Einsatz von FoReST
zur Entwicklung einer verteilten Lichtsteuerung

Reinhard Gotzhein, Christian Peper

Computer Networks Group
Fachbereich Informatik, Universität Kaiserslautern
URL: http://rn.informatik.uni-kl.de/~silicon/
Email: {gotzhein,peper}@informatik.uni-kl.de

FBT'2001 - Formale Beschreibungstechniken für verteilte Systeme
11. GI/ITG-Fachgespräch, Bruchsal, 21.-22. Juni 2001

1 Die Fallstudie

In der Fallstudie wird ein Ausschnitt der 4. Etage des Gebäudes 32 der Universität Kaiserslautern zugrunde gelegt (s. Abb. 1). Darin sind zwei Büroräume sowie ein Flurabschnitt dargestellt. Die Büroräume verfügen über je zwei Lichtgruppen, die separat über Schalter betätigt werden können. Weiterhin gibt es je zwei Fenster, die sich mit Sonnenblenden einzeln abdunkeln lassen. Bewegungsmelder und Türkontakte ergänzen die Basisinstrumentierung.

Als Zusatzausstattung ist ein User Control Panel vorgesehen, über das sämtliche Grundfunktionen des Raums zugänglich sind; zusätzlich sollen Lichtszenarien frei definierbar und abrufbar sein. Für das Gesamtmodell ist weiterhin ein Facility Manager Control Panel zu realisieren, über das sämtliche Einzelfunktionen der Gebäudeteile sowie zusätzlich Sammelfunktionen (z.B. Ein-/Ausschalten aller Lichter, Schließen aller Sonnenblenden) bereitzustellen sind.

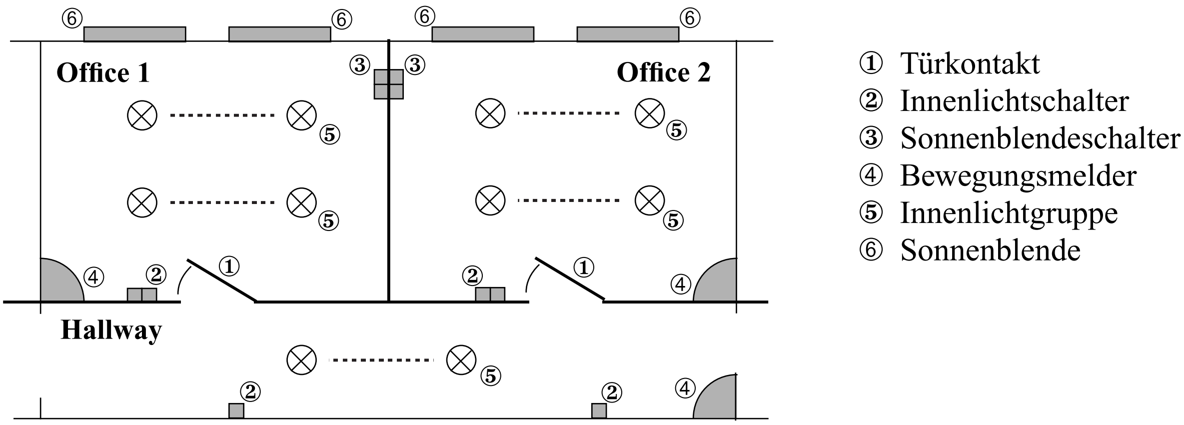

Abbildung 1: Grundriss des Gebäudemodells

Die untersuchten Anforderungen (s. Tab. 1) basieren auf einer vorangegangenen, im SFB 501 entwickelten Fallstudie [6], in der zusätzliche Anforderungen an die Lichtsteuerung sowie an die Temperatursteuerung der gesamten Etage gestellt werden. Ziel der in diesem Beitrag vorgestellten Fallstudie war der Nachweis, dass die im Teilprojekt "Generische Kommunikationssysteme" des SFB 501 entwickelten Ansätze syntaktisch und semantisch integrierbar sind. Dazu wurden sämtliche Entwicklungsphasen, beginnend mit der Anforderungsanalyse und endend mit der Bereitstellung des implementierten Gebäudemodells, durchlaufen. Die Ergebnisse der Fallstudie sind unter http://rn.informatik.uni-kl.de/~silicon/ dokumentiert.

1.	The general user needs for each kind of room are:	
	U1	If a person occupies a room, there has to be *safe illumination*, if nothing else is desired by the *chosen light scene*.
	U2	As long as the room is occupied, the chosen light scene has to be maintained.
	U3	If the room is reoccupied within T1 minutes after the last person has left the room, the chosen light scene has to be reestablished.
	U4	If the room is reoccupied after more than T1 minutes since the last person has left the room, the *default light scene* has to be established.
	U5	For each room, the chosen light scene can be set by using the user control panel.
	U6	...
2.	The user needs concerning the offices are:	
	U23	The ceiling light groups should be maintained by the control system depending on the *current light scene*.
	U24	A user control panel in an office should be movable as is a telephone.
	U25	...

Tabelle 1: Auszug aus der Problembeschreibung

2 Prozessmodell

In der Analysephase wurde das in Abb. 2 dargestellte Prozessmodell zur Ermittlung und Präzisierung von Anwendungsanforderungen eingesetzt [2] [5]. An dieser Stelle seien drei Aspekte des Prozessmodells besonders hervorgehoben:

- Die Anforderungsanalysephase verläuft iterativ, wobei jede Iteration neben der Formalisierung ein Review mit Kundenbeteiligung vorsieht. Das Eingabedokument für das Kunden-Review wird aus der Anforderungsspezifikation durch eine Projektion, bei der formale Anteile ausgeblendet werden, gewonnen. Um die Abweichung zwischen formaler und natürlichsprachlicher Version so klein wie möglich zu halten, wird letztere bei der Formalisierung durch einen quasi-Übersetzungsvorgang aus der formalen Version erzeugt.
- Ergebnis der Analysephase ist eine formale Anforderungsspezifikation, die direkt als Eingabe für die Entwurfsphase dient. Diese formale Spezifikation liegt bei positivem Abschluss des Kunden-Reviews bereits fertig vor, muss also nicht mehr entwickelt werden, so dass potentielle Fehlerquellen weitgehend vermieden werden.
- Zur Formalisierung der Anforderungsbeschreibung werden musterbasierte Ansätze eingesetzt. Ausgangspunkt ist eine Sammlung von Requirement Patterns, die aufgrund von Aussagen der Anforderungsbeschreibung selektiert, adaptiert und schließlich komponiert werden.

Das Prozessmodell ist unabhängig von der eingesetzten Spezifikationstechnik. Es wurde zur Durchführung von Fallstudien mit der Spezifikationstechnik FoReST [1] instanziiert.

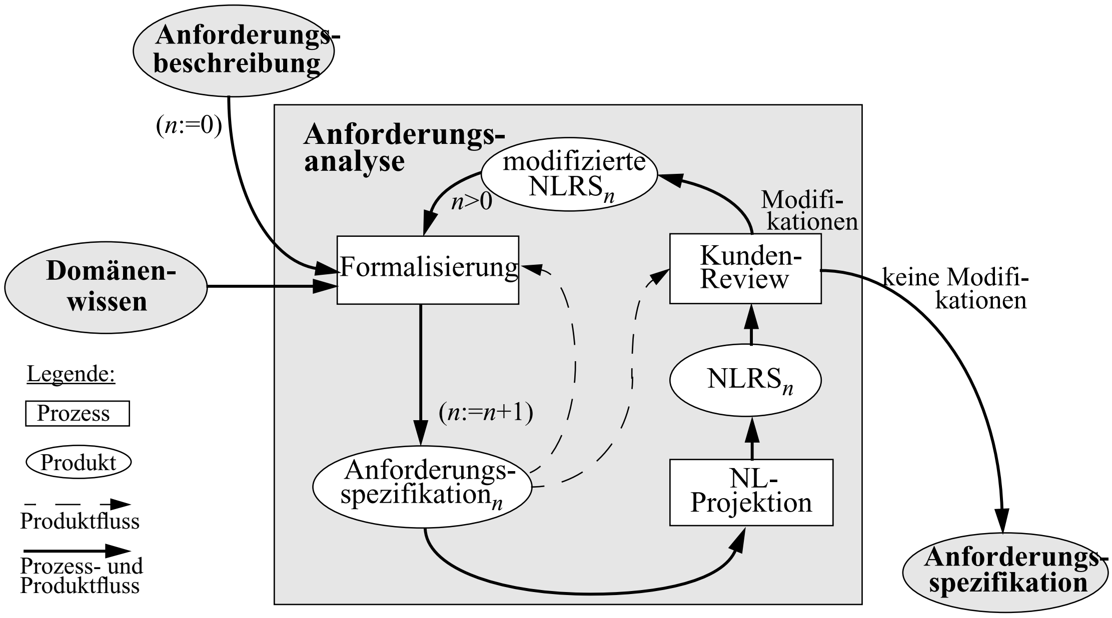

Abbildung 2: Anforderungsanalyseprozess

3 Formal Requirement Specification Technique (FoReST)

Zur formalen Spezifikation von Anwendungsanforderungen wurde die auf das Anwendungsfeld zugeschnittene *Formal Requirements Specification Technique* (*FoReST)* [1], die objektorientierte Konzepte mit der Realzeit-Logik tRTTL (tailored Real Time Temporal Logic) integriert, entwickelt und eingesetzt. FoReST-Spezifikationen bestehen aus einer Sammlung von Klassendefinitionen, wobei die aus der objektorientierten Modellierung bekannten Assoziationen wie Aggregation, Komposition und Spezialisierung (Vererbung) direkt unterstützt werden. Abweichend von den bekannten Techniken werden die Objekte einer Klasse jedoch nicht durch Operationen näher charakterisiert, sondern durch ihre Eigenschaften. Zur Spezifikation dieser Eigenschaften wird die bereits erwähnte Logik tRTTL eingesetzt. In Verbindung mit Spezialisierung und Vererbung hat diese Vorgehensweise u.a. den Vorteil, dass die Substitutionseigenschaft erhalten bleibt.

Die Struktur der Problemspezifikation "Verteilte Lichtsteuerung" ist in Abb. 3 als UML-Klassendiagramm repräsentiert. Der darin hervorgehobene Anteil "*SwitchedLight*", der sowohl in Büroräumen als auch im Flur auftritt, wird nun zur Illustration des FoReST-Ansatzes genauer vorgestellt. Das UML-Klassendiagramm wird hier lediglich der Übersicht halber angegeben, es ist nicht Bestandteil der FoReST-Spezifikation.

3.1 Spezifikation von Klassen mit FoReST

Der Einsatz objekt-orientierter Analysemethoden führt zu einem statischen Architekturmodell, das in der vorliegenden Fallstudie durch die Struktur der Systemumgebung geprägt ist. Für eine Menge von Objekten, die dieselben Merkmale aufweisen, wird jeweils eine Klasse eingeführt.

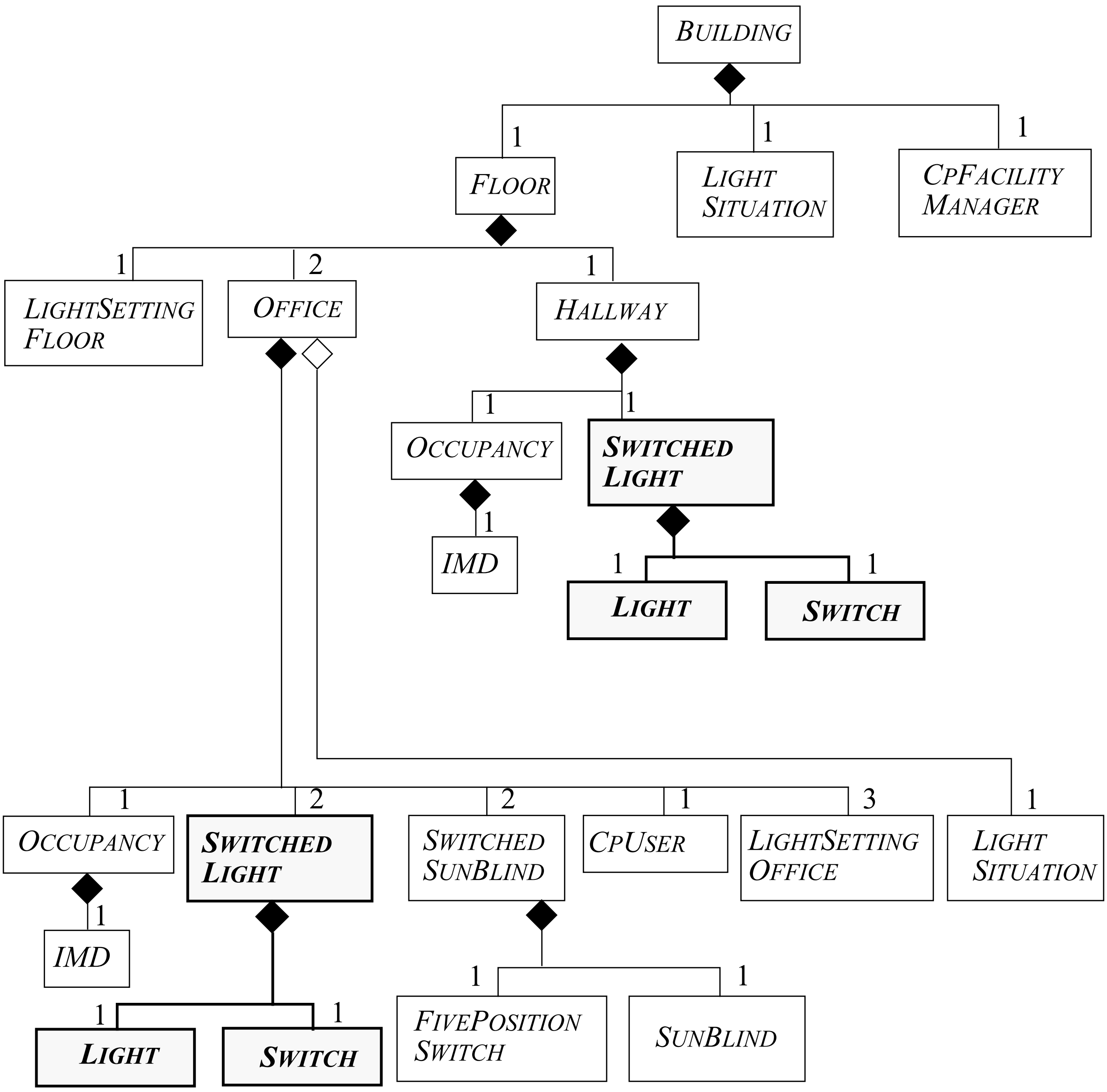

Abbildung 3: UML Klassendiagramm

In FoReST werden Klassen durch einen eindeutigen Klassenname, eine Signatur sowie Eigenschaften, die das zulässige Verhalten der Objekte beschreiben, charakterisiert.

Die *Signatur* einer Klasse ist eine Folge von Attributdeklarationen, wobei jede Deklaration einen Attributnamen, eine Klassifikation, eine Absicht sowie einen Bereich festlegt. Attribute können z.B. als Prädikate, Funktionen oder Objekte eingeführt werden. Funktionen und Prädikate sind weiter als statisch oder dynamisch klassifizierbar, abhängig davon, ob ihre Werte zeitabhängig sind.

In Tab. 2 wird die Klasse `LIGHT` mit der Absicht, von konkreten Lichtquellen zu abstrahieren, eingeführt. Alle Instanzen der Klasse besitzen das Attribut `lightValue`, das als zeitbehaftete

Description Class *LIGHT*	
Intention	: An abstract, dimmable light (actuator).
Signature	
Domain *LIGHTVALUES = {0..15}*	
Intention	: These are the possible values for setting the light: 0 = off, 15 = on
Timed Function *lightValue:* → *LIGHTVALUES*	
Intention	: The current light value.
Scope	: machine controlled, visible
Machine Statements	
Property *M1*	
Formal	: $\Box(lightValue = 15)$
NL	: Initially, the light is on, and remains on unless determined otherwise by properties in aggregating classes.

Tabelle 2: Klasse *LIGHT*

Funktion mit Werten im Intervall [0..15] klassifiziert wird. In Tab. 3wird die Klasse *SWITCH* mit zwei Attributen spezifiziert.

Description Class *SWITCH*	
Intention	: An abstract, binary switch (sensor).
Signature	
Domain *POSITIONS = {on,off}*	
Intention	: Positions of the switch.
Time Constant *ep = 0.1 sec*	
Intention	: Minimum time distance between two events (*event period*).
Timed Function *observedPosition:* → *POSITIONS*	
Intention	: The position of the switch.
Scope	: environment controlled, visible
Domain Knowledge	
Property *D1*	
Formal	: $\Box\ \forall p \in \mathit{POSITIONS}: ([observedPosition = p] \rightarrow \Box_{\leq ep}\ observedPosition = p)$
NL	: The minimum time distance between two events is *ep*. Here: it takes at least *ep* to change the position of the switch.

Tabelle 3: Klasse *SWITCH*

Zusätzlich zur Signatur wird das Verhalten der Objekte formal spezifiziert. Anders als in Entwicklungssprachen wie etwa UML, die lediglich eine Liste von Operationnamen angeben, wird hier das zulässige Verhalten implizit durch eine Menge von Eigenschaften spezifiziert. Eigenschaftsorientierte Spezifikationen sind zur Erfassung von Anforderungen besonders geeignet, da sie die Abstraktion von irrelevanten Details unterstützen. Ferner erhalten sie den direkten

Bezug zur ursprünglichen informellen Problembeschreibung, sofern diese in demselben Stil abgefasst ist (s. Kap. 1).

Die Eigenschaften sind im Beispiel in drei Kategorien klassifiziert, u.z. Domänenwissen, Anforderungen und Maschineneigenschaften. Diese Klassifikation resultiert aus dem zugrundegelegten Referenzmodell für Problembeschreibungen und wird in [3] näher erläutert. Die Definition der Klasse *LIGHT* enthält eine Maschineneigenschaft, die den Default-Wert für die zeitbehaftete Funktion *lightValue* festlegt.

3.2 Spezifikation von Komposition mit FoReST

Komplexe Objekte lassen sich vielfach als zusammengesetzte Objekte modellieren, wobei die Komposition der Teile zu neuen Merkmalen führen kann. Wie schon zuvor wird für komplexe Objekte, die dieselben Merkmale aufweisen, eine Klasse eingeführt. Die Teile eines komplexen Objekts werden dabei durch spezielle Attribute, die als "Objekte" klassifiziert und mit einer Klasse assoziiert sind, beschrieben. Die Instanziierung einer Klasse führt automatisch zur Instanziierung der Objektteile.

Description Class ***SWITCHEDLIGHT***		
Intention	:	An abstract, switched light.
Signature		
Object *light: LIGHT*		
Object *switch: SWITCH*		
Time Constant *rd = 0.3 sec*		
Intention	:	Maximum delay until positions of the switch must lead to a reaction of the light (*reaction delay*).
Machine Statements		
Property *M1*		
Formal	:	$\Box\, \forall v \in \mathit{LIGHTVALUES}: ([\mathit{lightValue} = v] \rightarrow (\mathit{lightValue} = v\ \mathcal{W}\ \exists p \in \mathit{POSITIONS}: [\mathit{switch.observedPosition}=p]))$
NL	:	If the light is set to some value, it remains so unless the position of the switch is changed.
Property *M2*		
Formal	:	$\Box\, ([\mathit{switch.observedPosition}=\mathit{off}] \rightarrow \Diamond_{\leq rd}\ \mathit{light.lightValue}=0)$
NL	:	When the switch enters position *off*, the light is switched off with a maximum delay of *rd*.
Property *M3*		
Formal	:	$\Box\, ([\mathit{switch.observedPosition}=\mathit{on}] \rightarrow \Diamond_{\leq rd}\ \mathit{light.lightValue}=15)$
NL	:	When the switch enters position *on*, the light is switched on with a maximum delay of *rd*.

Tabelle 4: Komposition: Klasse *SWITCHEDLIGHT*

In Tab. 4 wird die zusammengesetzte Klasse *SWITCHEDLIGHT* spezifiziert. Jede Instanz dieser Klasse aggregiert je ein Objekt der Klassen *LIGHT* und *SWITCH*. Dazu kommen ein weiteres Attribut sowie mehrere Maschineneigenschaften, die auf die Komponenten Bezug nehmen. Die

Eigenschaften *M1* bis *M3* spezifizieren Abhängigkeiten zwischen Attributen der Komponenten `switch` und `lightValue`, die erst durch die Komposition dieser Komponenten zustande kommen.

Aus Eigenschaft *M2* geht hervor, wie das Licht ausgeschaltet werden kann. Dies steht offensichtlich im Widerspruch zur Eigenschaft *M1* der Klasse *LIGHT*, die verlangt, dass das Licht immer eingeschaltet bleibt. Inkonsistenzen dieser Art kommen dadurch zustande, dass Eigenschaften aggregierter Objekte durch Eigenschaften aggregierender Klassen überlagert werden. Diese Inkonsistenzen werden im FoReST-Ansatz durch Priorisierung der Eigenschaften der aggregiereden Klassen aufgelöst.

4 Kontext

Die hier vorgestellte Problemspezifikation ist Teil einer durchgängigen Fallstudie, in der eine komplette verteilte Lösung maßgeschneidert wurde. Im Mittelpunkt standen dabei die Kommunikationslösungen, von der Kommunikationsmiddleware bis zur Kommunikationshardware incl. Treiber. Zur Maßschneiderung kamen neuartige Methoden zum Einsatz, die die Wiederverwendung von Entwurfs- und Implementierungslösungen zum Ziel haben, insbesondere musterbasierte Entwurfstechniken sowie Techniken zur automatischen Generierung von Implementierungen. Einzelheiten zu den Kommunikationslösungen der Fallstudie sind [7] zu entnehmen.

5 Literatur

[1] Kronenburg, M., Peper, C.: *Application of the FOREST Approach to the Light Control Case Study*, Journal of Universal Computer Science (J.UCS), Special Issue on "Requirements Engineering: The Light Control Case Study", Springer, 2000

[2] Gotzhein, R., Kronenburg, M., Peper, C.: *Reuse in Requirements Engineering: Discovery and Application of a Real-Time Requirement Pattern*, 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98), Lecture Notes in Computer Science 1486, Springer, Lyngby, Denmark, September 14-15, 1998. pp. 65-74

[3] Kronenburg, M., Peper, C.: *Definition and Instantiation of a Reference Model for Problem Specifications*, 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99), Kaiserslautern, 1999

[4] Peper, C.: *Transformations in Pattern-Based System Specifications*, 9. GI/ITG-Fachgespräch "Formale Beschreibungstechniken für verteilte Systeme (FBT'99), München, Herbert-Utz-Verlag, 1999

[5] Peper, C., Gotzhein, R., Kronenburg, M.: *A Generic Approach to the Formal Specification of Requirements*. Proceedings of the 1st IEEE International Conference on Formal Engineering Methods (ICFEM'97), Hiroshima, Japan, November 1997, pp. 252-261

[6] Queins, S., Zimmermann, G., Becker, M., Kronenburg, M., Peper, C., Merz, R., Schäfer, J.: *The Light Control Case Study: Problem Description*, Special Issue of the Journal of Universal Computer Science, Springer, July 2000

[7] Schaible, P., Thees, J.: *Maßschneiderung vs. Wiederverwendung in Kommunikationssystemen*, 11. GI/ITG-Fachgespräch "Formale Beschreibungstechniken für verteilte Systeme (FBT'2001), Bruchsal, 21.-22.6.2001