РАЗРАБОТКА МНОГОАГЕНТНОЙ МОДЕЛИ В ANYLOGIC (ЛАБОРАТОРНАЯ РАБОТА № 3)

Цель работы

Изучить методологию агентного моделирования. Приобрести практические навыки работы с системой AnyLogic при построении агентных моделей. Разработать поведенческую логику агента «Пациент» в мультиагентной системе «Больница».

1. Описание предметной области

Мультиагентная система «Больница» моделирует поток пациентов через различные этапы медицинского обслуживания. Целью модели является исследование загрузки персонала, времени ожидания пациентов, эффективности триажа и лечения, а также анализ отказов в обслуживании и госпитализаций.

1.1. Агенты системы

- Patient (Пациент) – Основной источник нагрузки. Имеет характеристики: тяжесть состояния, терпение, потребность в лечении и анализах.
- Doctor (Врач) – Обслуживает пациентов, ставит диагноз, принимает решение о лечении, анализах или госпитализации.
- Nurse (Медсестра) – Проводит первичный триаж, назначает приоритет обслуживания.
- Receptionist (Регистратор) – Оформляет пациента, заводит медицинскую карту.
- LabUnit (Лаборатория) – Выполняет анализы и диагностику, имеет очередь.
- Hospital/Main (Среда) – Управляет популяциями агентов, очередями, статистикой и правилами маршрутизации.

1.2. Взаимодействия агентов

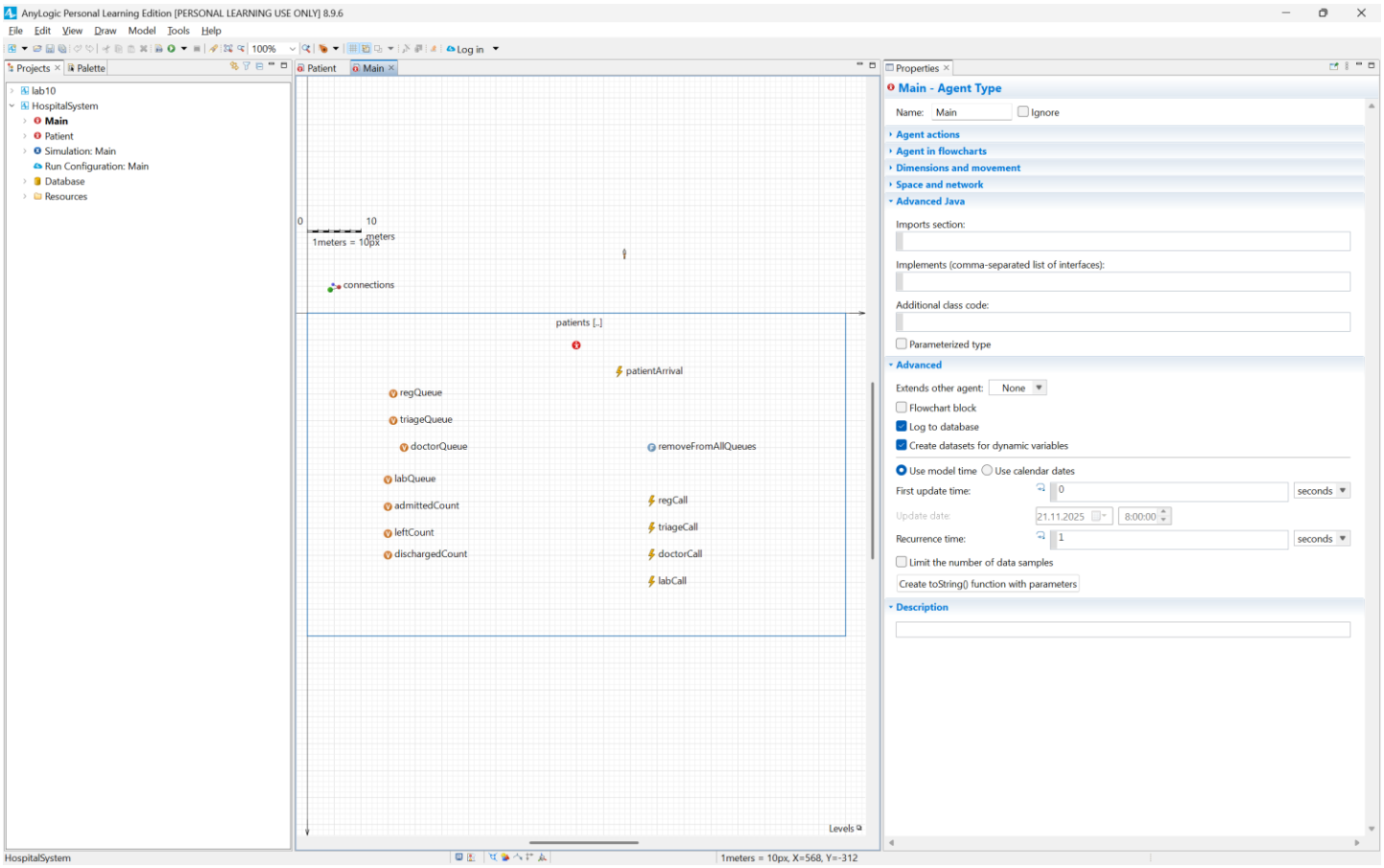
- Пациент отправляет запрос на обслуживание регистратору/триажу/врачу через сообщения
- Врач/медсестра вызывает следующего пациента из очереди
- Лаборатория возвращает результат анализа пациенту/врачу
- Среда (Main) управляет очередями и выдает следующего пациента по приоритету

2. Реализация поведенческой логики агента Patient

Поведение агента «Пациент» реализовано с помощью диаграммы состояний (Statechart), параметров, переменных и механизма обмена сообщениями.

2.1. Создание класса агента

1. Projects → New → Agent Type
2. Имя класса: Patient
3. Создана популяция агентов patients в классе Main



2.2. Параметры агента

Параметры задают индивидуальные характеристики каждого пациента:

Параметр	Тип	Значение по умолчанию	Описание
severity	int	uniform_discr(1, 5)	Тяжесть состояния (1-5)
patienceTime	double	triangular(0.5, 2, 6)	Время ожидания до ухода (ч)
serviceNeed	double	triangular(5, 15, 40)	Длительность лечения (мин)
needsLab	boolean	randomTrue(0.3)	Нужны ли анализы
admissionProb	double	0.1 + 0.15*severity	Вероятность

			госпитализации

2.3. Переменные агента

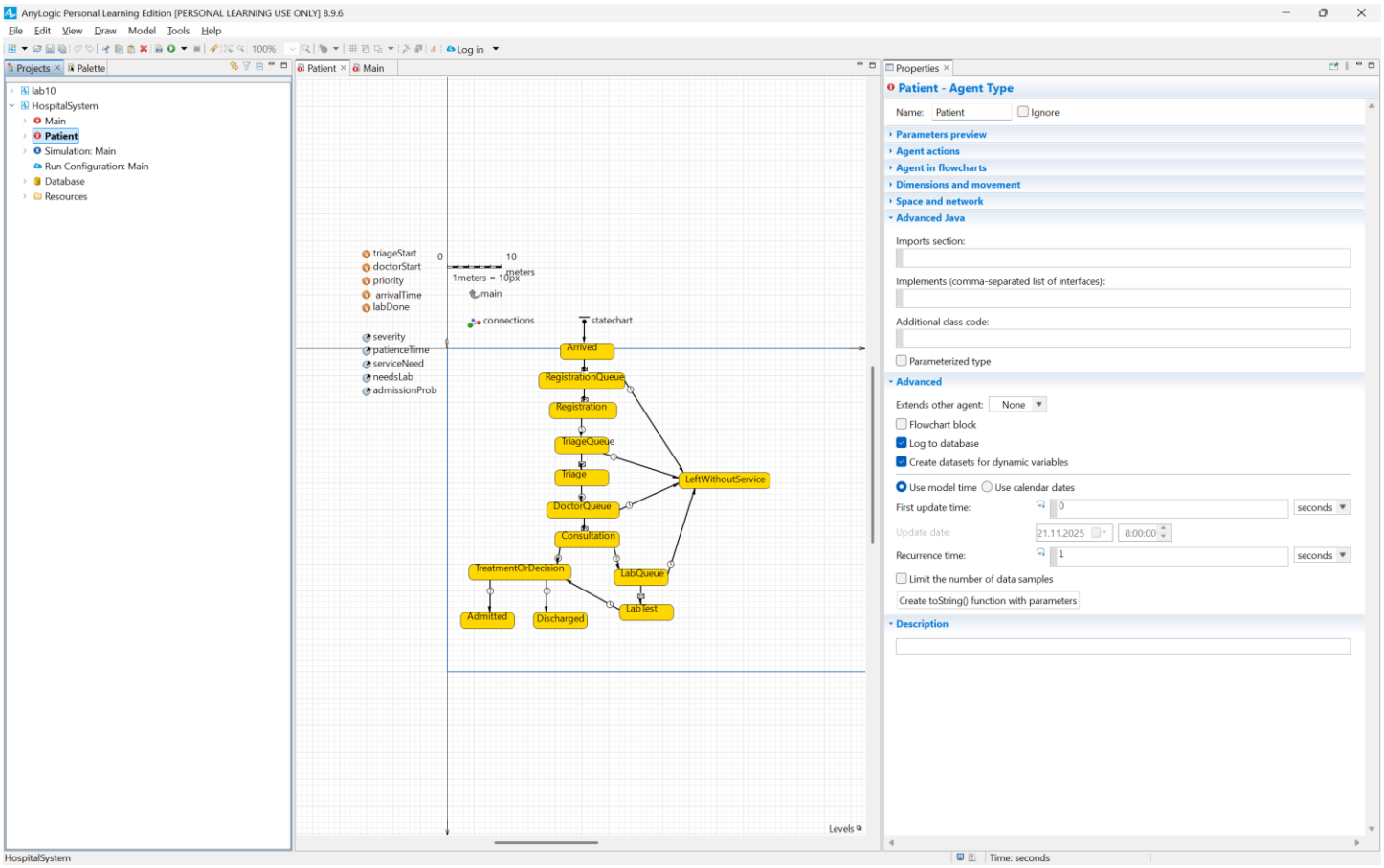
Переменные хранят состояние агента в процессе моделирования:

Переменная	Тип	Начальное значение	Описание
arrivalTime	double	time()	Время прибытия в больницу
triageStart	double	0	Время начала триажа
doctorStart	double	0	Время начала приема врача
priority	int	3	Приоритет (назначается триажем)
labDone	boolean	false	Завершены ли анализы

2.4. Диаграмма состояний (Statechart)

Поведение пациента моделируется с помощью диаграммы состояний flow, которая включает следующие состояния:

- Arrived – Пациент прибыл в больницу
- RegistrationQueue – Ожидание в очереди на регистрацию
- Registration – Процесс регистрации
- TriageQueue – Ожидание триажа
- Triage – Первичный осмотр медсестрой
- DoctorQueue – Ожидание приема врача
- Consultation – Консультация у врача
- LabQueue – Ожидание анализов (если требуются)
- LabTest – Проведение лабораторных анализов
- TreatmentOrDecision – Принятие решения о дальнейших действиях
- Admitted – Госпитализирован
- Discharged – Выписан
- LeftWithoutService – Покинул больницу из-за долгого ожидания


AnyLogic Personal Learning Edition [PERSONAL LEARNING USE ONLY] 8.9.6
File Edit View Draw Model Tools Help
Log in
Projects
Palette
lab10
HospitalSystem
Main
Patient
Simulation: Main
Run Configuration: Main
Database
Resources
Patient
Main
triageStart
doctorStart
priority
arrivalTime
labDone
severity
patienceTime
serviceNeed
needsLab
admissionProb
0
10
meters
1meters = 10px
main
connections
statechart
Arrived
RegistrationQueue
Registration
TriageQueue
Triage
DoctorQueue
Consultation
TreatmentOrDecision
LabQueue
LabTest
Admitted
Discharged
LeftWithoutService
Levels
Properties
Patient - Agent Type
Name: Patient
Ignore
Parameters preview
Agent actions
Agent in flowcharts
Dimensions and movement
Space and network
Advanced Java
Imports section:
Implements (comma-separated list of interfaces):
Additional class code:
Parameterized type
Advanced
Extends other agent: None
Flowchart block
Log to database
Create datasets for dynamic variables
Use model time
Use calendar dates
First update time:
0
seconds
Update date:
21.11.2025
8:00:00
Recurrence time:
1
seconds
Limit the number of data samples
Create toString() function with parameters
Description
HospitalSystem
Time: seconds

2.5. Описание ключевых переходов

Переход: Arrived → RegistrationQueue

```
Происходит: Немедленно
Действие:
get_Main().regQueue.add(this);
send("needReg", get_Main());
```

Пациент добавляется в очередь на регистрацию и отправляет сообщение в среду.

Переход: RegistrationQueue → Registration

```
Происходит: При получении сообщения
Тип сообщения: String
Условие: message equals "callReg"
```

Срабатывает, когда регистратор вызывает пациента из очереди.

Переход: Registration → TriageQueue

```
Происходит: По таймауту
Таймаут: uniform(1, 3) минуты
Действие:
get_Main().triageQueue.add(this);
send("needTriage", get_Main());
```

После регистрации пациент переходит в очередь на триаж.

Переход: Triage → DoctorQueue

```
Происходит: По таймауту
Таймаут: uniform(2, 5) минут
Действие:
priority = (severity >= 4) ? 1 : (severity == 3 ? 2 : 3);
get_Main().doctorQueue.add(this);
send("needDoctor", get_Main());
```

Медсестра назначает приоритет на основе тяжести состояния и направляет к врачу.

[Место для скриншота: Свойства перехода Triage → DoctorQueue]

Переход: Consultation → LabQueue (условный)

```
Происходит: По таймауту
Таймаут: serviceNeed
Условие: needsLab == true
Действие:
get_Main().labQueue.add(this);
send("needLab", get_Main());
```

Если пациенту нужны анализы, он направляется в лабораторию.

Переход: TreatmentOrDecision → Admitted / Discharged

```
Переход в Admitted:
Условие: randomTrue(admissionProb)
Действие: get_Main().admittedCount++;

Переход в Discharged:
Условие: !randomTrue(admissionProb)
```

На основе вероятности принимается решение о госпитализации или выписке.

Переходы в LeftWithoutService (из очередей)

```
Происходит: По таймауту
Таймаут: patienceTime
Действие:
get_Main().removeFromAllQueues(this);
get_Main().leftCount++;
```

Если пациент не дождался обслуживания, он покидает больницу.

2.6. Обработка сообщений

В свойствах класса Patient в поле «Действие при получении сообщения» (On message) указано:

```
flow.receiveMessage(msg);
```

Это обеспечивает корректную обработку сообщений, которые инициируют переходы в диаграмме состояний.

3. Логика среды Main

В классе Main реализованы очереди, счетчики статистики и события для вызова пациентов:

3.1. Очереди и переменные

```
ArrayList<Patient> regQueue = new ArrayList<>();
ArrayList<Patient> triageQueue = new ArrayList<>();
ArrayList<Patient> doctorQueue = new ArrayList<>();
ArrayList<Patient> labQueue = new ArrayList<>();

int admittedCount = 0;
int leftCount = 0;
int dischargedCount = 0;
```

3.2. События вызова пациентов

Созданы циклические события для вызова пациентов из очередей

Event: regCall

```
Таймаут: uniform(0.5, 1) минуты
Действие:
if (!regQueue.isEmpty()) {
    Patient p = regQueue.remove(0);
    p.send("callReg", p);
}
```

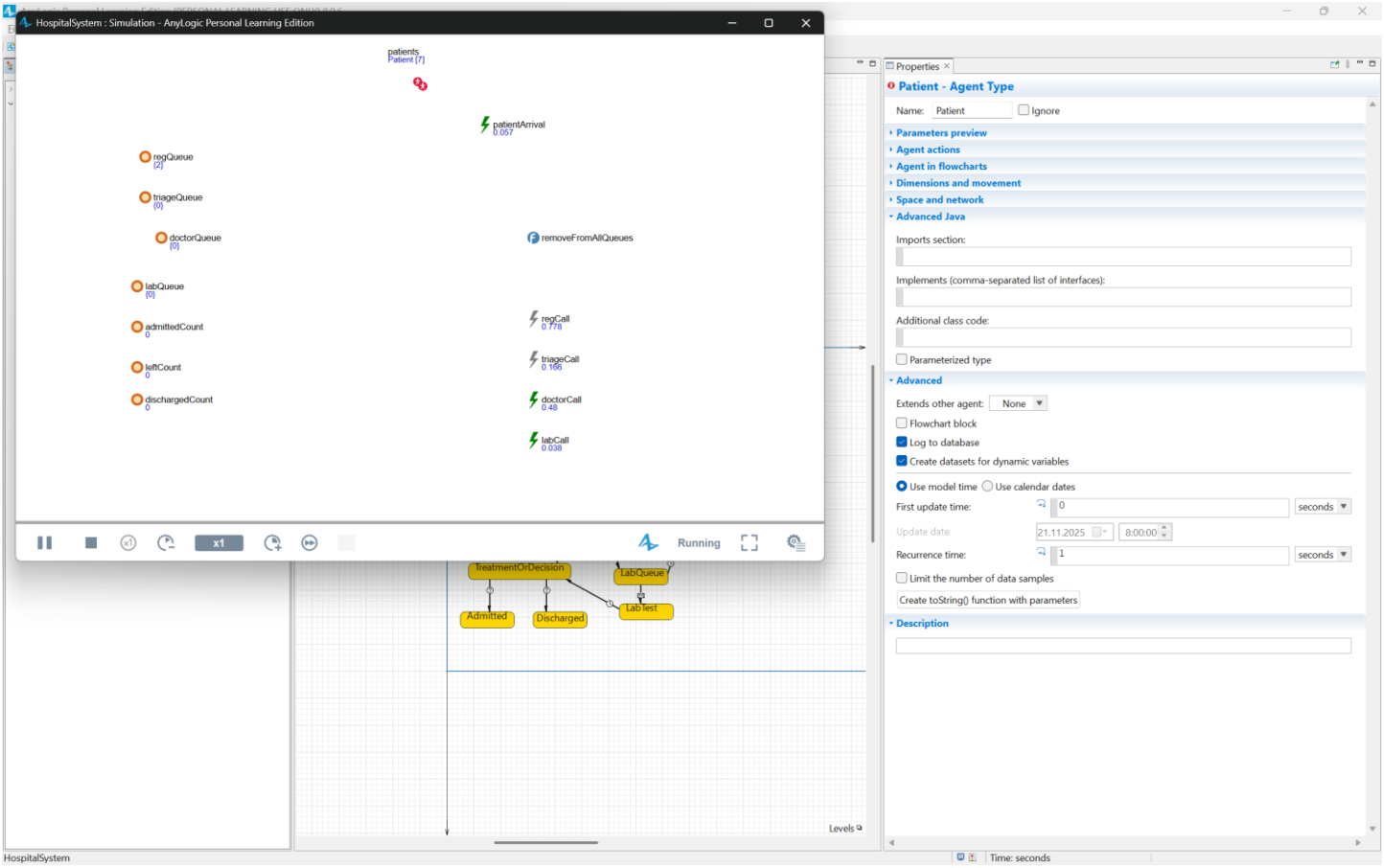
Event: doctorCall

```
Таймаут: uniform(1, 2) минуты
Действие (выбор по приоритету):
if (!doctorQueue.isEmpty()) {
    Patient best = doctorQueue.get(0);
    for (Patient p : doctorQueue)
        if (p.priority < best.priority) best = p;
    doctorQueue.remove(best);
    best.send("callDoctor", best);
}
```

4. Визуализация и статистика

Для анализа работы системы добавлены следующие элементы визуализации:

- Текстовые метки с отображением размеров очередей
- Счетчики госпитализированных, выписанных и ушедших пациентов
- Временной график (Time Plot) для отслеживания динамики очередей
- График загрузки персонала
- Среднее время пребывания пациента в системе



Выводы

1. Успешно реализована поведенческая логика агента «Пациент» в мультиагентной системе «Больница» с использованием AnyLogic.

2. Агент моделирует полный цикл обслуживания пациента: от прибытия и регистрации до выписки или госпитализации

3. Использованы ключевые элементы агентного моделирования: параметры для индивидуальных характеристик, переменные для хранения состояния, диаграмма состояний для описания поведения.

4. Реализован механизм обмена сообщениями между агентами через send() и receiveMessage(), что обеспечивает децентрализованное управление.

5. Модель позволяет анализировать загрузку системы, время ожидания пациентов, эффективность триажа и выявлять узкие места в процессе обслуживания.

6. Визуализация с изменением цвета агентов наглядно демонстрирует текущее состояние каждого пациента в системе.

7. Модель может быть расширена добавлением других агентов (врачей, медсестер, лаборантов) с собственной логикой поведения для более полного моделирования работы больницы.