Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Projekt 1
Sommersemester 2011

Andreas Basener

Entwicklung eines Skriptsimulators für das Living-Place

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

Zusammenfassung

In dieser Ausarbeitung wird eine Lösung für die Simulation von Sensordaten vorgestellt. Dazu werden die nötigen Voraussetzungen beschrieben und anschließend die einzelnen Bestandteile eines Scriptsimulators erklärt. Zum Schluss werden mögliche Verbesserungen des Scriptsimulators vorgeschlagen.

1 Einführung

Da im Living-Place-Projekt viele Teilprojekte zurzeit erst entstehen oder geplant werden, gleichzeitig aber bereits Sensordaten der fertigen Wohnung benötigt werden, ist es nötig, diese Daten zu simulieren.
Dazu habe ich im Rahmen des Masterprojekts 1 einen Simulator konzipiert und umgesetzt. Mit Hilfe dieses Simulators sollen beliebige Aktivitäten und Tagesabläufe innerhalb des Living-Places dargestellt werden können.
Ein weiterer Vorteil eines Simulators ist es, dass Änderungen leicht vorzunehmen sind, ohne Manipulationen an der realen Installation vornehmen zu müssen. Zudem können Konflikte zwischen verschiedenen Projekten durch den Einsatz eines Simulators vermieden werden. Es können auch von der realen Zeit abweichende Zeiten simuliert werden und Zeiträume beschleunigt oder verlangsamt abgespielt werden (Gipser, 1999).

Für den Simulator wird ein Framework benötigt, mit dem die verschiedenen Komponenten des Living-Places dargestellt werden können. In diesem Framework soll ein Script zur Verfügung gestellt werden, mit dem ein Tagesablauf simuliert werden kann. Das Script besteht dabei aus mehreren Einträgen, die die Komponenten im Living-Place wiedergeben. Beim Abspielen des Scripts sollen die einzelnen Einträge Sensordaten generieren. Um die erzeugten Sensordaten weiteren Projekten im Living-Place zur Verfügung stellen zu können, muss der Simulator an die bestehende Infrastruktur angeschlossen werden.

In den folgenden Kapiteln wird erläutert, wie das CASAS-Projekt mit Sensordaten umgeht. Danach stelle ich meine Lösung für einen Skriptsimulator vor. Zum Schluss wird dargestellt, was der Skriptsimulator bereits leisten kann und wie er in der Zukunft erweitert werden kann.

2 CASAS Projekt

Im Rahmen des CASAS-Projektes der Washington State University werden diverse Aufzeichnungen von Sensordaten zur Verfügung gestellt[1]. Diese Aufzeichnungen stellen unterschiedliche Szenarien dar. U.a. werden kurze Arbeitsabläufe dargestellt. Es stehen aber auch komplexe Aufzeichnungen zur Verfügung, die den Tagesablauf einzelner oder mehrerer Personen über einen längeren Zeitraum wiedergeben.

2.1 Sensordaten

Die Sensordaten in den Aufzeichnungen sind, wie in Listing 1 zu sehen, folgendermaßen aufgebaut: Jedes einzelne Sensorwert steht in einer separaten Zeile. In dieser Zeile wird zuerst das Datum und die Uhrzeit für den erzeugten Sensorwert angegeben, wann der Sensorwert erzeugt wurde. Danach ist die ID des Sensors angegeben, gefolgt von dem Sensorwert. Optional kann zum Schluss noch ein zusätzlicher Text angegeben werden. Die Bestandteile eines Datums sind durch Tabulatoren getrennt.
Die einzelnen Sensordaten werden chronologisch aufsteigend aufgelistet.

```
2008-02-27      12:49:55.470956 M17     ON
2008-02-27      12:49:55.470956 M15     OFF
2008-02-27      12:49:55.808938 M14     OFF
2008-02-27      12:49:57.548709 M16     OFF
2008-02-27      12:49:57.717712 M13     OFF
2008-02-27      12:49:58.10558  AD1-B   0.0332818
2008-02-27      12:49:59.197328 M17     OFF
2008-02-27      12:50:01.10764  AD1-B   0.302934
2008-02-27      12:50:10.25482  AD1-B   0.471413
2008-02-27      12:50:25.21543  AD1-B   0.538329
2008-02-27      12:50:38.708635 M17     ON
2008-02-27      12:50:40.5204   AD1-B   0.467429
2008-02-27      12:50:42.521531 M17     OFF     Wash_hands end
2008-02-27      12:50:43.533263 M17     ON      Cook begin
2008-02-27      12:50:56.242    I07     ABSENT
2008-02-27      12:51:01.601718 D01     OPEN
```

Listing 1: Sensordatenaufzeichnung

[1] http://ailab.wsu.edu/casas/datasets.html

2.2 Synthetic Data Generator

Neben den umfangreichen Aufzeichnungen realer Sensordaten wird das Werkzeug *Synthetic Data Generator*[2] zur Verfügung gestellt, mit dem Sensordaten nach Belieben erzeugt werden können (Mendez-Vazquez u. a.).
Für das Werkzeug wird eine Textdatei benötigt, in der festgelegt wird, wie die Sensordaten erzeugt werden sollen (s. Listing 2). Diese Textdatei wird von dem *Synthetic Data Generator* eingelesen, welcher anschließend eine neue Textdatei mit den gewünschten Sensordaten ausgibt. Die Sensordaten werden wie im vorhergehenden Kapitel formatiert.

```
3
lamp 2 on off
thermostat 10 65 66 67 68 69 70 71 72 73 74
television 11 on off 2 3 4 5 6 7 8 9 10
2
e1
daily 1
2
lamp on NoOrder
television 5 Uniform 10 20
e2
weekly 2
3
lamp off NoOrder
thermostat 72 Normal 10 2
television 10 NoOrder
2002-10-07T09:30
off 1 off
10000
```

Listing 2: Scriptanweisungen

[2] http://ailab.wsu.edu/casas/datasets/sdg.zip

3 Skriptsimulator

Die in Kapitel 2 vorgestellten Sensordaten sind leider nicht vollständig mit dem Living-Place kompatibel. Pro Zeile kann immer nur ein einzelner Sensorwert angegeben werden. Zusammengesetzte Werte, wie sie z.B. für Raumkoordinaten benötigt werden, sind nicht möglich. Weiterhin sind die Wertebereiche sowie die Genauigkeit der Sensoren nicht dokumentiert. Das Skriptformat für den *Synthetic Data Generator* ist zudem ziemlich unübersichtlich. Komplexe Tagesabläufe können damit nur sehr schwer dargestellt werden.

Daher habe ich mich entschieden einen, eigenen Skriptsimulator zu bauen, der auf die Bedürfnisse des Living-Places zugeschnitten ist.

Mit dem Skriptsimulator soll es möglich sein, beliebige Tagesabläufe zu erstellen und zu simulieren. Es soll möglich sein, die bestehende Infrastruktur des Living-Places abzubilden. Zudem soll der Aufwand möglichst gering gehalten werden, neue Komponenten in den Simulator zu integrieren.
Weiterhin soll sich der Skriptsimulator nahtlos in die bestehende Infrastruktur des Living-Places einfügen und von unabhängigen Projekten als Quelle für Sensordaten benutzt werden können.

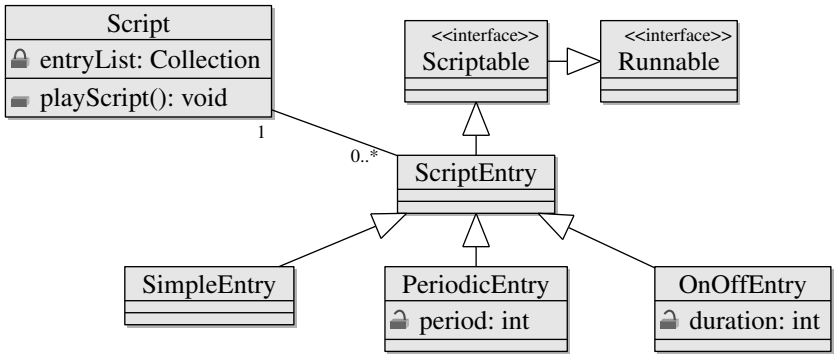

Abbildung 1: Skriptsimulator Klassendiagramm (vereinfacht)

Für die Umsetzung eines Frameworks für den Skriptsimulator werden nur wenige Komponenten benötigt. Wie der Name Skriptsimulator schon sagt, wird ein Skript benötigt das festhält, wie der genaue Simulationsablauf aussehen soll. Das Skript enthält mehrere Skripteinträge, welche die einzelnen Simulationsschritte beschreiben. Für diese Einträge wird eine Schnittstelle definiert. Diese Schnittstelle dient dazu, auf verschiedene Einträge mit den gleichen Methoden zugreifen zu können.

Für die Umsetzung des Skriptsimulators wird die Programmiersprache Java verwendet. Weiterhin werden die Softwarebibliotheken SWT, GSON, JFace, slf4j und ActiveMQ eingesetzt.

3.1 Skript

Die Reihenfolge und der Zeitpunkt, wann Nachrichten gesendet werden, werden über ein Skript gesteuert. In diesem Skript sind einzelne Skripteinträge enthalten, die jeweils einen bestimmten Simulationsschritt beschreiben. Ein Skripteintrag kann dabei einen einzelnen Sensorwert beschreiben, aber auch mehrere, über einen Zeitraum verteilte, Sensordaten umfassen. Z.B. kann ein Eintrag das Öffnen einer Tür, regelmäßige Temperaturdaten oder die Bewegung eines Objektes im Raum widerspiegeln.

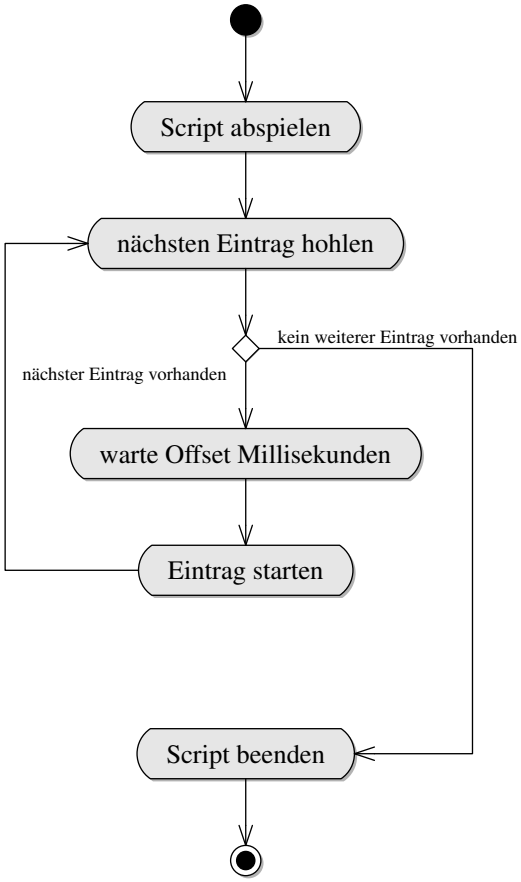

Abbildung 2: Skript Aktivitätsdiagramm

Das Skript arbeitet seine Einträge der Reihe nach ab. Wie in Abbildung 2 zu sehen, holt sich das Skript den nächsten Eintrag, wartet die in diesem Eintrag festgelegte Offsetzeit in Millisekunden und startet anschließend die run()-Methode des Skripteintrages. Das wird so lange wiederholt, bis der letzte Eintrag im Skript ausgeführt worden ist. Danach wird das gesamte Skript beendet.

3.2 Scriptable/ScriptEntry

Das Interface Scriptable erweitert das Runnable Interface von Java[3] . Dadurch können alle Skripteinträge, die das Interface Scriptable implementieren, ohne großen Aufwand als eigenständiger Thread ausgeführt werden. Somit können automatisch mehrere Skripteinträge parallel ausgeführt werden, ohne dass eine zusätzliche Verwaltung der Skripteinträge nötig ist.

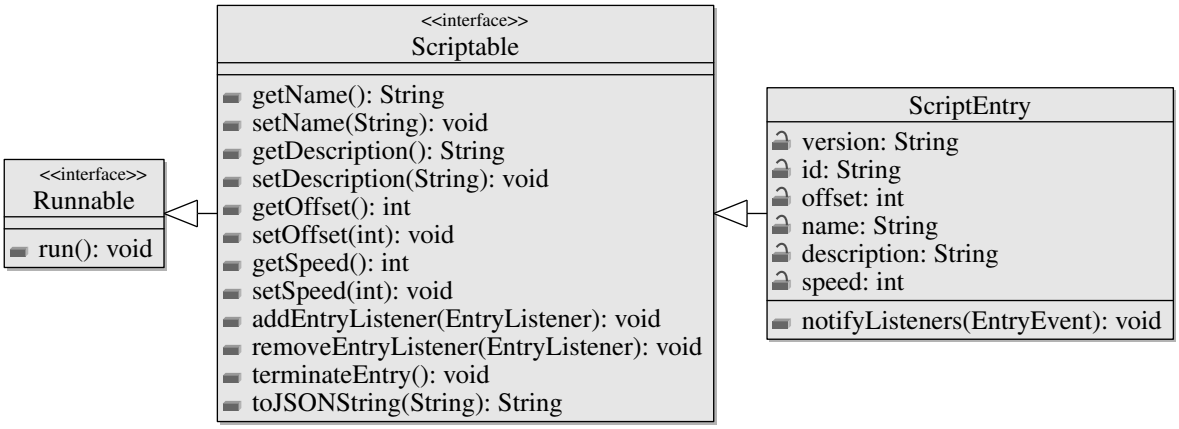

Abbildung 3: Scriptable/ScriptEntry Klassendiagramm

In Abbildung 3 ist das Klassendiagramm für das Skriptable-Interface zu sehen. Es werden diverse Getter und Setter definiert, die den Zugriff auf benötigte Felder regeln.
Mit den Methoden addEntryListener(EntryListener) und removeEntryListener(EntryListener) können sich interessierte Empfänger beim Skripteintrag ein- bzw. austragen, wenn sie neue Sensordaten des Skripteintrags empfangen wollen. Neue Sensordaten werden mittels des Eventmechanismus von Java verbreitet, welcher wiederum dem Observer-Pattern folgt (Gamma, 1995, S. 293 ff.).
Durch die Methode terminateEntry() kann dem Skripteintrag signalisiert werden, dass es seinen Thread terminieren soll.
Mit der toJSONString(String):String wird aus dem Scripteintragobjekt ein JSON-String erzeugt.

Die Standardimplementierung des Scriptable-Interfaces ist die ScriptEntry-Klasse. In der ScriptEntry-Klasse werden auch die benötigten Felder für die Getter und Setter angelegt. Konkrete Scripteintragsklassen können von der ScriptEntry-Klasse abgeleitet werden. Dadurch wird redundanter Code für z.B. die Getter und Setter vermieden. Wichtig ist jedoch, dass in einer neuen Klasse die run()-Methode und evtl. die toJSONString(String):String-Methode überschrieben werden.

[3] http://download.oracle.com/javase/1.4.2/docs/api/java/lang/Runnable.html

3.3 EntryListener/EntryEvent

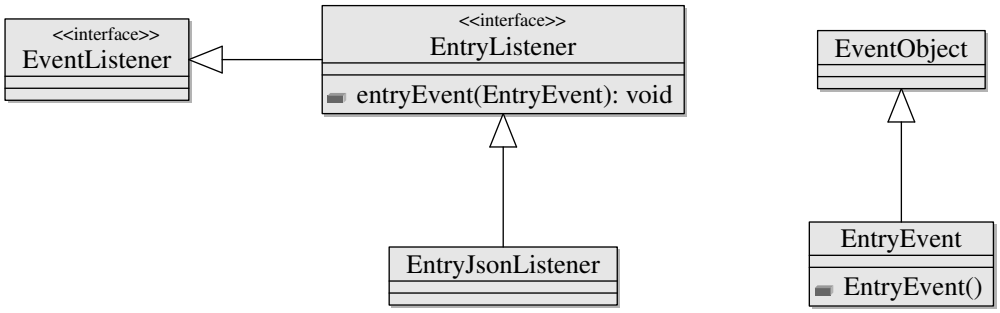

Abbildung 4: EntryListener/EntryEvent Klassendiagramm

Immer, wenn ein ScriptEntry einen neuen Sensorwert erzeugt, wird er mittels des Eventmechanismus von Java (Ullenboom, 2009) bekannt gegeben. Dazu wird ein neuer EntryEvent erzeugt, der eine Referenz auf den ScriptEntry als Eventquelle enthält.
Dieser Event wird den eingeschriebenen Listenern bekannt gegeben, die dann darauf reagieren können.

```
class EntryJsonListener implements EntryListener
{
  /**
   * Catch Event, generate JSON String
   * and publish it to ActiveMQ
   */
  @Override
  public void entryEvent(EntryEvent event){
    try{
      ScriptEntry entry = (ScriptEntry)event.getSource();
      String s = entry.toJSONString(connection.getClientID()
                                    + random.nextInt());
      System.out.println(s);
      sendJSONtoActiveMQ(s);
    }
    catch (JMSException e){
      e.printStackTrace();
    }
  }
}
```

Listing 3: EntryJsonListener

Eine Standardimplementierung für den EntryListener im Skriptsimulator ist die EntryJsonListener-Klasse (s. Listing 3). Sie implementiert das EntryListener-Interface und damit die entryEvent(EntryEvent)-Methode. In dieser Methode wird aus der Quelle des empfangenen EntryEvent-Objektes ein JSON-String erzeugt. Dieser String wird anschließend an das ActiveMQ gesendet.

3.4 GUI

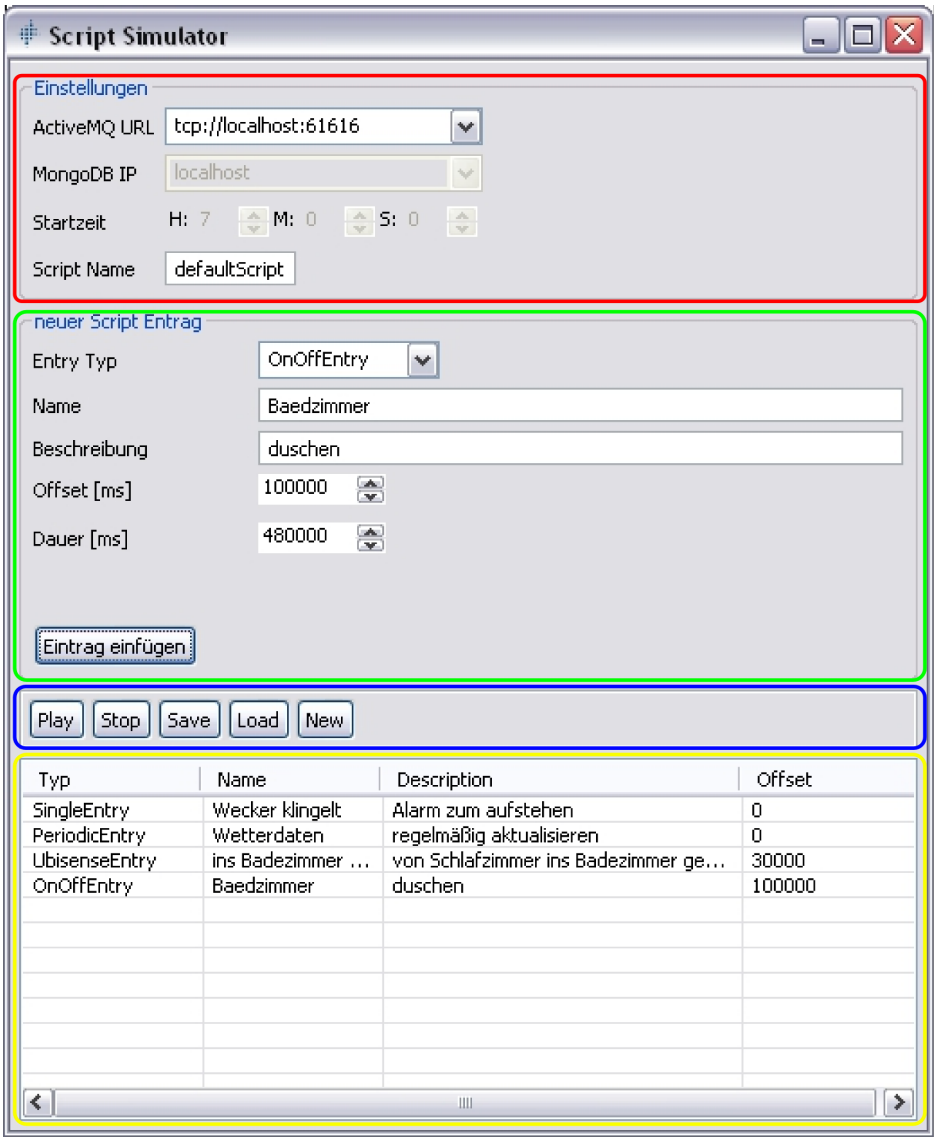

Abbildung 5: GUI Skriptsimulator

In Abbildung 5 ist die grafische Benutzeroberfläche des Skriptsimulators zu sehen. Die einzelnen Funktionsbereiche wurden farbig markiert, damit sie besser zu erkennen sind.

Im rot markierten Bereich können die IP-Adressen angegeben werden, die für die Kommunikation mit ActiveMQ und MongoDB verwendet werden sollen.
Weiterhin können die Startzeit für das Skript und der Skriptname angegeben werden.

Im grün markierten Bereich werden die Einstellungen für neue Skripteinträge vorgenommen. Über ein DropDown-Menü wird der Typ des Skripteintrages ausgewählt. Je nach Typ ändert sich der Bereich darunter und bietet weitere Einstellmöglichkeiten. Bei jedem Eintrag können zumindest Name, Beschreibung und Offset eingestellt werden.
Der in Abbildung 5 gewählte Typ OnOffEntry bietet zusätzlich an, eine Dauer einzustellen. Andere Typen bieten hier wiederum andere Einstellmöglichkeiten an.

Im blau markierten Bereich kann das Skript gestartet, gestoppt, gespeichert, geladen sowie ein neues Skript angelegt werden.

Im gelben Bereich sind die erzeugten ScriptEntrys aufgelistet. Hier werden Typ, Name, Beschreibung und Offset des Eintrages angezeigt. Weitere Informationen über einen Skripteintrag werden momentan nicht angezeigt.

3.5 Anbindung an das Living-Place

Die Kommunikation des Skriptsimulators mit dem Living-Place findet über ActiveMQ (Snyder u. a., 2011) und MongoDB (Membrey u. a., 2010) statt (Otto und Voskuhl). ActiveMQ dient hierbei als Middleware, um mit den anderen Projekten des Living-Places Informationen austauschen zu können. Als Datenformat wird auf JSON (Rieber, 2009, S. 658 ff.) gesetzt.

```
{
  "version":"0.1",
  "id":"ID:basener-8eaf320-4768-1314618855208-1:1144250018",
  "name":"Name des Entrys",
  "description":"Beschreibung des Entrys"
}
```

Listing 4: JSON Nachricht

In Listing 4 ist ein Minimalbeispiel für eine JSON-Nachricht aufgeführt.

Damit die erzeugten Sensordaten für eine spätere Verwendung persistent gespeichert werden können, wird die MongoDB verwendet. Dadurch wird ein zentraler Ablagepunkt für die Sensordaten ermöglicht und der Austausch vereinfacht.

4 Aussicht

Die Funktionen des Skriptsimulators sind noch nicht vollständig umgesetzt.
Die Möglichkeit, Skripte abzuspeichern bzw. zu laden, ist momentan noch nicht implementiert. Dazu ist es notwendig, die unterschiedlichen Skripteinträge in einem Datenformat abzubilden.

Zurzeit wird das Skript in Echtzeit abgespielt, wenn der Play-Knopf gedrückt wurde. Das Abspielen eines Skriptes in Echtzeit kann eine Weile dauern. Für kurze Tests ist es aber nicht wünschenswert, lange auf Ergebnisse warten zu müssen. Daher sollte der Skriptsimulator um einen Mechanismus erweitert werden, mit dessen Hilfe die Abspielgeschwindigkeit beschleunigt werden kann.
Andererseits kann aber auch eine Verlangsamung der Abspielgeschwindigkeit notwendig sein, z.B. wenn schnelle Handlungsabläufe nachvollzogen werden sollen.

Die Sensorwerte werden zurzeit reproduzierbar und deterministisch erzeugt. In der realen Welt kann es aber vorkommen, dass einzelne Sensorwerte gar nicht oder falsch übertragen werden. Weiterhin können Verzögerungen und weitere Störungen auftreten. Um den Simulator möglichst realitätsnah zu gestalten, sollte dieses Verhalten im Simulator berücksichtigt werden können.

Die grafische Benutzeroberfläche kann in einigen Punkten verbessert werden. U.a. werden in der Tabelle der Scripteinträge nur die allgemeinen Informationen der Einträge angezeigt. Außerdem können die Einträge nach dem Erstellen nicht mehr verändert werden.
Zu beachten ist auch, dass die Einträge zwingend in der Reihenfolge erzeugt werden müssen, in der sie später abgespielt werden sollen. Hier sollte eine Sortierung der Einträge, auch nachträglich, ermöglicht werden.

Neben den oben aufgeführten Verbesserungen des Skriptsimulators sind sicherlich noch weitere denkbar. Durch die übersichtliche Struktur des Skriptsimulators sind Änderungen aber ohne großen Aufwand möglich.

Literatur

[Basener 2011a] BASENER, Andreas: Ausarbeitung AW1 Drahtlose Sensornetzwerke im Kontext Ambient Assisted Living / HAW Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-aw1/basener/bericht.pdf, Februar 2011. – Forschungsbericht

[Basener 2011b] BASENER, Andreas: Ausarbeitung AW2 Erlernen und Erkennen von Verhaltensmustern in Sensordaten / HAW Hamburg. August 2011. – Forschungsbericht

[Gamma 1995] GAMMA, E.: *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995 (Addison-Wesley professional computing series). – ISBN 9780201633610

[Gipser 1999] GIPSER, M.: *Systemdynamik und Simulation.* Teubner, 1999. – ISBN 9783519027430

[von Luck u. a. 2010] LUCK, Prof. Dr. K. von ; KLEMKE, Prof. Dr. G. ; GREGOR, Sebastian ; RAHIMI, Mohammad A. ; VOGT, Matthias: Living Place Hamburg – A place for concepts of IT based modern living / Hamburg University of Applied Sciences. URL http://livingplace.informatik.haw-hamburg.de/content/LivingPlaceHamburg_en.pdf, Mai 2010. – Forschungsbericht

[Membrey u. a. 2010] MEMBREY, P. ; PLUGGE, E. ; THIELEN, W. ; HAWKINS, T.: *The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing*. Apress, 2010 (Definitive Guide Series). – ISBN 9781430230519

[Mendez-Vazquez u. a.] MENDEZ-VAZQUEZ, Andreas ; HELAL, Abdelsalam ; COOK, Diane: CHI Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research. In: *Simulating Events to Generate Synthetic Data for Pervasive Spaces*, URL http://eecs.wsu.edu/~cook/pubs/chi09.2.pdf

[Otto und Voskuhl] OTTO, Kjell ; VOSKUHL, Sören: Weiterentwicklung der Architektur des Living Place Hamburg / HAW Hamburg. – Forschungsbericht

[Rieber 2009] RIEBER, P.: *Dynamische Webseiten in der Praxis: PHP, MySQL, CSS, Javascript, XHTML und Ajax*. mitp-Verlag, 2009 (mitp Professional). – ISBN 9783826617829

[Snyder u. a. 2011] SNYDER, B. ; BOSANAC, D. ; DAVIES, R.: *ActiveMQ in Action*. Manning Publications, 2011 (Manning Pubs Co Series). – ISBN 9781933988948

[Ullenboom 2009] ULLENBOOM, Christian: *Java ist auch eine Insel*. Eighth. Galileo Computing, 2009. – URL http://openbook.galileocomputing.de/javainsel8/. – ISBN 978-3-8362-1371-4