Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Diplomarbeit

Michael Stickel

Konstruktion eines mobilen Systems zur Steuerung von Bühnenbeleuchtungsanlagen.

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Michael Stickel

Thema der Diplomarbeit

Konstruktion eines mobilen Systems zur Steuerung von Bühnenbeleuchtungsanlagen.

Stichworte

Bühnenbeleuchtungsanlagen, Beleuchtungssteuerung, Client-Server, mobile Clients

Kurzzusammenfassung

Für die Beleuchtungssteuerung in Theatern, Opernhäusern, Veranstaltungshallen und in Fernsehstudios werden Lichtstellpulte verwendet, die von der Software und der Hardware speziell auf die jeweilige Anwendung ausgelegt sind. Diese Pulte werden von Beleuchtern eingesetzt, um die Helligkeit, die Farbe sowie andere Parameter der Scheinwerfer zu steuern. In der Regel sind die Beleuchtungssysteme als zentrale Steuerungen aufgebaut, die ihre Steuerbefehle an die angebundenen Scheinwerfer weitergeben. Während der Proben und der Aufbauphasen einer Veranstaltung ist es die Aufgabe der Beleuchter, die Scheinwerfer so einzurichten, dass sie für die Veranstaltung genutzt werden können. Die Diplomarbeit befasst sich mit der Konstruktion eines Beleuchtungssystem, das, ausgerichtet an einem bereits existierenden System, den Beleuchter in seiner Arbeit unterstützen soll.Ziel ist es, zwecks Effizienzsteigerung Arbeitsschritte zu vereinfachen und neue technische Möglichkeiten aufzuzeigen. Konkret wird durch eine Client-Server-Architektur mit mobilen Clients die Unabhängigkeit des Beleuchters von der Position des zentralen Beleuchtungsstellwerks erreicht.
Michael Stickel

Title of the paper

Construction of a mobile system for the control of stagelighting-systems.

Keywords

Stagelightingsystem, Lightcontrol, Client-Server, mobile Clients

Abstract

For lightcontrol in theatres and opera houses, for events and in tv studios there are lightconsoles used, that are specially line up with the respective application. These lightconsoles are for controlling the brightness, the color and other functions of the spotlights. Generally the lighting systems are central controlled. From the central point of controlling the instructions are given to the individual lights. During rehearsals and before events or performances the lighting technicians have to adjust the spotlights. The dissertation looks into a construction of a lighting system to support the lighting technician in his work. The intention is on the one hand to simplify, on the other hand to show new technicial possibilities - both for the improvement of efficiency. The independence of the lighting technician from the position of the lightconsole is desirable and realized by a client server architecture with mobile clients.

Inhaltsverzeichnis

1. Einleitung

Denn die einen sind im Dunkeln

und die anderen sind im Licht

Und man siehet die im Lichte,

die im Dunkeln sieht man nicht.

Bertold Brecht

Die obige Strophe aus der Dreigroschenoper von Brecht soll als “Parabel” dienen für das Thema, mit dem sich diese Diplomarbeit beschäftigt. Es geht um den Bereich der Beleuchtung, genauer: der Bühnenbeleuchtung.

Die wenigsten Zuschauer, die heute ein Theater oder eine Oper betreten, um einer Vorstellung beizuwohnen, ahnen, wie komplex die “Maschinerie” ist, die zum Gelingen der Darbietung beiträgt. Ein ganzer “Apparat” von Mitarbeitern ist notwendig, um einem Stück den ihm gebührenden Auftritt zu verschaffen: Neben den - buchstäblich - vordergründigen Schauspielern und ihren Regisseuren haben Inspizienten, Bühnenarbeiter, Maskenbildner, Schneider, Tontechniker und Lichttechniker - um nur einige zu nennen - ihre Arbeit einfließen lassen.

Mit dem Tätigkeitsbereich der Letztgenannten wird sich im Rahmen dieser Arbeit näher zu beschäftigen sein.

Um die Bedürfnisse eines Beleuchters in Bezug auf die Technik mit der er arbeitet besser verstehen zu können, ist es wichtig, sein Arbeitsfeld näher in Augenschein zu nehmen. Weiterhin interessieren die Größenordnungen im Bereich der technischen Ausstattung heutiger Theater, Opern oder Konzertproduktionen.

Die technische Ausstattung im Lichtbereich einer modernen Produktion umfasst durchschnittlich:

- 2-3 Beleuchtungsstellwerke
- 200-300 Dimmer mit angeschlossenen Scheinwerfern.
- 15-30 Multifunktionsscheinwerfer (Moving-Lights)

Der durchschnittliche Leistungsbedarf liegt bei ungefähr 450.000 VA (Watt), was einer Anschlussleistung von etwa 10-15 Haushalten entspricht.

Die Dimmer zum Ansteuern von Scheinwerfern, die im Bereich der Bühnenbeleuchtung verwendet werden, weisen generell eine Leistung von mindestens 2000 Watt auf; in Theatern werden im allgemeinen Dimmer mit einer Leistung von 5000 Watt verwendet.

Abbildung 1.1.: Sicht auf die Seitengalerie einer Studiobühne

Abbildung 1.1[1] zeigt einen Blick auf die Seitengalerie einer Studiobühne. Zu erkennen sind die Scheinwerfer[2], die sich längs oberhalb des Zuschauerraums aufreihen.

In einem Theater befinden sich zusätzlich Scheinwerfer im Bühnenhaus, auf der Seitenbühne, der Hinterbühne sowie in den Beleuchtungszügen oberhalb der Bühne. In den meisten Theaterstücken werden zudem mobile Scheinwerfer eingesetzt, die stückspezifisch an entsprechenden Positionen zum Einsatz kommen.

Jeder einzelne Scheinwerfer, summa summarum oft 200 bis 300 an der Zahl, muss vor Beginn einer Vorstellung kontrolliert werden. Man überprüft die Funktion und stellt die Position, die Ausrichtung, die Farbe und die Größe des Lichtkegels ein. Es muss gewährleistet sein, dass bei allen Aufführungen desselben (Theater-)Stücks ein gleichbleibender visueller Eindruck entsteht.

Auf Abbildung 1.2[3] ist die Schaltzentrale der Beleuchtung zu sehen. Von dieser Stelle, von der sogenannten “Licht-Regie” aus, werden alle Scheinwerfer gesteuert. Die Licht-Regie befindet sich in den meisten Theatern in halber Höhe im hinteren Bereich des Zuschauerraums und besitzt ein Sichtfenster auf selbigen. Hier laufen alle Anfragen und Aktionen zum Thema “Licht” zusammen.

Insbesondere während der Proben kann es in diesem Bereich zu Engpässen kommen, wenn mehrere, konkurrierende Anfragen an die Person gestellt werden, die das Beleuchtungspult, das sogenannte “Beleuchtungsstellwerk”, bedient.

Der in Theatern unbedingt einzuhaltende, strikte Ablauf während der Vorbereitung einer Aufführung darf nicht durch Pannen oder defektes Material unterbrochen werden. Andernfalls

[1] Zur Verfügung gestellt mit freundlicher Unterstützung der Firma ADB Lighting Technologies.
[2] Zum Größenvergleich, die Scheinwerfer haben eine Länge von etwa einem Meter.
[3] Zur Verfügung gestellt mit freundlicher Unterstützung des Thalia-Theaters Hamburg.

Abbildung 1.2.: Die Schaltzentrale der Beleuchtung, das Stellwerk

riskiert man einen für das Publikum spürbaren - und damit öffentliche Kritik herausfordernden - zeitlichen Verzug.

Der Arbeits-Ablauf beginnt morgens mit dem Aufbau der Probe. Dazu gehört das Aufstellen der Bühnenteile, das Einrichten des Tons, sowie das Aufbauen und Einrichten der benötigten Scheinwerfer.

Die auf der Hauptbühne stattfindenden Proben dienen im Theater in der Regel nicht mehr dem Einüben des Textes und der Art seiner Präsentation, sondern sind, neben den Arbeiten mit dem Bühnenbild und den Kostümen, zur Prüfung der musikalischen Einspielungen und dem Durchgang und Test der Lichteinstellungen gedacht.

In den Lichtproben geht der Regisseur zusammen mit einem Light-Designer alle Szenen eines Theaterstückes Schritt für Schritt durch und bespricht mit ihm die einzelnen Lichteinstelllungen. Diese Lichtstimmungen werden dann von einem Beleuchter in ein Beleuchtungsstellwerk eingegeben.

Der Stellwerksbeleuchter befindet sich entweder in der Licht-Regie oder zusammen mit dem Beleuchtungsstellwerk in der sogenannten “Dritten Reihe”. Die “Dritte Reihe” ist nicht notwendigerweise mit der dritten Stuhlreihe identisch, dennoch ist der Begriff inzwischen in das Fachvokabular eingeflossen. Die “Dritte Reihe” als Arbeitsplatz des Stellwerksbeleuchters bietet den Vorteil, dass er hier in direktem Kontakt zum, in den Proben regelmäßig hier sitzenden, Light-Designer steht.

Doch der Stellwerksbeleuchter ist nicht der einzige “Licht-Schaffende”: Ein oder mehrere Beleuchter sitzen am Verfolger, einem speziellen, nur von Hand zu bewegenden Scheinwerfer,

um zu gegebener Zeit einzelne Schauspieler oder Objekte auf der Bühne ins rechte Licht setzen zu können.

Aufgabe weiterer ein bis zwei Beleuchter ist es, den Anweisungen des Light-Designers folgend, die oben bereits angesprochenen mobilen Scheinwerfer aufzustellen und einzurichten.

In der Proben-Phase zählen Flexibilität und vor allem zügiges Arbeiten zu den Ansprüchen, die an die Beleuchter gestellt werden.

Nach den zumeist anstrengenden und langwierigen Proben beginnt der Aufbau für die Abendvorstellung. Diese zweite Arbeits-Phase erstreckt sich über mehrere Stunden bis zum Beginn der Vorstellung. Die letzten Einstellungen der Scheinwerfer werden häufig bei heruntergelassenem Vorhang noch wenige Minuten vor Beginn der Aufführung vorgenommen. Ebenso finden die ersten Abbauten bereits während des Schlussapplauses statt, um Zeit - und damit Kosten - zu sparen.

Die Kenntnis über den geradezu maschinellen Ablauf einer Theatervorstellung und den dazugehörigen Arbeiten lässt eine gewisse Ernüchterung in puncto Schauspiel-Genuss aufkommen. Auch Theater sind letztlich Wirtschaftsunternehmen, die in allen Bereichen eine Kostenminimierung anstreben, möglichst ohne die Produktqualität leiden zu lassen - sprich in diesem Fall den künstlerischen Aspekt zu vernachlässigen.

Der immer größere (technische) Aufwand der Inszenierungen und die immer stärkere Einsparung von Arbeitskräften führt bei den Theatern notgedrungen zu einer Erhöhung der Effizienz, auch die Arbeitsmittel betreffend.

Die Arbeitsmittel eines Beleuchters bestehen hauptsächlich aus den Scheinwerfern, den Kabeln, den Dimmern und der alle Beleuchtungsgeräte ansteuernden Schaltzentrale: dem Beleuchtungsstellwerk.

Im Rahmen dieser Diplomarbeit wird sich mit der Analyse und dem Design eines Systems beschäftigt, das es ermöglicht, die Aufgabe des Abrichtens von Scheinwerfern und das Erstellen von Licht-Stimmungen unabhängig von der Arbeit an der Hauptkonsole, dem Beleuchtungsstellwerk, zu übernehmen.

Es wird konkret auf der Basis eines Beleuchtungssystems der Firma “ADB Lighting Technologies” diskutiert, doch sind die konzeptionellen Unterschiede zu anderen Systemen so gering, dass sie vernachlässigt werden können.

Im Kapitel “Analyse” wird zunächst auf die Anforderungen des zu entwerfenden Systems eingegangen: Was soll das System aus Sicht des Benutzers leisten? Dabei werden auch die beteiligten Akteure aufgeschlüsselt und die einzelnen Anwendungsfälle vorgestellt.

Nach der Diskussion der Anwendungsfälle beschäftigt sich das zweite Kapitel “Design und Realisierung” zum einen mit den verfügbaren Middleware-Technologien. Der Auswahl einer geeigneten Middleware folgt die Vorstellung einiger Klassenmodelle, die dann im weiteren Verlauf des Kapitels genauer besprochen werden.

In “Zusammenfassung und Ausblick” werden die in der Diplomarbeit benannten Ziele und

real Erreichtes gegenübergestellt. Es wird danach gefragt, inwieweit diese Gegenüberstellung konkludent ist, welche Erkenntnisse sich aus einer Übereinstimmung oder auch eventuellen Diskrepanzen ergeben. Des weiteren beinhaltet das Kapitel die Bewertung des Prototypen und es wird die prototypische Implementation des Systems angesprochen. Der zum Schluss gegebene Ausblick thematisiert das zukünftige Vorgehen sowie mögliche Erweiterungs- und Entwicklungsmöglichkeiten, die ein System wie das hier vorgestellte in Zukunft haben wird.

2. Analyse

Im folgenden Abschnitt wird auf die Problemstellung eingegangen und es werden die für das System relevanten Anwendungsfälle diskutiert. Im Rahmen der Anwendungsfälle werden auch exemplarisch erste Dialogprototypen vorgestellt und besprochen.

2.1. Problemstellung

Wie in der Einleitung beschrieben, ist die Zeit beim Aufbau eines Theaterstückes knapp bemessen. Innerhalb eines kurzen Zeitlimits muss jeder der eingesetzten Scheinwerfer auf seine Funktion hin überprüft werden, sowie auf Farbe und Ausrichtung, die er während der Vorstellung an- und einnehmen soll.

Der bereits angesprochene immer größere Aufwand der Inszenierungen beinhaltet auch eine permanent wachsende Zahl von Scheinwerfern. Dieser Umstand trägt zu einer Verlängerung der Zeitperiode bei, die zum Prüfen der Scheinwerfer notwendig ist.

Bei der bisher üblichen Kontroll-Methode wird der einzelne Scheinwerfer angewählt und - im Fachjargon - “reingeregelt”. Nach Überprüfung der Funktion, der Farbe und Ausrichtung wird der Scheinwerfer wieder “rausgeregelt”, um danach mit dem nächsten fortzufahren.

Diese Verfahrensweise beschränkt sich nicht nur auf Theaterveranstaltungen, sondern wird auch bei der Oper oder in Konzerthallen bzw. -arenen angewandt. Bei letzteren beiden steht allerdings mehr Zeit für Aufbau und Proben zur Verfügung und es entfällt zumeist die Neu-Zusammenstellung der verwendeten Geräte.

Ziel der hier vorliegenden Diplomarbeit ist es, eine Lösung hinsichtlich der umständlichen Arbeitsweise beim Ausrichten der Scheinwerfer und den daraus resultierenden Zeitproblemen zu finden. Das System soll dem Beleuchter als Hilfsmittel dienen und ihm die Übersicht bieten, die er benötigt, um seine Aufgaben trotz zeitlicher Einschränkungen zielstrebig und zuverlässig erfüllen zu können.Weiterhin soll das System die Anzeige und das Bedienen von Scheinwerfern unabhängig vom eigentlichen Stellwerk unterstützen.

Dadurch wäre beispielsweise ein Beleuchter, der während der Probe auf der Bühne einen Scheinwerfer einrichten möchte, in die Lage versetzt, diesen eigenständig mittels eines Tablet-PCs oder eines PDAs “reinzuregeln”. Diese Vorgehensweise würde den Beleuchter am Stellwerk entlasten - bei Proben ein nicht zu unterschätzender Zeitgewinn.

2.2. Zustandsaufname des gegebenen Systems

Das Zusammenspiel der einzelnen Beleuchter bei den meisten heute zur Verfügung stehenden Systemen ist in Abbildung 2.1 zu sehen. Die Beleuchter müssen mittels Funkgerät mit dem Stellwerksbeleuchter kommunizieren und ihm darüber den Wunsch zum “Reinregeln” eines Scheinwerfers, den sie gerade einleuchten wollen, mitteilen.

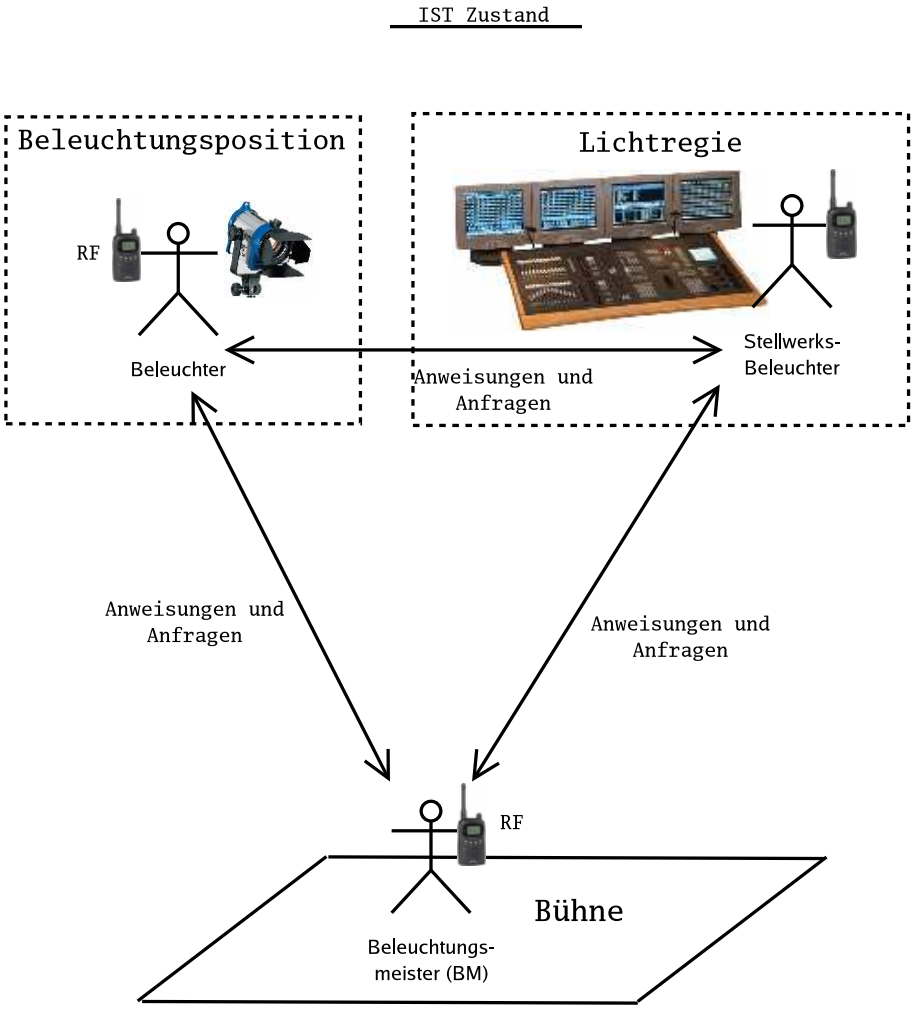

Abbildung 2.1.: Ist-Zustand bei Beleuchtungssystemen

Abbildung 2.2 zeigt eine mögliche Geräte-Konstellation[1] für eine Veranstaltung. Im oberen Bereich der Abbildung sind drei Rechner zu erkennen. In der Mitte und rechts sind zwei Beleuchtungsstellwerke zu sehen. Das Stellwerk in der Mitte ist der sogenannte Master, über den die Vorstellung gesteuert wird. Das Stellwerk auf der rechten Seite und das Notebook auf der linken Seite dienen als Slaves, mittels derer der Master fernbedient werden kann. Die Konsole im rechten Teil der Abbildung kann die Steuerung der Vorstellung übernehmen, sollte der

[1] Im Kontext dieser Arbeit ist “Gerät” gleichzusetzen mit dem Begriff Scheinwerfer.

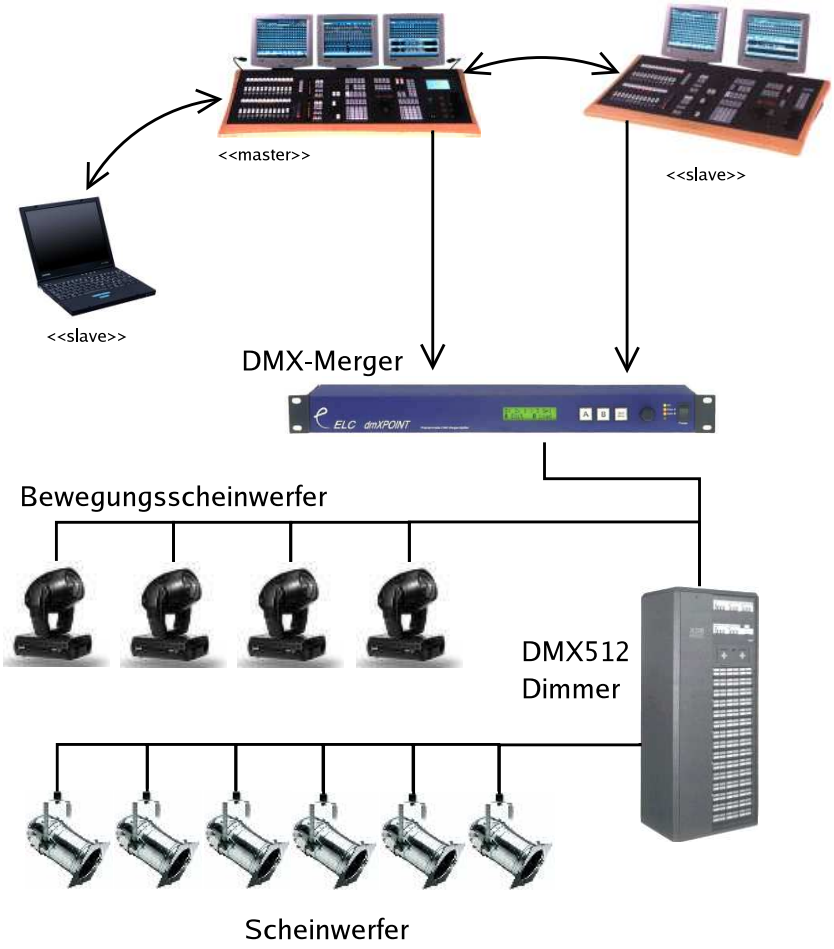

Abbildung 2.2.: Eine mögliche Konfiguration

Master ausfallen. Im unteren rechten Bereich ist ein Dimmer abgebildet; dazwischen befindet sich ein DMX-Merger[2], der die DMX-Daten der beiden Konsolen kombiniert und diese an den Dimmer weiterleitet. Auf der linken Seite des Dimmers sind Bewegungsscheinwerfer an den Merger angebunden; die konventionellen Scheinwerfer darunter sind an den Dimmer angeschlossen.

Zur Zeit existieren bei der ISIS Sofware, dem Stellwerkssystem der Firma ADB, vier Mechanismen zur Kommunikation mit den anderen Systemkomponenten:

ArtNet Die Konsolen können über angeschlossene ArtNet-Knoten weitere Endgeräte ansprechen, wenn die von der Konsole zur Verfügung gestellten DMX-Ausgänge nicht ausreichen. Eine weitere Verwendung des ArtNet-Protokolls bei der ISIS Software ist das Ermitteln von “WiFi-Remote” ^{TM}Fernsteuerungen.

[2] Zu Fachbegriffen siehe Glossar.

WiFi-Remote Zum ermitteln der WiFI-Remote ™Fernbedienungen wird das ArtNet-Protokoll verwendet. Findet eine Konsole eine solche Fernbedienung und ist diese in der Konsole aktiviert, so baut die Konsole eine Verbindung auf TCP-Ebene zu der Fernbedienung auf, über die dann Ereignisse und Statusmeldungen übertragen werden. Diese Fernbedienfunktion erhöht nicht die Skalierbarkeit des Systems, sondern bietet lediglich eine weitere Eingabemethode für das gleiche System, ähnlich dem Anschließen einer weiteren Tastatur an einen Computer. Die Wahrscheinlichkeit eines störenden Aufeinandertreffens zweier Beleuchter, die gleichzeitig über ihre Fernbedienung auf das System zugreifen, ist sehr hoch.

Konsolen Synchronisation Die Konsolen, auf denen die ISIS ™Software läuft, versenden im Sekundentakt ein Heardbeat Datagramm. Das Datagramm kann auch herangezogen werden, um zu ermitteln, welche Konsolen im Netz verfügbar sind und welcher Sitzung - als “Sub net” bezeichnet - sie beitreten möchten. Findet ein Slave einen Master für sein “Sub net”, nicht mit Sub-Netzen in TCP/IP-Netzwerken zu verwechseln, so versucht der Client eine TCP-Verbindung zu dem Master aufzubauen. Die über diese Verbindung übertragenen Informationen sind die Ereignisse, die auf der Ebene der Benutzerschnittstelle ausgelöst werden. Es handelt sich also um Informationen wie zum Beispiel Tastendrücke und die Werte von Reglern und Digitalstellern.

SMB Nachdem ein Slave die Verbindung zu einem Master aufgebaut hat, liest ersterer über SMB das aktuelle Vorstellungs-Repository vom Master, damit beide über die gleichen Daten verfügen. Anschließend findet die Synchronhaltung der Daten über die zuvor beschriebene Konsolen-Synchronisation statt.

In Abbildung 2.3 ist ein Klassendiagramm zu sehen, das die beim Patching, der Zuordnung der Geräte zu den DMX-Kanälen, beteiligten Dateien und ihre Relation darstellt.

Die Dateien sind patch.def, chandef.asg, channbr.asg, patchin.def, dimlaws.def und die Dateien die die Geräte beschreiben. Diese Dateien müssen gelesen werden, um die korrekte Zuordnung der Kreisnummern und deren Funktionen wie Helligkeit und Farbe zu den DMX-Kanälen zu gewährleisten.

Die Zuordnung wird von der Serverseite gelesen und vom Ausgabe-Dienst behandelt. Dies bedeutet, dass der Client lediglich Parameter-Werte der Geräte setzt und nicht direkt auf die DMX-Ausgabe zugreift. Es wird aber einen Dienst geben müssen, der es ermöglicht, die Zuordnung verändert zu können und die aktuelle Gerätezuordnung ermitteln zu können.

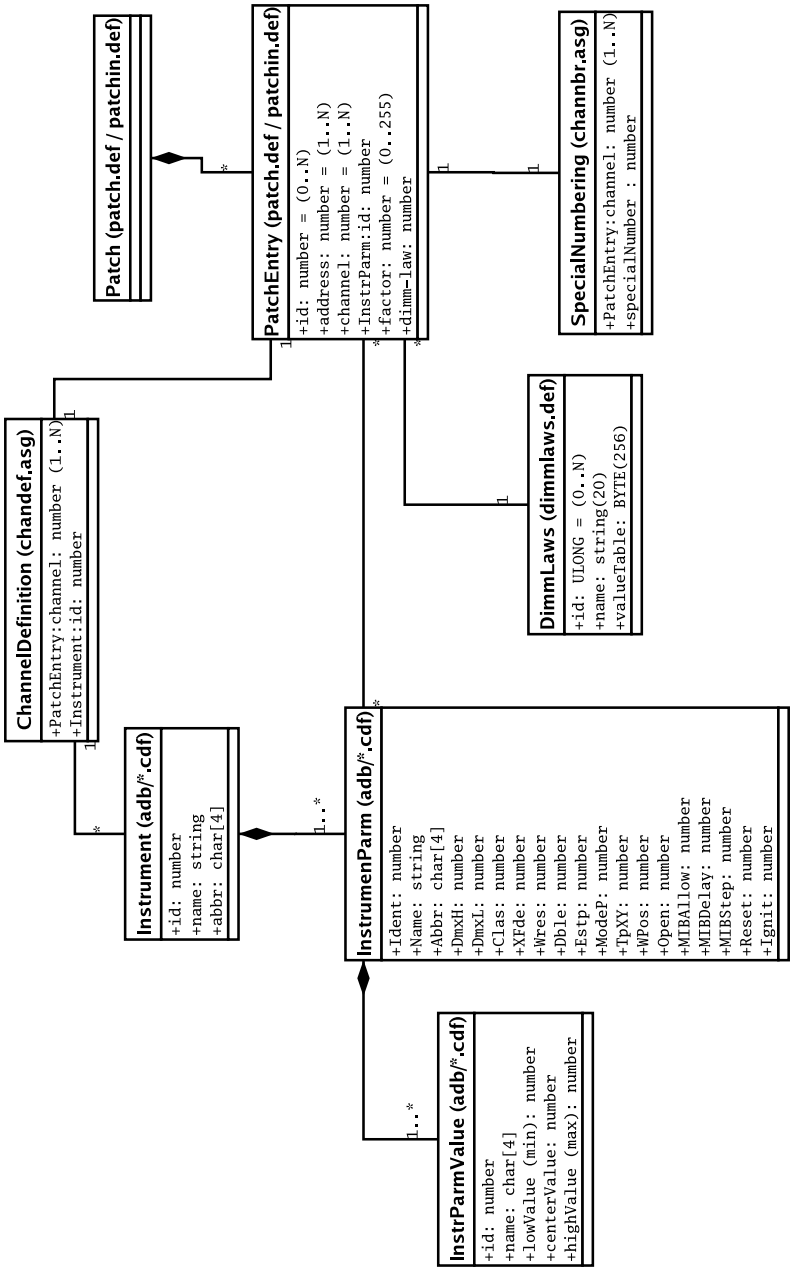

Abbildung 2.3.: Klassenmodell des vorhandenen Patching

2.3. Soll-Zustand

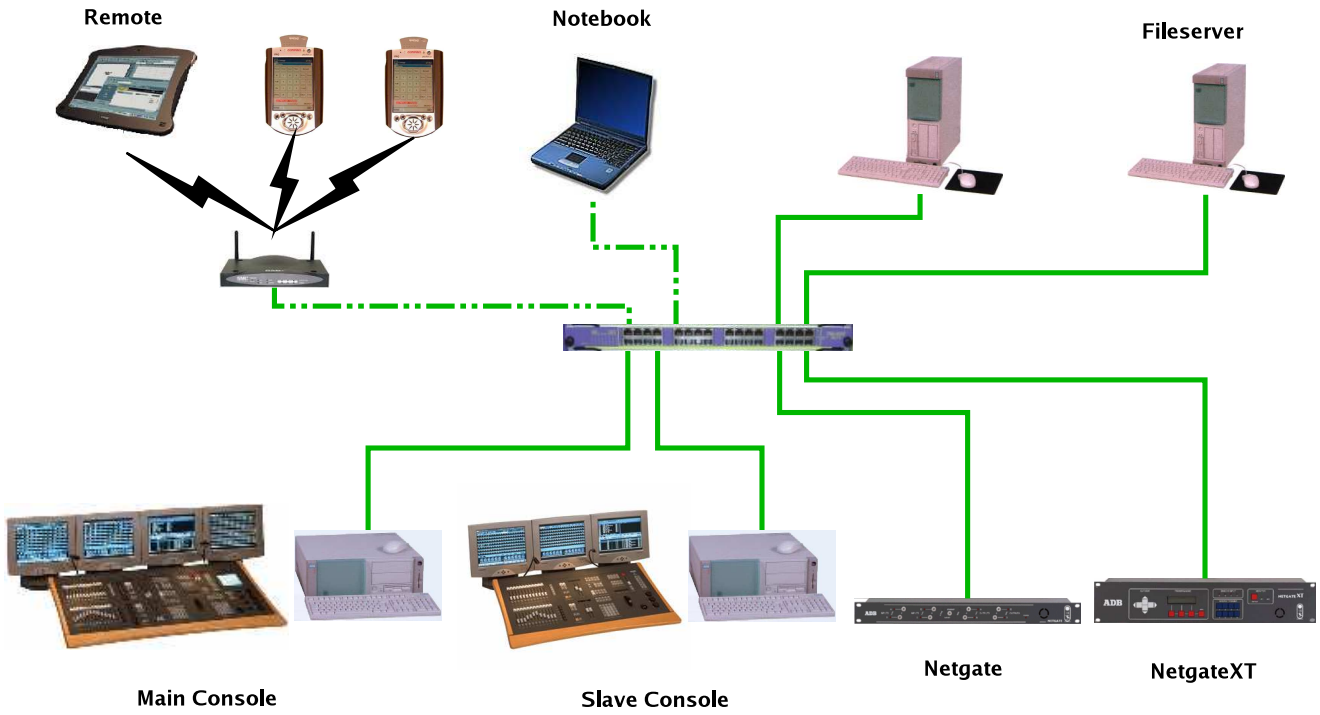

Abbildung 2.4.: Möglicher Aufbau eines Beleuchtungssystem

Abbildung 2.4 zeigt, wie sich die zusätzlichen Komponenten, in Form der PDA's sowie des Tablet-PCs und des Notebooks, im linken oberen Viertel in das System einfügen.

In Abbildung 2.5 ist die geänderte Situation zum vorigen Kapitel zu sehen.

Wie in der zuvor angesprochenen Abbildung zu erkennen ist, stellt der Wunsch nach Skalierbarkeit eine Anforderung an das System dar. Es muss die Möglichkeit bestehen, ohne Änderung des Systems weitere Clients in dieses einzubinden, ohne dass diese aufwendig konfiguriert werden müssten.

Die Skalierbarkeit von Beleuchtungssystemen ist in der Vergangenheit lange Zeit nicht gefordert worden. Ein Grund hierfür war die mangelnde Verfügbarkeit von bezahlbaren drahtlosen Kommunikationsmedien mit adäquater Übertragungsleistung, wie das heutige Wireless LAN. Ein weiterer Grund bestand in den vormals hohen Preisen mobiler Clients.

Zudem war die Nachfrage nach mobilen Einheiten seitens der Anwender nur sehr gering, da einem Großteil der Beleuchter die sich daraus ergebenden Möglichkeiten nicht bewusst bzw. bekannt waren.

Weitere Anforderungen an das System sind die Robustheit, die Performanz, die Verfügbarkeit, die Wartbarkeit, die Prägnanz und die Eleganz aus Sicht des Anwenders. Aus Sicht des Entwicklers sollte das System zu den oben genannten auch noch die Anforderungen der

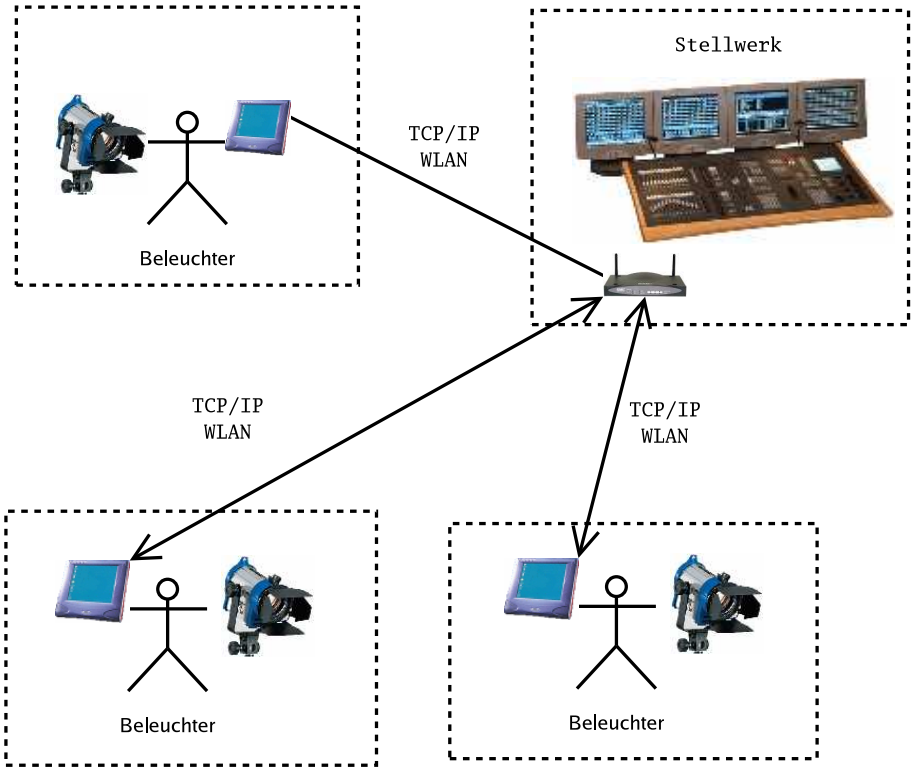

Abbildung 2.5.: Mögliche Kommunikation eines modernen Beleuchtungssystems

Erweiterbarkeit, Sicherheit und der Änderbarkeit (Agilität) erfüllen.

2.4. Use Cases

In der folgenden Tabelle sind alle Anwendungsfälle (UseCases) samt dazugehöriger Beschreibung aufgelistet (UseCase-Katalog).

Bezeichnung	Kurzbeschreibung
Anmelden	Dient der Anmeldung des Benutzers am System
Ebene erstellen/bearbeiten	Der Benutzer erstellt/bearbeitet eine neue Ebene
Scheinwerfer abrichten	Dient dem Abrichten (Prüfen) von Scheinwerfern
Kreise anzeigen / bedienen	Dient dem Regeln der Helligkeit von Scheinwerfern

Die folgende Tabelle ist ein Verzeichnis aller Akteure, die in den nachfolgend beschriebenen UseCases vorkommen.

Bezeichnung	Kurzbeschreibung
Beleuchter	Der Beleuchter richtet Scheinwerfer ein, bedient den bzw. die Verfolgerscheinwerfer und das Beleuchtungsstellwerk
Administrator	Anwender mit erweiterten Rechten, wie dem Bearbeiten von Ebenen und dem Einstellen von Parametern
Stellwerkssoftware	Die Stellwerkssoftware ist das Softwaresystem, das die Scheinwerfer ansteuert und Kommandos von einem Beleuchter entgegennimmt

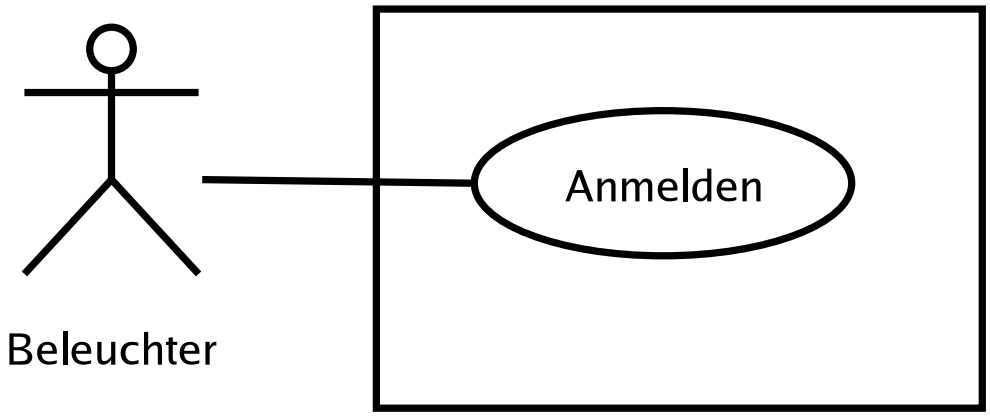

Abbildung 2.6.: Anwendunggsfall - Anmelden am System

2.4.1. Anwendungsfall - Anmelden am System

Der Anwendungsfall dient dem Anmelden des Benutzers am System.

Will sich der Benutzer an einer Sitzung anmelden, so erhält er den in Abbildung 2.7 gezeigten Dialog, mittels dessen er eine der verfügbaren Sitzungen auswählen kann. Sobald er die Auswahl getätigt hat, wird der Dialog angezeigt, der in Abbildung 2.8 zu sehen ist. Hier kann der Benutzer sich mittels seines Benutzernamens und seines Passworts anzumelden. Ist diese Anmeldung erfolgreich, so nimmt er an der Sitzung, unter Berücksichtigung seiner Rechte, teil.

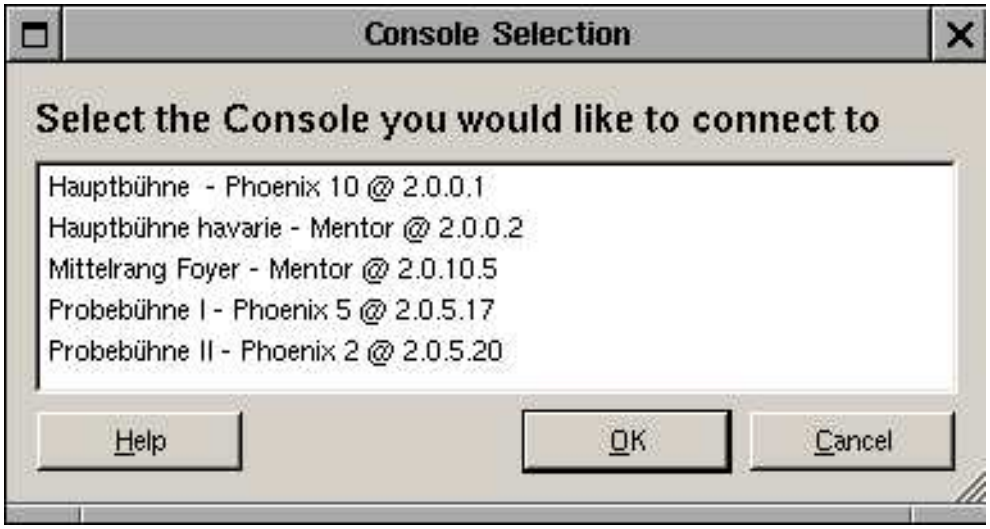

Abbildung 2.7.: Dialog zur Auswahl der gewünschten Sitzung

Abbildung 2.8.: Dialog zur Anmeldung an der ausgewählten Sitzung

2.4.2. Ebene erstellen/bearbeiten

Aufgrund der von Theater zu Theater unterschiedlichen Darstellungen der Scheinwerfer, der ständig neu hinzukommenden Geräte oder auch dem Fortfallen von Positionen, muss der Beleuchter die Möglichkeit erhalten, die Darstellungen selbstständig bearbeiten zu können.

Der hier besprochene Anwendungsfall dient dem Erstellen und Bearbeiten der Masken bzw. Ebenen. Er kommt beispielsweise zur Anwendung, wenn der Beleuchter einige neu hinzugekommene Kreise in die Darstellung aufnehmen möchte. Der Beleuchter kann entweder eine neue Ebene erstellen oder die Geräte in eine bereits vorhandene Ebene einfügen. So kann es vorkommen, dass für ein bestimmtes Theaterstück einzelne Geräte an Positionen aufgestellt werden, an denen bei anderen Stücken keine Geräte verwendet werden.

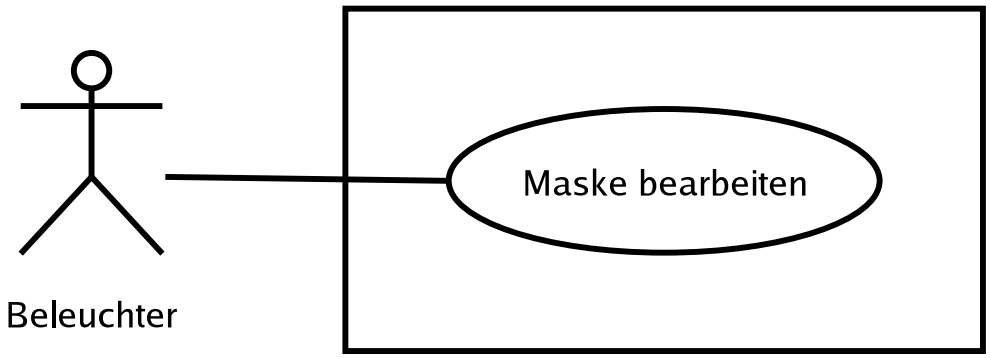

Abbildung 2.9.: UseCase-Diagramm zum Bearbeiten der Ebenen

Um die Übersichtlichkeit zu verbessern, ist die Darstellung der Scheinwerfer in einzelne Ebenen unterteilt. Jede Ebene stellt einen Teil des Gesamtbereichs dar, in dem sich Scheinwerfer befinden. Es besteht die Möglichkeit, eine Ebene zu erstellen, die eine Übersicht über alle vorhandenen Ebenen bietet und von der aus die anderen Ebenen aufrufbar sind.

Der Systemeinrichter ist in der Lage, neue Ebenen zu erstellen und in ihnen neue Kreise oder Verweise bzw. Quicklaunches zu Ebenen einzurichten. Bei Quicklaunches handelt es sich um Komponenten ähnlich der Gerätedarstellungen. Wird die Komponente durch einen Mausklick aktiviert, findet ein Wechsel auf die assoziierte Ebene statt. Der Systemeinrichter kann auch Texte erstellen, die der Orientierung dienen sollen.

Abbildung 2.10 zeigt den Dialog zum Bearbeiten der Ebenen. Auf der linken Seite ist eine Liste der verfügbaren Ebenen zu sehen. Die Ebene, die im mittleren Feld bearbeitet werden soll, kann auf dieser Liste angewählt werden.

Darunter befinden sich einige Kontrollelemente, mittels derer eine neue Ebene erzeugt oder die angewählte Ebene gelöscht werden kann. Des weiteren ist es möglich, den Namen der Ebene zu ändern. Wird das System angewiesen, eine neue Ebene zu erstellen, fragt das System in einem weiteren Dialog (Abbildung 2.11) nach den initialen Eigenschaften der Ebene, wie zum Beispiel ihrem Namen.

Daraufhin erscheint im mittleren Feld (Abbildung 2.10), dem Arbeitsbereich, die neu erstellte Ebene.

Dieser Maske können nun Elemente hinzugefügt oder in ihr bereits vorhandene Elemente gelöscht, verschoben oder in ihrer Größe verändert werden. Die einzelnen Elemente verfügen über Attribute, die abhängig vom Typus des Elements sind.

Folgende Element-Typen sind vorgesehen:

- Gerät - Stellt eine einen Scheinwerfer repräsentierende Komponente dar.
- Quicklaunch - Gibt Informationen zu einer Ebene aus und ermöglicht die Anwahl der Ebene.
- Text - Stellt einen vom Benutzer definierbaren Text dar.
- Button (optional) - Ein Knopf, dem vom Benutzer ein Ereignis zugewiesen werden kann, das an das Stellwerk weitergegeben und von ihm ausgeführt wird.
- Regler (optional) - Gibt seinen Zustand an das Stellwerk weiter und löst ein Ereignis aus, das einen linearen Wert erwartet (z. B. Submaster-Wert).

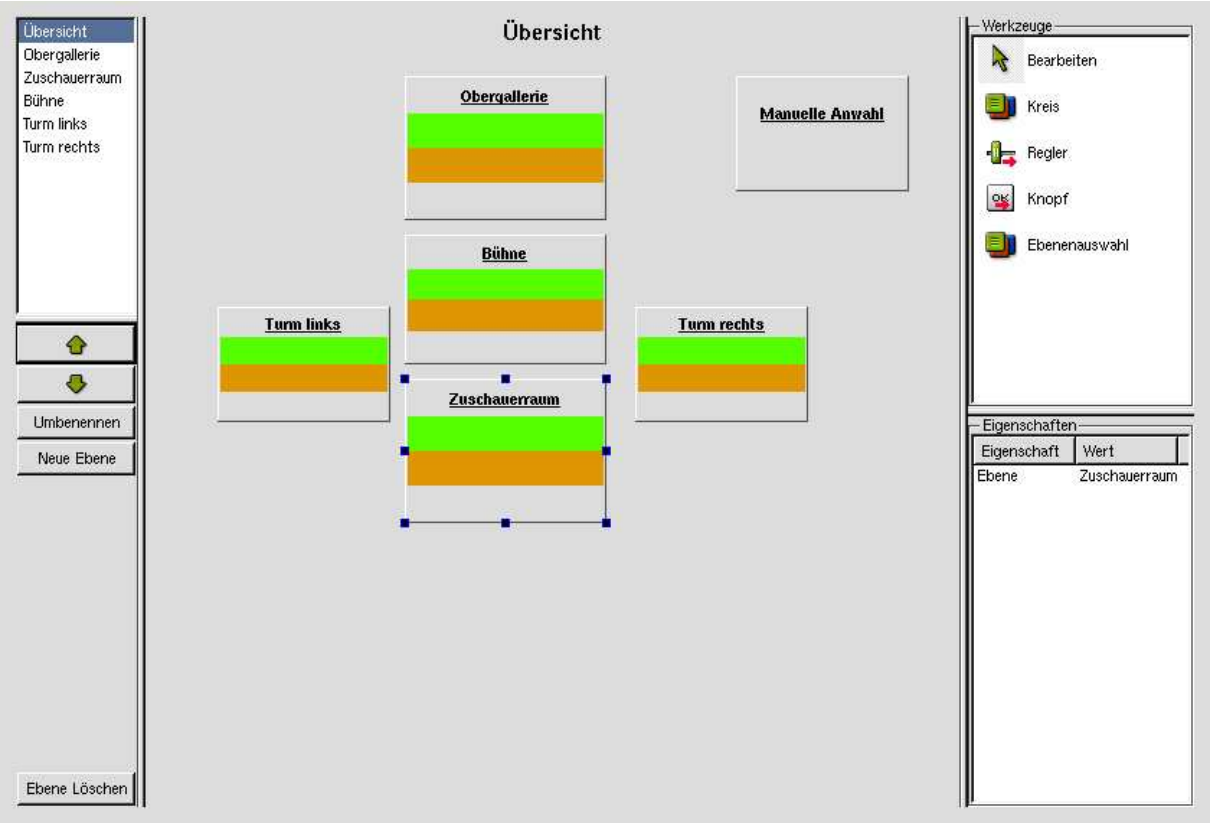

Abbildung 2.10.: Bearbeiten der Übersichts-Ebene

Abbildung 2.11.: Dialog zum Erfragen des Namens einer Ebene.

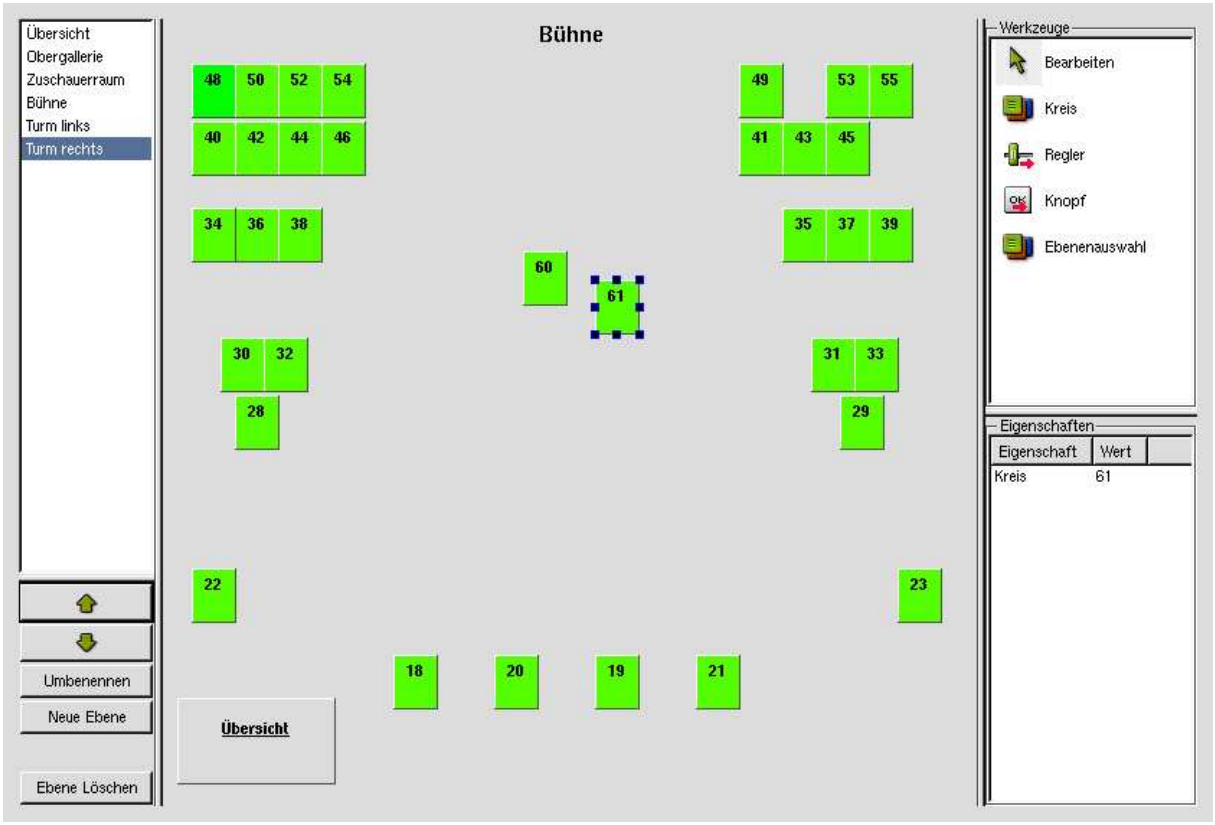

Abbildung 2.12.: Dialog zum Bearbeiten der Bühnenebenen.

2.4.3. Scheinwerfer abrichten

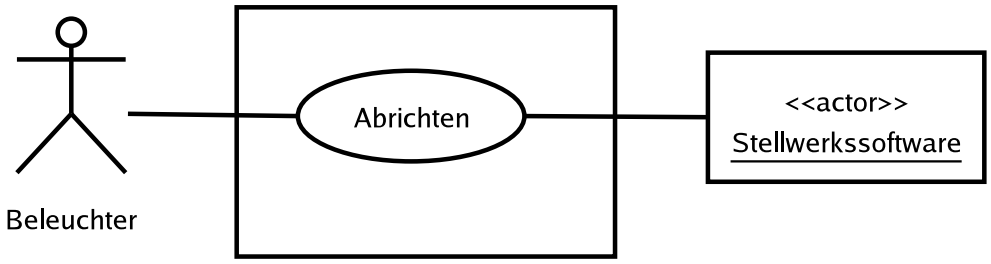

Abbildung 2.13.: UseCase Abrichten

Das Abrichten der Scheinwerfer dient dem Kontrollieren der Funktion, der Farbe und der Position des Scheinwerfers.

Abbildung 2.14 zeigt die Darstellung, die dem Beleuchter eine Übersicht der Ebenen liefert, in der sich Scheinwerfer befinden. Aus der Übersichtsebene kann mittels Anwahl des entsprechenden Übersichtselements in die Ebene gewechselt werden, die dem jeweiligen Übersichtselement zugeordnet ist.

Weiterhin kann eine Ebene durch das Anwählen ihres Namens in der Liste auf der linken Seite der Maske aktiviert werden. Eine Ebene, die Geräte enthält, ist in der Abbildung 2.15 zu sehen.

Zum Abrichten der Scheinwerfer ist neben dem Arbeitsfeld nur die Auswahl der Ebenen sowie die Statusanzeige sichtbar. Wird die Vorstellung geladen, werden alle Geräte mit dem Status “ungeprüft” vorbelegt. Während des Abrichtens durchlaufen die Geräte zudem die Stati “aktiv” und ”eingerichtet”. Der Status “ungeprüft” zeigt an, dass das Gerät noch nicht geprüft wurde und bedient sich dazu der Farbe Grün. Alle orange markierten Geräte erhalten den Status “aktiv” sobald sie angewählt wurden. Wurde ein Gerät auf seine Funktion und korrekte Ausrichtung hin überprüft und dies bestätigt, so wechselt sein Status auf “eingerichtet” und seine Farbe auf Grau. Ein bereits geprüftes Gerät kann jederzeit wieder angewählt werden, um erneut kontrolliert zu werden. Wechselt ein Scheinwerfer seinen Status, so nimmt er auch eine für den Status vorkonfigurierte Helligkeit an. Diese befindet sich in der Regel bei “ungeprüft” auf 10%, bei “aktiv” auf 80% und bei “eingerichtet” auf 0%.

In der Statusübersicht auf der linken Seite ist eine Summe der Stati aller Ebenen dargestellt. Sobald ein Gerät seinen Zustand wechselt, ändert sich auch die Anzahl der Geräte in der Übersicht zu der Ebene. In der Übersicht befinden sich drei Zeilen für jeden der drei möglichen Zustände der Geräte. Die Farben der Zeilen sind denen der Geräte-Zustände angepasst (Grün, Orange, Grau).

In der Übersichtsebene sind in jeder Ebenenauswahl bzw. -Komponente die Stati der Ebene zu sehen.

Um Geräte anzuwählen, die in keiner Ebene vorhanden sind und nicht in eine der Ebenen aufgenommen werden sollen, existiert eine weitere Ebene. Sie ist in Abbildung 2.16 zu sehen.

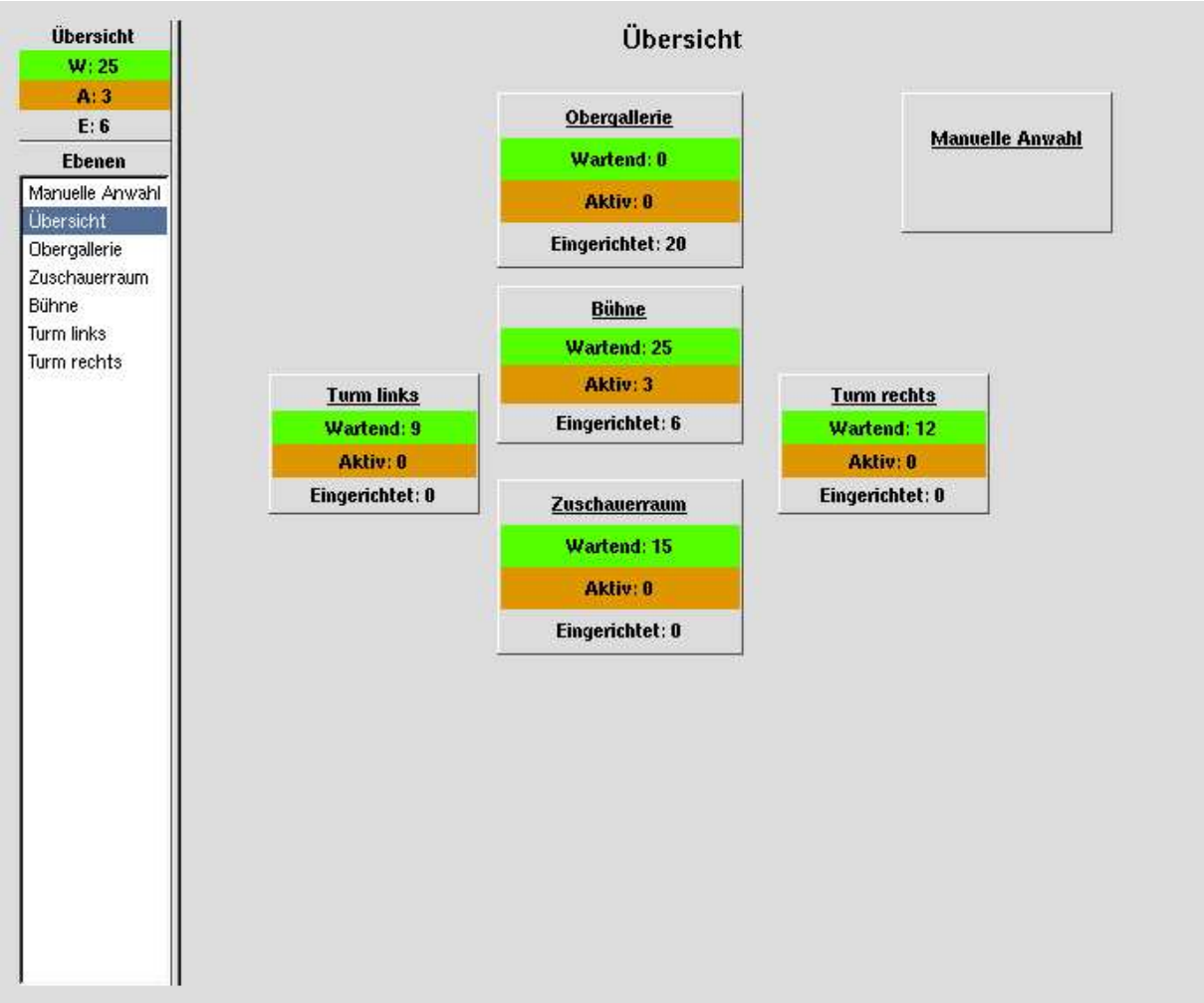

Abbildung 2.14.: Übersichts-Dialog zum Abrichten der Scheinwerfer

Über diese Ebene können alle Kreise angewählt werden. Auch hier werden die Kreise entsprechend ihres Status' farblich gekennzeichnet. Die sich im Status "ungeprüft" befindenden Kreise sind grün hinterlegt, die Kreise im Status "Aktiv" haben einen orangefarbenen Hintergrund und die eingerichteten Geräte nehmen die Farbe Grau an. Alle übrigen Kreise sind einheitlich schwarz eingefasst.

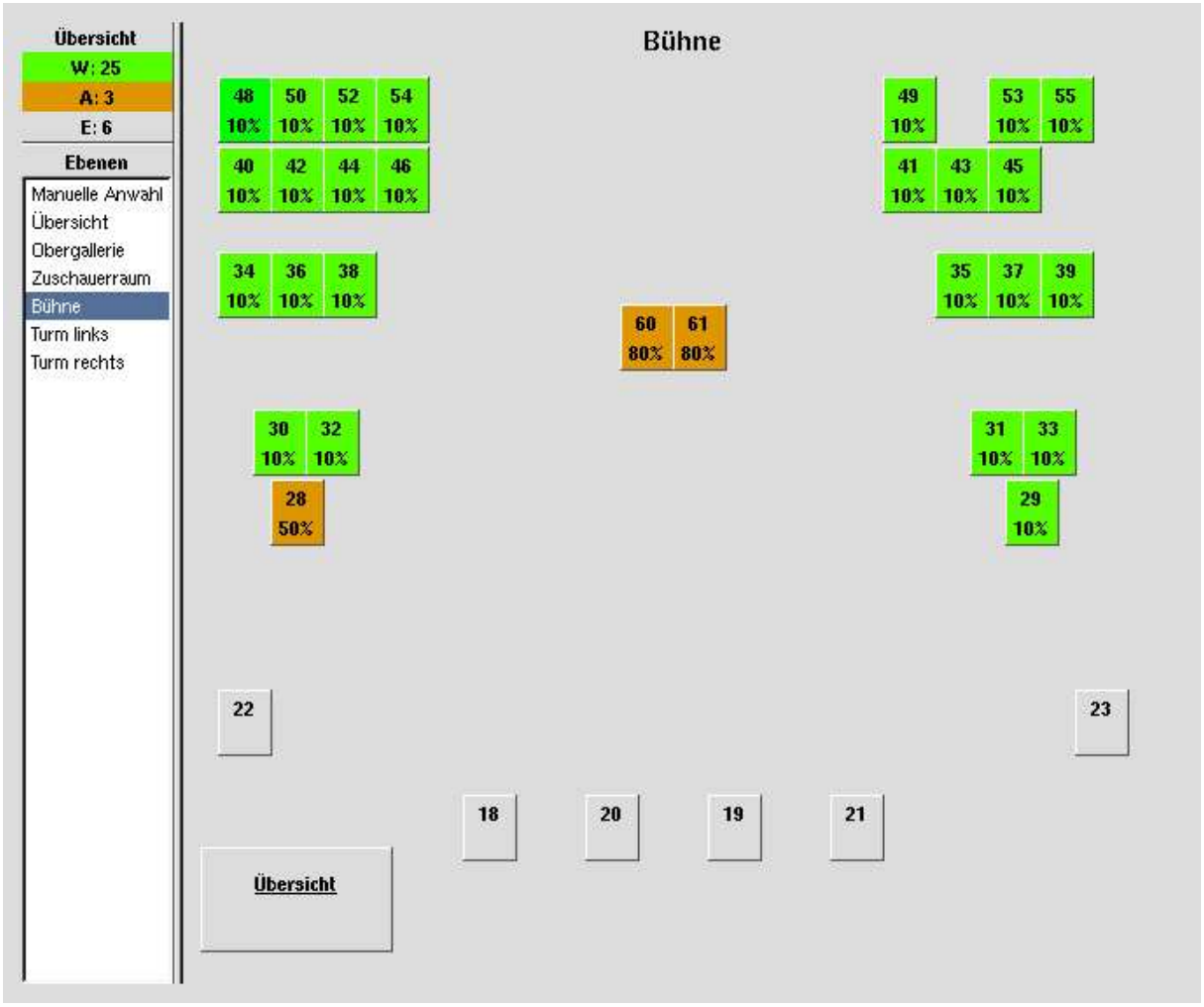

Abbildung 2.15.: Ansicht einer Ebene mit Kreisen in unterschiedlichen Zuständen

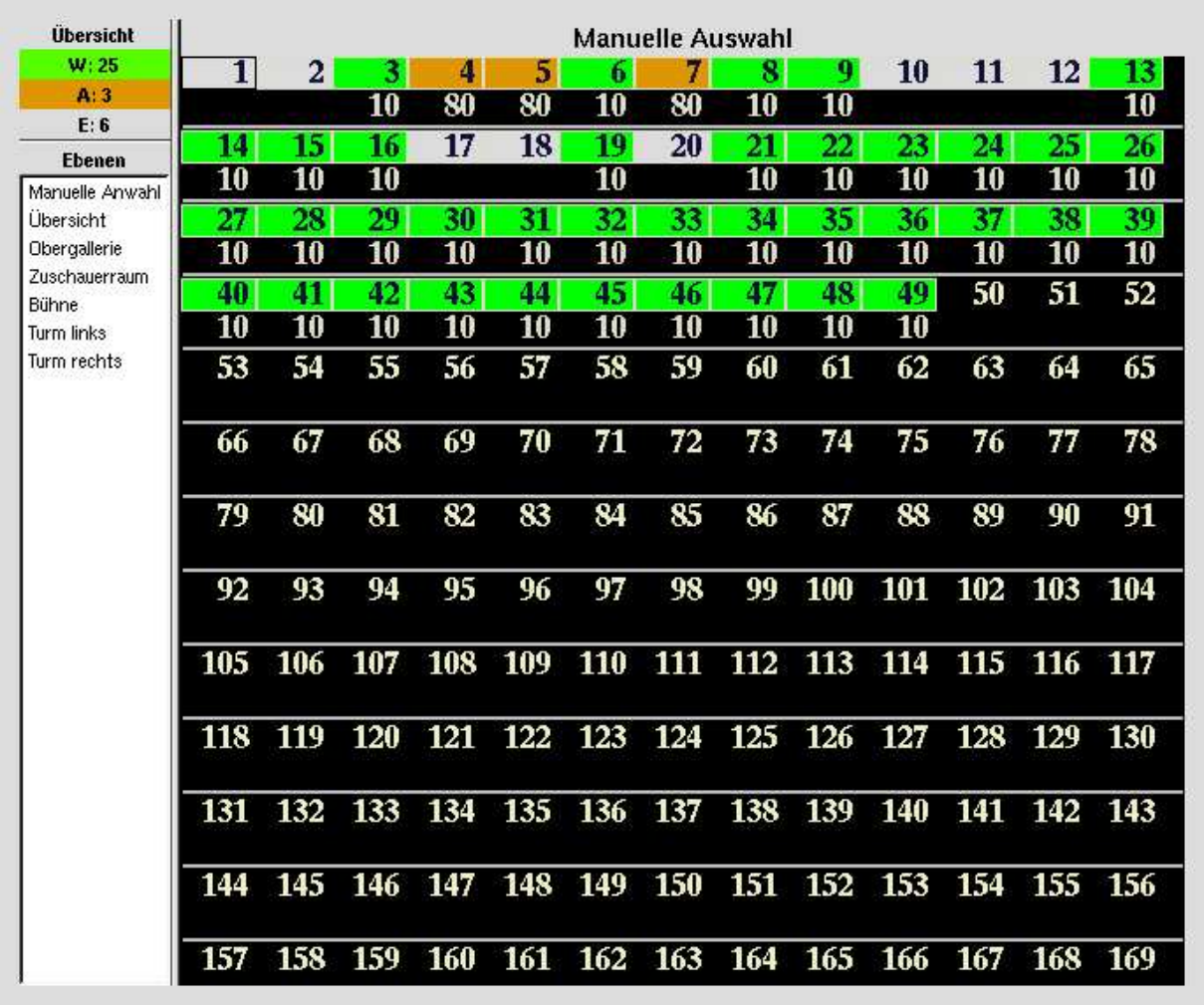

Abbildung 2.16.: Ansicht der Anwahl über die Kreisansicht

2.4.4. Kreise anzeigen / bedienen / ziehen

Mit Hilfe dieses Anwendungsfalls (Abbildung 2.17) ist der Beleuchter in der Lage, einzelne Scheinwerfer “reinzuregeln”. Es besteht die Möglichkeit, die Intensität der Kreise zu beeinflussen, unabhängig von den Einstellungen des Beleuchters am Stellwerk. Die vom Stellwerk ausgegebene Intensität der einzelnen Geräte wird in der Darstellung 2.19 angezeigt.

Anhand der Übersicht, wie in Abbildung 2.18 zu sehen, ist der Beleuchter in der Lage, eine der Ebenen anzuwählen. Innerhalb der Ebenen, die Geräte enthalten, können letztere angewählt und in ihrer Intensität verändert werden.

Die manuelle Bedien-Ebene, zu sehen in Abbildung 2.20 und Abbildung 2.21, ist für die Nutzung von Geräten vorgesehen, die in keiner der Ansichten zur Verfügung stehen.

Bei der Verwendung eines Tablet-PCs mit Touchscreen kann es sinnvoll sein, eine Anzahl von Tasten zur Verfügung zu haben, über die eine schnelle Anwahl von Werten ermöglicht wird. Als Beispiel sei auf Abbildung 2.21 verwiesen.

Ist ein statischer Lichtzustand einmal eingestellt, so kann dieser in einer Stimmung abgelegt werden.

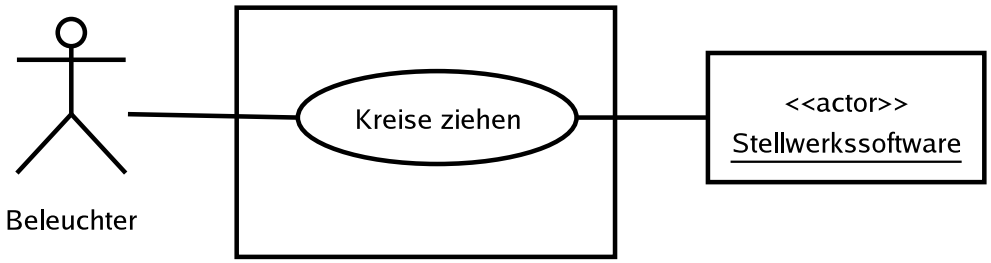

Abbildung 2.17.: UseCase Diagramm zum Anwendungsfall Bedienen

Abbildung 2.18.: Übersichtsdarstellung der Bedienung

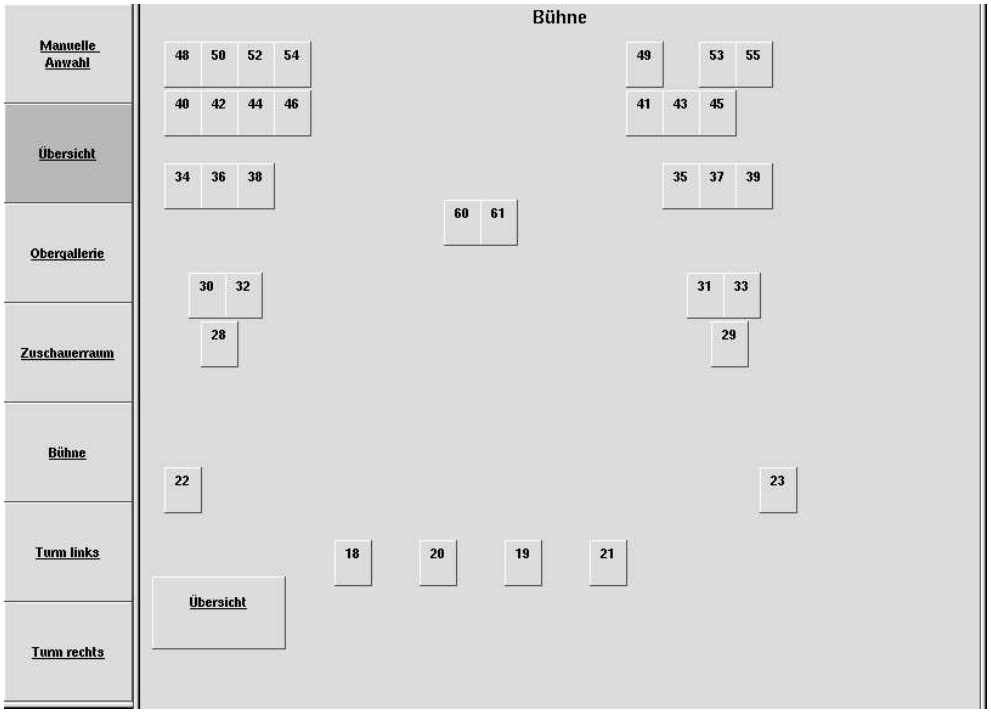

Abbildung 2.19.: Detailsicht einer Ebene im Anwendungsfall Bedienen

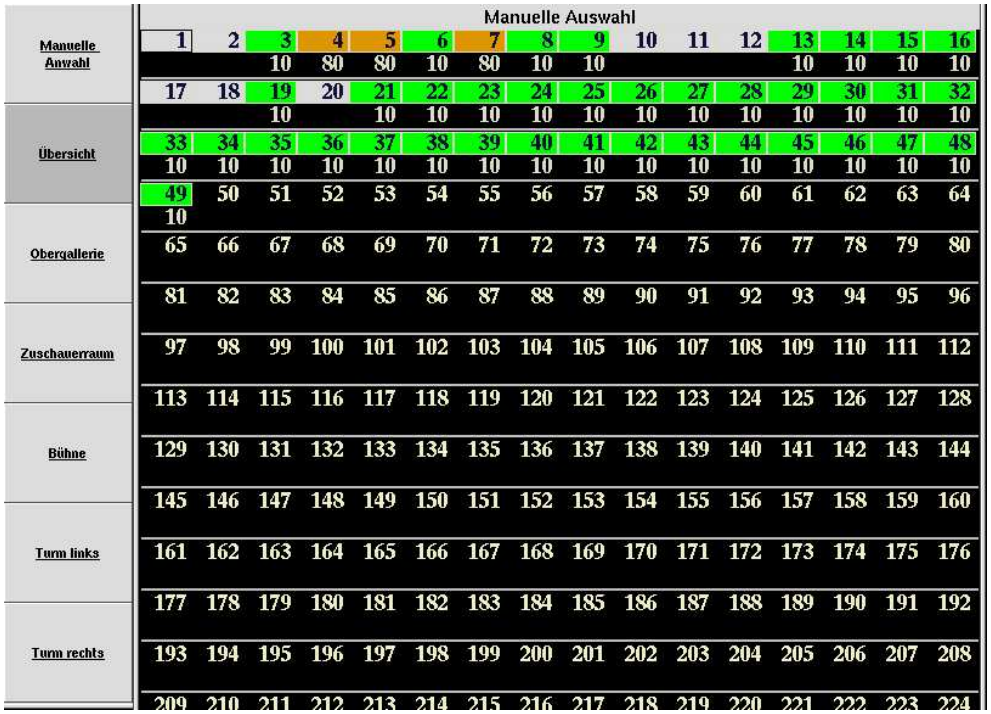

Abbildung 2.20.: Ansicht aller Kreise im Anwendungsfall Bedienen

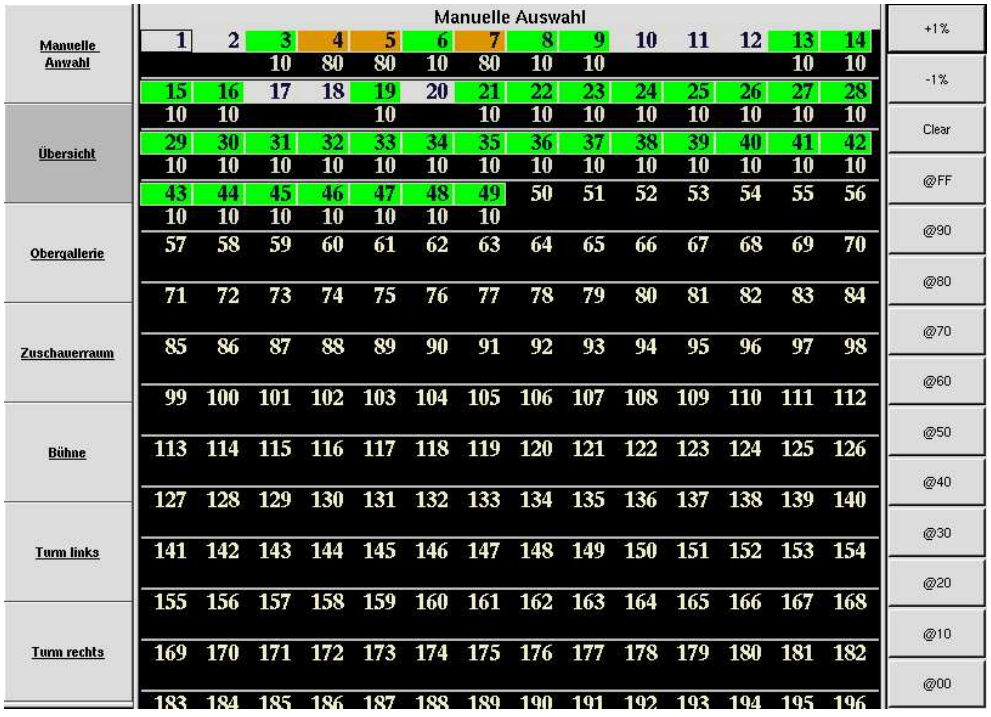

Abbildung 2.21.: Seitlich angebrachte Schalter zur schnellen Anwahl von Werten (Tablet-PC)

2.4.5. Bearbeiten von Stimmungen

In diesem konkreten Anwendungsfall hat der Beleuchter die Möglichkeit, eine vorhandene Stimmung aufzurufen und die Helligkeiten der einzelnen Geräte sowie die Überblendzeiten zu verändern. Die Dialoge ähneln dabei denen zum Bedienen von Kreisen.

2.5. Zusammenfassung

Das Kapitel Analyse beschäftigte sich damit, was das System leisten soll. Das existierende System der Firma ADB wurde analysiert und die Möglichkeiten der Anbindung herauskristallisiert.

Es wurden die einzelnen Anwendungsfälle diskutiert und Diagrammprototypen vorgestellt, die die Interaktion des Benutzers mit dem System zeigten.

3. Design und Realisierung

Nachdem sich das Kapitel “Analyse” mit den Anforderungen an das System beschäftigt hat, geht es in diesem Kapitel um Design und Realisierung: Wie soll das System die Aufgaben erfüllen, die zuvor festgelegt wurden?

Da eine Client-Server-Applikation entwickelt wird, besteht ein erster Schritt darin, sich mit der Middleware zu beschäftigen. Daraufhin werden die Geschäftsklassen identifiziert, die zum Lösen der im vorangegangenen Kapitel genannten Aufgaben notwendig sind. Zudem werden die Relationen zwischen den Klassen näher betrachtet.

3.1. Die Middleware / Entscheidung für eine Middleware

Ziel dieses Abschnittes ist es zunächst, den Begriff der “Middleware” zu (er)klären sowie einige der verfügbaren Middleware-Konzepte vorzuführen und einander gegenüberzustellen. Die daraus gewonnenen Erkenntnisse werden als Grundlage für die Wahl einer Middleware verwendet, die für das zu konstruierende System am besten geeignet ist.

Was verbirgt sich hinter dem Begriff “Middleware”?

Auf den Internet-Seiten des “Software Engineering Institute” der “Carnegie Mellon” Universität findet sich folgende Definition der Bezeichnung “Middleware”:

Middleware is connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network. Middleware is essential to migrating mainframe applications to client/server applications and to providing for communication across heterogeneous platforms.

Die Middleware ist also ein Stück Software, das sich in der Mitte, zwischen der dienstanbietenden und der dienstnutzenden Applikation, befindet und diese miteinander verbindet. Dabei abstrahiert die Middleware auch von den konkreten Methoden, die zur Nutzung der Netzwerkdienste auf den eingesetzten Betriebssytemen notwendig sind.

Bei einigen Middleware-Systemen ist sogar durch die Definition einer einheitlichen Programmierschnittstelle, auch “language binding” genannt, sichergestellt, dass die Nutzung der Middleware durch die Applikation auf Quelltextebene unabhängig von dem eingesetzten Betriebssystem ist. Das ermöglicht es dem Anwendungsprogrammierer, das Programm ohne Mo-

difikationen auf Seiten der Anbindung an die Middleware auf anderen Betriebssystemen zu übersetzen.

Da die Middleware eine einheitliche Kommunikationsschnittstelle auf Seiten der Netzwerkschicht definiert, ist es unterschiedlichen Applikationen, die möglicherweise in verschiedenen Programmiersprachen geschrieben wurden, möglich, auf die gleichen Dienste zuzugreifen.

Diese Tatsache vereinfacht den Aufbau einer heterogenen Systemlandschaft innerhalb der Applikation. Auf diese Weise kann ein Teil der Applikation, bei dem ein hoher Wert auf Verfügbarkeit gelegt wird, auf Mainframe-Systemen wie VMS, S/390, AS/400 oder auf Unix-Systemen laufen. Als Betriebssysteme für die Benutzerschnittstelle können Desktop-Systeme wie Windows, MacOS-X oder OS/2 zur Anwendung kommen. Bei prozessnahen Teilen der Anwendung können auch Echtzeitbetriebssysteme wie zum Beispiel QNX oder VxWorks eingesetzt werden. Dadurch lässt sich die Applikation in unterschiedliche Domänen aufteilen, für die ihrerseits die bestmöglichen Systeme zum Einsatz kommen.

Die Relevanz des Begriffs der Middleware wird in [Hei03] folgendermaßen beschrieben:

Wir sind um eine einheitliche Kommunikationsschnittstelle bemüht, die die darunterliegende Implementation verbirgt, um eine Austauschbarbeit der beteiligten Teilsysteme zu ermöglichen.

Das Diagramm veranschaulicht nochmals das zuvor Beschriebene.

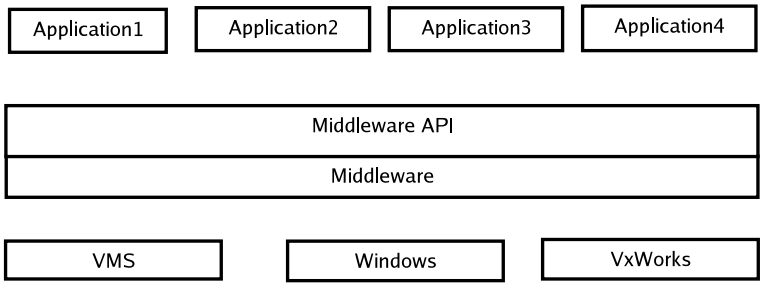

Abbildung 3.1.: Rolle der Middleware

3.2. Relevanz der Auswahl der passenden Middleware

Die Auswahl der Middleware hat nicht nur Auswirkung auf die technische Implementation. Faktoren, die oft unterschätzt werden, sind die Verfügbarkeit, die Einfachheit der Installation und die Schulung des Service-Personals. Eine entwickelte Systemlösung steht niemals separiert und autark im Raum; immer sind viele andere Stellen mitbetroffen.

Der zukünftige Anwenderkreis und die Systemumgebung sind ein Auswahlkriterium für die zu verwendende Middleware. Eine kontrollierte Umgebung in einem Callcenter oder einem Rechenzentrum beispielsweise bietet gänzlich andere Möglichkeiten bzw. birgt andere Gefahren in sich, als eine eher unkontrollierte Umgebung wie sie häufig im Beleuchtungsbereich vorzufinden ist.

Fragen, die sich stellen sollten:

Sind die Personen, die das System einrichten, auch diejenigen, die es nutzen? Oder nehmen das Einrichten spezielle Techniker und Administratoren vor?

Ist es möglich, das System aufwendig zu konfigurieren, da es die folgenden Jahre unverändert laufen wird? Oder ist vorgesehen, das System ständig auf- und wieder abzubauen, was eine einfache Konfiguration des Systems erforderlich macht?

Wer wird das System warten? Das Service-Personal muss eingewiesen werden und sich mit den betroffenen Komponenten auskennen. Ein zu fehleranfälliges System kann technisch möglicherweise überaus elegant konstruiert sein oder aber besonders günstig, ist dafür aber im Service außerordentlich teuer.

Solcherlei nicht-technische Kriterien sollten bei der Auswahl der "richtigen"Middleware ebenfalls betrachtet werden, da die Auswahl der Middleware eine strategische Entscheidung ist, die vom Unternehmen für einen langen Zeitraum gefällt wird.

In die Überlegungen mit einbezogen werden sollten auch die Kosten, die für die Schulung der Entwickler, der Service-Techniker und des Marketings investiert werden müssen.

3.3. Die Entscheidungskriterien für eine Middleware

Zunächst müssen die Kriterien festgelegt werden, die für die Auswahl der am besten geeigneten Middleware in bezug auf das gegebene Problem in Frage kommen. Zur Aufstellung der bewussten Kriterien muss nochmals der Fokus auf den Anwenderkreis gerichtet werden. Die Anwender sind in der Regel diejenigen, die zumindest einen Teil des Systems, den Client, installieren. Sie besitzen im allgemeinen nur geringe Kenntnisse im Bereich Netzwerke; die Namen der gängigen Protokolle sind den Anwendern zwar oftmals geläufig, doch fehlt das tiefergehende Wissen.

Das System muss daher auf Seiten des Clients möglichst ohne Konfiguration auskommen. Die Serverseite ist die Konsole und verfügt in der Regel über eine einheitliche Standard-Installation. An dieser Stelle muss die Installation des System-Teils in die Installation der gesamten Konsole eingebunden werden, wobei die Einstellungen der Konsole ausreichen müssen.

Die Kenntnis darüber, wer das System installieren und wer es nutzen wird, führt zum ersten Kriterium:

1. Simpel in der Konfiguration; wenn möglich ohne Konfigurationsaufwand von Seiten der

Anwendung (Plug And Play), da kein Einfluss auf die Konfigurierbarkeit des Systems genommen werden kann.

Die vom Anwender ausgeführten Aktionen sollen einen unmittelbaren Einfluss auf das System haben. Es ist nicht wünschenswert, dass das System auf eine Anfrage, die ihm aktuell gestellt wird, erst eine Minute später antwortet; außer das Ereignis, um das in der Anfrage gebeten wurde, tritt erst zu einem späteren Zeitpunkt ein. Daraus folgt direkt die zweite Anforderung an das zu erstellende System: Die Komponenten des Systems sollten eng gekoppelt sein.

2. Enge Kopplung (Synchrone Kommunikation).

   Das Netzwerk, das für die Kommunikation der Komponenten unseres Systems verwendet wird, dient ausschließlich dem eigens dafür vorgesehenen Zweck. Es ist nicht notwendig, über das Internet auf die Daten einer Beleuchtungskonsole zugreifen zu können. Es wird sogar gewünscht, dass unter keinen Umständen direkt aus dem Internet auf die Konsole zugegriffen werden kann. Dies führt direkt zur nächsten Anforderung.

3. Lokales Netzwerk. Es ist nicht notwendig, den Dienst im Internet zur Verfügung zu stellen. Die bereitgestellten Dienste werden nur im LAN verwendet.

   Da die Betriebssysteme der Konsole, die den Server darstellt, und des Clients feststehen, und weiterhin für den Server Linux und für den Client Windows zum Einsatz kommt, folgt daraus unsere vierte Anforderung.

4. Betriebssystemübergreifend. Es muß eine Lösung zumindest für Windows und UNIX, speziell Linux, verfügbar sein.

   Die Tatsache, dass die Hauptaufgabe der Konsole das Ansteuern der Endgeräte ist, die Kommunikation nur einen untergeordneten Stellenwert einnimmt und die Middleware dem restlichen System nicht unnötig Ressourcen rauben sollte, führt uns zur fünften Anforderung:

5. Schmale Implementation (schlanke Middleware-Implementation)

   Da auf Seiten des Servers C++ als Programmiersprache zum Einsatz kommt - auch um eine spätere Integration in die ISIS-Software zu ermöglichen - es aber vorbehalten sein soll, einen Client auch in Java realisieren zu können, steht fest, dass die Middleware eine Anbindung für unterschiedliche Programmiersprachen zur Verfügung stellen sollte.

6. Sprachübergreifende Implementation (zumindest C++ und JAVA).

   Da es sich bei den hier relevanten Daten um komplexe und verkettete Objekte handelt, sollte die Middleware auch Unterstützung bei der Realisierung von komplexen Objektzusammenhängen bieten. Die Middleware sollte in einem Methodenaufruf auch Objekte und nicht lediglich simple Elemente zurückliefern können.

7. Zugriff auf komplexe Objekte, nicht nur einfache entfernte Funktionsaufrufe.

3.4. Middleware-Systeme im Überblick

In diesem Abschnitt werden einige Middleware-Konzepte und -Systeme vorgestellt und auf unsere Kriterien hin untersucht. Anschließend wird die vorteilhafteste Lösung gewählt.

Zunächst die Vorstellung der Middleware-Konzepte:

3.4.1. Transaction processing (TP) monitors

Transactions Monitore werden häufig in Flugbuchungssystemen in Reisebüros eingesetzt. Der Transaktionsmonitor nimmt die Anfragen der einzelnen Clients der Reihe nach an und verarbeitet diese als elementare, nicht-trennbare Anfragen. Ein Transaktionsmonitor fasst mehrere nicht elementare Operationen zu einer Elementar-Operation zusammen. Dies ist beispielsweise notwendig, wenn mehrere Zugriffe auf unterschiedliche Tabellen einer Datenbank notwendig sind, um einen Flug zu buchen ohne eine versehentliche Überbuchung vorzunehmen. Wenn trotzdem Überbuchungen vorkommen sollten, wäre es in diesem Fall kein Fehler des Systems, sondern Absicht der Fluggesellschaft.

Ein weiteres Beispiel beschreibt die Überweisung von einem Bank-Konto auf ein anderes. Die Überweisung sollte sich ebenfalls als eine Elementar-Operation verhalten und die Geldabnahme des einen Kontos mit der Geldzunahme des anderen Kontos fest verknüpft sein - selbst wenn es sich auf Datenbankebene um zwei getrennte Operationen handelt. Auch dieser Vorgang kann durch einen Transaktionsmonitor überwacht werden, womit allerdings eine aufwendigere Gestaltung eingeschlossen ist, um sicherzustellen, dass kein Geldbetrag abhanden kommt.

Da Transaktionsmonitore ihrerseits auf RPC, MOM oder anderen synchronen oder asynchronen Kommunikationsmechanismen aufbauen, bedarf es ihrer hier keiner weiteren Betrachtung.

Weitergehende Informationen zu Transaktionsmonitoren sind unter [Bro04] und [Sad04] zu finden.

3.4.2. Remote Procedure Calls (RPCs)

Im Folgenden werden kurz die “Remote Procedure Calls”, kurz RPCs, beschrieben. Die RPCs bieten den Mechanismus eines Funktionsaufrufs. Die auszuführende Funktion muss dabei nicht auf dem lokalen Rechner liegen, sondern kann sich auch auf einem anderen befinden. Der Client ruft einen so genannten Stub auf, der die Aufgabe besitzt, die Parameter zu verpacken, was als “marshaling” bezeichnet wird.

Als nächstes schickt der Stub die Parameter mit der Angabe der aufzurufenden Funktion an den Server. Dort angekommen werden die Parameter wieder ausgepackt, was als “unmarshaling” bezeichnet wird. Anschließend erfolgt das Aufrufen der Implementation der Funktion mit den ausgepackten Parametern und das Resultat wird zurück an den Client versendet. Der

Stub des Clients liefert das Ergebnis an den Aufrufer der Funktion. Die Funktionen mit ihren Parametern und den Rückgabewerten werden in einer Definitionsdatei beschrieben. Aus dieser Datei wird mittels eines speziellen Übersetzers der Stub für die Client-Seite und der Code zum Auspacken der Parameter, der dann die eigenliche Funktion aufruft, generiert.

ONC/RPC (Open Network Computing / Remote Procedure Calls)

Die ONC/RPC werden häufig auch als SunRPC bezeichnet, da sie von der Firma SUN entwickelt und erstmalig eingesetzt wurden.

Die Kommunikation erfolgt synchron und einseitig. Der Client ist der Auftraggeber, der Server ist der Auftragnehmer.

Die Beschreibung der Funktionsaufrufe und der Parameter geschieht mittels der eXternal Data Representation, kurz als XDR bezeichnet. Ein IDL-Compiler liest die Datei in der sich die Definitionen befinden und generiert programmiersprachenspezifisch den Code-Teil des Servers und des Clients, der das “marshaling” und “unmarshaling” der Parameter vornimmt.

Der Programmierer muss dabei nicht wissen, wie die Daten über das Netzwerk übertragen werden, wenngleich die Kommunikation definiert und offen zugänglich ist. ONC/RPC wird unter anderem für das Network Filesystem (NFS) verwendet, das bei verteilten Dateisystemen im UNIX-Umfeld häufig Einsatz findet.

IDL-Übersetzer, die die XDR in programmiersprachenspezifischen Code überführen, existieren für die unterschiedlichsten Programmiersprachen. Bei Bekanntsein der Signatur ist es auch möglich, die Aufrufe dynamisch - ohne die Definitionsdatei - zu generieren.

Eine Beispielanwendung zu ONC/RPC findet sich in Anhang A.

ONC/RPC in Kürze:

- Die Aufrufstruktur wird mittels XDR (eXternal Data Representation) beschrieben.
- Die Form der Netzwerkkommunikation kann als “Black Box” angesehen werden.
- Multithreading im Client und Server wird unterstützt.
- Unterschiedliche Formen der Authentifikation werden unterstützt.
- Der Aufruf im Client blockiert und kehrt erst wieder zurück, wenn die entfernte Prozedur ausgeführt wurde oder ein Fehler (z.B Timeout) aufgetreten ist.
- Die Nachrichten an den Server enthalten eine ID, die im Client generiert wurde und vom Server in der Antwort mitgeliefert wird. Dadurch kann der Client die Antworten zuordnen und Client und Server können doppelt versendete Nachrichten erkennen und entsprechend reagieren.
- Mögliche Aufrufsemantiken: exactly once, at most once, at least once.

Abbildung 3.2.: Beispiel eines XML-RPC Aufrufes in PERL mittels Ken MacLeod's "Frontier::Client" Moduls:

```
use Frontier::Client;
$server = Frontier::Client->new(url => 'http://squareserver.org/RPC2');
$result = $server->call('examples.square', 10);
print "square of 10 is $result\n";
```

Abbildung 3.3.: Kodierung eines Aufrufs in XML-RPC

```
<methodCall>
  <methodName>example.square</methodName>
  <params>
    <param><value><int>10</int></value></param>
  </params>
</methodCall>
```

- Parameter mit variabler Länge sind möglich; bei der Beschreibung in der IDL wird die maximale Länge angegeben.

Als Referenzen zu ONC/RPC sind [Ste98] und [FM96] zu nennen.

XML-RPC

Eine weitere Implementation des RPC Paradigmas sind die XML-RPC, die den zuvor beschriebenen ONC/RPC vom Prinzip her gleichen, jedoch als Sitzungsschicht bzw. Transportschicht HTTP verwenden und sich in der Darstellungsschicht der "eXtensible Markup Language" (XML) bedienen. Da die XML-RPCs XML verwenden und damit die Anfragen und Parameter in textueller Form übertragen werden, können diese problemlos durch eine Firewall geschleust werden, ohne deren Sicherungsfunktion zu beeinträchtigen. Ein RPC-Client kann als Web-Service (CGI-Script) implementiert werden. Es existieren Implementationen für diverse Programmiersprachen wie PERL, Java, Python, C, C++, PHP und andere, sowie für diverse Betriebssysteme wie UNIX, Windows und MacOS-X (UNIX).

Die Abbildung 3.3 zeigt die Kodierung des XML-Dokuments, das für den Aufruf erzeugt und über den Datenkanal an den Server übertragen wird.

Verfügbare Typen bei XML-RPC sind:

- int (32-bit vorzeichenbehaftet)
- string (auch Unicode)
- boolean

- double
- dateTime.iso8601
- base64
- array
- struct

Die Schnittstelle (Parameter) kann dynamisch vom Dienst erfragt werden.

Siehe hierzu auch [US03] sowie [Kid01]

3.4.3. Message-Oriented Middleware (MOM)

"Message Oriented Middleware", kurz MOM, ermöglicht den asynchronen Austausch von Nachrichten in verteilten Systemen. Es ist eine lose Kopplung der einzelnen Komponenten vorgesehen. Durch den asynchronen Austausch von Nachrichten ist gewährleistet, dass Nachrichten auch für den Fall zugestellt werden können, wenn zeitweilig keine Verbindung zum Internet besteht. Zur näheren Information siehe auch [Hei03].

3.4.4. Web Services - SOAP

Das "Simple Object Access Protocol", kurz SOAP, ist ein von der W3C[1] definierter Standard zur Kodierung von entfernten Prozeduraufrufen, ähnlich der RPCs. Es bedient sich bei der Kodierung der Nachrichten der XML, der "eXtended Markup Language". SOAP definiert den Envelope (Umschlag), der das Zustellen der Nachricht regelt, die Darstellung von anwendungsspezifischen Datentypen sowie die Darstellung von Methodenaufrufen. Zum Transport von SOAP-Nachrichten wird in der Regel HTTP verwendet, obgleicht SOAP nicht auf eine bestimmte Transportschicht angewiesen ist. Nähere Informationen zu SOAP sind in [Mit03], [Pir03] sowie [TM01] zu finden.

3.4.5. CORBA (ORB)

Die "Common Object Request Broker Architecture" (CORBA) ist eine von der "Object Management Group" (OMG) festgelegte Architektur, die eine objektorientierte Middleware beschreibt. Als Erweiterung zu den bisher beschriebenen RPCs wird hier der Begriff des Objektes eingeführt. Aus der Sicht des Anwendungsprogrammierers verhält es sich wie bei den zuvor beschriebenen RPCs. Mittels einer IDL wird in einer Datei der Aufbau, in diesem Fall eines Objektes mit seinen Methoden, beschrieben. Aus dieser Beschreibung wird mittels eines IDL-Compilers ein Client-Stub und ein Server-Skelett generiert. Der Client-Stub kann wie ein

[1] W3C: WWW Consortium http://www.w3.org/

normales Objekt verwendet werden, sobald eine Referenz auf das auf dem Server liegende Objekt generiert wurde. Der Client-Stub verhält sich wie ein Proxy, der die Methodenaufrufe an das auf dem Server liegende Objekt weiterleitet. Hierbei dient das “Internet Inter-ORB Protocol” (IIOP) der Kommunikation zwischen Client und Server. Für die Serverseite generiert der IDL-Compiler ein Server-Skelett, das eine abstrakte Klasse implementiert, von der eine konkrete Klasse abgeleitet werden muss. In der konkreten Klasse werden dann die Methoden implementiert, die zu der Klasse in der IDL existieren.

Ein Objekt wird durch die “Interoperable Object Reference” (IOR) beschrieben, eine Referenz auf ein CORBA-Objekt. Ein Objekt kann eine Referenz in Textform erzeugen, die daraufhin zum Beispiel in einer E-Mail oder auf einer Diskette versendet werden könnte. In der Regel werden die Referenzen auf Objekte aber mittels des CORBA Namensdienstes (NameService) ermittelt. Hierzu wird ein CORBA-Nameserver auf einem Rechner zur Verfügung gestellt, an dem die Parteien ihre Objekte an einen Namen binden können. Ist ein solcher Namensdienst und der Name des gewünschten Objektes bekannt, ist es einem Client möglich, seine Objektreferenz zu erfragen und damit Zugriff auf das Objekt zu erhalten.

3.4.6. Zusammenfassung der Middleware-Konzepte

Beim Betrachten der zuvor festgelegten Anforderungen (siehe Kapitel 3.1.2: “Die Entscheidungskriterien für eine Middleware”), scheidet MOM als mögliche Middleware auf Grund der zweiten Anforderung “Enge Kopplung” aus. RPC bietet zwar Funktionsaufrufe, unterstützt aber nicht die Verwaltung komplexer Objekte, was in Anforderung 7 gefordert wurde. Da SOAP den RPCs hinzugerechnet werden kann, muss es ebenfalls als mögliche Middleware gestrichen werden. Folglich bleibt nur das ORB-Konzept als in diesem Fall geeignete Middleware bestehen. Die Betrachtung soll sich dabei nur auf CORBA richten, da es sich um die am weitesten verbreitete, ORB-basierte Middleware handelt. Zudem existieren für CORBA diverse - auch freie - Implementationen für C++ und Java, die es wiederum für diverse Betriebssysteme gibt, was in den Anforderungen 4 und 6 zur Bedingung gemacht wurde.

Weitere Literatur, die sich mit dem Thema Middleware auseinandersetzt, ist auch [Wen04] und [Hei03]. Im Zusammenhang mit CORBA und C++ sei auf [HV99] verwiesen.

3.5. CORBA ORB’s

Da auf der Serverseite C++ als Programmiersprache verwendet wird und kein Java zur Verfügung steht, sind nur solche CORBA ORB’s in Betracht gezogen worden, die in C oder C++ implementiert sind.

Von den zur Verfügung Stehenden wurden nur die Open-Source CORBA ORBs TAO, MICO, Orbit und omniORB untersucht, da sie frei verfügbar sind.

Bei der Überprüfung der vorliegenden CORBA ORB’s hat sich bei Orbit herausgestellt, dass er nicht der Spezifikation des “CORBA C++ Language Bindings” folgt. Somit wäre zum

einen eine spezielle API zu erlernen, und zum anderen hätte ein späterer Wechsel auf einen anderen CORBA ORB einen größeren Aufwand zur Folge.

Bei der Wahl zwischen den verbleibenden ORBs fiel die Entscheidung für TAO, da er die vollständigste Implementation liefert und dank des ACE Frameworks auf einer großen Zahl Plattformen lauffähig ist.

Es ist jedoch unter Einhaltung gewisser Regeln möglich, später noch einen Wechsel auf andere ORB's vorzunehmen. Ebenso besteht die Möglichkeit, auf dem Server einen anderen ORB als auf dem Client zu verwenden. Auf diese Weise könnte für den Client zu einem späteren Zeitpunkt auch Java mit JacORB als CORBA ORB zum Einsatz kommen.

MICO bietet bisher noch nicht die Echtzeitfähigkeit, die von TAO geliefert wird. Diese Tatsache kann später ein entscheidendes Kriterium darstellen und bestärkte daher die Entscheidung für TAO.

Sollte sich in Zukunft herausstellen, dass MICO oder omniORB vorteilhafter sind, so ist ein Wechsel mit geringem Aufwand möglich, da MICO und omniORB in Hinblick auf das "Language Binding" in großen Teilen kompatibel sind.

3.6. CORBA Bootstrapping - Ermitteln des CORBA NameService

Nachfolgend soll erläutert werden, wie im vorliegenden Fall der CORBA Nameservice ermittelt wird. Für CORBA existiert noch keine einheitliche Methode des "Bootstrapings". Der Prozess des Bootstrapings benennt hier den Vorgang des Ermittelns eines Nameservices - bei dem es sich auch um ein CORBA-Objekt handelt - ohne Kenntnis der anderen Systeme im Netz.

Daher muss eine Möglichkeit gefunden werden, die Referenz auf das "NameService" Objekt zu erhalten.

Es existieren proprietäre Implementationen solcher Bootstrap-Protokolle zum Ermitteln eines CORBA-Nameservices. Allerdings können sie im vorliegenden Fall nicht verwendet werden, da bei der Implementation dieses Systems auch Wert auf die Austauschbarkeit der verwendeten Middleware gelegt wird.

Einer der ORBs, die ein eigenes Bootstrap-Protokoll implementieren, ist TAO. TAO bedient sich hierzu einer Multicast-Anfrage. Eine ähnliche Methode gebraucht auch das erstmalig im RFC2165 vorgestellte "Service Location Protocol" (SLP). Das SLP stellt ein Protokoll zum Auffinden von Diensten in einem Netz zur Verfügung. SLP bietet dabei den Vorteil, dass es sich um einen offenen Standard handelt. Die Firma Novel zum Beispiel bedient sich des SLPs zum Auffinden von Servern und Diensten (Druckdienst, Dateidienst, et cetera).

Für SLP existiert eine OpenSource-Implementation namens OpenSLP, die sowohl für UNIX als auch für Windows (WIN32) verfügbar ist. Es ist auch möglich, OpenSLP für die PocketPC-Betriebssysteme von MicroSoft zu übersetzten.

Aufgrund der Tatsache, dass es sich bei SLP um einen Standard handelt und eine schlanke und funktionsfähige Implementation vorhanden ist, bietet sich dieses Protokoll zum Auffinden von CORBA-NameServices an.

Ein weiterer Vorteil SLPs ist, dass es im Rahmen von ACN (Advanced Control Network) eingesetzt wird. ACN ist eine Sammlung von Protokollen, die im Unterhaltungssektor (Bühnentechnik) Verwendung finden und speziell auf dessen Anforderungen ausgelegt wurden. SLP dient auch dort dem Auffinden von Diensten.

Bestätigt und bestärkt wird die Entscheidung für SLP durch das Wissen, dass bei ACN der Übergang vom Status des Drafts in den Status der Verabschiedung unmittelbar bevorsteht. Die ersten Implementationen sind voraussichtlich Ende 2004 zu erwarten.

3.7. Die wesentlichen Geschäftsklassen

In diesem Abschnitt sollen die Geschäftsklassen identifiziert werden, was durch die Beschreibung der Klassen, Objekte und Sachverhalte aus den realen Gegegenheiten heraus erfolgt.

Eine “Veranstaltung” hat einen Namen und verwendet “Geräte” wie Scheinwerfer, Dimmer und weitere zum Lichtbereich gehörige Technik. Die Veranstaltung hat Kenntnis darüber, welche Geräte an ihr teilnehmen und ob diese schon eingeleuchtet wurden. Ein “Lichtpult”, auch Konsole genannt, verfügt über Submaster, denen ein Effekt, eine Stimmung, ein Chaser oder aber eine Gruppe zugeordnet werden kann. Eine “Gruppe” ist eine anonyme Stimmung. Eine “Stimmung” enthält den Zustand aller Scheinwerfer. Die Stimmung hat einen Namen und ihr sind diverse Zeiten zugeordnet: Wartezeit, Einblendzeit, Haltezeit, Ausblendzeit. Ein Gerät beschreibt ein bestimmtes Exemplar eines Gerätetypen. Das Gerät besitzt möglicherweise eine Seriennummer und ein Anschaffungsdatum. Darüber hinaus kann ein Gerät in einer Veranstaltung eingesetzt sein. Ein Gerätetyp verfügt über bestimmte Parameter (Eigenschaften), die beeinflusst bzw. gesetzt werden können und benötigt eine bestimmte Menge an DMX-Kreisen, damit er angesteuert werden kann. Die “Gerätezuordnung” beschreibt eine Zuordnung von DMX-Kreisen an einen DMX-Ausgang zu einem Gerät. Ein DMX-Anschluss ermöglicht es, bis zu 512 “DMX-Slots” auszugeben. Ein DMX-SLot ist ein 8-Bit-Wert und entspricht dabei in der Regel einem Dimmerkanal. Allerdings benötigt man gerade bei Bewegungsscheinwerfern für ein Gerät mehr als einen DMX-Slot, um all seine Funktionen abbilden zu können. Zu den Funktionen gehören die Helligkeit, Pan und Tilt, die Farbe sowie weitere Parameter. Insbesondere Pan und Tilt benötigen häufig zwei Slots, um einen 16-Bit-Wert als Winkel abzubilden. Ein DMX-Ausgang kann sich dabei direkt am Gerät befinden oder über einen Netzwerkknoten repliziert werden. Der Netzwerkknoten wird über das ARtNet-Protokoll angesprochen und stellt ein bis mehrere DMX-Ausgänge, häufig auch mehrere DMX-Eingänge, zur Verfügung. Ein DMX-Ausgang, auch DMX-Universum genannt, kann bis zu zu 512 DMX-Slots ausgeben. In Erweiterungen des DMX-Protokolls ist es einem Lichtpult auch möglich, die angeschlossenen Geräte zu erfragen.

Ein Gerät kann einen der nachfolgenden Zustände annehmen: “ungeprüft”, “aktiv” oder “eingerichtet”. Zu Anfang einer Veranstaltung haben alle Geräte den Status “ungeprüft”, da sie

noch nicht geprüft wurden. Beim Anwählen eines Gerätes wird es auf eine zuvor festgelegte Helligkeit geregelt und sein Zustand wechselt zu “aktiv”. Wird das Gerät vom Beleuchter als funktionstüchtig markiert, so wechselt sein Zustand auf “eingerichtet”; andernfalls wird der Zustand “ungeprüft” angenommen.

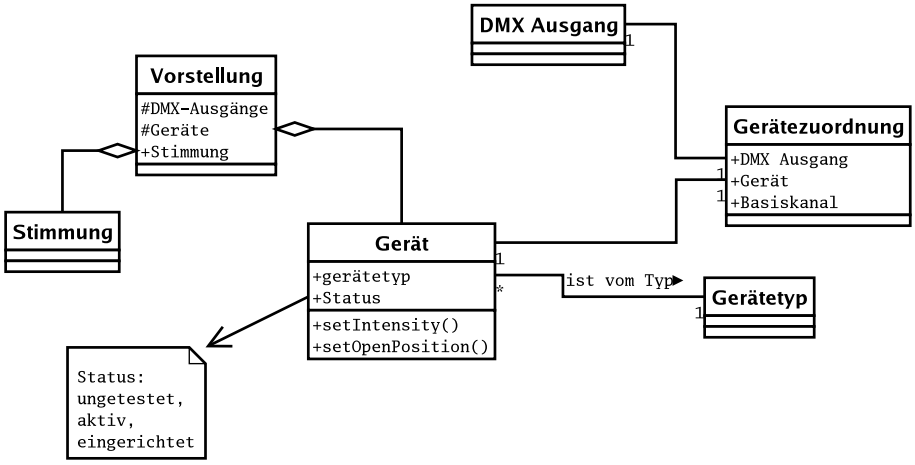

Abbildung 3.4.: Die Geschäftsklassen

In der Abbildung 3.5 ist eine schematische Darstellung der Anordnung der Bedienkomponenten zu sehen. In der Mitte sind die Ebenen übereinander angeordnet, wobei nur eine, die angewählte Ebene, sichtbar ist. Auf der linken Seite stehen untereinander: eine Statusanzeige, die Ebenenauswahl sowie die Hauptsteuerung des Anwendungsfalles, sofern der Anwendungsfall über eine solche verfügt. Beim Bearbeiten zum Beispiel wird der Hauptsteuerbereich (Steuerbereich 1) für die Bedienelemente zum Erstellen, Löschen, Umbenennen, Kopieren und Verschieben von Ebenen verwendet. Der anwendungsfallspezifische Steuerbereich (Steuerbereich 2) wird für die erweiterte Bedienung im Anwendungsfall genutzt. Hier finden sich beim Bedienen eine Liste der Werkzeuge und Komponententypen sowie komponentenspezifische Optionen wieder.

Die Abbildung 3.6 zeigt das dazugehörige Klassendiagramm.

3.7.1. Anmelden

Durch den Vorgang des Anmeldens erhält der Anwender das Recht, auf die Objekte des Servers zugreifen zu dürfen. In der ersten Version des zu erstellenden Prototypen wird auf eine komplizierte Rechtevergabe verzichtet. Es gibt lediglich zwei Berechtigungsklassen. Erstere erlaubt alle Vorgänge, bis auf die Konfiguration des Systems sowie das Bearbeiten der Ebenen. Diese ist für den Akteur “Beleuchter” gedacht. Die zweite Berechtigungsklasse erlaubt den uneingeschränkten Zugriff auf das System, das Bearbeiten der Ebenen und den Zugriff auf die Konfiguration mit eingeschlossen.

Die Rechtevergabe wird im ersten Schritt derart simpel gewählt, um einen hohen Aufwand

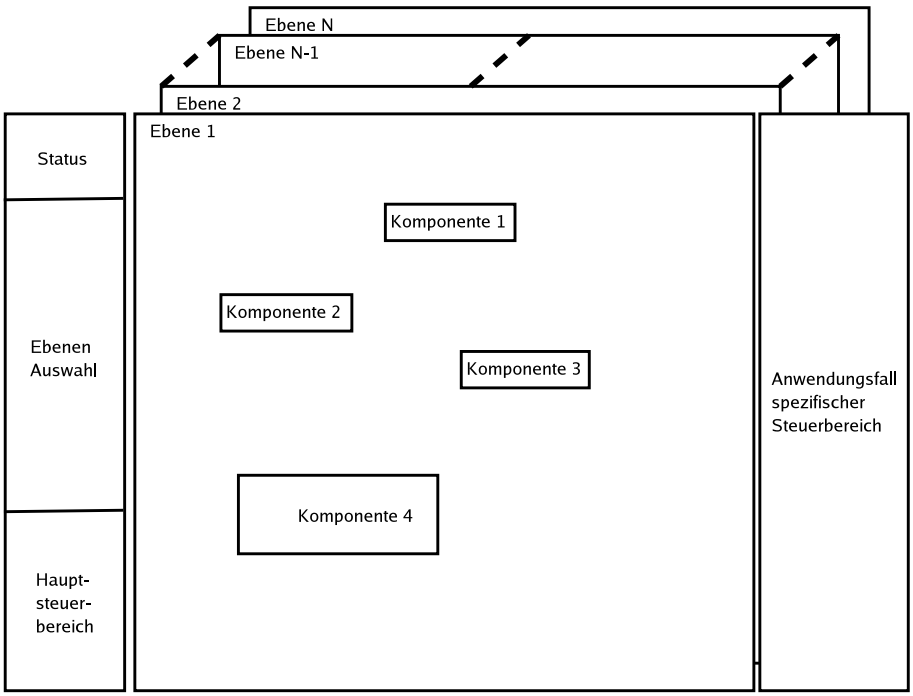

Abbildung 3.5.: Eine Schematische Darstellung der Ansicht

für das Erstellen eines komplexen Rechtesystems zu vermeiden und dem Benutzer des Prototypen die Möglichkeit zu geben, sich mit dem System vertraut zu machen und Erfahrungen zu sammeln. Erst der folgende Prototyp erhält ein aufwendigeres und leistungsfähigeres Rechtesystem, in das Erfahrungen und Anregungen der Anwender mit einfließen. So kann es sinnvoll sein, dass ein Benutzer zwar neue Geräte einfügen und dessen Eigenschaften verändern kann, dass er aber keine bestehenden Geräte verändern darf.

3.7.2. Bearbeiten

Bei dem Bearbeiten der Gerätedarstellung ist zu beachten, dass im Falle des Löschens einer Ebenen-Ansicht, auch die auf sie verweisenden Ebenen-Komponenten gelöscht werden müssen.

In der zuvor besprochenen schematischen Darstellung ist auf der linken Seite eine Ebenen-Auswahl zu sehen. Diese gilt auch hier zur Auswahl der Ebene, die aktuell sichtbar sein soll und somit bearbeitet werden kann. In dem darunter befindlichen Bereich sind die auch schon im UseCase “Ebene erstellen/bearbeiten” angesprochenen Bedienknöpfe angeordnet.

Im Bereich rechts vom zentralen Arbeitsbereich sind die Werkzeuge und Komponenten dargestellt, die zum Bearbeiten der Ebene eingesetzt werden können. Darunter befinden sich die Eigenschaften der gerade angewählten Komponente.

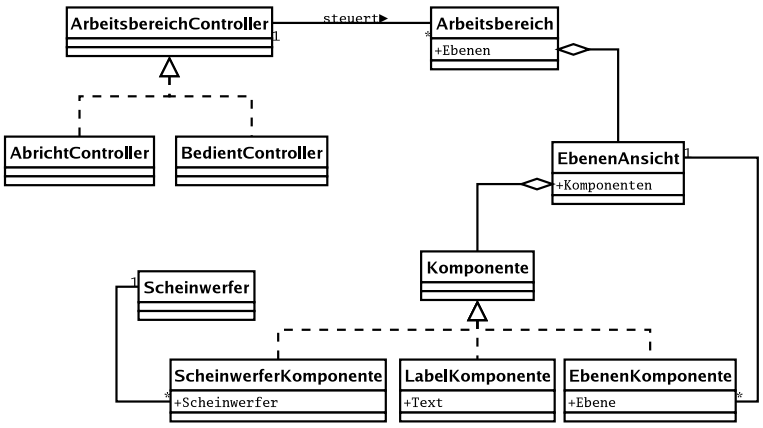

Abbildung 3.6.: Klassendiagramm der Benutzerschnittstelle

Die Komponenten besitzen eine Standardschnittstelle, mit der auf ihre Eigenschaften zugegriffen werden kann. Eine Komponenteneigenschaft stellt einen Editor zur Verfügung, mit dem sich die Eigenschaft bearbeiten lässt. So ist es möglich, auch komplexere Eigenschaften wie Farben zu bearbeiten oder Scheinwerfer oder Ebenen auszuwählen, wie dies in Abbildung 3.7 zu erkennen ist.

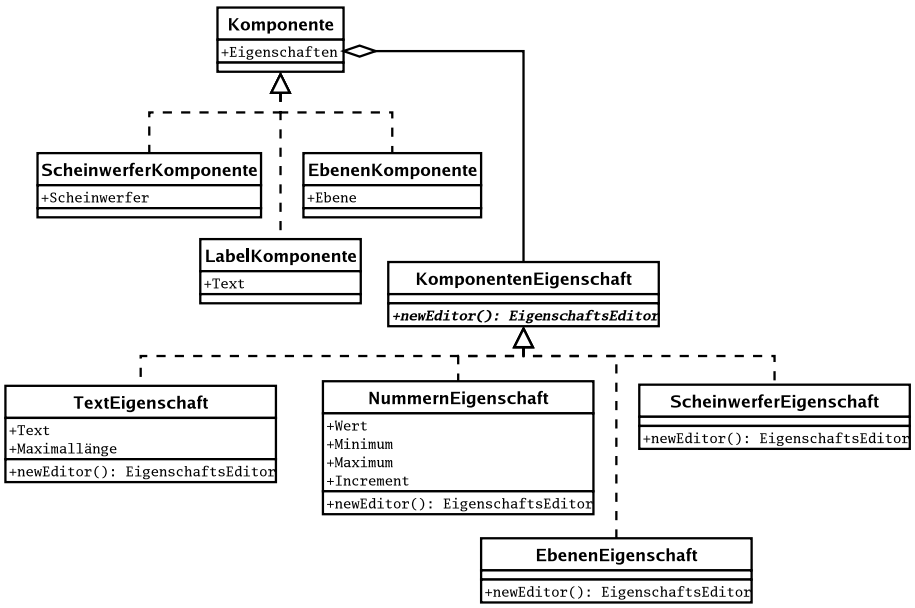

Abbildung 3.7.: Klassendiagramm - Bedienkomponenten und Eigenschaften

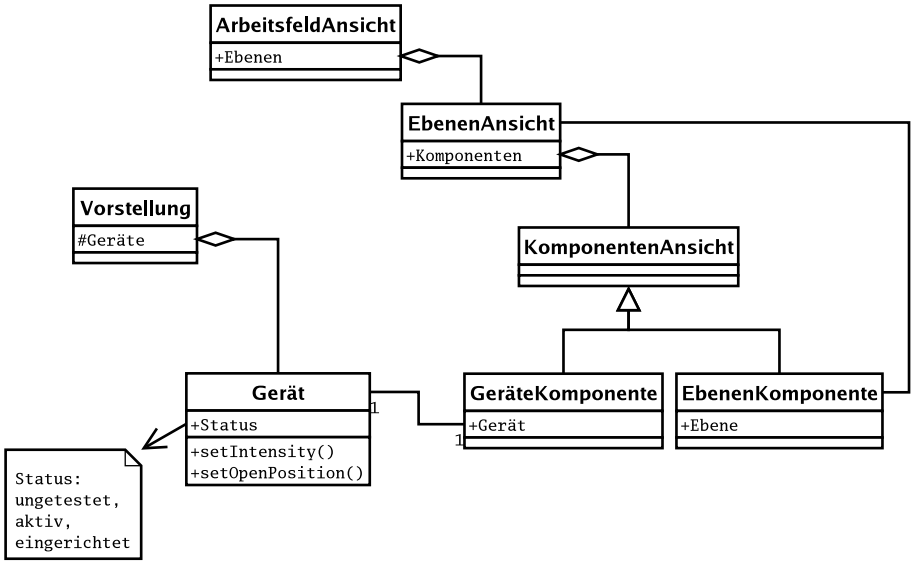

Abbildung 3.8.: Die für das Abrichten in Frage kommenden Klassen

3.7.3. Abrichten

In der Statusübersicht auf der linken Seite ist eine Summe der Stati aller Ebenen dargestellt. In der Übersichtsebene sind in jeder Ebenenauswahl bzw. -Komponente die Stati der Ebene zu sehen.

Da sich das Geräteobjekt auf dem Server befindet, wird die Statusänderung zentral an dem einen Objekt vorgenommen. Bei einer Statusänderung werden die anderen Gerätekomponenten, auf anderen Clients oder demselben, mittels des Observer-Pattern benachrichtigt. Auf diese Weise ist der Beleuchter im Stellwerk oder der Beleuchtungsmeister direkt über den Fortschritt des Einleuchtens im Bilde. Dafür berechnet der Controller, der die Statusänderung der Scheinwerfer steuert, nach einer solchen Änderung die Anzahl der zu einem Status gehörenden Geräte neu.

3.7.4. Bedienen

Beim Vorgang “Bedienen” werden in den Gerätekomponenten die Helligkeitswerte der Scheinwerfer angezeigt. Dabei sind sowohl die Soll- als auch die tatsächlich ausgegebenen Ist-Werte abgebildet. Es erfolgt die Weitergabe der Soll-Werte an das Beleuchtungspult, durch das sie geschleust werden, bis sie am Schluss an die Dimmer und Scheinwerfer ausgegeben werden. Die Ist-Werte werden an die Gerätekomponente in der Darstellung geleitet. Das Informieren der Komponente über die Wertänderung geschieht mittels Observer-Pattern.

An dieser Stelle ist darauf zu achten, das der Client nicht direkt die DMX-Kanäle ansteuert, sondern auf die Eigenschaften des Gerätes, wie die Helligkeit, zugreift. Der Zugriff auf die DMX-Kanäle erfolgt dann auf Seiten des Servers über den entsprechenden Adapter. Dadurch

ist gewährleistet, dass der Client auch für andere Zwecke eingesetzt werden kann, bei denen keine DMX-Ausgänge zur Verfügung stehen, wie bei ACN. Siehe auch Abbildung 3.9

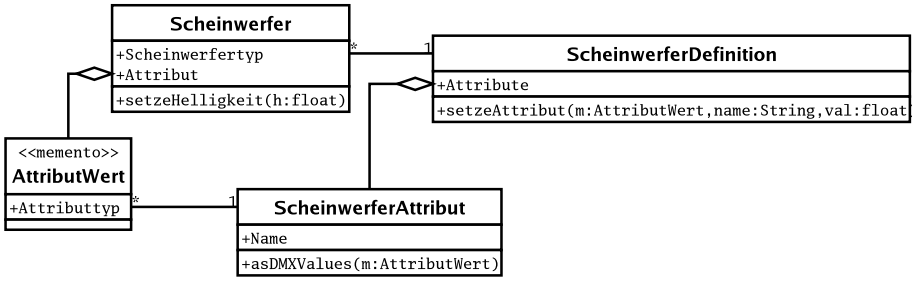

Abbildung 3.9.: Klassendiagramm - Scheinwerfer und Attribute

3.7.5. Stimmung bearbeiten

Eine zu bearbeitende Stimmung liegt in der Regel auf dem Server. Bei der Verwendung von CORBA spielt es prinzipiell keine Rolle, auf welchem Server sich die Stimmung befindet. So könnten die Stimmungen auch auf mehrere Rechner verteilt sein. Eine Stimmung ist sich nicht bewusst über die den Aufbau der Gerätezustände, die sie speichert. An dieser Stelle kann das Memento-Pattern angewendet werden.

Im Hinblick auf spätere Erweiterungen sollte jedoch beachtet werden, dass in einer Lichtsteuerung die Werte, die zu den Geräten in den Stimmungen gespeichert werden, mit den Werten in anderen Stimmungen verknüpft und anschließend auf die DMX-Ausgabe geleitet werden.

Abbildung 3.10 zeigt, wie die Datenflüsse innerhalb einer Lichtsteuerung zusammengeführt werden, um am Ende, meist über DMX, auf die Scheinwerfer zu treffen. Die Stimmungen, die Chaser und andere Effekte sind identisch mit den Datenquellen.

Bei Änderung des Helligkeitswertes eines Gerätes in der Darstellung werden die Wertänderungen an das Stimmungsobjekt auf dem Server geleitet, das dann die anderen Darstellungen mittels Observer-Pattern über die Änderung benachrichtigt. Auch die Datenflusseinheit muss über die Änderung des Zustandes informiert werden, um eine Neuberechnung der Ausgabedaten auszulösen.

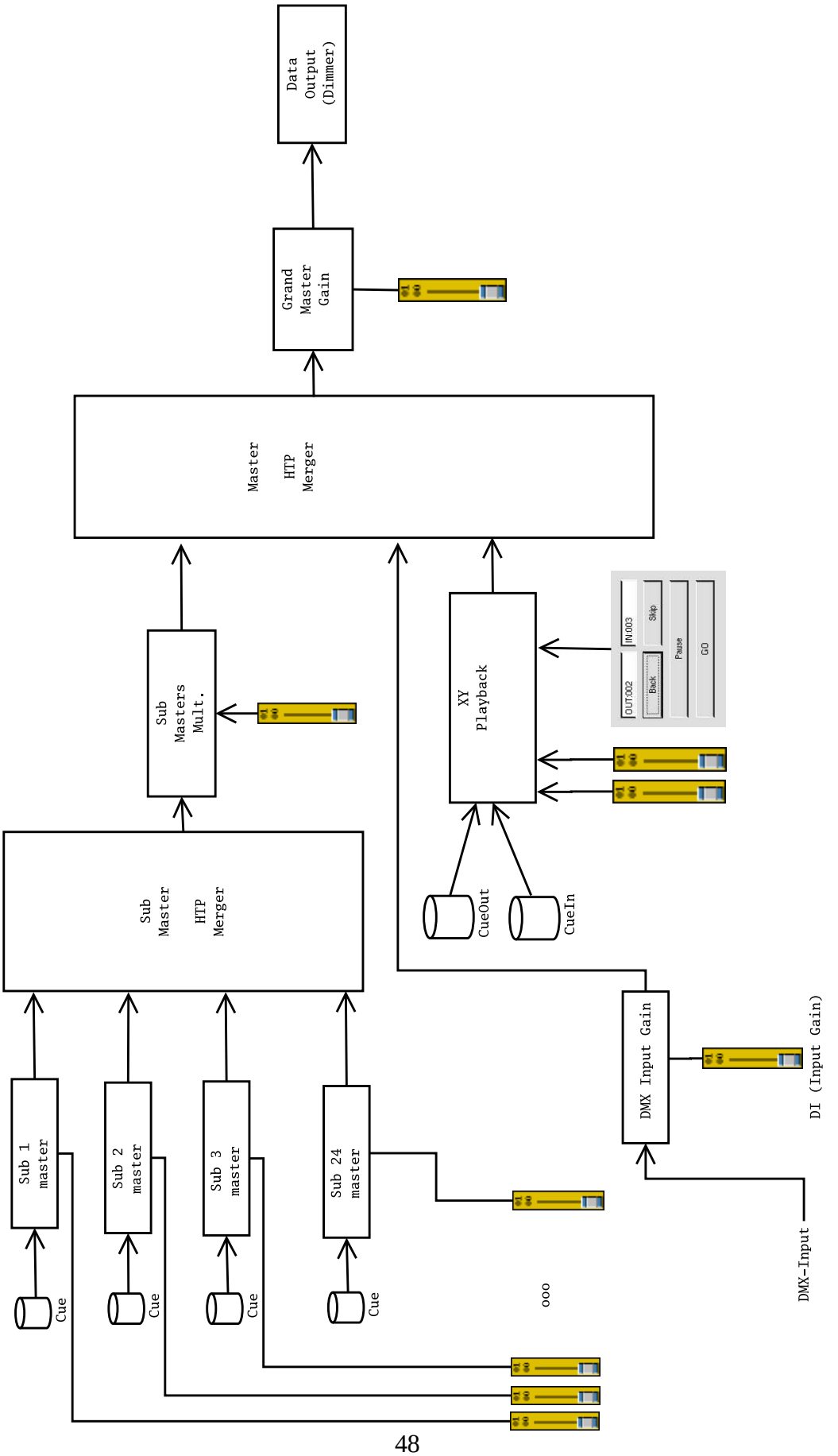

Abbildung 3.10.: Exemplarischer Datenfluss innerhalb eines Beleuchtungsstellwerks

3.8. Anbindung an das Beleuchtungsstellwerk

In Abbildung 3.11 ist zu erkennen, dass die Anbindung des Systems an das Beleuchtungspult über diverse Adapter stattfindet. Letztere abstrahieren den Zugriff auf die DMX-Ausgabe, die Gerätezuordnung, die Stimmungsdaten und die Steuerung von Teilen des angebundenen Stellwerks. Das ermöglicht den späteren Austausch des Beleuchtungsstellwerks.

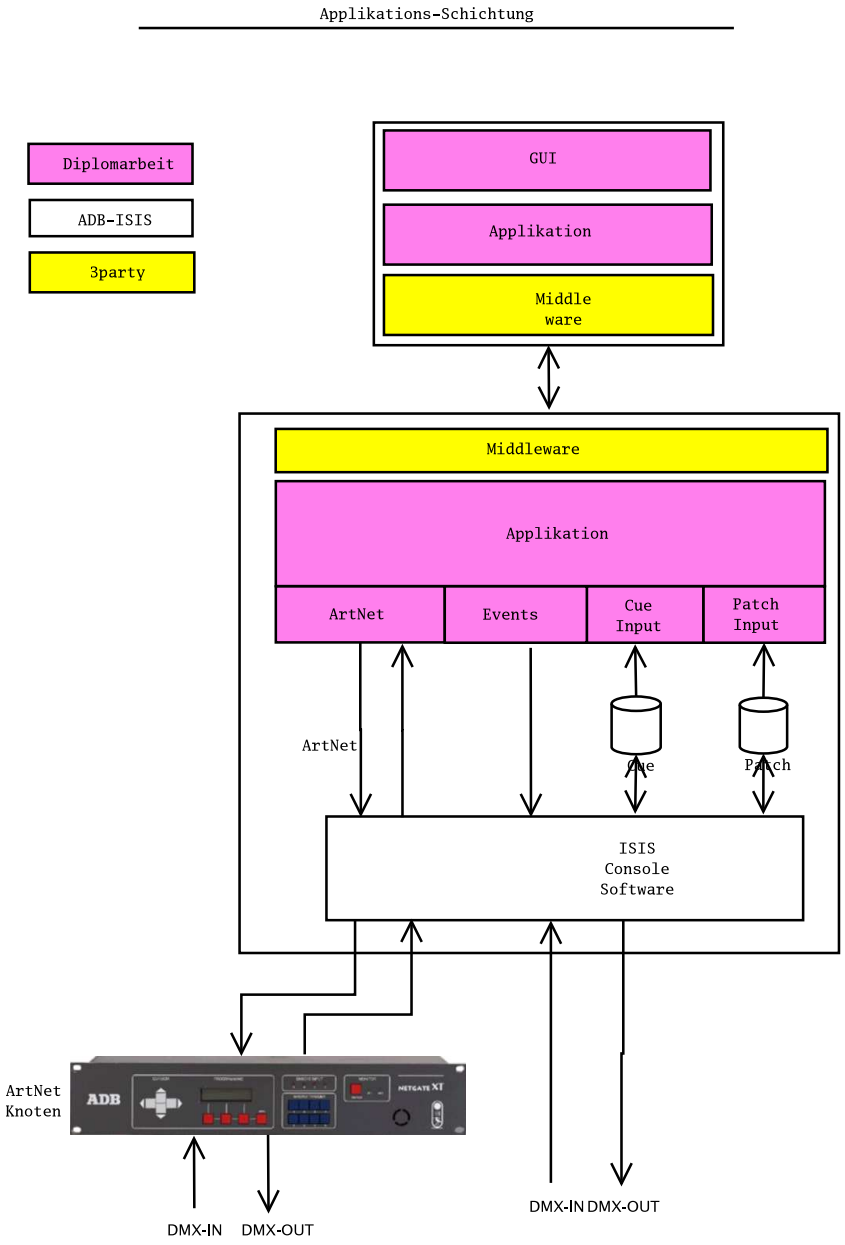

Abbildung 3.11.: Übersicht über die beteiligten Komponenten.

3.9. Realisierung

Zu den für die Realisierung verwendeten Werkzeugen zählen unter anderem TAO, openSLP und Qt (GUI/Application-Framework der Firma Trolltech).

TAO wurde auf Seiten des Servers für Linux übersetzt, was sich als problemlos erwies. Ebenso ergaben sich keinerlei Schwierigkeiten beim Übersetzen von openSLP.

Auf der Client-Seite kam Windows auf einem Tablet-PC zum Einsatz. Auch hier gelang nach ersten Anlaufschwierigkeiten das Übersetzen von TAO und openSLP.

3.10. Zusammenfassung

Die Beschäftigung mit Design und Realisierung in bezug auf das zu entwerfende System hat zu verschiedenen Entscheidungen geführt. Nach Untersuchung unterschiedlicher Middleware-Konzepte fiel die Wahl auf CORBA und speziell auf TAO. Das sich bei CORBA bezüglich des initialen NameService ergebende Bootstrap-Problem wurde mittels SLP gelöst.

Durch eine systematische Analyse der Anwendungsfälle entstand ein Klassenmodell. In diesem Zusammenhang wurde auch das Zusammenspiel der einzelnen Klassen untersucht.

4. Zusammenfassung und Ausblick

Ziel der Arbeit war es, ein mobiles verteiltes System für den Beleuchtungsbereich zu konstruieren. Das System sollte den Beleuchter beim Abrichten von Geräten vor einer Veranstaltung unterstützen, ihm die Möglichkeit bieten, einzelne Geräte “reinzuregeln” und Teile des Beleuchtungsstellwerks fernzusteuern.

Bei der Konstruktion des Systems wurde auf die Modularität geachtet, wodurch eine Austauschbarkeit einzelner Komponenten gewährleistet ist. Durch die Austauschbarkeit der Komponenten ergibt sich im späteren Einsatz der Vorteil der besseren Wartbarkeit, sowie der Änderbarkeit, um die Applikation an andere Beleuchtungsstellwerke anbinden zu können. Dem Anwender gegenüber würde sich das System in einem solchen Fall unverändert präsentieren.

Auch die Client-Applikation ist solchermaßen geartet, dass sie jederzeit ausgetauscht werden kann.

Während der Analyse hat sich gezeigt, dass das bestehende Beleuchtungstellwerk nur recht einfache Schnittstellen zur Kommunikation zur Verfügung stellt. Es ist zum Beispiel nicht möglich, auf vom Beleuchtungsstellwerk verwaltete Daten wie Stimmungen zuzugreifen.

Soll das System um weitere Funktionen, wie das Bearbeiten von Stimmungen und anderen Objekten, die das Beleuchtungsstellwerk verwaltet, erweitert werden, ist es notwendig, dass das Stellwerk eine erweiterte Schnittstelle zur Verfügung stellt. Auch könnten die Adapter und die Server-Applikation dieses Systems direkt in die Software des Beleuchtungsstellwerks integriert werden.

Ein weiteres - erreichtes - Ziel dieser Arbeit war es, zu zeigen, dass die heutigen Standards und Methoden der Informatik, auf denen diese Ausarbeitung ebenso basiert wie auf offenen Protokollen und Open-Source-Implementationen von verwendeten Komponenten, auch in der Entwicklung von Software für den Beleuchtungssektor anwendbar sind.

Der im Rahmen der Ausführungen vorgestellte Prototyp bietet durch seine Erweiterbarkeit eine geeignete Basis für spätere Ergänzungen.

Die in dieser Diplomarbeit diskutierten Aspekte zum Einsatz mobiler Systeme in der Beleuchtungstechnik stellen nur einen Teil der technischen Perspektiven in diesem Bereich dar. Eine weitere Möglichkeit bestünde zum Beispiel darin, Funktionen einzubinden, mit denen sich Stimmungen, Effekte, Chaser und Submaster verteilt programmieren ließen. Auch wäre es möglich, mehrere Konsolen solcherart miteinander zu verbinden, dass sich nicht nur die Redundanz erhöhen würde, sondern ebenfalls die Skalierbarkeit des gesamten Systems. Das

Resultat wäre eine weitere Steigerung der Effektivität im Arbeitsbereich der Anwender.

Wichtig ist es, eine klare Trennung zwischen den Ebenen einer N-Tier-Architektur vorzunehmen. Auf diese Weise können mobile Systeme wie Tablet-PC's nicht nur für wenige spezielle Aufgaben eingesetzt, sondern beispielsweise auch zum Programmieren von Stimmungen verwendet werden, oder gar den Betrieb der Vorstellung übernehmen.

Die Hybridkonsolen, die sowohl konventionelle als auch bewegte Scheinwerfer ansprechen können, bergen die Gefahr, entweder zu sehr mit Funktionen überfrachtet zu sein oder nicht ausreichende Funktionen zu bieten.

Es wird abzuwarten sein, ob sich CORBA im Bühneneinsatz bewährt und für diesen doch sehr "rauen" Bereich geeignet ist. Es bedurfte im Beleuchtungssektor immer mehr Zeit als in anderen Bereichen, bis sich eine Technologie etabliert hatte. Der Einsatz von Computernetzwerken stellt in dieser Hinsicht keine Ausnahme dar.

Alle großen Hersteller von Beleuchtungspulten bieten mittlerweile Netzwerkanbindungen ihrer Geräte an. Allerdings nutzen die meisten noch proprietäre Protokolle zur Kommunikation der Pulte untereinander. Durch die Anwendung dieser Protokolle wird auf viele Möglichkeiten verzichtet, wie sie sich zum Beispiel im Bereich des Routermarktes durch öffentlich zugängliche, von allen Firmen implementierte Protokolle bieten.

Es ist abzusehen, dass es noch einer langen Entwicklungsperiode bedarf, bis sich ACN bewährt, die ersten "Kinderkrankheiten" hinter sich gelassen und im Beleuchtungsbereich etabliert hat. Auch ist es ungewiss, ob bei den Herstellern ein Umdenken betreffs herstellerunabhängiger Protokolle einsetzen wird und sie diese zukünftig nutzen und unterstützen werden.

Da die Technik im Lichtbereich höchsten Ansprüchen in Bezug auf die Verfügbarkeit genügen und gleichzeitig von Anwendern installierbar sein muss, deren Kernwissen nicht im Bereich des Netzwerkmanagements liegt, ist es wichtig, sich selbst organisierende Systeme zu konstruieren, mittels derer Systemlandschaften mit minimalem Aufwand zu erstellen sind.

A. Beispiel zu ONC/RPC

Abbildung A.1.: Makefile Datei zum ONC/RPC Beispiel

```
#============================
# M A K E F I L E
#============================
all: server client

server : server.o test_svc.o test_xdr.o

client : client.o test_clnt.o test_xdr.o

server.o : server.c test.h

client.o : client.c test.h

test.h test_clnt.c test_svc.c test_xdr.c : test.x
        rpcgen  $<

clean:
        -rm -f *.o test.h test\_clnt.c test\_svc.c test\_xdr.c
```

Abbildung A.2.: square.x Datei zum ONC/RPC Beispiel

```
/*============================
 * S Q U A R E . X
 *============================*/
struct square_in { long arg1; };
/* Input (argument) */

struct square_out { long res1; }; /* output (result) */

program SQUARE_PROG {
  version SQUARE_VERS {
    square_out SQUAREPROC (square_in) = 1; /* procedure number 1 */
  } = 1; /* version 1 */
} = 0x31230000; /* program number */
```

Abbildung A.3.: server.c Datei zum ONC/RPC Beispiel

```
/*============================
 * S E R V E R . C
 *============================*/
#include "square.h"
square_out * squareproc_1_svc(square_in *parm, struct svc_req *req)
{
  static square_out ret;
  ret.res1 = parm->arg1 * parm->arg1;
  return &ret;
}
```

Abbildung A.4.: client.c Datei zum ONC/RPC Beispiel

```
/*============================
 * C L I E N T . C
 *============================*/
#include "square.h"
int main (int argc, char **argv)
{
  CLIENT      *cl;
  square_in    in;
  square_out  *out;
  if (argc != 3)
    {
      fprintf (stderr, "usage: %s <hostname> <integer-value>\n", argv[0]);
      return 1;
    }
  cl = clnt_create (argv[1], SQUARE_PROG, SQUARE_VERS, "tcp");
  in.arg1 = atol(argv[2]);
  out = squareproc_1(\&in, cl);
  if (out == NULL)
    {
      fprintf (stderr, "error in RPC call\n");
      return 1;
    }
  printf ("result: %ld\n", out->res1);
}
```

B. Abbildungen

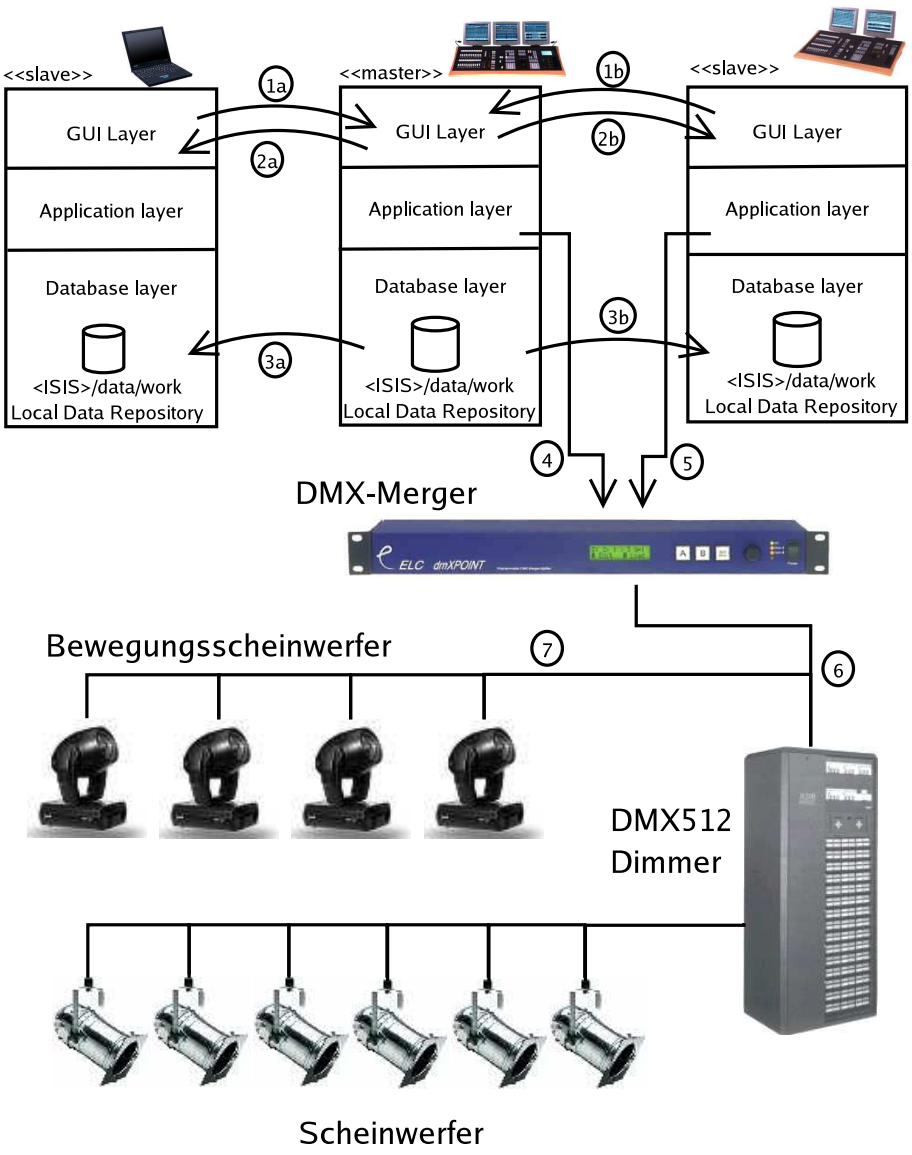

Abbildung B.1.: Bisherige Kommunikation I

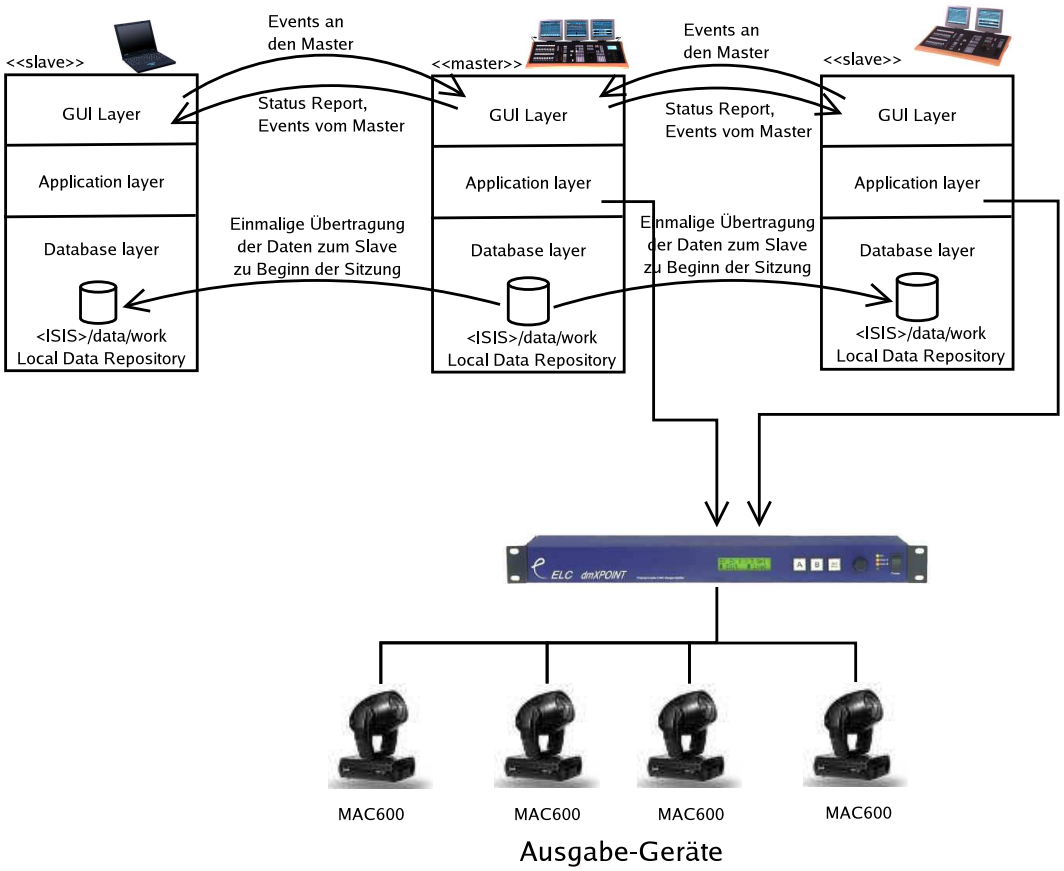

Abbildung B.2.: Bisherige Kommunikation II

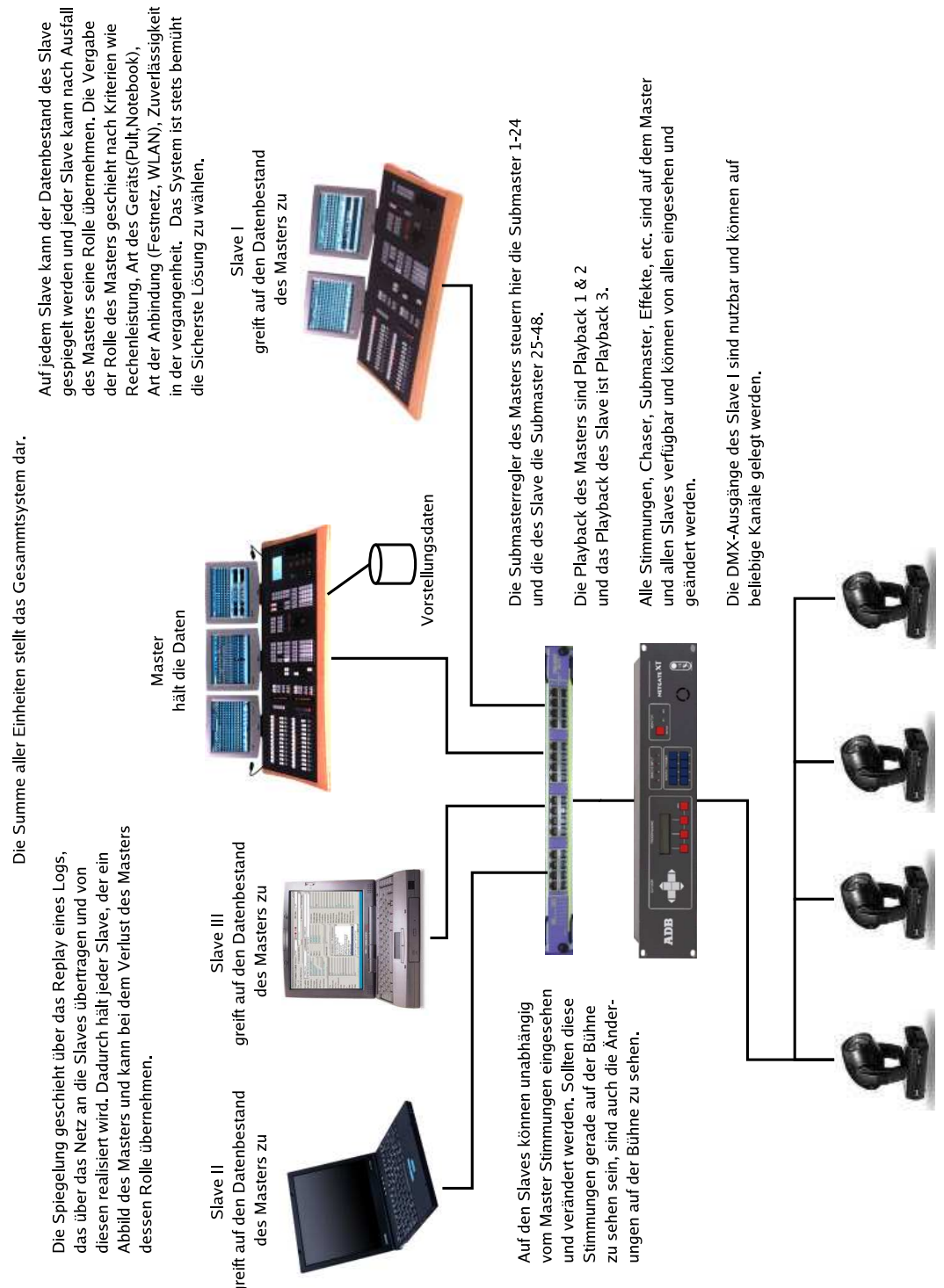

Abbildung B.3.: Mögliche Kommunikation eines modernen Beleuchtungssystems I

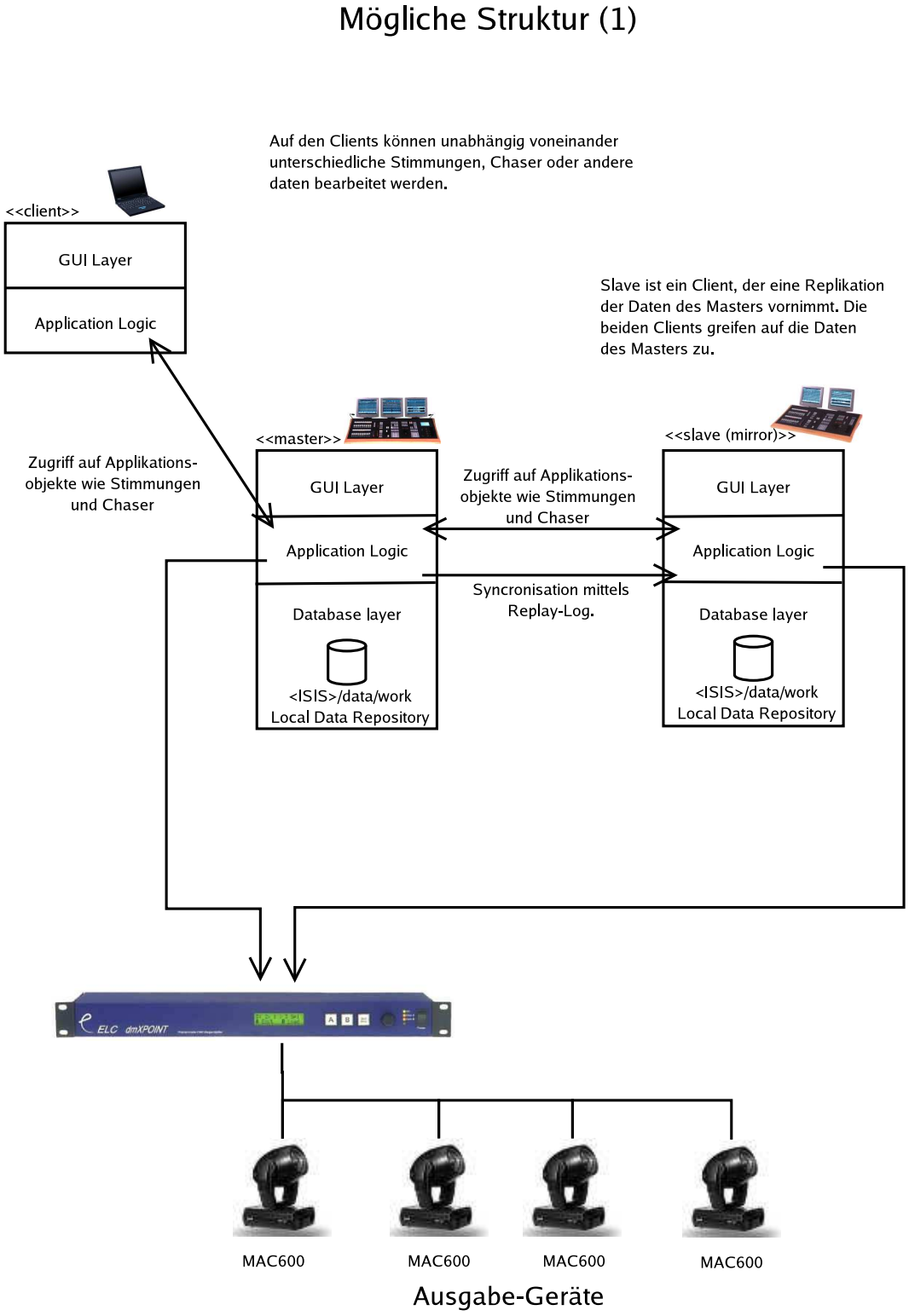

Abbildung B.4.: Mögliche Kommunikation eines modernen Beleuchtungssystems II

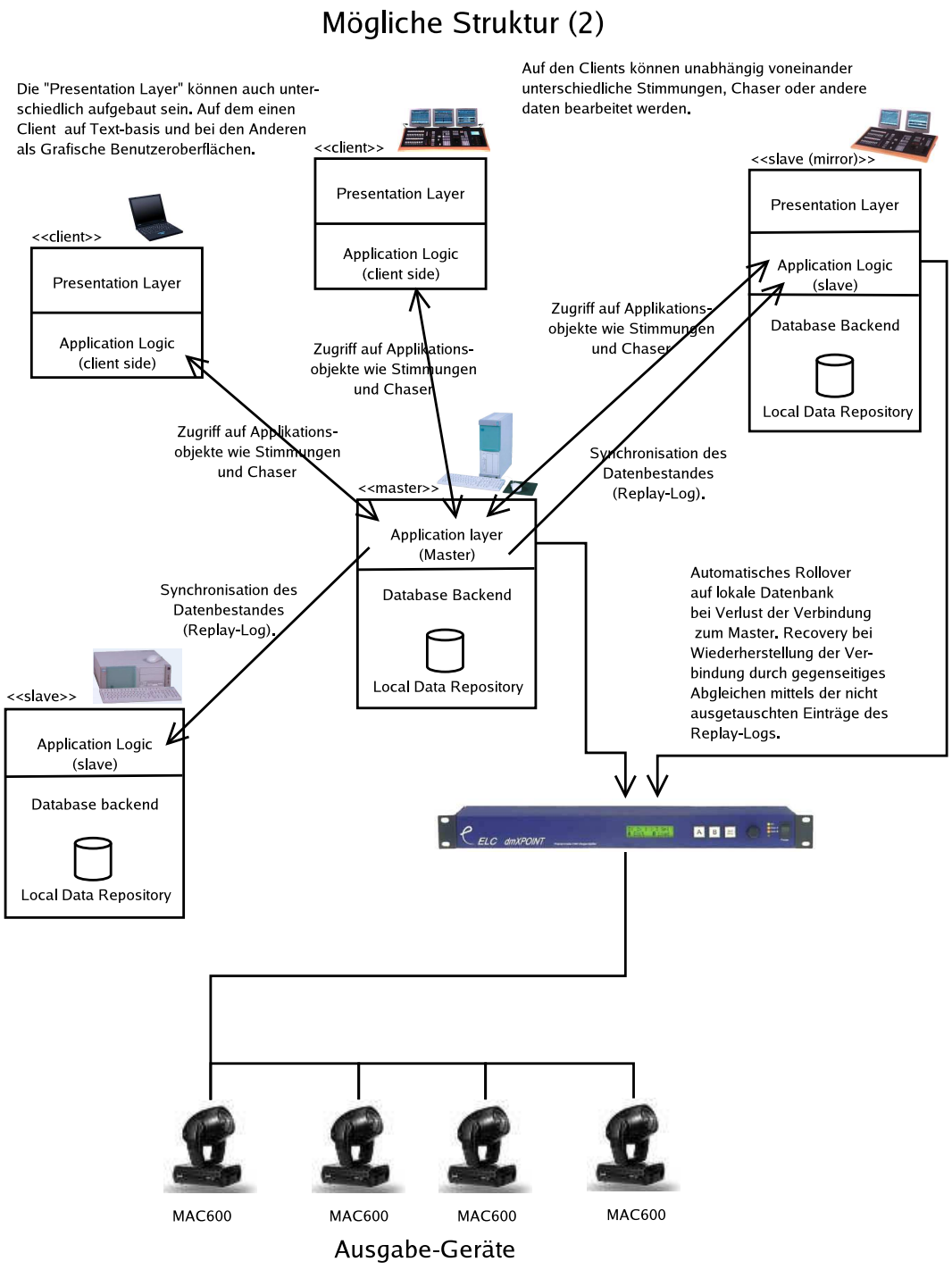

Abbildung B.5.: Mögliche Kommunikation eines modernen Beleuchtungssystems III

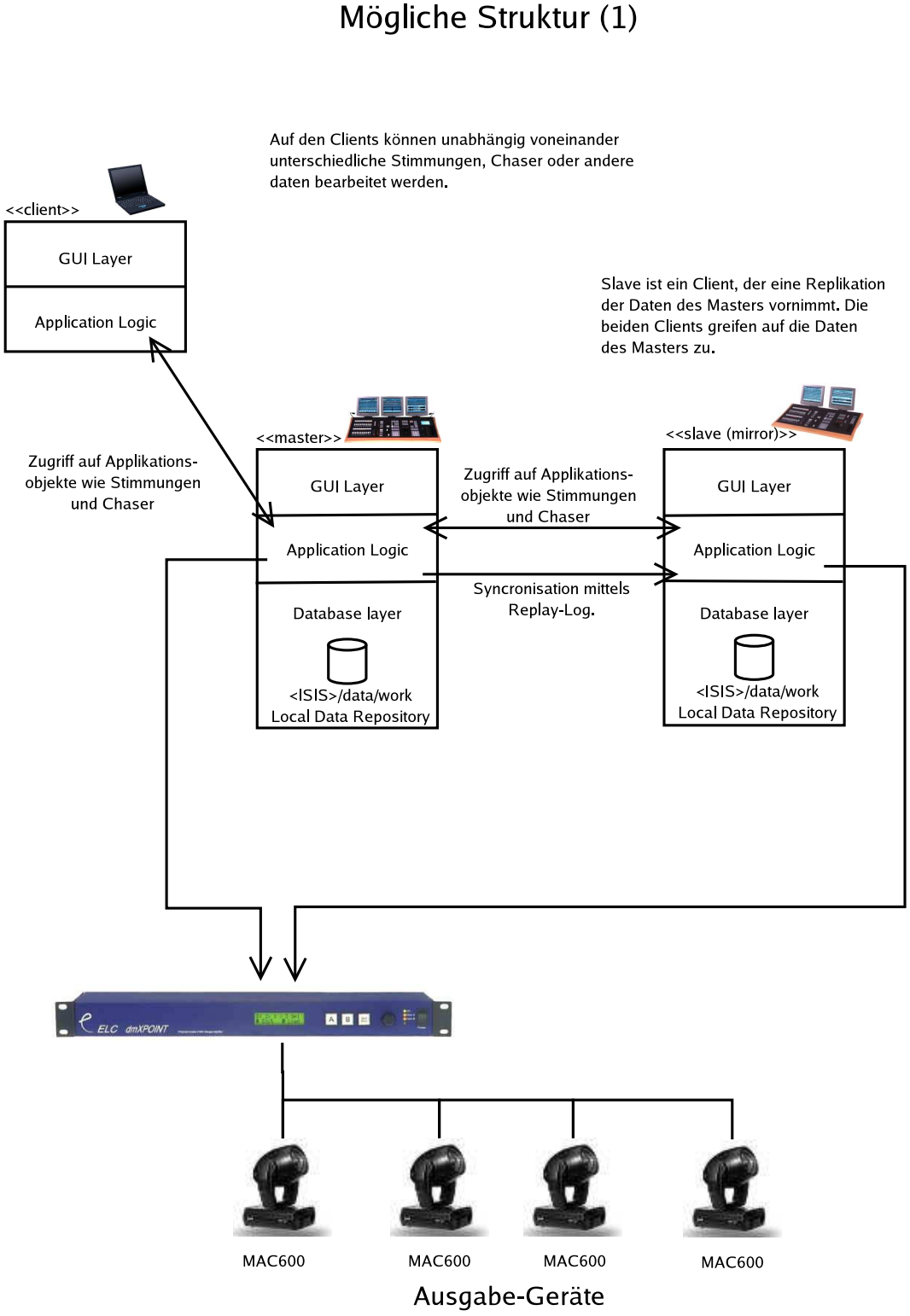

Abbildung B.6.: Mögliche Kommunikation eines modernen Beleuchtungssystems IV

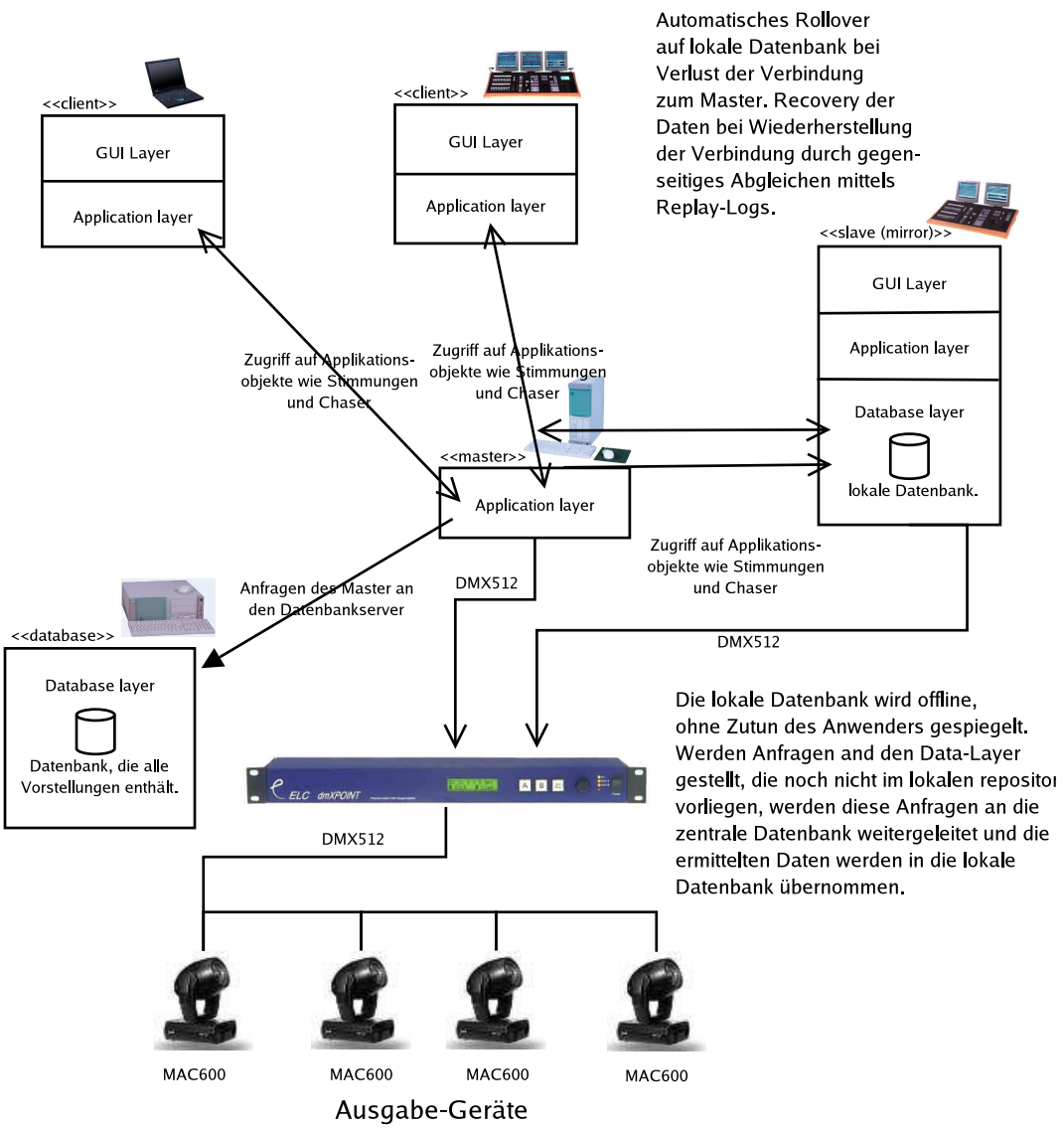

Abbildung B.7.: Mögliche Kommunikation eines modernen Beleuchtungssystems V

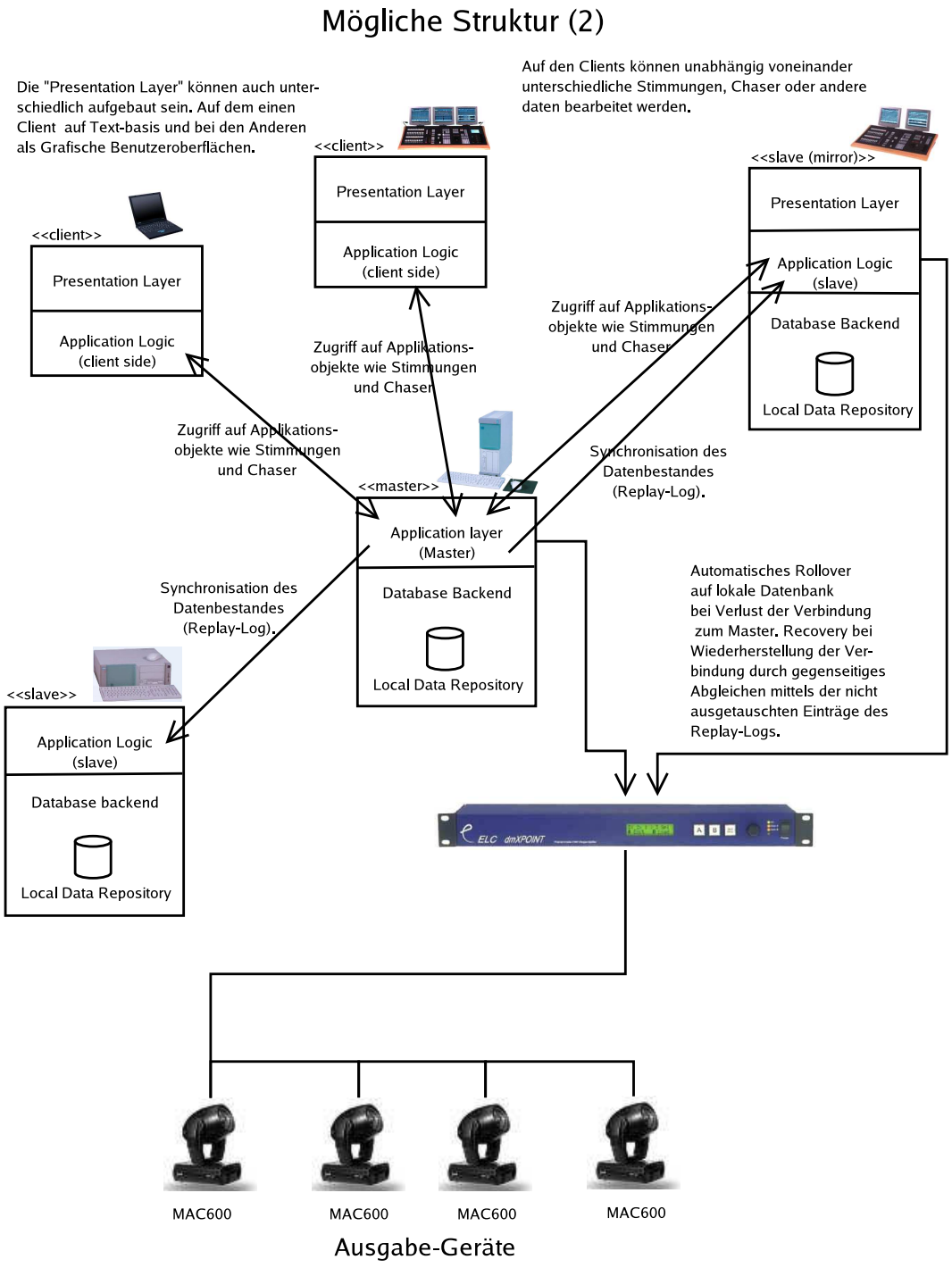

Abbildung B.8.: Mögliche Kommunikation eines modernen Beleuchtungssystems VI

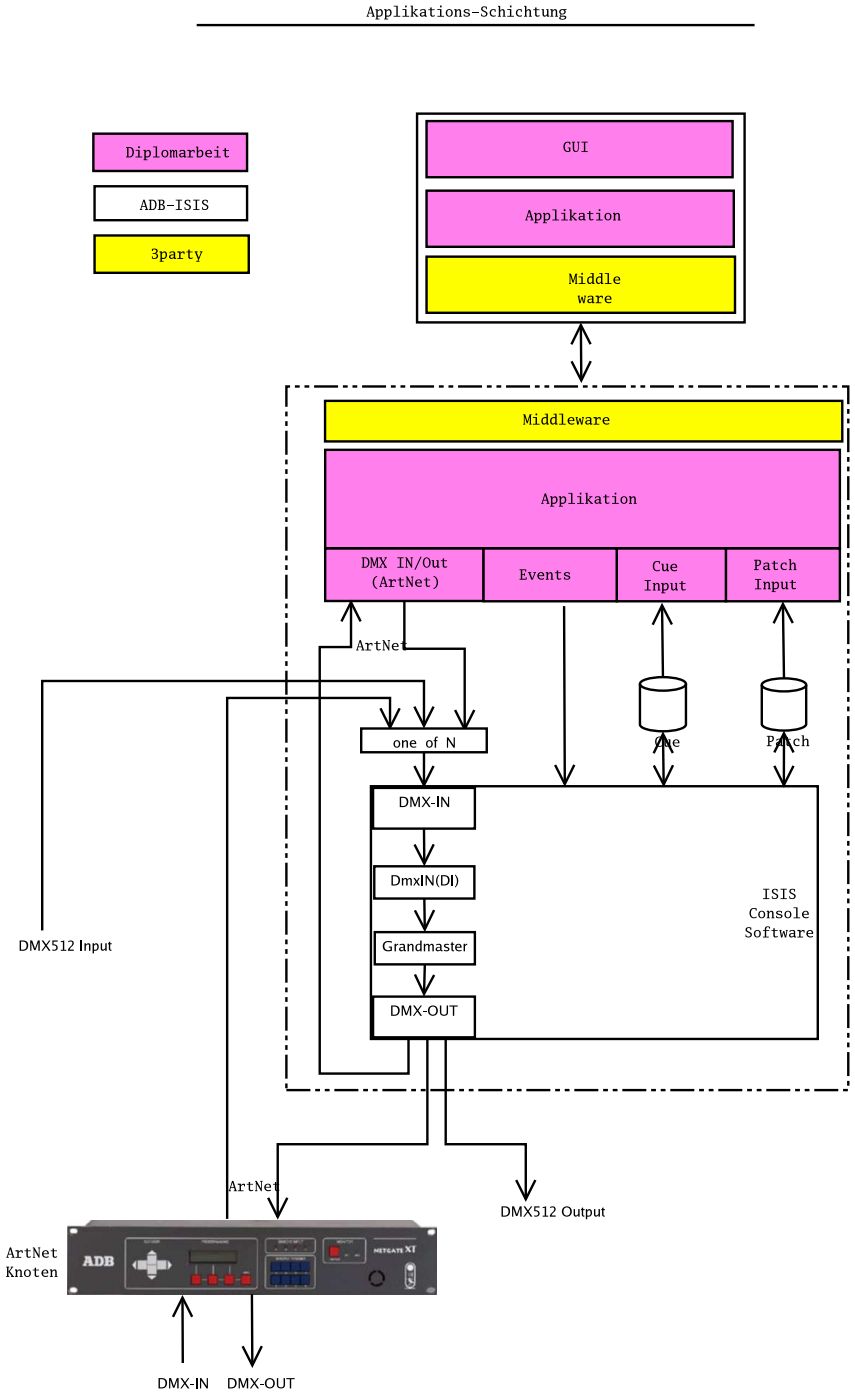

Abbildung B.9.: Übersicht über die beteiligten Komponenten.

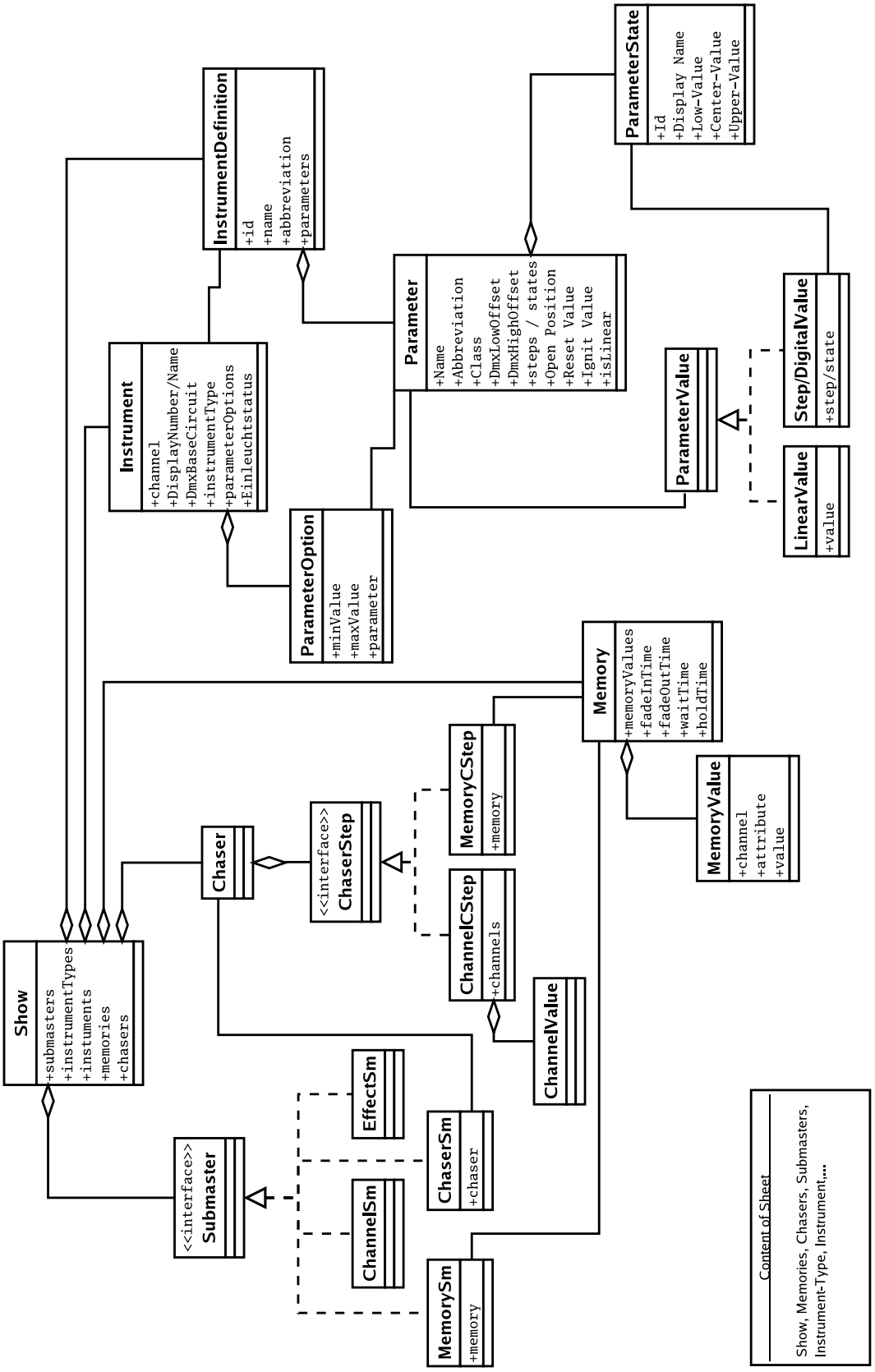

Abbildung B.10.: Klassenmodell I.

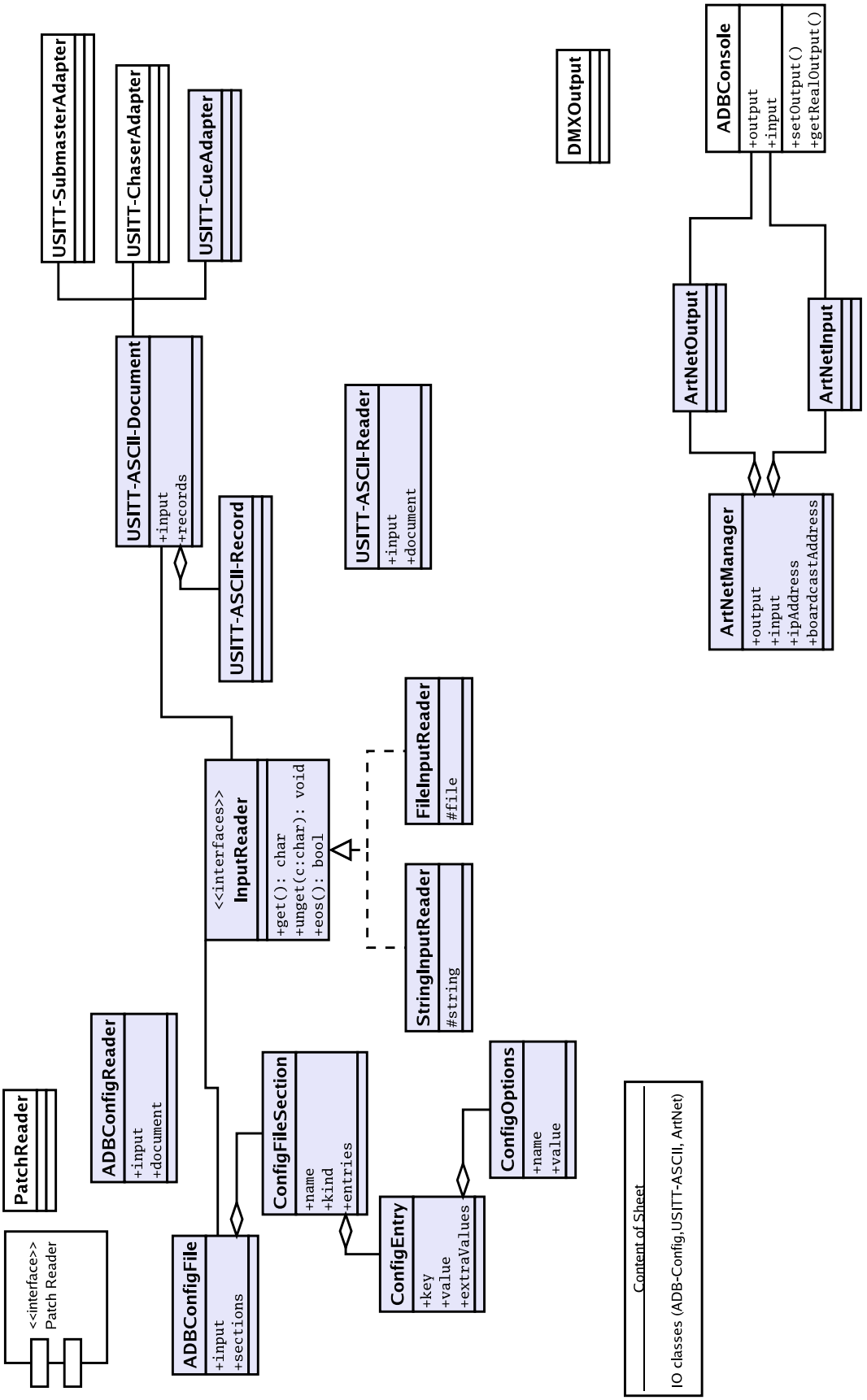

Abbildung B.11.: Klassenmodell II.

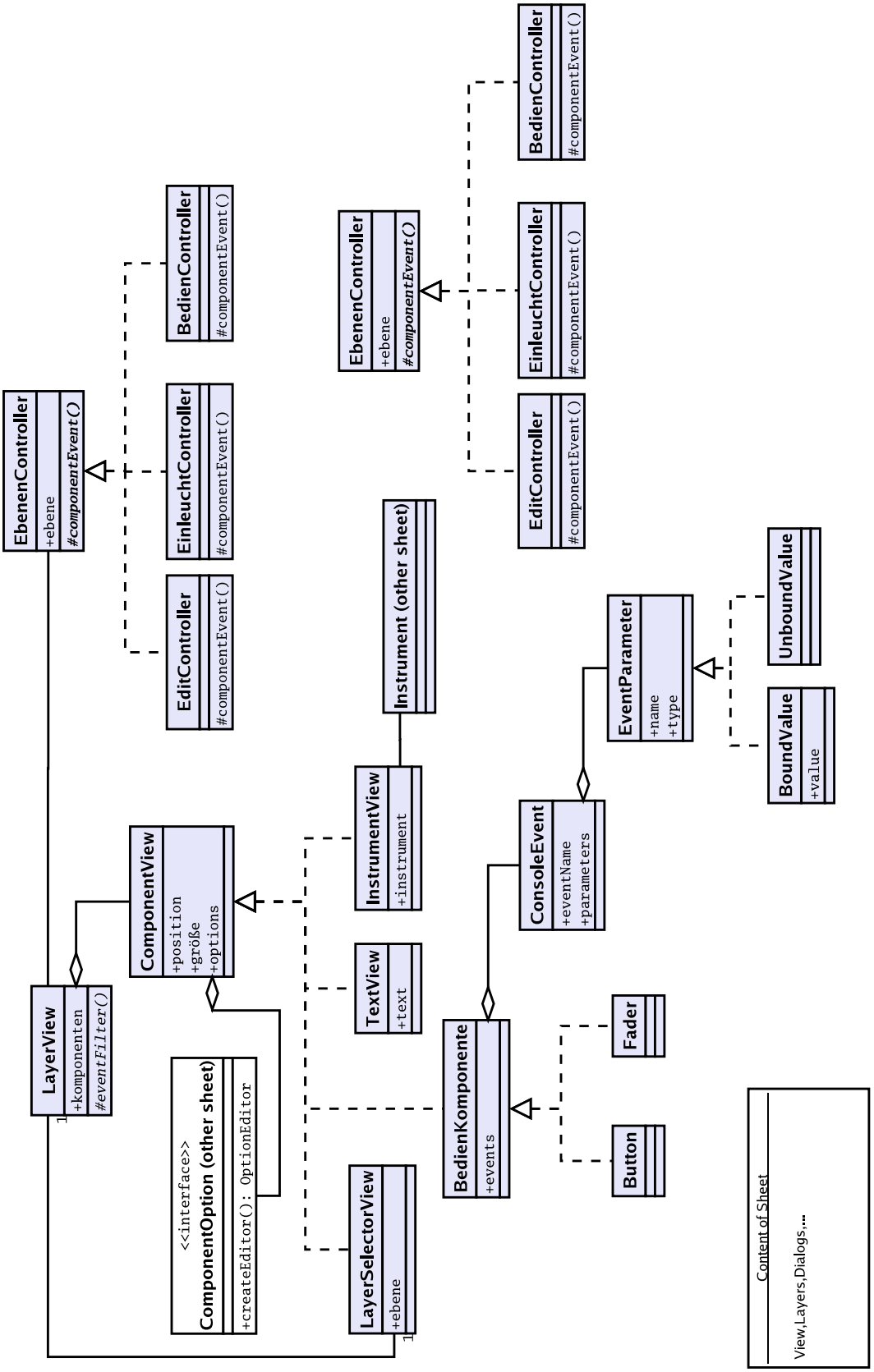

Abbildung B.12.: Klassenmodell III.

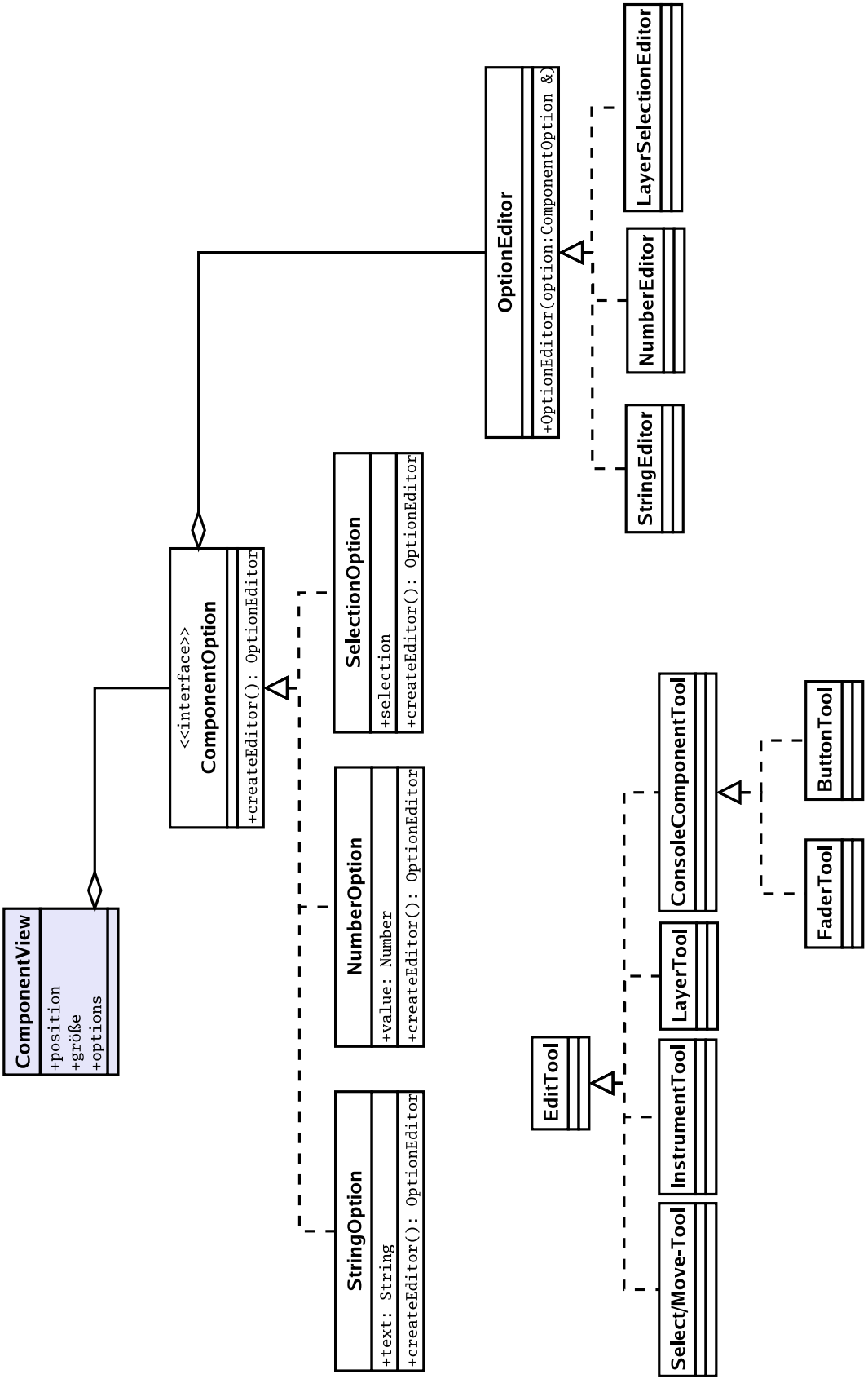

Abbildung B.13.: Klassenmodell IV.

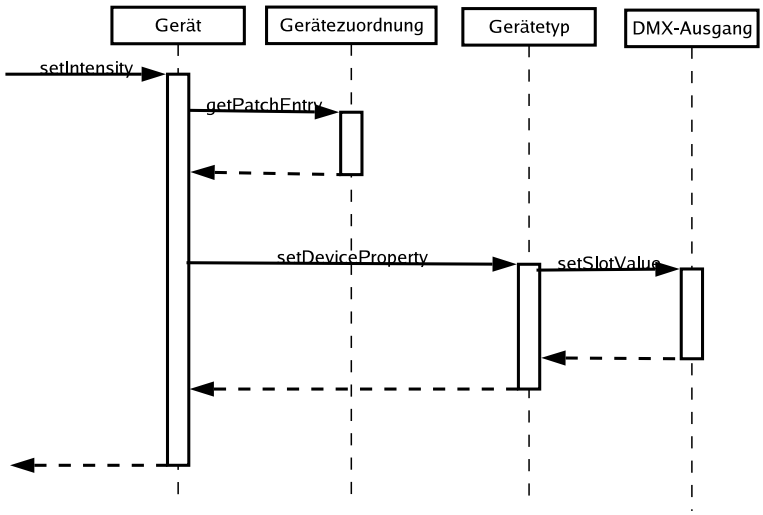

Abbildung B.14.: Sequenzdiagramm zur Gerätemethode setIntensity

C. Glossar

ACN: Advanced Control Network, kurz ACN, ist eine Sammmlung von Protokollen zur Steuerung und Verwaltung von Komponenten und wurde hauptsächlich für den Beleuchtungs- und den Tonbereich entwickelt. ACN ist noch nicht verabschiedet, steht aber kurz vor der Fertigstellung.

Analoge Ausgabe (0..10V): Vor der Einführung digitaler Übertragungsprotokolle wie DMX512 oder Netzwerkprotokollen wie ArtNet oder ACN wurden die Informationen, meist die Helligkeit betreffend, mittels analoger Spannungen übertragen. Für 24 Kanäle wurde ein Kabel mit 25 Adern verlegt: eine Ader pro Kanal sowie eine gemeinsame Ader als “common ground” (0V).

ArtNet: Protokoll zu Übertragung von DMX512 Daten über Ethernet. Dieses Protokoll ist von der Firma Artistic Licence entwickelt worden und die Spezifikation ist frei verfügbar.

Blende / Torblende: Absperreinheit, die dazu führt, dass der Lichtstrahl partiell abgedeckt wird. Sie wird eingesetzt, um klare Abgrenzungen an Kanten, wie zum Beispiel der Bühnenkante oder einer Wandecke, zu erreichen.

Chaser: Eine Abfolge von Stimmungen mit festgelegten Pausenzeiten. Der Chaser wird häufig für Lauflichteffekte eingesetzt.

D54, AMX192: Ein analoges Verfahren zum Übertragen von Dimmerwerten, bei dem die Analogwerte zeitlich gemultiplext auf einer Leitung übertragen werden. Hinzu kommt ein common ground, sowie eine Taktleitung bei D54 und zwei symmetrisch ausgelegte Taktleitungen bei AMX192. D54 benötigt daher ein 3-adriges Kabel und AMX192 ein 4-adriges Kabel. Über D54 können 384 Kanäle und über AMX192 192 Kanäle übertragen werden.

DMX512 / DMX512-A: Protokoll zur Übertragung von bis zu 512 Dimmerwerten über eine serielle Verbindung, die mit einer Geschwindigkeit von 250KBaud (250.000 Bits / Sekunde) übertragen werden. DMX512 ist ein reiner “Master Slave” Bus auf RS485 Basis. DMX512-A liefert die Möglichkeit einer bidirektionalen Kommunikation, um eine Enumeration der vorhandenen Geräte sowie das Abfragen von Statusinformationen zu ermöglichen.

DMX-Merger: Eine Komponente, die die Daten mehrerer DMX-Universen zu einem zusammenfasst. Als Verfahren zur Kombination der einzelnen DMX-Universen werden von den Geräten häufig HTP, LoTP und LaTP angeboten.

DMX-Universum: Ein DMX-Universum umfasst bis zu 512 DMX-Slots und verwendet das DMX512 Protokoll zur Übertragung der DMX-Slots. Es existieren diverse Netzwerk-Protokolle zum Übertragen von DMX-Slots, wie beispielsweise “ArtNet” von der Firma Artistic Licence oder “ShowNet” von der Firma Strand Lighting.

DMX-Slot: Ein einzelner 8-Bit Wert bei dem DMX512-Protokoll. Bei Bewegungsscheinwerfern werden oft zwei SLots zu einem 16-Bit Wert zusammengefasst.

Einleuchten: Vorgang, der vor jeder Veranstaltung stattfindet und bei dem alle Geräte auf ihre Funktion und ihre korrekte Ausrichtung (Position, Größe, Farbe, et cetera) hin überprüft werden.

Gruppe: Zustand der beteiligten Scheinwerfer zu einem bestimmten Zeitpunkt.

Helligkeit: Intensität des Lichtstrahls, der bei den Beleuchtungs-Konsolen zumeist in Prozent (%) angegeben wird.

HTP: Highest Takes Place (HTP) ist ein Verfahren zum Mergen (Zusammenfügen) mehrerer Quellen zu einer Information. Hierbei wird die “max” Funktion verwendet, also die vom Wert her größte Information an das Ziel weitergegeben.

Hybridkonsole: Ein Beleuchtungsstellwerk, das darauf ausgelegt ist, sowohl konventionelle Scheinwerfer als auch bewegte Scheinwerfer (Moving Lights) anzusteuern.

Konsole: Systemeinheit, die die Steuerung der Beleuchtungsgeräte übernimmt.

Kontente: Verzeichnis aller Scheinwerfer mit Hinweisen zum Aufbau. Die Kontente wird zum Aufbau der Scheinwerfer vor dem Beginn der Veranstaltung benötigt.

Bühnenbuch:

Kanal: Der Träger eines Helligkeitswertes, zumeist in Prozent (%) angegeben.

LoTP: Lowest Takes Place (LoTP, manchmal auch LTP) ist ein Verfahren zum Mergen (Zusammenfügen) mehrerer Quellen zu einer Information, wie auch HTP oder LaTP. Hierbei wird die “min” Funktion verwendet, also der kleinste Wert herangezogen und an das Ziel weitergegeben.

LaTP: Latest Takes Place (LaTP, vereinzelt auch LTP) ist ein Verfahren zum Mergen (Zusammenfügen) mehrerer Quellen zu einer Information, wobei im Gegensatz zu HTP und LoTP, der zuletzt gesetzte Wert berücksichtigt wird.

Mergen: Das Mergen hat in der Beleuchtungstechnik die Aufgabe des Zusammenfügens mehrerer Quellinformationen zu einer Zielinformation. Gängige Verfahren sind hierbei HTP, LoTP, LaTP. LoTP oder LaTP werden auch häufig LTP genannt, da der Name nicht einheitlich festgelegt ist.

Merger: Siehe DMX-Merger.

Moving Light: Auch "Bewegungsscheinwerfer". Scheinwerfer, der über die Möglichkeit verfügt, seine Position mittels motorischer Antriebe zu verändern, wobei bezüglich der Position meist nur Pan und Tilt beeinflusst werden. Des Weiteren können viele Moving Lights auch die Farbe, Größe sowie andere Eigenschaften des austretenden Lichtstrahls beeinflussen.

Nullstimmung: Eine Nullstimmung ist eine Stimmung, in der alle an der Vorstellung beteiligten Kanäle (Geräte) eine definierte Intensität haben, zum Beispiel 3%.

Open Position: ADB (eine Firma für Beleuchtungstechnik) spezifisch. Angabe einer Position für Bewegungsscheinwerfer, in der der Scheinstrahl in der Mittelposition steht und alle vorhandenen Filter oder Shutter offen sind, den Lichtstrahl also nicht beeinflussen.

Patch: Ein Patch ist in der Beleuchtungstechnik die Zuordnung eines Kanals in einer Quellmenge zu einem Kanal in einer Zielmenge. Es kann dabei mehrere Patches geben. Ein Ausgabe-Patch zum Beispiel kann dazu genutzt werden, die logischen Kreise physikalischen Kanälen zuzuordnen. Ein Eingabe-Patch legt die Zuordnung von physikalischen Kanälen zu logischen Eingangskreisen fest. Die Implementation und Nutzung ist von System zu System unterschiedlich. Die Kardinalitäten der Zuordnungen (1:1), (1:N) sind üblich, aber von System zu System verschieden.

Position: Ein Attribut eines Scheinwerfers. Bei Moving Lights ist es möglich, die Position über die Konsole zu beeinflussen.

RDM: Remote Device Management (RDM) ist ein Protokoll zur Verwaltung von Beleuchtungsgeräten. Es wird sowohl bei DMX512-A als auch bei ACN angewendet.

Scheinwerfer: Beleuchtungskörper.

Shutter: Absperreinheit, die dazu führt, dass bei eingeschalteter Lampe kein Licht aus dem Lampengehäuse austritt.

SLP: Das Service Location Protocol (SLP) ist ein Protokoll zum Anbieten und Erfragen von Diensten und wird im RFC 2608 (siehe auch RFC3105) spezifiziert. Mit Hilfe von SLP kann sich ein Dienst an einem oder mehreren SLP-Servern im Netz anmelden. Dienstsuchende werden in die Lage versetzt, nach allen oder bestimmten Diensten zu suchen. SLP wird unter anderem in der TCP/IP Implementation von Novel Netware und in ACN verwendet.

Stimmung: Zustand der beteiligten Scheinwerfer zu einem bestimmten Zeitpunkt. Hinzu kommen, je nach Stellwerk, Eigenschaften wie Wartezeit, Einblendzeit, Ausblendzeit oder eine Folgestimmung. Dies ist prinzipiell eine Erweiterung der Gruppe um zusätzliche Eigenschaften bzw. Attribute.

Literaturverzeichnis

[Abo04] B. Aboba. Rfc 3723 - securing. http://www.networksorcery.com/enp/rfc/rfc3723.txt, 2004.

[Bro04] Christopher Browne. Transaction processing monitors. http://cbbrowne.com/info/tpmonitor.html, 2004.

[Fle04] P. Fleming. Rfc 3712 - lightweight directory access protocol (ldap): Schema for printer services. http://www.networksorcery.com/enp/rfc/rfc3712.txt, 2004.

[FM96] Stefan Fischer and Walter Müller. *Netzwerkprogrammierung unter LINUX und UNIX*, volume 1. Carl Hanser Verlag, 1996.

[Gut99] E. Guttman. Rfc 2608 - service location protocol, version 2. http://www.networksorcery.com/enp/rfc/rfc2608.txt, 1999.

[Gut01a] E. Guttman. Rfc 3059 - attribute list extension for the service location protocol. http://www.networksorcery.com/enp/rfc/rfc3059.txt, 2001.

[Gut01b] E. Guttman. Rfc 3111 - service location protocol modifications for ipv6. http://www.networksorcery.com/enp/rfc/rfc3111.txt, 2001.

[Gut02] E. Guttman. Rfc 3224 - vendor extensions for service location protocol, version 2. http://www.networksorcery.com/enp/rfc/rfc3224.txt, 2002.

[Hei03] Hayat Heinrich. Auswahl einer middleware für mobile clients. Master's thesis, Hochschule für Angewandte Wissenschaften Hamburg, jul 2003.

[HV99] M. Henning and S. Vinoski. *Advanced CORBA©Programming with C++*, volume 1. Addison-Wesley, 1999.

[Kem99] J. Kempf. Rfc 2614 - an api for service location. http://www.networksorcery.com/enp/rfc/rfc2614.txt, 1999.

[Kem00] J. Kempf. Rfc 2926 - conversion of ldap schemas to and from slp templates. http://www.networksorcery.com/enp/rfc/rfc2926.txt, 2000.

[Kem01a] J. Kempf. Rfc 3082 - notification and subscription for slp. http://www.networksorcery.com/enp/rfc/rfc3082.txt, 2001.

[Kem01b] J. Kempf. Rfc 3105 - finding an rsip server with slp. http://www.networksorcery.com/enp/rfc/rfc.txt, 2001.

[Kid01] Eric Kidd. Xml-rpc howto. http://xmlrpc-c.sourceforge.net/xmlrpc-howto/xmlrpc-howto.html, 2001.

[KR01] James F. Kurose and Keith W. Ross. *Computernetze, Ein Top-Down-Ansatz mit Schwerpunkt Internet*, volume 1. Addison Wesley Longman, 2001.

[Mit03] Nilo Mitra. Soap version 1.2 part 0: Primer. http://www.w3.org/TR/2003/REC-soap12-part0, 2003.

[Pir03] Vito Piraino. Soap:rpc und asynchrones soap. http://n.ethz.ch/student/jodaniel/37-310/reports/seminar_vito_piraino.p%df, 2003.

[Sad04] Darleen Sadoski. Transaction processing monitor technology. http://www.sei.cmu.edu/str/descriptions/tpmt.html, 2004.

[Ste98] W. Richard Stevens. *UNIX Network Programming / Interprocess Communications*, volume 2. Prentice Hall PTR, 1998.

[TM01] TU-München. Asynchrone rpc's. http://www11.informatik.tu-muenchen.de/lehre/lectures/ss1994/va/chap_3/%async.html, 2001.

[US03] Inc. UserLand Software. Xml-rpc home page. http://www.xmlrpc.com/, 2003.

[Vei97] J. Veizades. Rfc 2165 - service location protocol. http://www.networksorcery.com/enp/rfc/rfc2165.txt, 1997.

[Wen04] Piotr Wendt. Entwicklung corba-basierter middleware für mobile anwendungen. Master's thesis, Hochschule für Angewandte Wissenschaften Hamburg, jun 2004.

[Zha02] W. Zhao. Rfc 3421 - select and sort extensions for the service location protocol (slp). http://www.networksorcery.com/enp/rfc/rfc3421.txt, 2002.

[Zha03] W. Zhao. Rfc 3528 - mesh-enhanced service location protocol (mslp). http://www.networksorcery.com/enp/rfc/rfc3528.txt, 2003.

Abbildungsverzeichnis