Notarzt-Simulation

Erzeugt von Doxygen 1.8.12

Inhaltsverzeichnis

Kapitel 1

Hierarchie-Verzeichnis

1.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

Kapitel 2

Klassen-Verzeichnis

2.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

Kapitel 3

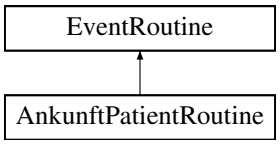
Klassen-Dokumentation

3.1 AnkunftPatientRoutine Klassenreferenz

Routine für die Ankunft des Notarztes beim Patienten.

`#include <AnkunftPatientRoutine.h>`

Klassendiagramm für AnkunftPatientRoutine:



Öffentliche Methoden

- AnkunftPatientRoutine (Notarzt ∗arzt, EventList ∗eList, NotfallWarteschlange ∗warteschlange)
  Konstruktor.
- void execute (Event ∗event)
  Spezifizierung der Zustandsänderungen.

3.1.1 Ausführliche Beschreibung

Routine für die Ankunft des Notarztes beim Patienten.

Diese Routine beschreibt die Zustandsänderungen in der Notarzt-Simulation wenn der Notarzt beim Patienten ankommt.

3.1.2 Beschreibung der Konstruktoren und Destruktoren

3.1.2.1 AnkunftPatientRoutine::AnkunftPatientRoutine (Notarzt ∗ *arzt,* EventList ∗ *eList,* NotfallWarteschlange ∗ *warteschlange*)

Konstruktor.

Abhängigkeiten werden injiziert und eine spezielle EventRoutine mit dem Event-Typ ANKUNF_PATIENT wird erstellt.

3.1.3 Dokumentation der Elementfunktionen

3.1.3.1 void AnkunftPatientRoutine::execute (Event * *event*) [virtual]

Spezifizierung der Zustandsänderungen.

Die Ankunft des Notarztes beim Patienten verändert den Zustand des Notarztes und entfernt den zugehörigen Notfall, der nun behandelt wird, aus der Notfall-Warteschlange. Ein neues Event für das Ende der Behandlung wird der Simulation hinzugefügt, der Zeitpunkt dieses Events entsprechend berechnet.

Implementiert EventRoutine.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

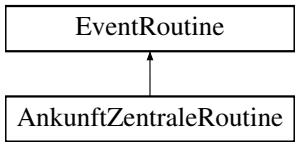
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/AnkunftPatientRoutine.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/AnkunftPatientRoutine.cpp

3.2 AnkunftZentraleRoutine Klassenreferenz

Routine für die Ankunft des Notarztes in der Zentrale.

`#include <AnkunftZentraleRoutine.h>`

Klassendiagramm für AnkunftZentraleRoutine:



Öffentliche Methoden

- AnkunftZentraleRoutine (Notarzt *notarzt, EventList *eventList, NotfallWarteschlange *notfallWarteschlange)
  Konstruktor.
- void execute (Event *event)
  Spezifizierung der Zustandsänderungen.

3.2.1 Ausführliche Beschreibung

Routine für die Ankunft des Notarztes in der Zentrale.

Diese Routine beschreibt die Zustandsänderungen in der Notarzt-Simulation wenn der Notarzt in der Zentrale ankommt.

3.2.2 Beschreibung der Konstruktoren und Destruktoren

3.2.2.1 AnkunftZentraleRoutine::AnkunftZentraleRoutine (Notarzt * *notarzt,* EventList * *eventList,* NotfallWarteschlange * *notfallWarteschlange*)

Konstruktor.

Abhängigkeiten werden injiziert eine spezielle EventRoutine mit dem Event-Typ ANKUNFT_ZENTRALE wird erstellt.

3.2.3 Dokumentation der Elementfunktionen

3.2.3.1 void AnkunftZentraleRoutine::execute (Event * *event*) [virtual]

Spezifizierung der Zustandsänderungen.

Die Ankunft des Notarztes in der Zentrale verändert den Zustand des Notarztes auf wartend und erzeugt bei bei Bedarf ein neues Event. Das neue Event vom Typ Abfahrt_Patient ist abhängig davon, ob bereits ein weiterer Notfall vorliegt oder nicht.

Implementiert EventRoutine.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

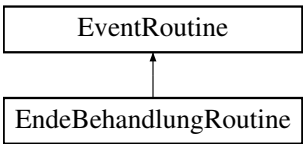
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/AnkunftZentraleRoutine.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/AnkunftZentraleRoutine.cpp

3.3 EndeBehandlungRoutine Klassenreferenz

Routine für das Ende einer Behandlung eines Notfalls.

`#include <EndeBehandlungRoutine.h>`

Klassendiagramm für EndeBehandlungRoutine:



Öffentliche Methoden

- EndeBehandlungRoutine (Notarzt *notarzt, EventList *eventList, NotfallWarteschlange *notfallWarteschlange)

  Konstruktor.
- void execute (Event *event)

  Spezifizierung der Zustandsänderungen.

3.3.1 Ausführliche Beschreibung

Routine für das Ende einer Behandlung eines Notfalls.

Diese Routine beschreibt die Zustandsänderungen in der Notarzt-Simulation wenn die Behandlung eines Notfalls abgeschlossen ist.

3.3.2 Beschreibung der Konstruktoren und Destruktoren

3.3.2.1 EndeBehandlungRoutine::EndeBehandlungRoutine (Notarzt * *notarzt,* EventList * *eventList,* NotfallWarteschlange * *notfallWarteschlange*)

Konstruktor.

Abhängigkeiten werden injiziert und eine spezielle EventRoutine mit dem Event-Typ ENDE_BEHANDLUNG wird erstellt.

3.3.3 Dokumentation der Elementfunktionen

3.3.3.1 void EndeBehandlungRoutine::execute (Event * *event*) [virtual]

Spezifizierung der Zustandsänderungen.

Das Ende der Behandlung eines Notfalls bzw. Patienten verändert den Zustand des Notarztes. Dabei muss zuerst entschieden werden, ob der Notarzt gleich zu einem weiterem Notfall fahren muss oder zurück in die Zentrale fährt. Einentsprechendes Event entweder vom Typ ABFAHRT_ZU_PATIENT oder ABFAHRT_ZU_ZENTRALE wird erstellt. Dieses abhängige Event (abhängig von ENDE_BEHANDLUNG) bekommt als Zeitpunkt der Erscheinung den selben Zeitpunkt wie das Event vom Typ ENDE_BEHANDLUNG, welches diese Routine ausgelöst hat. Es passiert also in der Simulation augenblicklich ohne Zeitverzögerung.

Implementiert EventRoutine.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

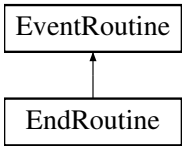
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/EndeBehandlungRoutine.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/EndeBehandlungRoutine.cpp

3.4 EndRoutine Klassenreferenz

Routine für das Ende der Simulation.

```
#include <EndRoutine.h>
```

Klassendiagramm für EndRoutine:



Öffentliche Methoden

- EndRoutine ()

  Konstruktor Erstellt eine spzeielle EventRoutine mit dem Event-Typ ENDE.
- void execute (Event ∗event)

  Spezifizierung der Zustandsänderungen.

3.4.1 Ausführliche Beschreibung

Routine für das Ende der Simulation.

Diese Routine verändert keine Objekte in der Simulation sondern dient der Kontrolle der Simulationsschleife. Die Simulationsschleife bricht ab (und beendet somit die Simulation) sobald diese Routine ausgeführt werden soll.

3.4.2 Dokumentation der Elementfunktionen

3.4.2.1 void EndRoutine::execute (Event ∗ *event*) [virtual]

Spezifizierung der Zustandsänderungen.

Keine Änderung an Objekten der Simulation.

Implementiert EventRoutine.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/EndRoutine.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/EndRoutine.cpp

3.5 Event Klassenreferenz

Klasse für die Erstellung von Ereignissen.

```
#include <Event.h>
```

Öffentliche Methoden

- Event (int executionTime, EventType type)

  Konstruktor.
- int getExecutionTime ()

  Rückgabe der Ausführungszeit.
- EventType getType ()

  Rückgabe des Event-Types.

3.5.1 Ausführliche Beschreibung

Klasse für die Erstellung von Ereignissen.

Implementierung eines allgemeinen Ereignises im Rahmen der diskreten, ereignisgesteureten Simulation.↩ Implementierung eines Ereignisses (engl. Event) im Rahmen der diskreten, ereignisgesteuerten Simulation.

3.5.2 Beschreibung der Konstruktoren und Destruktoren

3.5.2.1 Event::Event (int *exeTime,* EventType *t*)

Konstruktor.

Erstellt ein Event-Objekt zur Verwendung innerhalb einer Simulation (bsp. durch das Hinzufügen zur EventList). Die Angabe von Ausführungszeit und Typ sind notwendig, um die Behandlung des Events im Rahmen der Simulationsschleife zu ermöglichen. Üblicherweise sollte es auch eine EventRoutine mit dem entsprechendem Typ existieren.

Implementierung eines allgemeinen Ereignises im Rahmen der diskreten, ereignisgesteureten Simulation.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/Event.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/Event.cpp

3.6 EventList Klassenreferenz

Verwaltet Ereignisse.

`#include <EventList.h>`

Öffentliche Methoden

- EventList ()
  Konstruktor.
- Event ∗ popEvent ()
  Nächstes Ereignis liefern und löschen.
- Event ∗ getNextEvent ()
  Nächstes Ereignis liefern.
- int removeEventByType (EventType type)
  Löscht das nächste Ereignis eines bestimmten Types.
- void addEvent (Event ∗event)
  Ereignis hinzufügen.
- void printList ()
  Drucken der aktuellen Liste auf die Ausgabe.

3.6.1 Ausführliche Beschreibung

Verwaltet Ereignisse.

Ereignisliste (engl. event list) für die diskrete, ereignisgesteuerte Simulation. Die Ereignisliste ist ein Besatndteil jeder ereignisgesteuerten Simulation und defniert das nächste Ereignis der Simulation. Ereignisse können hinzugefügt und wieder entfernt werden. Die Liste sortiert die Ereignisse nach ihrer Ausführungszeit.

3.6.2 Beschreibung der Konstruktoren und Destruktoren

3.6.2.1 EventList::EventList ( )

Konstruktor.

Ereignisliste für die diskrete, eventgesteruerte Simulation.

3.6.3 Dokumentation der Elementfunktionen

3.6.3.1 void EventList::addEvent (Event * *event*)

Ereignis hinzufügen.

Fügt das übergebene Event der Ereignisliste hinzu und sortiert es in die richtige Stelle ein. Die Liste bleibt somit sortiert.

3.6.3.2 Event * EventList::getNextEvent ( )

Nächstes Ereignis liefern.

Liefert jenes Ereignis, dass von der Ausführungszeit her das nächste Ereignis ist. Kein Löschen, d.h. der wiederholte Aufruf liefert das gleiche Ereignis.

3.6.3.3 Event * EventList::popEvent ( )

Nächstes Ereignis liefern und löschen.

Liefert jenes Ereignis, dass von der Ausführungszeit her das nächste Ereignis ist. Das Ereignis wird zugleich von der Liste entfernt.

3.6.3.4 int EventList::removeEventByType (EventType *type*)

Löscht das nächste Ereignis eines bestimmten Types.

Löscht jenes Ereignis aus der Ereignisliste, dass den selben Typ entspricht wie im übergebenem Parameter. Löscht NICHT alle Events dieses Typs, sondern nur jenes von dem Typ, das als nächstes zur Ausführung kommen würde.

Gibt zurück, ob ein Event gelöscht wurde (1) oder nicht (0)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

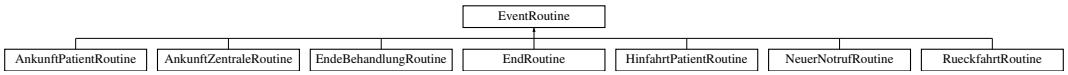
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/EventList.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/EventList.cpp

3.7 EventRoutine Klassenreferenz

Abstrakte Erignis-Routine.

`#include <EventRoutine.h>`

Klassendiagramm für EventRoutine:



Öffentliche Methoden

- EventRoutine (EventType type)
  Konstruktor.
- virtual void execute (Event *event)=0
  Funktionssignatur Ausführung einer Routine.
- EventType getType ()
  Liefert Ereignistyp.

3.7.1 Ausführliche Beschreibung

Abstrakte Erignis-Routine.

Abstrakte Klasse für die minimale Definition einer Ereignisroutine in einer diskreten, eventgesteuerten Simulation.

3.7.2 Beschreibung der Konstruktoren und Destruktoren

3.7.2.1 EventRoutine::EventRoutine (EventType *type*)

Konstruktor.

Eine Routine besitzt immer einen spezifischen Typ, der von der Art der Simulation abhängig ist. Dieser Typ ermöglich der Simulationsschleife, die passende Routine zu einem Event zu finden.

3.7.3 Dokumentation der Elementfunktionen

3.7.3.1 virtual void EventRoutine::execute (Event * *event*) [pure virtual]

Funktionssignatur Ausführung einer Routine.

Abstrakte Funktion, die die Erwartungshaltung an alle implementierten Ereignisroutinen definiert. Das ermöglicht eine allgemeine Definition der Simulationsschleife ohne dass spezielle Wissen über die implementierten Ereignisroutinen notwendig ist.

Muss von jeder spezialisierenden Klasse implementiert werden

Implementiert in HinfahrtPatientRoutine, NeuerNotrufRoutine, EndeBehandlungRoutine, AnkunftPatientRoutine, RueckfahrtRoutine, AnkunftZentraleRoutine und EndRoutine.

3.7.3.2 EventType EventRoutine::getType ( )

Liefert Ereignistyp.

Fertig implementierte Funtkion zur Rückgabe des Eriegnistyps.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

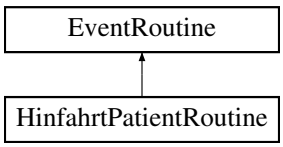
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/EventRoutine.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/EventRoutine.cpp

3.8 HinfahrtPatientRoutine Klassenreferenz

Routine für den Beginn der Hinfahrt zum Patienten/Notfall.

`#include <HinfahrtPatientRoutine.h>`

Klassendiagramm für HinfahrtPatientRoutine:



Öffentliche Methoden

- HinfahrtPatientRoutine (Notarzt ∗notarzt, EventList ∗eventList, NotfallWarteschlange ∗notfallWarteschlange, Zufall ∗randomGenerator)

  Konstruktor.
- void execute (Event ∗event)

  Spezifizierung der Zustandsänderungen.

3.8.1 Ausführliche Beschreibung

Routine für den Beginn der Hinfahrt zum Patienten/Notfall.

Diese Routine beschreibt die Zustandsänderungen in der Notarzt-Simulation wenn der Notarzt den Weg zum Patienten/Notfall einschlägt.

3.8.2 Beschreibung der Konstruktoren und Destruktoren

3.8.2.1 HinfahrtPatientRoutine::HinfahrtPatientRoutine (Notarzt ∗ *notarzt,* EventList ∗ *eventList,* NotfallWarteschlange ∗ *notfallWarteschlange,* Zufall ∗ *randomGenerator*)

Konstruktor.

Abhängigkeiten werden injiziert und eine spezielle EventRoutine mit dem Event-Typ ABFAHRT_ZU_PATIENT wird erstellt.

3.8.3 Dokumentation der Elementfunktionen

3.8.3.1 void HinfahrtPatientRoutine::execute (Event * *event*) [virtual]

Spezifizierung der Zustandsänderungen.

Das Losfahren des Notarztes zum Patienten verändert den Zustand des Notarztes und berechnet die die Ankunftzeit beim Patienten, um ein neues Event mit dem Typ ANKUNFT_PATIENT zu erstellen. Die Berechnung der Ankunftzeit geht aus einer stochastisch ermittelten Fahrtzeit und der aktuellen Simulationszeit hervor. Zudem wird ein mögliches Ereignis gelöscht, das die Ankunft des Notarztes in der Zentrale ankündigt. Dies ist dann der Fall, wenn diese Routine aufgerufen wird, während sich der Notarzt auf dem Rückweg zur Zentrale befindet, also ein Notfall mit hoher Priorität eingegangen ist.

Implementiert EventRoutine.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

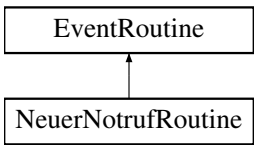
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/HinfahrtPatientRoutine.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/HinfahrtPatientRoutine.cpp

3.9 NeuerNotrufRoutine Klassenreferenz

Routine für den Eingang eines Notrufes.

`#include <NeuerNotrufRoutine.h>`

Klassendiagramm für NeuerNotrufRoutine:



Öffentliche Methoden

- NeuerNotrufRoutine (NotfallWarteschlange *n, Notarzt *arzt, EventList *eList, Zufall *randomGen)
  Konstruktor.
- void execute (Event *event)
  Spezifizierung der Zustandsänderungen.

3.9.1 Ausführliche Beschreibung

Routine für den Eingang eines Notrufes.

Diese Routine beschreibt die Zustandsänderungen in der Notarzt-Simulation wenn ein neuer Notfall in der Zentrale eintrifft.

3.9.2 Beschreibung der Konstruktoren und Destruktoren

3.9.2.1 NeuerNotrufRoutine::NeuerNotrufRoutine (NotfallWarteschlange * *n*, Notarzt * *arzt*, EventList * *eList*, Zufall * *randomGen*)

Konstruktor.

Abhängigkeiten werden injiziert und eine spezielle EventRoutine mit dem Event-Typ NEUER_NOTRUF wird erstellt.

3.9.3 Dokumentation der Elementfunktionen

3.9.3.1 void NeuerNotrufRoutine::execute (Event * *event*) [virtual]

Spezifizierung der Zustandsänderungen.

Trifft ein neuer Notruf in der Zentrale ein, so muss ein neuer Notfall erstellt werden und der Simulation hinzugefügt werden, indem der Notfall der NotfallWarteschlange hinzugefügt wird. Die für die Erstellung eines Notfall benötigten Attribute werden stochastisch ermittelt. Des weiteren wird der akutelle Zustand des Notarztes überprüft und möglicherweise ein Event generiert, dass den Notarzt anweist, den aktuellen Notfall zu behandeln.

Implementiert EventRoutine.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

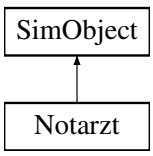
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/NeuerNotrufRoutine.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/NeuerNotrufRoutine.cpp

3.10 Notarzt Klassenreferenz

Repräsentiert einen Notarzt.

`#include <Notarzt.h>`

Klassendiagramm für Notarzt:



Öffentliche Methoden

- Notarzt (NotarztStates state, int place)
  Konstruktor.
- int **getTimestamp** ()
- NotarztStates **getNotarztState** ()
- int **getNotarztPlace** ()
- void **setTimestamp** (int newTimestamp)
- void **setNotarztState** (NotarztStates newState)
- void **setNotarztPlace** (int newPlace)
- void getState ()
  Funktionssignatur für das Abspeichern des Objektes.

3.10.1 Ausführliche Beschreibung

Repräsentiert einen Notarzt.

Klasse für die Erstellung von Notarzt-Objekten, die in der Notarzt-Simulation verwendet werden können. Der Notarzt besitzt keine eigen Logik sondern repräsentiert immer nur einen bestimmten Zustand des Notarztes in der Notarzt-Simulation.

3.10.2 Beschreibung der Konstruktoren und Destruktoren

3.10.2.1 Notarzt::Notarzt (NotarztStates *state*, int *place*)

Konstruktor.

Mit den gegebene Parametern wird der Anfangszustand des Notarztes in der Notarzt-Simulation defniert. Timestamp wird initial immer auf 0 gesetzt.

3.10.3 Dokumentation der Elementfunktionen

3.10.3.1 void Notarzt::getState ( ) [virtual]

Funktionssignatur für das Abspeichern des Objektes.

Diese Schnittstelle stellt die minimale Erwartunhshaltung an Zustands-Objekten innerhalb einer Simulation dar. Sie muss foglich von jedem speziellem Zustands-Objekt implementiert werden. Diese Funktion speichert den aktuellen Zustand des Objektes.

TODO Wird in der aktuellen Notarzt-Implementierung nicht verwendet. Die aktuelle Architektur lässt keine allgemeinen Speicherung von SimObjekten zu.

Implementiert SimObject.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

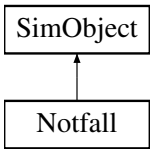
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/Notarzt.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/Notarzt.cpp

3.11 Notfall Klassenreferenz

Repräsentiert einen Notfall.

`#include <Notfall.h>`

Klassendiagramm für Notfall:



Öffentliche Methoden

- Notfall (int callTime, int prio, int treatmentDuration, int place)

  Konstruktor.
- int **isUrgent** ()
- int **getCallTime** ()
- int **getTreatmentDuration** ()
- int **getPlace** ()
- void getState ()

  Funktionssignatur für das Abspeichern des Objektes.

3.11.1 Ausführliche Beschreibung

Repräsentiert einen Notfall.

Klasse für die Erstellung eines Notfall-Objektes, dessen Zustand im Laufe der Simulation verändert werden kann. Besitzt keine eigene Logik, sondern repräsentiert nur den aktuellen Zustand.

Erstellt werden Notfälle im Rahmen der Notfall-Simulation in der Regel nachdem ein neuer Notruf bei der Zentrale eingetroffen ist.

Notfälle werden von der NotfallWarteschlange verwaltet.

3.11.2 Beschreibung der Konstruktoren und Destruktoren

3.11.2.1 Notfall::Notfall (int *callTime,* int *prio,* int *treatmentDuration,* int *place*)

Konstruktor.

Mit den gegebenen Parametern wird der Anfangszustand des Notfalls in der Notarzt-Simulation defniert.

3.11.3 Dokumentation der Elementfunktionen

3.11.3.1 void Notfall::getState ( ) [virtual]

Funktionssignatur für das Abspeichern des Objektes.

Diese Schnittstelle stellt die minimale Erwartunhshaltung an Zustands-Objekten innerhalb einer Simulation dar. Sie muss foglich von jedem speziellem Zustands-Objekt implementiert werden. Diese Funktion speichert den aktuellen Zustand des Objektes.

TODO Wird in der aktuellen Notarzt-Implementierung nicht verwendet. Die aktuelle Architektur lässt keine allgemeinen Speicherung von SimObjekten zu.

Implementiert SimObject.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/Notfall.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/Notfall.cpp

3.12 NotfallWarteschlange Klassenreferenz

`#include <NotfallWarteschlange.h>`

Öffentliche Methoden

- NotfallWarteschlange (StateStorage *storage)

  Konstruktor.
- void add (Notfall *notfall)

  Fügt einen Notfall hinzu.
- Notfall * pop ()

  Liefert und entfernt den nächsten Notfall.
- Notfall * front ()

  Liefert den nächstenNotfall.
- void printList ()

  Druckt die aktuelle Warteschlange auf die Ausgabe.

3.12.1 Ausführliche Beschreibung

Verwaltet Notfälle

In der Notarzt-Simulation werden Notfälle in einer Warteschlange verwaltet. Die Warteschlange defniert dabei den nächsten Notfall, den der Notarzt behandeln muss. Für die Sortierung der Notfälle werden dabei zwei Kriterien heran gezogen: den Zeitpunkt des Anrufes (ältere Anrufe zuerst) und die Priorität (hohe Priorität immer zuerst).

3.12.2 Beschreibung der Konstruktoren und Destruktoren

3.12.2.1 NotfallWarteschlange::NotfallWarteschlange (StateStorage * *storage*)

Konstruktor.

Erstellt ein NotfallWarteschlangen-Objekt. Benötigt ein StateStorage Objekt zur Registrierung von Notfällen, damit diese automatisch abgespeichert werden.

3.12.3 Dokumentation der Elementfunktionen

3.12.3.1 void NotfallWarteschlange::add (Notfall * *notfall*)

Fügt einen Notfall hinzu.

Der übergebene Notfall wird der Warteschlange hinzugefügt. Seine Position innerhalb der Warteschlange bestimmen seine Attribute der Priorität und der Anrufzeit.

3.12.3.2 Notfall * NotfallWarteschlange::front ( )

Liefert den nächstenNotfall.

Liefert den nächsten Notfall, ohne ihn aus der Warteschlange zu entfernen. D.h. ein direktes, wiederholtes Aufrufen dieser Funktion liefert den gleichen Notfall.

3.12.3.3 Notfall * NotfallWarteschlange::pop ( )

Liefert und entfernt den nächsten Notfall.

Liefert den nächsten zu behandelden Notfall zurück und löscht diesen aus der Warteschlange.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

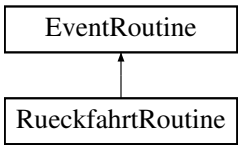
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/NotfallWarteschlange.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/NotfallWarteschlange.cpp

3.13 RueckfahrtRoutine Klassenreferenz

Routine für die Rückfahrt zur Zentrale.

`#include <RueckfahrtRoutine.h>`

Klassendiagramm für RueckfahrtRoutine:



Öffentliche Methoden

- RueckfahrtRoutine (Notarzt *notarzt, EventList *eventList, Zufall *randomGenerator)
  Konstruktor.
- void execute (Event *event)
  Spezifizierung der Zustandsänderungen.

3.13.1 Ausführliche Beschreibung

Routine für die Rückfahrt zur Zentrale.

Diese Routine beschreibt die Zustandsänderungen in der Notarzt-Simulation wenn der Notarzt sich auf den Weg zurück zur Zentrale macht.

3.13.2 Beschreibung der Konstruktoren und Destruktoren

3.13.2.1 RueckfahrtRoutine::RueckfahrtRoutine (**Notarzt** ∗ *notarzt,* **EventList** ∗ *eventList,* **Zufall** ∗ *randomGenerator*)

Konstruktor.

Abhängigkeiten werden injiziert und eine spezielle EventRoutine mit dem Event-Typ ABFAHRT_ZU_ZENTRALE wird erstellt.

3.13.3 Dokumentation der Elementfunktionen

3.13.3.1 void RueckfahrtRoutine::execute (**Event** ∗ *event*) [virtual]

Spezifizierung der Zustandsänderungen.

Die Rückfahrt des Arztes zur Zentrale ändert den Zustand des Notarztes und generiert ein neues Event, dass die Ankunft des Arztes in der Zentrale ankündigt. Der Zeitpunkt, wann der Arzt in der Zentrale ankommt, wird aus der stochastisch ermittelten Fahrtzeit und der aktuellen Simulationszeit berechnet.

Implementiert EventRoutine.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

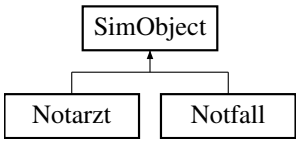
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/RueckfahrtRoutine.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/RueckfahrtRoutine.cpp

3.14 SimObject Klassenreferenz

Abstrakte Klasse für Zustands-Objekte der Simulation.

`#include <SimObject.h>`

Klassendiagramm für SimObject:



Öffentliche Methoden

- virtual void getState ()=0

  Funktionssignatur für das Abspeichern des Objektes.

3.14.1 Ausführliche Beschreibung

Abstrakte Klasse für Zustands-Objekte der Simulation.

Diese abtrakte Klasse definiert die Schnittstelle zu den Objekten einer Simulation, die in der Gesamtheit den Zustand der Simulation repräsentieren. Die Idee dahinter ist, dass das abspeichern der Zustände nach jedem Durchlauf der Simulationschleife unabhängig von der spezzifischen Implementierung ist.

3.14.2 Dokumentation der Elementfunktionen

3.14.2.1 virtual void SimObject::getState ( ) [pure virtual]

Funktionssignatur für das Abspeichern des Objektes.

Diese Schnittstelle stellt die minimale Erwartunhshaltung an Zustands-Objekten innerhalb einer Simulation dar. Sie muss foglich von jedem speziellem Zustands-Objekt implementiert werden. Diese Funktion speichert den aktuellen Zustand des Objektes.

TODO Wird in der aktuellen Notarzt-Implementierung nicht verwendet. Die aktuelle Architektur lässt keine allgemeinen Speicherung von SimObjekten zu.

Implementiert in Notfall und Notarzt.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/SimObject.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/SimObject.cpp

3.15 SimulationManager Klassenreferenz

Steuert die Simulation.

`#include <SimulationManager.h>`

Öffentliche Methoden

- SimulationManager (EventList ∗eList, EventRoutine ∗roList[ ], int numRoutines, StateStorage ∗storage)
  Konstruktor.
- void **run** (Event ∗initlaEvents[ ], int sizeEvents, int endTime)

3.15.1 Ausführliche Beschreibung

Steuert die Simulation.

Objekte dieser Klasse sind verantwortlich für den grundsätlichen Ablauf einer diskreten, ereignisgesteuerten Simulation. Diese besteht aus eine Initialisierungspahse, wo die initale Ereignisliste mit den initialen Ereignissen gefüllt wird. Anschließend beginnt die Simulationsschleife zu laufen.

3.15.2 Beschreibung der Konstruktoren und Destruktoren

3.15.2.1 SimulationManager::SimulationManager (EventList * *eList*, EventRoutine * *roList[ ]*, int *numRoutines*, StateStorage * *storage*)

Konstruktor.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/SimulationManager.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/SimulationManager.cpp

3.16 StateStorage Klassenreferenz

Öffentliche Methoden

- void **saveState** ()
- void **registerNotfall** (Notfall *notfall)
- void **unregisterNotfall** (Notfall *notfall)
- void **registerNotarzt** (Notarzt *notarzt)
- void **unregisterNotarzt** (Notarzt *notarzt)
- void **check_error** ()
- void **dbconnect** ()
- void **dbdisconnect** ()
- void **dbtest** ()
- int **max_idNotarzt** ()
- int **max_idNotfall** ()
- void **storeNotarzt** (Notarzt *notarzt)
- void **storeNotfall** (Notfall *notfall)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/StateStorage.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/StateStorage.cpp

3.17 Zufall Klassenreferenz

Öffentliche Methoden

- void **getRandomExpNotruf** (int Zeitraum, vector< int > *vec, int *size)
- int **getPrio** ()
- int **versorgungszeit** (int Prio)
- int **getStadtbezirk** ()
- int **Fahrzeit** (int Bezirk1, int Bezirk2)

Statische öffentliche Attribute

- static int **Bevoelkerung** [ ] = {10000,30000,50000,31000,50000,10000,66000,15000,4000,1000}
- static int **BevArraySize** = 10
- static std::time_t **now** = std::time(0)
- static boost::random::mt19937 **gen** = boost::random::mt19937(static_cast<uint32_t>(now))

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/include/Zufall.h
- /home/mkemp/Studium/DiskreteSimulation/notarzt-sim/source/src/Zufall.cpp

Index