

Builder.build() - The Main Entry Point

The simulation begins when Repast Simphony calls `Builder.build(Context<Object> context)`. This method orchestrates the entire initialization process.

Step-by-step initialization:

1. **Get the Simulation Schedule**
    - Retrieves the Repast Simphony schedule that manages all time-based events
    - The schedule represents time as a double precision numeric. Each day is `1` unit, and events may be scheduled at any point between one "tick" and the next. `10.0` is the beginning of the 10th day. `10.5` is the middle of the 10th day.
2. **Create the Hospital**
    - Creates the main `Hospital` object with configured capacity
    - Hospital capacity: `120` beds total (configurable via `hospitalCapacity`)
    - ICU capacity: `20` beds (configurable via `icuCapacity`)
    - Ward capacity: `100` beds (hospitalCapacity - icuCapacity)
    - The Hospital is added to the simulation context
3. **Initialize Hospital Internal State** During Hospital construction, the following are initialized:
    - **Patient containers:**
        - `patients` - all patients currently in the hospital

  - `inIcu` - patients currently in ICU
  - `notInIcu` - patients currently in ward
  - `patientsNeedingOt/Pt/Rt` - patients requiring therapy visits
- **Discharge processes:**
  - `icuDischarger` - handles ICU patient discharges using LogNormal(scale=`0.820`, shape=`0.916`)
  - `nonIcuDischarger` - handles ward patient discharges using LogNormal(scale=`0.768`, shape=`1.253`)
- **Transfer process:**
  - `transferer` - handles ICU-to-Ward transfers using LogNormal(scale=`1.0`, shape=`0.5`)
- **Data collection buffers:**
  - `visitData` - StringBuffer accumulating all HCW visit records
  - `admissionData` - StringBuffer accumulating all admission records
  - `dischargedPatients` - ArrayList storing DischargedPatient records
- **Network for HCW-Patient assignments:**
  - `hospitalnet` - network tracking which HCWs are assigned to which patients

4. **Start the Admission Process**

   - Creates an `Admission` process with mean inter-arrival time of `0.05` days
   - The admission process uses an Exponential distribution
   - `admissionProcess.start()` schedules the first admission event

5. **Schedule the Builder's Recurring Methods**

   - Schedules `Builder.daily()` to run every `1.0` tick starting at tick `1.0`
   - Schedules `Builder.perShiftOperations()` to run every `0.5` ticks starting at tick `0.5`
   - Schedules `Builder.endOfRun()` to run once at tick `365`

6. **Build All Healthcare Workers**

   - Calls `buildHealthCareWorkers()` to create the HCW workforce

7. **Create Networks**

   - Creates two network structures for visualization/analysis (ICU and wards)

Healthcare Worker Creation

The `buildHealthCareWorkers()` method creates all HCWs and assigns each a PatientVisit process:

Ward Doctors

- **Count:** (hospitalCapacity - icuCapacity) × physiciansPerPatient = `100` × `0.2` = `20` doctors
- **Type:** DOCTOR
- **Visit pattern:** Each doctor gets a PatientVisit process that:
  - Uses Gamma(shape=`0.52`, scale=`90.7`) distribution for inter-visit intervals (in minutes)
  - Adds `6.6` minutes for room visit duration
  - Starts immediately with `pv.start()`
- **Infection control attributes:**
  - Hand hygiene compliance (pre-visit): `0.5`
  - Hand hygiene compliance (post-visit): `0.5`
  - PPE/glove compliance: `0.5`
- **Assignment:** Added to `wardContext` with `icu=false`

ICU Doctors

- **Count:** icuCapacity × icuPhysiciansPerPatient = `20` × `0.3` = `6` doctors
- **Type:** DOCTOR
- **Visit pattern:**
  - Uses Gamma(shape=`0.52`, scale=`35.3`) distribution (shorter intervals than ward)
  - Starts immediately
- **Assignment:** Added to `icuContext` with `icu=true`

Ward Nurses

- **Count:** (hospitalCapacity - icuCapacity) × nursesPerPatient = `100` × `0.2` = `20` nurses
- **Type:** NURSE
- **Visit pattern:**
  - Uses Gamma(shape=`0.54`, scale=`55.1`) distribution
  - Starts immediately
- **Assignment:** Added to `wardContext` with `icu=false`

ICU Nurses

- **Count:** icuCapacity × icuNursesPerPatient = `20` × `0.5` = `10` nurses
- **Type:** NURSE
- **Visit pattern:**
  - Uses Gamma(shape=`0.54`, scale=`20`) distribution (much more frequent visits)
  - Visit check interval: `1/3` day (`8` hours)
- **Assignment:** Added to `icuContext` with `icu=true`

ICU Respiratory Therapists

- **Count:** icuCapacity × icuRtsPerPatient = `20` × `0.1` = `2` ICU RTs
- **Type:** ICURT
- **Visit pattern:**
  - Uses Gamma(shape=`0.54`, scale=`20`) distribution
  - Visit check interval: `1/3` day
- **Assignment:** Added to `icuContext` with `icu=true`

Ward Therapists

All three types (RT, PT, OT) currently have staffing ratios set to `0.1` (default), so: - **Respiratory Therapists (RT):** hospitalCapacity × `0.1` = `12` RTs - **Physical Therapists (PT):** hospitalCapacity × `0.1` = `12` PTs - **Occupational Therapists (OT):** hospitalCapacity × `0.1` = `12` OTs

Key difference for therapists: - They visit patients from specific "needs" lists: `hospital.patientsNeedingRt/Pt/Ot` - Visit pattern: Gamma(shape=`0.62`, scale=`61.7`) distribution

Simulation Execution - The Event Loop

Once initialization completes, the Repast Simphony scheduler begins executing events. The simulation runs for `365` ticks (days).

Recurring Scheduled Events

1. Builder.perShiftOperations() - Every 0.5 Ticks

When: Runs at ticks `0.5`, `1.0`, `1.5`, `2.0`, etc. (twice per day, representing shift changes)

What it does: - Calls `hospital.setPatientNurseAssignments()` to reassign nurses to patients

Nurse Assignment Algorithm:

1. **Clear all existing nurse-patient assignments**
   - Removes all edges in the hospital network between nurses and patients
   - Doctor-patient assignments remain intact
2. **For each patient in the hospital:**
   a. **Get appropriate nurse pool**
      - If patient is in Ward: get all ward nurses from `wardContext`
      - If patient is in ICU: get all ICU nurses from `icuContext`
   b. **Sort nurses by current workload**
      - Nurses are sorted by their network degree (number of assigned patients)
      - This implements load balancing
   c. **Find nurses with minimum workload**
      - Find the minimum degree among available nurses
      - Create list of all nurses tied for minimum degree
   d. **Assign two nurses to the patient**
      - If `2`+ nurses tied for minimum: randomly select `2`
      - If only `1` nurse at minimum: assign that nurse, then find next-lowest workload nurse
      - Creates edges in `hospitalnet` connecting nurses to patient
      - Ensures each patient has `2` nurse assignments (when possible)

2. Builder.daily() - Every 1.0 Tick

When: Runs at ticks `1.0`, `2.0`, `3.0`, etc. (once per day)

What it does: - Calls `hospital.resetTherapyNeeds()` to refresh therapy assignment lists

Therapy Needs Reset:

1. **Clear existing therapy lists**
   - `patientsNeedingOt.clear()`

- `patientsNeedingPt.clear()`
- `patientsNeedingRt.clear()`

2. **Rebuild therapy lists from current patient population**
   - Loop through all current patients
   - If `patient.needsOt` is true, add to `patientsNeedingOt`
   - If `patient.needsPt` is true, add to `patientsNeedingPt`
   - If `patient.needsRt` is true, add to `patientsNeedingRt`

Why daily reset? Ensures therapy assignment lists stay synchronized with current patient population, removing discharged patients and adding newly transferred patients.

Stochastic Event Processes

In addition to the regular scheduled methods, several stochastic processes fire at random intervals:

3. Admission Process

Frequency: Exponential(`0.05`) distribution - mean of `0.05` days (~`72` minutes) between admissions

Process flow:

1. **Fire event**
   - Increments total admissions counter
   - Calls `hospital.createAndAdmitPatient()`
   - Reschedules itself for next admission

2. **Create and Admit Patient**

   Capacity check
   - Only admit if `patients.size() < bedCount` (`120`)

   Patient creation
   - Create new `Patient()` object
   - Patient gets unique `agentId` from Agent base class
   - Patient gets new `AgentDisease` object tracking disease state

   ICU vs Ward admission
   - Random draw: probability `icuAdmitProbability` (`0.15`)

- Additional check for ICU: ICU must have available beds (`inIcu.size() < icuBedCount`)
- **Default:** `85%` of admissions go to Ward

If admitted to Ward:

a. **Disease importation check**
   - Probability `admitImportationInfectionProbability` (`0.01`)
   - If true: patient starts with INFECTED disease state
   - Mark as imported, record infection date, start disease process

b. **Update patient attributes**
   - Add to tracking lists: `patients`, `notInIcu`
   - Set location: `admitLocation="Ward"`, `currentLocation="Ward"`
   - Set admission time to current tick
   - Mark `icuAdmit=false` attribute

c. **Schedule discharge**
   - `nonIcuDischarger.scheduleDischarge(p)` creates one-time event
   - Time drawn from LogNormal(scale=`0.768`, shape=`1.253`)
   - Schedules call to `hospital.dischargePatient(p)`

d. **Assign therapy needs**
   - Randomly assign based on probabilities:
     - `needsOt`: probability `needsOt` (`0.1`)
     - `needsRt`: probability `needsRt` (`0.1`)
     - `needsPt`: probability `needsPt` (`0.1`)

e. **Add to ward context**
   - Add patient to `wardContext` for ward-specific operations

f. **Record admission**
   - Append to `admissionData`: patientId, admitTime, icuAdmit=false, importation status

If admitted to ICU:

Process differs from Ward admission in these ways:

a. **Disease importation**
   - Uses `admitImportationInfectionProbabilityICU` (`0.01`) - same probability as ward

b. **Tracking lists and location**
   - Added to `inIcu` list instead of `notInIcu`
   - Location set to "ICU" instead of "Ward"
   - Mark `icuAdmit=true` attribute

c. **Discharge schedule**
   - Uses `icuDischarger` with different distribution: LogNormal(scale=`0.820`, shape=`0.916`)

d. **Additional: Schedule potential transfer**
   - **Key difference:** ICU patients get a transfer event scheduled
   - `transferer.scheduleTransfer(p)` creates one-time event
   - Time drawn from LogNormal(scale=`1.0`, shape=`0.5`)
   - Schedules call to `hospital.transferPatient(p)`
   - Note: Both transfer and discharge are scheduled; whichever fires first will execute

e. **Therapy needs**
   - Different probabilities than ward:
     - `needsOt`: probability `needsOtIcu` (`0.1`) - same as ward
     - `needsRt`: probability `needsRtIcu` (`1.0`) - **ALL ICU patients need RT**
     - `needsPt`: probability `needsPtIcu` (`0.1`) - same as ward

f. **Context assignment**
   - Added to `icuContext` instead of `wardContext`

Add to therapy lists if needed

- If patient needs OT: add to `patientsNeedingOt`
- If patient needs PT: add to `patientsNeedingPt`
- If patient needs RT: add to `patientsNeedingRt`

Assign doctor

- Calls `setPatientDoctorAssignments(p)`
- Uses similar load-balancing algorithm as nurse assignment
- Finds doctor (from appropriate context) with fewest current patients
- Adds edge in `hospitalnet` connecting doctor to patient
- Doctor assignment persists until patient discharge (unlike nurses who reassign every shift)

4. PatientVisit Process

One process instance per HCW - Each doctor, nurse, and therapist has their own visit process

Frequency: Varies by HCW type and location, using Gamma distributions

Process flow:

1. **Start**

- Calculate next event time from distribution
- Schedule one-time event to call `fire()`

2. **Fire**

   - Calls `hcw.makeAVisit()`
   - Reschedules itself with `start()` - creating continuous loop

3. **Next Event Time Calculation**

   - Sample from HCW-specific Gamma distribution (in minutes)
   - Add fixed room visit duration: `6.6` minutes
   - Convert to days: multiply by `TimeUtils.MINUTE` constant
   - Return: currentTime + elapsedTime

4. **Make a Visit** - Implementation varies by HCW type

   The base class `HealthCareWorker.makeAVisit()` is overridden in each subclass:

   Doctors:

   - Get the `hospitalnet` network from hospital
   - Check if doctor has any assigned patients (`getDegree(this) > 0`)
   - If yes: select random patient from those connected to this doctor (`getRandomAdjacent(this)`)
   - **Key point:** Doctors only visit their ASSIGNED patients (assigned at patient admission)
   - Check for transmission
   - Record visit to `hospital.visitData`

   Nurses:

   - Identical logic to doctors
   - Get `hospitalnet` network
   - Check if nurse has assigned patients
   - Select random patient from those connected to this nurse
   - **Key point:** Nurses only visit their ASSIGNED patients (assigned every 0.5 ticks at shift change)
   - Check for transmission
   - Record visit

   Therapists (RT/PT/OT):

   - Check if `needsArray` has any patients (size > 0)
   - `needsArray` is set to `hospital.patientsNeedingOt/Pt/Rt` for each therapist type
   - Select random patient from the needs list
   - **REMOVE patient from needs list** after selection

- **Key point:** Each patient is visited at most once per day by each therapist type
- The needs list is repopulated daily by `Builder.daily()` → `hospital.resetTherapyNeeds()`
- Check for transmission
- Record visit

ICU Respiratory Therapists (IcuRt):

- Get list of all current ICU patients (`hospital.inIcu`)
- If list is empty, return (no visit)
- Select random patient from ALL ICU patients (not using assignment network)
- **Key point:** ICU RTs can visit any ICU patient, not limited to assigned patients
- Check for transmission
- Record visit

Transmission check:

- Logic for all HCW types:
  - If HCW not contaminated AND patient not colonized/infected: no transmission possible
  - If HCW contaminated AND patient clean: potential HCW→patient transmission
  - If HCW clean AND patient colonized/infected: potential patient→HCW transmission
  - If both contaminated/infected: no additional transmission
- Note: Actual transmission logic is not fully implemented

Visit recording - All HCW types append to `hospital.visitData`:

- Format: `hcwId, hcwType, hcwDiseaseState, patientId, patientDiseaseState, patientLocation, visitTime`
- Creates comprehensive audit trail of all visits

5. Discharge Process

Scheduled individually for each patient upon admission

Timing: - ICU patients: LogNormal(scale=`0.820`, shape=`0.916`) - Ward patients: LogNormal(scale=`0.768`, shape=`1.253`)

Process flow:

1. **Schedule discharge**

   - Called when patient is admitted
   - Sample discharge time from distribution

- Schedule one-time event: `hospital.dischargePatient(p)`

2. **Discharge patient**

   Collect timing data

   - Record admit time and current time for length-of-stay calculation

   Remove from hospital

   - Remove from main context
   - Remove from `patients` list
   - Remove from `inIcu` or `notInIcu` list

   Create discharge record

   - Create `DischargedPatient` object with:
     - agentId
     - admitTime
     - dischargeTime (current tick)
     - died flag (always false in current implementation)
     - icuAdmit flag (whether admitted to ICU)
     - transferTime (if transferred from ICU)
     - admitLocation ("ICU" or "Ward")
     - dischargeLocation (current location at discharge)

   Store record

   - Add to `dischargedPatients` list
   - Add to context (for potential Repast probes/data collection)

6. Transfer Process

Scheduled only for ICU-admitted patients

Timing: LogNormal(scale=`1.0`, shape=`0.5`)

Process flow:

1. **Schedule transfer**

   - Called when patient is admitted to ICU
   - Sample transfer time from distribution
   - Schedule one-time event: `hospital.transferPatient(p)`

2. **Transfer patient**

   Update location lists

- Remove from `inIcu` list
- Add to `notInIcu` list

Update patient state

- Set `currentLocation = "Ward"`
- Record `transferTime` (current tick)

Note: Patient's original discharge event (scheduled upon ICU admission) is NOT cancelled. If the ICU discharge event fires after transfer, it will still discharge the patient from their current location (Ward).

Model Termination and Output

Builder.endOfRun() - At Tick 365

When: Single execution at tick `365`

What it does:

1. **Write output files**
   - Calls `writeSingleRunFiles()`
2. **End simulation**
   - Calls `RunEnvironment.getInstance().endRun()`
   - Stops the scheduler

Output File Generation

The model creates three output files:

1. discharged_patients.txt

Format: CSV with header

```
agentId,admitTime,dischargeTime,icuAdmit,transferTime,admitLocation,dischargeLocation
```

Content: - One row per discharged patient - `transferTime` is `0.0` if patient was never transferred - `icuAdmit` indicates if patient was initially admitted to ICU - Enables length-of-stay analysis, transfer analysis, ICU vs ward utilization

Generation: - Loops through `hospital.getDischargedPatients()` list - Calls `DischargedPatient.toString()` for each record

2. visit_data.txt

Format: CSV with header

```
hcwId,hcwType,hcwDiseaseState,patientId,patientDiseaseState,patientLocation,visitTime
```

Content: - One row per HCW visit to a patient - Records disease states of both HCW and patient at time of visit - Includes visit location (ICU or Ward) and exact time - Enables visit pattern analysis, HCW workload analysis, transmission chain reconstruction

Generation: - Writes the accumulated `hospital.visitData` StringBuffer - This buffer is appended to throughout the simulation

3. admission_data.txt

Format: CSV with header

```
patientId,admitTime,icuAdmit,importation
```

Content: - One row per patient admission - `icuAdmit`: true if admitted to ICU, false if admitted to Ward - `importation`: true if patient arrived with imported MRSA infection - Enables admission pattern analysis and importation tracking

Generation: - Writes the accumulated `hospital.admissionData` StringBuffer - This buffer is appended to during admission

Key Model Parameters

All parameters are defined in Builder.java and accessible via Repast GUI:

Hospital Structure

- `hospitalCapacity` = `120` beds
- `icuCapacity` = `20` beds
- Ward capacity = `100` beds (derived)

Admission-Discharge-Transfer (ADT)

- `admissionsRate` = `0.05` (mean inter-arrival time in days)
- Ward discharge: LogNormal(scale=`0.768`, shape=`1.253`)
- ICU discharge: LogNormal(scale=`0.820`, shape=`0.916`)
- `icuAdmitProbability` = `0.15` (`15%` of admissions go to ICU)
- `icuTransferProbability` = `0.1` (affects transfer timing, not probability)
- ICU→Ward transfer: LogNormal(scale=`1.0`, shape=`0.5`)

Therapy Needs

- `needsRt` = `0.1` (ward), `needsRtIcu` = `1.0` (ICU) - respiratory therapy
- `needsPt` = `0.1` (both ICU and ward) - physical therapy
- `needsOt` = `0.1` (both ICU and ward) - occupational therapy

Staffing Ratios (HCWs per bed)

- Ward nurses: `0.2`
- ICU nurses: `0.5`
- Ward doctors: `0.2`
- ICU doctors: `0.3`
- ICU respiratory therapists: `0.1`
- Ward therapists (RT/PT/OT): `0.1` each

HCW Visit Patterns (Gamma distributions, shape and scale parameters)

All times in minutes, converted to days via `TimeUtils.MINUTE`

Ward: - Nurse visits: Gamma(`0.54`, `55.1`) + `6.6` min room time - Doctor visits: Gamma(`0.52`, `90.7`) + `6.6` min room time - Therapist visits: Gamma(`0.62`, `61.7`) + `6.6` min room time

ICU: - Nurse visits: Gamma(`0.54`, `20`) + `6.6` min room time - Doctor visits: Gamma(`0.52`, `35.3`) + `6.6` min room time

Disease Parameters

- `admitImportationInfectionProbability` = `0.01` (ward)
- `admitImportationInfectionProbabilityICU` = `0.01` (ICU)
- `hhAdherenceBase` = `0.5` (hand hygiene compliance for all HCW types)
- `ppeAdherenceIfCp` = `0.5` (PPE/glove compliance)

Disease Transmission Model (Partially Implemented)

The model includes disease tracking infrastructure but transmission logic is not fully implemented:

Disease States (DiseaseStates.java)

- SUSCEPTIBLE - not colonized or infected
- COLONIZED - carrying MRSA but asymptomatic
- INFECTED - active MRSA infection

Current Implementation

- Patients can be imported with INFECTED state on admission
- HCWs can be marked as contaminated
- Visit checking logic exists in HealthCareWorker class
- **However:** Actual transmission during visits is not fully implemented
  - `checkTransmissionToPatient()` returns true if HCW performs hand hygiene
  - `checkTransmissionToHcw()` returns false
  - `checkTransmission()` is empty

Data Collection for Transmission Analysis

Even though transmission is not implemented, the model records: - HCW contamination status at each visit - Patient disease state at each visit - Visit timing and location This allows post-hoc transmission analysis or model extension.

Summary of Model Dynamics

The model simulates a continuous flow hospital over `365` days:

1. **Patients arrive** stochastically (~`20` admissions per day on average)
2. **Admitted to ICU (`15%`) or Ward (`85%`)** based on probability and bed availability
3. **Assigned to doctors** (once, at admission) using load-balancing
4. **Assigned to nurses** (twice daily, at shift changes) using load-balancing
5. **Visited by HCWs** throughout their stay according to stochastic schedules
6. **ICU patients may transfer** to ward after some time
7. **All patients eventually discharge** after length-of-stay drawn from location-specific distributions
8. **Visit data** and **discharge data** accumulate throughout simulation

9. **Output files** written at day `365` for analysis

The model generates realistic admission-discharge-transfer patterns and HCW workload distributions that can be analyzed using the R/Quarto scripts in the `R/` directory.