Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht 2

Bastian Karstaedt

Entwicklung und Integration des *Indoor Spatial Information Services* in den Living Place Hamburg

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einführung

Dieses Projektsemester wurde dazu genutzt um – basierend auf den Arbeiten des vorangegangenen Projektes sowie der theoretischen Arbeiten – den *Indoor Spatial Information Service* (im Folgenden *ISIS*) zu entwickeln. Der *ISIS* wird als Service in intelligenten Umgebungen verstanden, der auf Basis eines dreidimensionalen IFC Modells Informationen über den Zustand physischer Entitäten sammelt (vgl. Szenarien *IIa*, *IIb* und *IIc*) und semantische Informationen u.a. in Form räumlicher Relationen als Dienst bereitstellt (Details zum ISIS siehe [5]).

1.1 Anwendungsszenarien

Die Anwendungsmöglichkeiten des ISIS sollen durch einige Szenarien verdeutlicht werden, die im Folgenden vorgestellt werden. Das erste Szenario befasst sich mit dem ISIS als Serviceanbieter für **räumliche Anfragen**.

> **Szenario I – Das Gebäudemodell als Serviceanbieter**
> „Wo ist meine Sonnenbrille?!“ – Um diese Frage des Bewohner beantworten zu können sind mehrere technische Hürden zu nehmen. Mit Hilfe von Bilderkennungsverfahren (z.B. SIFT oder SURF) wird zunächst die Position des Gegenstandes im Raum berechnet. Durch eine Service-Anfrage an den *Indoor Spatial Information Service* werden die Gegenstände in unmittelbarer Umgebung und ihre räumlichen Relationen zurückgegeben und der suchenden Person daraufhin eine verbale Beschreibung des Fundortes geliefert (“Die Sonnenbrille befindet sich jetzt auf dem Stuhl unter dem Fenster.”). Das Gebäudemodell wird hier also als Dienst zur Auflösung räumlicher Relationen verstanden.

In den folgenden Szenarien werden Informationen der Realität in den ISIS integriert. Dies dient insbesondere der **Konsistenzwahrung** zwischen realer Welt und Modell.

> **Szenario IIa – Integration von Positionsänderungen des Mobiliars**
> Damit die Rückgabemenge des ISIS in *Szenario I* auch dann noch korrekt ist, wenn der Stuhl verschoben wurde, muss im Gebäudemodell die Position des Gegenstandes aktualisiert werden, sobald dieser bewegt wurde. Hierzu wird im Living Place Hamburg das Echtzeit Ortungssystem *Ubisense*[1] eingesetzt.

[1] Webseite: http://www.ubisense.com

Szenario IIb – Integration von Fenster-Zustandsinformationen
In der intelligenten Wohnung werden Fenster automatisch geöffnet und geschlossen [3]. Über ein Steuersignal kann ein Fenster oder eine Gruppe von Fenstern[2] (`ALL`, `KITCHEN` etc.) geöffnet bzw. geschlossen werden. Der Zustand der Fenster ist im Modell festzuhalten und in der Visualisierung darzustellen.

Szenario IIc – Integration von Ortsinformationen des Bewohners
Bewegt sich ein Bewohner durch die Wohnung von A nach B werden diejenigen Bildschirme, die in seiner Nähe sind an- und die anderen abgeschaltet. Hierbei ist der Aufenthaltsort des Bewohners durch das bereits erwähnte Ortungssystem und die Positionen der Bildschirme durch das Gebäudemodell bekannt. Der ISIS dient hier in erste Linie als Lieferant von metrischen Informationen, deren Interpretation den Konsumenten dieser Informationen obliegt. In [8] wurde dieses Szenario bereits umgesetzt, allerdings ohne ein Modell und mit statisch festgelegten Positionsdaten.

Im Folgenden werden die *Anforderungen an den ISIS* diskutiert, dann die *Softwarearchitektur* vorgestellt und anschließend Details der *Realisierung* besprochen. Dort wird insbesondere auf die Umsetzung der Szenarien *IIa* und *IIb* genauer eingegangen.
Danach wird eine Auswahl weiterer *Arbeiten im Living Place Projektumfeld* geschildert und anschließend ein Fazit des Projektsemesters gezogen und ein Ausblick gegeben.

[2] Wiki: http://livingplace.informatik.haw-hamburg.de/wiki/index.php/Fenster&Heizungssteuerung

2 Realisierung des Indoor Spatial Information Services

Dieses Kapitel beschreibt die Realisierung des ISIS im Rahmen der Projektarbeit im Living Place Hamburg.

2.1 Anforderungen an den ISIS

Die *funktionalen Anforderungen* ergeben sich aus den bereits erwähnten Szenarien. Ohne im Detail auf einzelne Anforderungen einzugehen, werden im Folgenden die hierzu erforderlichen Komponenten kurz erläutert:

Kommunikation Ziel ist die Einbettung von ISIS in die Blackboard-Architektur des Living Place Hamburg. ISIS fungiert als Konsument (*Subscriber*) und Produzent (*Publisher*) von Informationen, und in Form eines Dienstleisters für semantische Anfragen (vgl. *Pull-Mechanismus*[1]).

Visualisierung Eine Visualisierung des Gebäudemodells soll ein optisches Feedback für den momentanen Zustand und die aktuellen Änderungen am Modell bieten.

Persistenz Zum Systemstart ist das Gebäudemodell vom IFC-Modellserver zu laden. In regelmäßigen Abständen und vor dem Beenden sind Änderungen zurückzuschreiben.

Analyse und Modifikation Es ist ein Modul zu realisieren, das Änderungen *an* und Abfragen *auf* ein Gebäudemodell ermöglicht.

Die *nicht-funktionalen Eigenschaften* des ISIS richten sich an den Bedürfnissen einer möglichst wartungs- und konfigurationsfreien Umgebung – dem *Smart Home* – aus. Dazu gehören (nach DIN 66272) vor allem Zuverlässigkeit und Effizienz. Zudem ist aus Entwicklersicht eine einfache Test- und Änderbarkeit unabdingbar.

[1] http://livingplace.informatik.haw-hamburg.de/wiki/index.php/Kommunikationsschnittstelle

2.2 Softwarearchitektur

Die im Rahmen des Projektes konzipierte und realisierte Architektur (siehe Abb. 2.1) ist das Resultat aus den Entwürfen theoretischer Vorarbeiten und den Erfahrungen der vorangegangenen Projektarbeit.

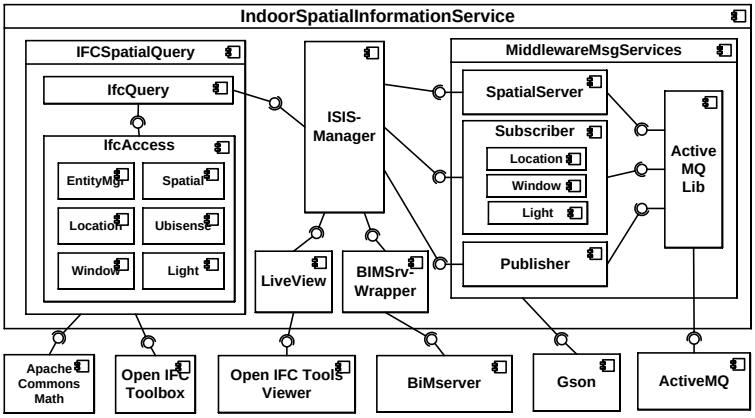

Abbildung 2.1: Architektur des *Indoor Spatial Information Service*

Der ISIS ist modular aufgebaut, sodass einzelne Komponenten in anderen Projekten eingesetzt werden können. Bei der Konzeption wurde auf eine möglichst geringe Kopplung der Komponenten untereinander Wert gesetzt.

Technologien, Tools und Bibliotheken

Im Architekturdiagramm finden sich die in den Anforderungen formulierten Bereiche *Kommunikation*, *Visualisierung*, *Persistenz* und *Analyse* in Form von internen oder externen Komponenten wieder.

Die **Kommunikation** (Komponente `MiddlewareMsgServices`) wird auf Basis des projektweit eingesetzten Message Brokers *ActiveMQ*[2] realisiert. In [9] wurde ein Wrapper zur Einbindung des ActiveMQ in Java Anwendungen entwickelt, der auch im ISIS zum Einsatz kommt. Hierüber werden z.B. Ortsinformationen von *Ubisense*[3] über die Position von Gegenständen oder den Aufenthaltsort von Personen an ISIS weitergereicht. Das projektweit

[2] Webseite: http://activemq.apache.org

[3] Der Aufenthaltsort von aktiven *Ubisense Tags* wird mittels laufzeitbasierte Distanzmessung ermittelt. Näheres siehe unter http://www.ubisense.net.

eingesetzte Nachrichtenformat ist JSON. Im ISIS wird zur (De-)Serialisation von JSON die offene *GSON* Bibliothek[4] verwendet.

Die **Visualisierung** erfolgt mit dem *Open IFC Tools Viewer* [10] der Bauhaus Universität Weimar. Der Zugang zur (beta) Version der Visualisierungs-API wird derzeit nur Forschungsprojekten gewährt. Die API stellt IFC-Modelle mittels Java3D dar und ermöglicht die interaktive Betrachtung des Modells. Im Rahmen des ISIS wurde die Visualisierung um Ortsänderungen von Mobiliar und die Darstellung der Position von Ubisense Tags erweitert. Das Hinzufügen von *neuen* Entitäten (z.B. Ubisense-Tags) in das Modell hat derzeit noch einen kompletten *Reload* des Views zu Folge, da dieser Anwendungsfall in der Toolbox bisher nicht berücksichtigt wurde.

Als **Persistenzebene** wird der Open Source *BiMserver* [11] in der derzeit aktuellsten Version 1.0 eingesetzt. Dieser verwaltet zentral das IFC-Modell und lässt sich über die SOAP Schnittstelle ansprechen. Er wurde bereits in [4] erfolgreich eingesetzt (damals in einer .NET Umgebung). Im Laufe des aktuellen Projektes wurde ein Shellskript geschrieben, dass bei Ausführung den aktuellen BiMserver („Nightlybuild“) auf der virtuellen Maschine installiert.

Für die **Analyse und Modifikation** des IFC Gebäudemodells steht die *IFCSpatialQuery*-Komponente bereit. Hierzu stellt die *Open IFC Java Toolbox* (ebenfalls von der Universität Weimar) einen objektorientierten Zugriff auf IFC Dateien bereit. Modelle können damit eingelesen, bearbeitet und gespeichert werden (eine räumliche Analyse ist damit *nicht* möglich). Im ISIS wurde sie zusammen mit einer Bibliothek für mathematische Anwendungen von Apache[5] in der *IFCSpatialQuery*-Komponente verwendet. Derzeit können mit dieser Komponente bspw. Entitäten (inkl. Attributierungen, Repräsentation usw.) hinzugefügt und positioniert werden und Entitäten auf Basis projektspezifischer Attribute gefiltert werden. Eine räumliche Analyse ist derzeit nicht möglich (siehe hierzu Kap. *Ausblick*).

Architekturdetails

Der Zugriff auf die Analyse- und Modifikationskomponente (und somit auf das Gebäudemodell) erfolgt über eine *Fassade*, die den Zugriff bündelt und somit die Schnittstelle vereinfacht und die Kopplung des Systems verringert. Diese Fassade ist zudem ein *Singleton*, um sicherzustellen, dass alle Instanzen auf dasselbe *Modell* zugreifen.

Um Informationen von Datenquellen zu erhalten fungiert der ISIS im Rahmen der Nachrichteninfrastruktur des Living Place als *Subscriber* auf mehrere *Topics* (z.B. das Topic

[4] Webseite: http://code.google.com/p/google-gson/
[5] Webseite: http://commons.apache.org/math

`UbisenseTracking`). Im ISIS gibt es für jedes Topic einen Subscriberthread. Jeder Subscriber erbt von einer abstrakten Oberklasse. Durch das dadurch geerbte gemeinsame Interface können die einzelnen Subscriber z.B. in einer Liste gehalten werden und ähnlich wie bei dem Strategy-Pattern verwaltet werden. D.h. auf alle Subscriberthreads in der Liste werden die gleichen Operationen ausgeführt werden (z.B. das Beenden oder Wiederaufnehmen der Kommunikation).

Serviceanfragen werden vom Modul *SpatialServer* auf Basis des in [8] vorgestellten *Pull-Mechanismus* realisiert. Hierbei werden Anfragen über eine Queue an den ISIS gesendet. Anschließend wird über ein Topic allen Subscribern des Topics geantwortet – der eigentliche Empfänger weiß auf Grund der Nachrichten-Id, dass die Nachricht eine Antwort auf seine Anfrage darstellt. Dies wurde beispielhaft durch eine `Ping` Anfrage implementiert.

Änderungen am *Model* werden an den *View* des Open Ifc Viewers durch vorhandene *ActionListener* weitergereicht – hierbei existiert im Viewer zu jeder IFC Entität eine korrespondierende, geometrisch angepasste Java3D Version. Änderungen am Modell führen deshalb *nicht* automatisch zu Änderungen im View, sondern müssen programmatisch herbeigeführt werden. Dies funktioniert z.B. bei Positionsänderungen einzelner Entitäten im Modell sehr gut, beim *Hinzufügen* neuer Entitäten ist es aber problematisch. Hier müsste die korrespondierende Java3D Version der neuen Entität in den Szene Graphen des 3D Viewers eingetragen werden. Dies wird derzeit noch nicht von der Bibliothek unterstützt. Da die Bibliothek nicht im Quellcode vorliegt, konnte keine eigene Erweiterung stattfinden. Wie bereits erwähnt findet derzeit daher beim Hinzufügen einer neuen Entität ein kompletter *Reload* der 3D Modells statt.

Zur Qualitätssicherung wurden zum einen Unit-Tests geschrieben, zum anderen wurde die Architektur mittels SonarJ[6] untersucht. Es konnte dadurch sichergestellt werden, dass keine Architekturverletzungen vorhanden sind. Jedoch wurden mit Hilfe des Tools Zyklen entdeckt und anschließend erfolgreich behoben.

2.3 Realisierung

Der ISIS wurde als *multithreading Applikation* in Java realisiert. Dahingehend erfolgt stündlich der Upload des IFC-Gebäudemodells in einem Timer-Thread. Auch die Kommunikation wird Thread-basiert gelöst, da durch blockierendes Warten auf Nachrichten das System zum Stillstand käme. Da diese Threads nicht ausschließlich lesend auf eine gemeinsame Datenstruktur – das Gebäudemodell – zugreifen, erfolgt der Zugriff hierauf nur innerhalb synchronisierter Code-Blöcke.

[6] http://www.hello2morrow.com/products/sonarj

Im Folgenden werden an Hand verschiedener Systemphasen Realisierungsdetails erörtert. In der *Initialisierungsphase* wird das System vorbereitet, indem das Gebäudemodell geladen und die Kommunikationsinfrastruktur hergestellt wird. Zur *Laufzeit* werden aktualisierte Zustandsinformationen in das Modell übertragen und Anfragen an das Modell gestellt. Die *Finalisierungsphase* führt vor dem Beenden des Systems abschließende Arbeiten zur Konsistenzwahrung durch.

Initialisierungsphase

Zum Systemstart wird das IFC-Gebäudemodell vom *BiMserver* geladen und der Analysekomponente zum Parsen übergeben. Letztere stellt daraufhin den zentralen Zugriff auf das Modell bereit.

Nun werden Attribut-Informationen einzelner Gebäudemodell-Entitäten extrahiert. Hierzu wurden dem IFC Modell in Autodesk Revit® weitere Attribute (sog. *IfcProperties*) hinzugefügt (siehe Abb. 2.2).

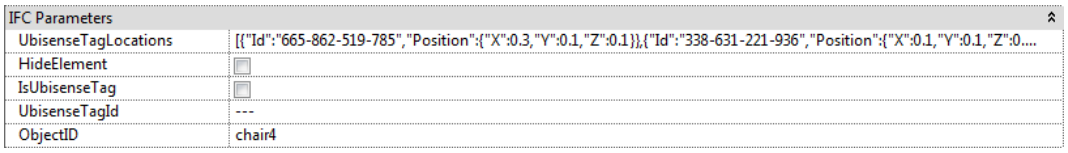

Abbildung 2.2: Neue Attribute im IFC Modell

Beim Start der *Visualisierung* werden alle Entitäten versteckt, deren `HideElement`-Attribut positiv belegt ist (z.B. Stockwerke, Deckenbalken, Wände etc.). Dadurch ist eine ungehinderte Sicht in die gewünschten Räume gewährleistet (s. Abb. 2.3).

Sobald das Ortungssystem die Position neuer *Tags* übermittelt, werden diese dem Modell hinzugefügt und anschließend in der Visualisierung dargestellt. Hierzu gibt es im Modell eine Entität, deren Repräsentation (Form und Farbe) für alle neuen Tags im IFC Objektmodell *dieselbe* ist – diese ist mit `IsUbisenseTag` markiert. Neue Tags werden u.a. mit dem entsprechend belegten Attribut `UbisenseTagId` versehen (z.B. „442-142-562-179“).

Das Attribut `ObjectID` wird projektübergreifend definiert und richtet sich nicht nach der im IFC-Modell eingesetzten GUID (*Globally Unique Id*). Die Verwendung des Attributs `UbisenseTagLocations` wird in der folgenden Phase beschrieben.

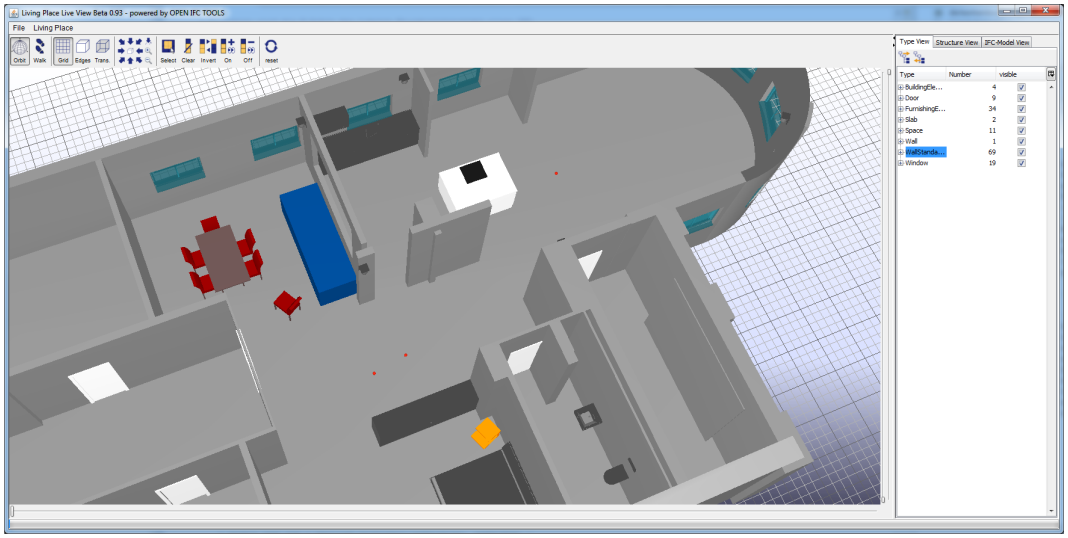

Abbildung 2.3: In den ISIS integrierte Visualisierung des Gebäudemodells (Ubisense Tags werden rot dargestellt)

Laufzeitphase

Zur Laufzeit werden zum einen Zustandsinformationen an den ISIS übergeben und zum anderen räumliche Anfragen an das Modell gestellt. Da das zu letzterem wichtige Analysemodul für *Szenario I* noch „work-in-progress“ ist, wird im Folgenden die Integration verschiedener Datenquellen in den ISIS beschrieben. Die Realisierungsdetails werden hier an Hand der Szenarien *IIa* und *IIb* besprochen.

Konsistenzwahrung auf Basis des Ortungssystems *Ubisense*

Szenario IIa befasst sich mit der Konsistenz von Realität und Modell, d.h. dass Ortsänderungen von Mobiliar an das Modell weitergeleitet werden müssen. Dies wird dadurch erreicht, dass Mobiliar mit je zwei Ubisense Tags ausgestattet wird, um somit den Aufenthaltsort sowie die Rotation um die z-Achse zu ermitteln[7].

In Abb. 2.4 ist links ein Stuhl dargestellt unter dem zwei Ubisense Tags platziert sind. Wie rechts in der Grafik zu sehen ist, hat jeder Tag je eine globale Position $\vec{t}_{global}$, welche durch Ubisense gegeben ist und außerdem eine lokale Position $\vec{t}_{lokal}$ im lokalen Koordinatensystem der Entität. Letztere muss ausgemessen und über das bereits erwähnte IFC-Attribut `UbisenseTagLocations` als JSON String in der Modellierungssoftware eingetragen werden (s. Abb. 2.2).

[7] Um die Rotation um alle Achsen zu ermitteln werden drei Tags benötigt.

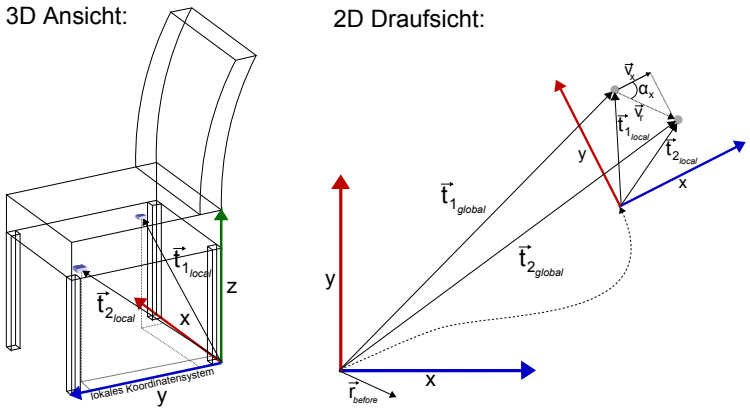

Abbildung 2.4: Platzierung von zwei Ubisense Tags unter einem Stuhl

Außerdem ist rechts in Abb. 2.4 zu sehen, wie sich das lokale Koordinatensystem einer IFC Entität an ihrem umgebenden Containerelement orientiert. Hierzu wird zum einen der Koordinatenursprung des lokalen Koordinatensystems (`IfcLocalPlacement`[8]) und zum anderen die Richtung der x-Achse in den Koordinaten des umgebenden Systems benötigt (`RefDirection`[9]).

Durch die Kenntnis der lokalen und globalen Positionen der Ubisense-Tags kann nun die Position der IFC-Entität im Modell berechnet werden.

Zunächst wird der Winkel α_x zwischen $\vec{v}_r$ und der x-Achse $\vec{v}_x = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ berechnet:

$$\vec{v}_r = \vec{t}_{1_{lokal}} - \vec{t}_{2_{lokal}} \tag{2.1}$$

$$\cos(\alpha_x) = \frac{\vec{V_1}\,\vec{V_2}}{|\vec{V_1}|\,|\vec{V_2}|} \tag{2.2}$$

Anschließend wird mit Hilfe einer Drehmatrix der Richtungsvektor $\vec{r}_{before}$ um den Winkel gedreht, um den das lokale Koordinatensystem zum umgebenden Koordinatensystem rotiert

[8] Siehe auch http://buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcgeometricconstraintresource/lexical/ifclocalplacement.htm
[9] Siehe auch http://www.iai-tech.org/ifc/IFC2x4/alpha/html/ifcgeometryresource/lexical/ifcaxis2placement3d.htm

ist:

$$R(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{2.3}$$

$$\vec{r} = \vec{t}_{1_{global}} - \vec{t}_{2_{global}} \tag{2.4}$$

$$\vec{r'} = R(\alpha) * \vec{r} \tag{2.5}$$

Die Berechnung von `RefDirection` ist somit abgeschlossen. Nun folgt die Verschiebung des lokalen Koordinatensystems (`IfcLocalPlacement`) im umgebenden System – dies geschieht auf ähnliche Weise (jedoch unter Zuhilfenahme der lokalen und globalen Position *eines* Ubisense-Tags) und wird hier nicht näher erläutert.

Integration weiterer Zustandsinformationen: Fenstersteuerung

Neben Ortsinformationen werden vom ISIS auch andere Zustandsinformationen in das Modell übertragen (vgl. *Szenario IIc*). Beispielhaft wurde hier der Öffnungswinkel der Fenster, welche durch ein intelligentes Steuerungssystem von Benedikt Johannsen und Alexander Pautz [3] ausgerichtet werden, im Modell abgebildet.

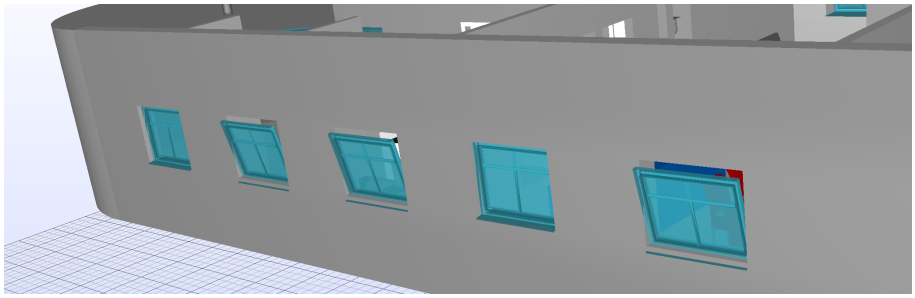

Abbildung 2.5: Zustandsinformationen über Fenster im IFC-Modell

Hierzu wird auf dem Topic `WINDOW.CONTROL` auf Steuernachrichten gewartet. Anschließend werden die Informationen über das eigens erstellte IFC Attribut `WindowPosition` den entsprechenden Entitäten zugewiesen. Wie in Abb. 2.5 S. 12 zu sehen ist, wird eine Öffnung symbolisch im Viewer dargestellt.

Finalisierungsphase

Vor dem Beenden des ISIS werden alle Threads abgeschlossen, überflüssige Elemente (wie z.B. Ubisense-Tags) aus dem Modell entfernt und das Modell auf den BiMserver geladen.

3 Arbeiten im Living Place Projektumfeld

3.1 Ubisense – das Ortungssystem für den Living Place

Genaue Ortsinformationen im Living Place [2] sind für einige Projekte – neben diesem – von erheblicher Bedeutung [1],[7],[9]. Hierzu wurde von Sören Voskuhl und Kjell Otto [9] die Installation des Ortungssystems Ubisense im Living Place vorgenommen.

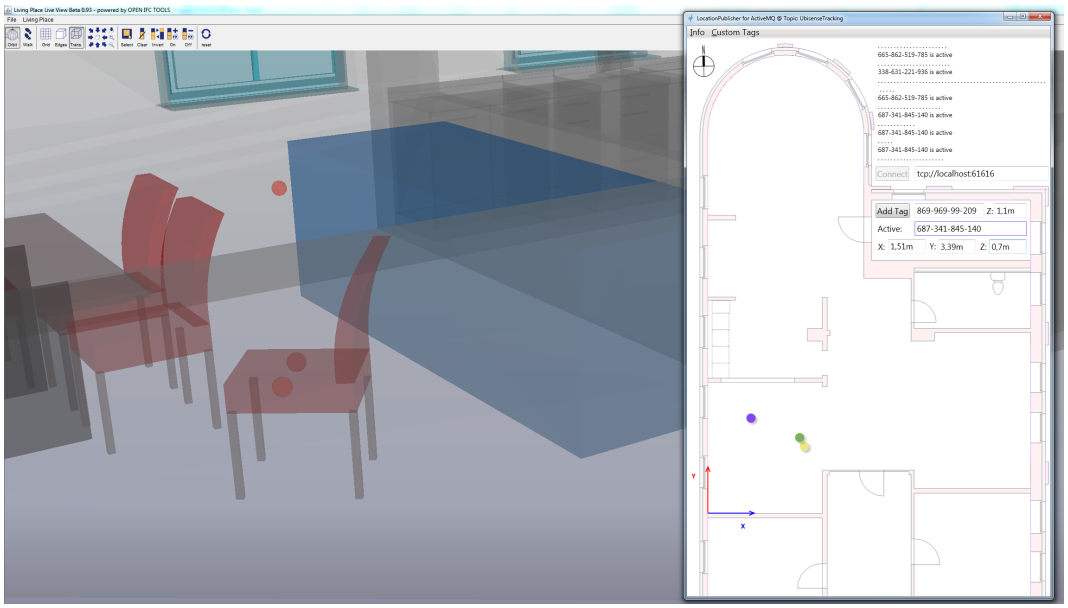

Abbildung 3.1: Der `MouseMoveLocationPublisher` (r.) ist ein *Mockup* für Ortsnachrichten – die Living Place Visualisierung (l.) verwendet diese um z.B. die Position von Mobiliar zu bestimmen

Der Autor hat auf Basis eines von Sören Voskuhl entwickelten Programms zum Auslesen der Ubisense-Ortsinformationen die Integration des ActiveMQ vorgenommen. Die .NET Anwendung *LocationPublisher* sendet nun über das Topic `UbisenseTracking` Positionsdaten von Ubisense-Tags (je eine Nachricht mit einer Angabe in *Metern* und eine in *Zentimetern*)[1] und ist bereits im Living Place aktiv.

[1] Details unter http://livingplace.informatik.haw-hamburg.de/wiki/index.php/ActiveMQ_Messages

Für *lokales Testen* von Anwendungen, die auf Ubisense Ortsinformationen zurückgreifen, wurde in [4] bereits ein Mockup erstellt, das in diesem Projekt erweitert wurde (s. r. in Abb. 3.1). Da in der aktuellen Projektphase der Koordinatenursprung sowie der Projektnorden (ebenfalls rechts in Abb. 3.1 zu sehen) gesetzt wurden, fand eine generelle Umstrukturierung statt. Mit Hilfe der Visualisierung und des LocationPublisher-Mockups konnte bereits ein Fehler in den Routinen zur Positionsberechnung entdeckt und behoben werden. Das Mockup ist somit für alle Entwickler interessant, die ihre Orts-basierte Anwendung lokal testen wollen.

3.2 Erweiterung des Living Place Modells

Das Living Place IFC-Gebäudemodell wurde mit *Revit Structure* und *Revit Architecture* bereits in den Vorarbeiten realisiert. Das Eigenstudium dieser Tools reichte für einige Anforderungen jedoch nicht gänzlich aus. Die Teilnahme des Autors am Kurs „Autodesk Revit Structure Schulung“[2] der Fakultät für Bau- und Umweltingenieurwissenschaften der Ruhr Universität Bochum vermittelte die notwendigen Kenntnisse. Im Folgenden werden die Bereiche angesprochen, die vom Besuch profitierten.

Der Living Place wurde während des Kurses komplett neu auf Basis eines DWG Stockwerksplans modelliert. Durch die vektorielle Darstellung dieses Planes konnten die dreidimensionalen Wände nun präzise auf dem zweidimensionalen Plan platziert werden.

Es Bestand im Vorfeld die Frage wie *Räume* im Modell definiert werden können. Räume haben im IFC Standard keine visuelle Ausprägung, sind aber Teil der topologischen Struktur – ihre Ausmaße sind zwar hinterlegt, werden aber nicht explizit aus den anliegenden Wänden definiert. Insofern konnten nun Bereiche im Living Place als Räume (*IfcSpace*) deklariert werden (siehe Abb. 3.2), die auch als gemeinsame begriffliche Grundlage in der Ontologie festgehalten wurden.

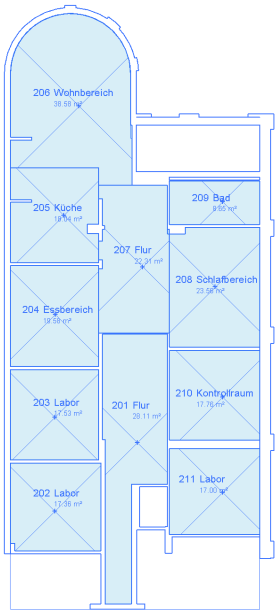

Abb. 3.2: Raumaufteilung des Living Places

Folgende Punkte konnten außerdem geklärt werden:

- Modellierung eigener Gegenstände (z.B. Mobiliar)
- Verschieben und Rotieren des Koordinatensystems

[2] http://www.inf.bi.ruhr-uni-bochum.de/index.php?option=com_content&task=view&id=260&Itemid=81

- Modellierung der Deckenbalken für eine verbesserte Lichtsimulation
- Erkennen und Beheben von *Clashes* (Überschneidungen)
- Weitere Stockwerke erstellen
- ...

Im Gespräch mit Dr. Kai Erlemann über das Konzept von Property-Sets in IFC wurde u.a. deutlich, wie man eigene Projektparameter in das Modell einbetten und verwalten kann.

Ohne den Besuch des Kurses an der RUB wäre es dem Autor erheblich schwerer gefallen die Fragen zur Modellierung und zum IFC Standard zu klären – insofern kann der Besuch als Erfolg gewertet werden. Es gilt nun die erworbenen Kenntnisse für nachfolgende Studentengenerationen an der Hochschule für Angewandte Wissenschaften Hamburg zu bewahren.

3.3 Kameras im Living Place

Zusammen mit Hosnia Najem und Sebastian Rudolf wurden die Positionen der drei 180°-Panorama-Kameras und der sechs steuerbaren PTZ-Dome-Kameras[3] vermessen. Alle Ortsangaben beziehen sich auf den vereinbarten Koordinatenursprung – gemessen wurde jeweils an unterster Stelle der Okulare (siehe Abb. 3.3).

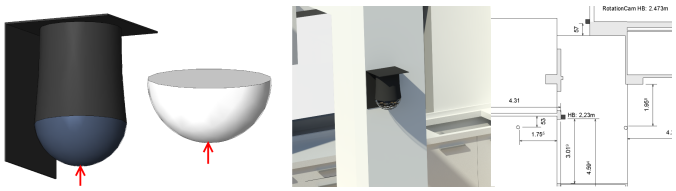

Abbildung 3.3: v.l.n.r.: Messpunkte, Rendering, Modelleintrag

Diese wurden anschließend in 3D modelliert und dem Living Place Modell hinzugefügt. Ein Rendering der Kameras im Living Place wurde zusätzlich auf Anfrage angefertigt. Außerdem wurden die Kameras und ihre Positionen der Projektontologie mit dem Tool *Protege* hinzugefügt.

Genaue Messungen der Kamerapositionen sind insbesondere für *Szenario 1* erforderlich, da hier auf Grund von Bilderkennungsverfahren wie SIFT oder SURF [6] eine möglichst exakte Erkennung der Position eines Gegenstandes im dreidimensionalen Raum erfolgen muss.

[3] PTZ: Pen Ten Zoom

4 Schluss

Der *Indoor Spatial Information Service* wurde in diesem Projektsemester in großen Teilen fertiggestellt. Hierzu zählt insbesondere die komplette Realisierung von *Szenario IIa*. Es wurden weiterhin beispielhaft Zustandsinformationen über *Fenster* (vgl. *Szenario IIb*) in das Modell übertragen – zukünftig können auf einfache Weise zusätzliche Datenquellen integriert werden (z.B. Türen [offen, geschlossen], Lichtquellen und Haushaltsgeräte [ein-, ausgeschaltet] etc.).

Die Grundlagen für das recht komplexe System *Indoor Spatial Information Service* sind gelegt. Die Evaluation im Living Place wird zeigen ob es dem Praxistest standhält – der Autor ist da guter Dinge.

4.1 Ausblick

Auf Grund der Komplexität von *Szenario I* wurde von einer „schnell gestrickten“ Implementierung räumlicher Operatoren abgesehen. Stattdessen wird die Realisierung räumlicher Operatoren als Ziel der Masterarbeit angegangen. Auf Basis einer Graphen-basierten Datenbank und dem Octree Algorithmus wurde in [5] vom Autor bereits ein Lösungsansatz vorgestellt.

Außerdem ist die Implementierung von Kenntnissen aus dem Bereich der *Spatial Cognition* interessant, um bspw. die Rückgabemenge einer räumlichen Anfrage auf prominente Objekte zu filtern, um damit dem Bewohner eine menschlichere, intuitivere Antwort auf eine Suchanfrage geben zu können.

Des Weiteren wird untersucht werden, in wie weit der ISIS in Form eines Brokers für semantische Informationen Verwendung finden kann – auch hierfür wurden die Grundlagen (in Form des Pull-Mechanismus) mit diesem Projekt bereits gelegt.

Literaturverzeichnis

[1] Jens Ellenberg. Klassifizierung von kontext in einer intelligenten wohnung. Technical report, HAW Hamburg, 2011.

[2] S. Gregor, M. Rahimi, M. Vogt, T. Schulz, and K.v.Luck. Tangible computing revisited: Anfassbare computer in intelligenten umgebungen. Technical report, Hochschule für Angewandte Wissenschaften Hamburg, 2009.

[3] Benedikt Johannsen and Alexander Pautz. Fenster & heizungssteuerung. Wikieintrag, Hochschule für Angewandte Wissenschaften Hamburg, 2011.

[4] Bastian Karstaedt. Visualisierung von ifc gebäudemodellen unter verwendung einer game-engine. Präsentation, HAW-Hamburg, 2010.

[5] Bastian Karstaedt. Entwicklung eines indoor spatial information services für smart homes auf basis der industry foundation classes. Technical report, HAW-Hamburg, August 2011.

[6] Hosnia Najem. Modellbasiertes suchen von objekten. Technical report, Hochschule für Angewandte Wissenschaften Hamburg, 2010.

[7] Hosnia Najem. Projekt 1 – aufbau der infrastruktur für die modell-basierte objektsuche im living place hamburg. Technical report, Hochschule für Angewandte Wissenschaften Hamburg, 2011.

[8] Kjell Otto and Sören Voskuhl. Projektbericht sommersemester 2010 – entwicklung einer architekture für den living place hamburg. Technical report, HAW-Hamburg, 2010.

[9] Kjell Otto and Sören Voskuhl. Projektbericht wintersemester 10/11 – weiterentwicklung der architektur des living place hamburg. Technical report, HAW-Hamburg, 2010.

[10] Jan Tulke, Eike Tauscher, and Michael Theiler. Open ifc tools – processing / visualisation / 4d. Zuletzte gesichtet am 22.11.2010.

[11] Leon van Berlo. Bimserver – open source building information modelserver. Webseite, Juli 2010. Letzter Aufruf: 12.08.2010.