TU CHEMNITZ

Fakultät für Informatik

Studienarbeit Projektor-Management

Lars Wolf

Betreuer: Ralf König

Präambel

Ziel dieser Arbeit ist die Untersuchung der Eigenschaften von managebaren Projektoren. Dabei ist zu beachten, welche Management Standards es gibt und wie diese von den einzelnen Herstellern umgesetzt werden. Der Schwerpunkt dieser Arbeit - die Analyse der Hardwareschnittstellen, der proprietären Protokolle und ihrer Verbreitung – bildet die Grundlage zum Entwurf einer Software, die aufgezeigte Management Probleme behandelt. Weiterhin ist zu klären, inwieweit existierende Lösungen dies umsetzen und welche Verbesserungen möglich sind. In das, während dieser Arbeit initiierte, Open-Source Softwareprojekt, sollen die hier gewonnen Erkenntnisse zukünftig einfliesen. Dazu hilft ein abschließender Ausblick auf erweiterbare Technologien.

1 Aspekte des Projektor Managements

1.1 Begriffsdefinition

Ein **Projektor** (lat. proicere "vorwärtswerfen, hinwerfen") ist ein optisches Gerät, das ein vergrößertes Bild eines Gegenstandes auf einer Projektionsfläche erzeugt [W01]. Ein Projektor dient allgemein dem Zweck, kleinformatige, visuelle Inhalte so zu vergrößern, dass sie für ein größeres, körperlich anwesendes Publikum gleichzeitig sichtbar und nachvollziehbar werden. Seit Beginn des 19.Jahrhunderts, zu dem das Projektionsprinzip erstmalig in so genannten Filmtheatern zum Einsatz kam, entwickelte sich die Technologie weiter und spezialisierte Geräte eroberten den Markt. Ein Gerät, das den Bildschirm von einem PC oder Laptop an die Wand oder auf eine Leinwand projiziert, wird als Beamer bezeichnet. Dabei ist zu beachten, dass die Bildquelle und der Projektionsapparat nicht in einem Gerät vereint, sondern wie heute üblich getrennt existieren.

Das **Management** (lat. manum agere "an der Hand führen") im funktionalen Sinn, bezeichnet eine Professionalisierung oder eine zielgerichteter Steuerung und bedingt ökonomisches Handeln. Informationen und Kommunikation sind essentielle Grundlagen für ein erfolgreiches Management [W03].

Bei dem Begriff **Projektor Management** verschmelzen beide vorgenannten Begriffe. Basierend auf einer effektiven Bereitstellung von Informationen und einer verbesserten Kommunikation ergeben sich neue Wege für einen effizienteren Umgang mit Projektionsgeräten. Thema hier ist Sinn und Notwendigkeit dieser Wege aufzuzeigen und praktische Lösungsansätze darzulegen.

1.2 Projektorpraxis

1.2.1 Installationsszenarien

Wachsende Anwendungsgebiete sowie die Popularität von Projektoren steigerten die Produktionsstückzahlen und ließen die Preise fallen. Folglich ist es Unternehmen, Schulen und Universitäten möglich, ihre Räume mit einer Vielzahl meist unterschiedlicher Beamer auszustatten und selbst im Privat- und Heimkinobereich finden Projektoren nun ihren Einsatz. Sie unterstützen Vorträge und Konferenzen, Trainingsmaßnahmen sowie Großveranstaltungen und Messen. Dabei ist festzustellen, dass für die meisten Einsätze ausschließlich eine Bildquelle und ein Projektionsgerät benötigt werden. Ausnahmen bilden hier synchronisierte Kombinationen, die das Gesamtbild auf mehrere Beamer übertragen oder sogar splitten.

Bezüglich des Managements von Projektoren ergeben sich jedoch weitere Installationsszenarien, die hier eine fundamentale Rolle einnehmen. Das Szenario bzw. der Anwendungsmodus gibt nicht allein Auskunft über die Anzahl der Geräte, die einer Verantwortlichkeit (nachfolgend Autorität genannt) unterliegen, sondern beschreibt zusätzlich wie viele Geräte parallel einem Management zugänglich gemacht werden können. Folgende zwei Modi sind differenzierbar:

- Mit **Single Mode** werden Szenarien bezeichnet, bei denen genau ein Beamer einer Autorität zugeordnet werden kann und dieser einem Management zugänglich ist. Beispiele findet man häufig im privaten Einsatzbereich.

- **Distributed Mode** beschreibt den Fall, dass mehrere Geräte einer Autorität unterliegen und gleichzeitig einem Management zugänglich gemacht werden.

Multiple Installationen ohne multiplen Management Zugang werden somit zwangsläufig dem Single Modus zugeordnet. Installationen im Distributed Mode findet man unter anderem an größeren Lehreinrichtungen und Unternehmen.

1.2.2 Wartung

Regelmäßige Wartung ist für jeden Beamer ein Muss. Staub und Dreck rütteln an der Funktionalität und können zu vorzeitigen Ausfall führen. Zur Werterhaltung sollte ein dauerhaft genutztes Gerät üblicherweise aller 18 Monate gereinigt werden. Lampen- und Filterwechsel sowie - bei Bedarf - eine Innenreinigung sind die Obliegenheiten eines Services. Folgen sporadischer oder unterlassener Pflege sind Hitzeschäden an Panels, kürzere Lebensdauer der Lampen und verminderte Bildqualität. Jedoch kann unnötige Wartung zu nicht notwendigen Geldausgaben führen.

Ziel des Projektor-Managements ist es daher, die Wartungsintervalle flexibel zu gestalten und somit Kosten zu sparen. Grundlage für eine zeitlich und umfänglich angepasste Pflege sind die nachstehend aufgelisteten Informationen

- **Lamp Time:** Gibt die aktuelle Brenndauer der Lampe an. Jede Leuchte hat eine bestimmte Lebenszeit und die Brenndauer sollte die vorgegebene Lebenszeit nicht überschreiten. Für eine effiziente Nutzung sollte die Lampe erst bei erreichen der maximalen Brenndauer gewechselt werden.

- **Remaining Lamp Time:** Ist ähnlich wie die vorherige Kenngröße, jedoch wird hier die aktuelle Brenndauer von der Maximalen subtrahiert und es ergibt sich die noch zu verbleibende Nutzungszeit der Lampe.

- **Operating Time:** Entspricht den Betriebsstunden des Projektors und ist eine Anhaltspunkt für Wartungen wie Filterwechsel oder Reinigung. Im Zusammenhang mit dem Alter des Projektors ist es eine Kenngröße für die Ausnutzung des Gerätes.

- **Temperature:** Zeigt den Zustand der Innentemperatur des Beamers. Prinzipiell lassen sich durch überhöhte Temperaturen, Fehler und Probleme, wie Filterverschmutzung oder Lüfterausfall, schon im laufenden Betrieb erkennen.

- **Status:** Je nach Gerätetyp, lassen sich weitere und detailliertere Fehler ermitteln. Beispielsweise können Probleme mit der Stromversorgung, mit der Linse, dem Lüfter usw. erkannt werden.

1.2.3 Steuerung und Überwachung

Weitere sinnvolle Aspekte eines Projektor Managements sind einerseits die klassische Fernsteuerung und anderseits die Überwachung von Nutzungsdauer/-rechten. Daher erlaubt es die für eine Wartung nötige Management-Verbindung, nicht nur die schon angeführten Informationen über Lebensdauer der verschiedenen Projektorkomponenten abzufragen, sondern sie gestattet auch weitere folgende Möglichkeiten der Nutzung.

Die so genannte **Remote Control** ermöglicht die Veränderung einer Vielzahl an Projektoreinstellungen ohne direkt mit dem Gerät in Kontakt treten zu müssen. Vorstellbar ist auch eine gleichzeitige Steuerung von mehreren Geräten, die einer Gruppierung von Projektoren entspräche. Infolgedessen verhalten sich gruppierte Projektoren für die Steuerung als ein Gerät und verändern miteinander die gewünschten Werte.

Projector Monitoring ermöglicht einen Einblick in den aktuellen Nutzungsstatus. Durch eine eindeutige Identifizierung der fest installierten Geräte ergibt sich die Aussicht, Diebstahl frühzeitig festzustellen, Nutzungsdauer zu überwachen sowie in Kombination mit Remote Control Projektoren zu sperren.

1.3 Schnittstellen und Protokolle

1.3.1 Welche Schnittstellen existieren?

Neben den bekannten Schnittstellen zur Bildübertragung, wie etwa RGB oder S-Video, stellen viele Projektoren eine oder mehrere Management-Schnittstellen zur Verfügung:

- **Serielles RS-232 Interface** als am häufigsten verwendete Anschlussart, bietet den einfachsten, aber auch unterschiedlichsten Zugang. Verschiedene Stecker komplizieren den Anschluss unnötig, spielen jedoch durch zwangsläufigen Zubehörerwerb den Herstellern Gelder in die Kassen. Auf Grund sehr beschränkter Anzahl von seriellen Anschlüssen, ist ein ökonomisches Projektor-Management, basierend auf einer seriellen Schnittstelle, nur schwer vorstellbar.

- **USB Anschluss** ist eher eine Seltenheit, sollte der Vollständigkeit halber aber nicht vergessen werden. Beide Anschlussarten führen ohne eine weitere Änderung zu dem Problem, dass jeweils nur ein Projektor an den Management PC auf eine begrenzte Distanz angeschlossen werden kann.

- **RJ-45 Ethernet Zugang** löst dieses Problem. Multiple Beamer in einem Netzwerk lassen sich anhand ihrer IP-Adresse identifizieren und ermöglichen eine gleichzeitige Verbindung zu einer zentralen Steuerung. Die ausgereifte und zuverlässige Netzwerktechnologie sowie die Standardisierung von Anschlüssen und Steckern rückt diese Anschlussart in den **Mittelpunkt der Betrachtungen** und zeichnet sich als zukünftig am stärksten vertretene Schnittstelle ab.

Zusammenfassend ist festzustellen, dass die Ethernet Schnittstelle die wohl vorteilhafteste Anschlusslösung darstellt. Verstärkt durch das vermehrte Aufkommen von kommerziellen COM-LAN Adaptern [B02] und diversen professionellen Selbstbauanleitungen für selbige, ist das primäre Ziel die Nutzung von LAN-Komponenten zur Kommunikation. Eine Ansteuerung der noch so häufig vertretenen RS-232 Schnittstellen über ein Netzwerk mittels Umsetzer würde, bei entsprechender Konfiguration der Projektoren eine weltweite Steuerung über das Internet erlauben. Das gilt natürlich nicht nur für die mit Adapter betriebenen Geräte, sondern für alle in ein Netzwerk integrierbaren Anschlussvarianten

Es sei nochmals darauf hingewiesen, dass der Trend für ein multiples Management eindeutig zu Ethernet Anschluss Lösungen geht und jegliche Umsetzertechnologie nur eine Brücke zur Integration vorhandener Geräte darstellen sollte.

1.3.2 Protokolle ohne einheitliche Standards

Ein Management sollte zweifelsfrei verschiedene Beamertypen betrachten, auswerten und steuern können. Jedoch setzt jeder Hersteller auf sein eigenes proprietäres Kommunikationsprotokoll. Somit sind Normalisierungen bezüglich der Protokolle der wohl schwierigste, aber auch wichtigste Teil dieser Arbeit. Nicht nur Befehlssätze oder Attribute sind unterschiedlich, es werden ferner auch verschiedene Protokollphilosophien und differente Übertragungsformate (**ASCII** bzw. **Hexadezimal**) eingesetzt. Folgende Tabelle soll anhand dreier Beispiele eine erste kurze Darstellung der Vielfalt der Kommunikationsstandards geben und gleichzeitig aufzeigen, worin die größten Unterschiede liegen.

Projector	**Digital Projection**	**Nec LT245 Serie**	**Epson Standard Code**
Interface	RS-232	Ethernet	RS-232
Command Format	Binär	ASCII	Binär
Command Type	Get / Set / Increment / Decrement / Execute	Get / Set	Set / Get / Initiliaze / Response
Command Syntax	1Byte_Operation_Type +2Byte_Operation +2Byte_Validation +4Byte_Target +4Byte_Value +4Byte_Lower_Limit +4Byte_Upper_Limit +4Byte_Increment	Command [Parameter 1] [Parameter 2]	"ESC" +1Byte_Size +1Byte_Attribute +1Byte_Classification +1Byte_Function +nBytes Data +"CS"
Return Codes	PAK = Packet Acknowledge	"ok" | Command: Value	ACK
Request Format	7 Byte Header	Command + Line Feed	Command | ACK
Response Format	7 Byte Header	Result + Line Feed	ACK + Result | ACK

Hauptschwierigkeit stellen eindeutig die unterschiedlichen Befehl-Syntax/Befehlsformate zusammen mit den verschiedenen Befehlstypen dar. Hier sind Standardisierungen gefragt. Der Fokus

gilt Lösungsansätzen zur **Abbildung der proprietären Protokolle** in ein einheitliches Format. Letzte Hürde sind die, beispielsweise bei *Epson Standard Code* eingesetzten **Befehlsfolgen**, welche abhängig von der Response eine zusätzliche Antwort schicken. Das heißt, eine Kommunikation besteht nicht nur aus Anfrage und Antwort, sondern kann abhängig von Erfolg oder Misserfolg weitere Antworten beinhalten.

1.3.3 Funktionsumfang

So vielfältig wie die Protokolle, sind auch die Funktionsumfänge der Geräte. Neben Ein- und Ausschalten, Bildquelle wählen, Bildeinstellung und Bildkorrektur werden noch unzählige andere, meist typspezifische, Funktionen unterstützt. Hier ist es jedoch relativ simpel einen gemeinsamen Nenner zu finden und primäre Funktionen, deren Eigenschaften durchaus wohl bekannt sind, als Basic Functions und alle weiteren, noch Unbekannten, als Extended Functions zu bezeichnen.

- **Basic Functions** repräsentieren die primäre Projektorfunktionalität, die alle Grundeinstellungen realisiert und von einem Großteil aller existierenden Geräte angeboten wird. Die Grundfunktionen können nochmals, ihrer Hauptaufgabe entsprechend, in weitere Funktionsgruppen unterteilt werden: **General, Screen, Audio, Detail**

- **Extended Functions** sind hauptsächliche typabhängige Funktionen, die auf Grund ihrer Spezialisierung im Projektor Management eher unbedeutend sind.

1.4 Existierende Lösungen

1.4.1 Überblick

Projektoren computergestützt steuern ist kein neues Thema. Die Hersteller lassen ihre Softwareentwickler kleine kostenlose Utilities in Form von Toolbars, Kontrollkästchen und integrierten Webservern entwickeln. Dabei geht auch hier der Trend eindeutig zum in den Beamer integrierten Webserver. Er ermöglicht die einfache Fernsteuerung des entsprechenden Gerätes über den Browser des Client PCs. Jedoch haben alle Steuerungslösungen eines gemeinsam: Sie sind herstellerabhängig

Es steht fest, dass ein Management nicht nur aus Fernkontrolle eines einzigen Beamers besteht, sondern vielmehr, wie in Punkt 1.1 erläutert, eine Symbiose aus Überwachen und Steuern mehrerer Entities ist. Diesbezüglich bietet der Markt eine zwar sehr beschränkte Anzahl von Softwareprodukten, diese hingegen besitzen aber umfangreiche Leistungen. Stichworte hierzu sind Monitoring, Diagnosing, Controlling und Scheduling als Hauptschwerpunkte. Leider existiert auch hier dasselbe Problem wie im vorangegangen Abschnitt: Alle Programme unterstützen nur eine beschränkte Anzahl von Projektoren und sind folglich genauso herstellerabhängig.

1.4.2 Beispiele für Management Software

Name	**HITACHI Projector Managment Software**
Schnittstellen	Ethernet RJ-45, Seriell RS-232
Leistungen	Steuerung, Scheduling, Monitoring und Diagnose für einzelne Projektoren oder Gruppen, Email Information bei Routine Wartungen
Projektoren	nur Hitachi Projektoren, CP und ED Serie
Betriebssystem	Windows
Kosten	kostenlos

Name:	**SONY PJNet!™**
Schnittstellen:	Ethernet RJ-45
Leistungen:	Steuerung, Diagnose, Scheduling, Gruppenoperationen, Email und SMS Benachrichtigung
Projektoren:	SuperSmart™ and SuperBright™ network projectors
Betriebssystem:	Windows
Kosten:	500 US-Dollar

Name	**InFocus ProjectorNet 2.0**
Schnittstellen	Ethernet RJ-45
Leistungen	Steuerung, Monitoring, SNMP, Ansteuerung von COM-LAN Adaptern mit seriellen Kommandos, 1-150 Projektoren, Email Benachitigung bei Projektor Events
Projektoren	InFocus Corporation Projektoren (InFocus®, Proxima®, ASK Projektoren)
Betriebssystem	Windows
Kosten	ab 999 US-Dollar

1.4.3 InFocus ProjectorNet® 2.0

Die Management-Software ***InFocus ProjectorNet***®, die in diesem Abschnitt ausführlicher betrachtet werden soll**,** ist eine auf dem Client/Server-Modell basierende Netzwerk Lösung, welche es den Administratoren erlaubt, mehrere Projektoren über ein Netzwerk zu steuern. ProjectorNet®, laut dem Hersteller entwickelt für mittelgroße bis große Firmen, Universitäten und andere Organisationen, bieten seinem Käufer die Möglichkeit Beamer so einfach wie andere Netzresourcen zu managen. Weitere Gründe zur Anschaffung sind Zuverlässigkeitsverbesserungen, Sicherheitserhöhung und eine Reduzierung der absoluten Fixkosten.

Im Detail ist Projectornet® eine Management Software, welche eine Fernkontrolle von InFocus Corparation (InFocus®, Proxima®, ASK) Projektoren über Standard TCP/IP Netwerke ermöglicht. Sie fügt sich, wenn man den werbestrategischen Beschreibungen Glauben schenkt, ideal in Firmen ein, welche ihre Projektoren verteilt über mehrere Räume, Etagen und Gebäude installiert haben. Grundlegend ist die Software aus folgenden drei **Hauptkomponenten** aufgebaut.

- NT-Service ***Tunnel-Vision*****:** Kommunikation mit dem Projektionsgerät
- SNMP Daemon ***Spin-SNMP*****:** Hintergrunddienst für SNMP Anfragen
- Webserver ***GoAhead*****:** Stellt die Projektorinformationen per HTTP zur Verfügung

Tunnel-Vision ist ein Betriebssystem Service, welcher im Hintergrund die proprietäre Kommunikation mit den Beamern übernimmt. Über Tunnel-Vision beziehen der SNMP Daemon **SpinSNMP** und der Webserver **GoAhead** ihre Daten. Beide leiten die an sie gerichteten Anfragen an Tunnel-Vision in einer unbekannten Form weiter (im Bild 1.1 gekennzeichnet mit „Data 1“ und „Data 2“), um die somit in Erfahrung gebrachten Informationen wiederum an die anfragenden Clients zurückzugeben. Mögliche Clients für den SNMP Daemon sind Netzwerk Management Programme und im Fall des Webservers können die Clients einerseits Pocket-PCs sein und andererseits die Windows GUI von ProjectorNet®. Alles in allem eine gut funktionierende und sehr komfortable Software, die aber wie alle anderen Lösungen durch die beschränkte Anzahl von unterstützten Projektoren nicht universell einsetzbar ist.

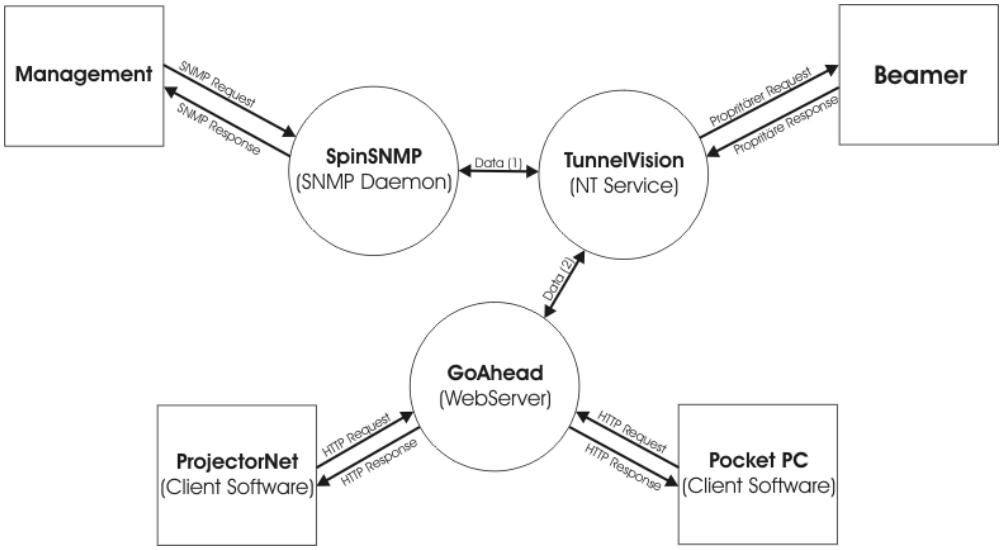


Bild 1.1: Schematischer Aufbau der Software InFocus ProjectorNet® 2.0

1.4.4 Resümee des Marktes

Softwarelösungen existieren, jedoch nur vereinzelt. Darüber hinaus neigen die Projektorhersteller dazu, ihre eigenen Ansätze zu verfolgen und keine globalen Aspekte in die Softwareentwicklung einzubeziehen. Eine herstellerseitige Standardisierung zeichnet sich im Projektorbereich nicht ab. Die vorhandenen Tools, sei es zur Steuerung oder sei es ein Management Programm, sind auf Grund ihrer Geräteabhängigkeit nicht leicht in existierende Systeme zu integrieren. Kosten und Nutzenfaktoren stehen daher in keinem Verhältnis. Auch wenn einige Produkte kostenlos erhältlich sind, ziehen sie, für ein vollständiges und effektives Management, die Anschaffung neuer Projektoren nach sich. Daher lohnt sich die Anschaffung und Inbetriebnahme der durchaus guten Software lediglich in Verbindung mit neuen Hardwarekomponenten, vorausgesetzt man betreibt ein Windows Betriebsystem.

1.5 Fazit

Aspekte des Projektor Managements sind vielfältig. Fakt ist, dass durchaus ein Verwaltungsbedarf für Geräte in Unternehmen, in größeren Lehreinrichtungen und selbst im Privatbereich existiert. Deutlich ist aber auch geworden, dass die Steuerungsprinzipien der Projektoren unterschiedlicher nicht sein können und obendrein die Hersteller keinen Grund für generelle Lösungen sehen. Im Gegenteil, es wird versucht, mit wenigen proprietären Management Programmen eine Kundenbindung zu erzielen. Als Gegenpol zu diesem Trend ist es dringlich, eine plattform- und projektorunabhängige **Management Software** zu entwerfen sowie ein **Open Source** Projekt zu initiieren, welches langfristig hilft, ein vollständiges Programm umzusetzen. Das Ziel dabei ist, nicht in einer ökonomische Untergrabung der erhältlichen Programme, sondern eher als Ansporn zu Verbesserungen hinsichtlich Kosten-Nutzen Faktoren, Flexibilität und Portabilität zu sehen.

2 Entwurf der Software

2.1 Informale Spezifikation

2.1.1 Zielstellung

Aufgabe der zu entwerfenden Software ist es, eine einfache Fernwartung bzw. -diagnose für eine bestimmte Anzahl von verteilten Projektoren zu ermöglichen. Die Software als zentrale Einheit, mit den Projektorgeräten über ein Netzwerk verbunden, gestattet dem Nutzer, Einstellungen und Konfigurationen durchzuführen, ohne unmittelbar mit dem gewünschten Gerät in Kontakt zu treten. Weiterhin bietet eine solche Lösung den Vorteil, mehrere Geräte zu verwalten sowie Gruppenoperationen durchzuführen. Computergestützte universelle „Fernbedienung“ würde man umgangssprachlich als Handhabung charakterisieren. Natürlich ist die effektive Konfiguration und Steuerung der Projektoren nur ein Aspekt bei der Entwicklung. Für einen überzeugenden Nutzwert ist der Fokus darüber hinaus auf das Management der Geräte gerichtet. Was das heißt und wie das im Endeffekt umgesetzt werden kann, ist Inhalt dieses Kapitels.

2.1.2 Programmiersprache

Die Implementierung der Software wird in der **objektorientierten Skriptsprache Python** geschehen. Python verbindet Vorteile traditioneller OO-Sprachen mit Merkmalen von Skriptsprachen und ermöglicht somit einen klassenbasierten Systementwurf. Darüber hinaus bietet Python hohe Portabilität und plattformübergreifender Einsetzbarkeit. Neben diesen zwei Hauptschwerpunkten ist der hervorragende Bibliothekssupport von Netzwerktechnologien, GUI Werkzeugen und die Mächtigkeit der eingebauten Objekttypen zweckmäßig. All dies und viele weitere Gründe lassen Python zu einer idealen Lösung für die hier auftretenden Aufgaben werden.

2.1.3 Hauptaufgaben

Zu den meist genutzten Funktionen zählen das Abfragen servicerelevanter Daten über das Netzwerk, wie z.B. Lampenbrenndauer, das Ein- und Ausschalten der Beamer und das Einstellen von Bildparametern. Anzumerken ist dabei, dass eine Einstellung von visuellen Optionen sicher nur soweit sinnvoll erscheint, solange diese keinen Sichtkontakt zum Beamer erfordert. Hinsichtlich der physisch gegeben Beschränkungen, fokussiert die Software primär auf Service und Wartung einzelner Projektoren und Projektorgruppen. Dabei kann mit Hilfe genauer Statusdaten die Koordination von Aufgaben, wie zum Beispiel Staubfilter reinigen und Lampen wechseln, erleichtert werden. Festlegung und Überwachung von Nutzungsdauer und Ruhephasen der Geräte werden erleichtert und gleichzeitig wird Missbrauch vorgebeugt.

2.1.4 Interaktion mit dem Programm

Nachdem die vorhandenen Projektoren konfiguriert sind, kann der User über ein **grafisches Nutzerinterface (GUI)** alle angebotenen Einstellungen vornehmen. Zur Erstellung eines plattformübergreifenden Users Interfaces hilft die *Tkinter* Bibliothek der Programmiersprache Python. Sie ermöglicht die Lauffähigkeit in verschiedenen Betriebssystemumgebungen sowie ein einheitliches

Erscheinungsbild der Oberfläche. Dazu können vordefinierte Fenster, Menüs und weitere grafische Elemente genutzt werden. Für Spezialisten ist darüber hinaus eine **Terminalversion** eine interessante Umsetzung. Sie soll ein ähnliches Projektor-Management ermöglichen, wobei hier die Einfachheit und schnelle Erweiterbarkeit sowie die Transparenz der Funktionalität im Vordergrund steht.

2.2 Auftretende Use Cases

2.2.1 Projektor

Der Projektor Use Case (Bild 2.1) verdeutlicht alle Handhabungen, die mit einem Projektor durchgeführt werden können. Um einen neuen Projektor hinzuzufügen ist es unentbehrlich, den entsprechenden Projektortyp auszuwählen, eine Verbindung zu spezifizieren und bei Bedarf das Gerät einer Gruppe zuzuordnen. Um spätere Änderungen vornehmen zu können, ist es möglich einen bereits angelegten Projektor zu bearbeiten. Dabei kann die Verbindung neu konfiguriert oder die Projektorgruppe neu zugewiesen werden. Einfacher verhält sich der Anwendungsfall Projektor löschen, dieser entfernt nach vorheriger Bestätigung den Projektor aus dem System. Hinter Projektor aktualisieren, was normalerweise regelmäßig im Hintergrund geschieht, verbirgt sich eine erneute Abfrage aller wichtigen Projektordaten.

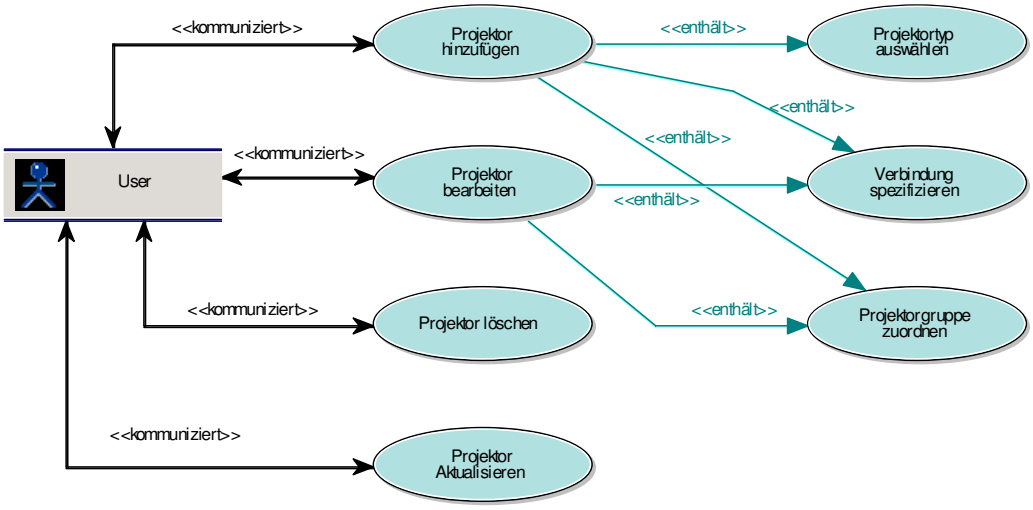


Bild 2.1: Projektor Use Case

2.2.2 Gruppierung

Die Gruppierung von Projektoren (Bild 2.2) setzt voraus, dass Gruppen angelegt, bearbeitet und gelöscht werden können. Eine Projektorgruppe ist nichts weiter als ein Container für Projektoren und repräsentiert diese nach Außen. Da ein Gruppencontainer nur eine Hülle ist, kann er einfach angelegt und bearbeitet werden. Für das Löschen muss gewährleistet werden, dass kein Projektor mehr dieser Gruppe zugeordnet ist.

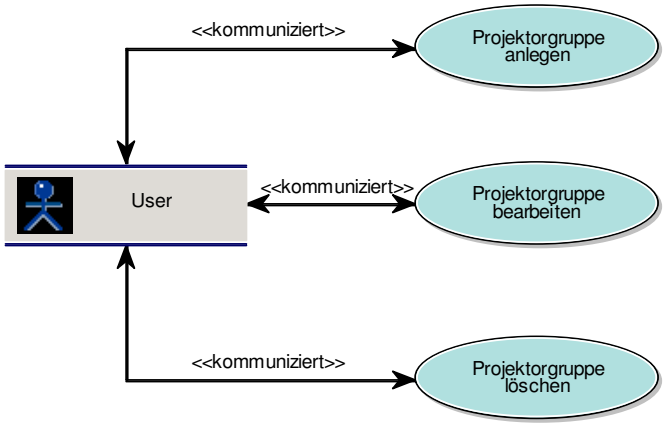


Bild 2.2 Gruppierung Use Case

2.2.3 Status abfragen

Die Voraussetzung für ein erfolgreiches Management sind detailierte und gleichzeitig schnelle Informationen. Dazu dient der Projektorstatus. Wenn dieser ermittelt werden soll, was gewissermaßen sofort nach dem Anlegen/Hinzufügen des Beamers ins System sowie durch das automatische Aktualisieren geschieht, werden, die unter Punkt 1.2.2 aufgeführten, Eigenschaften abgefragt. Diese Daten geben dem Nutzer einen präzisen Überblick und bilden die Grundlage für weiteres Handeln.

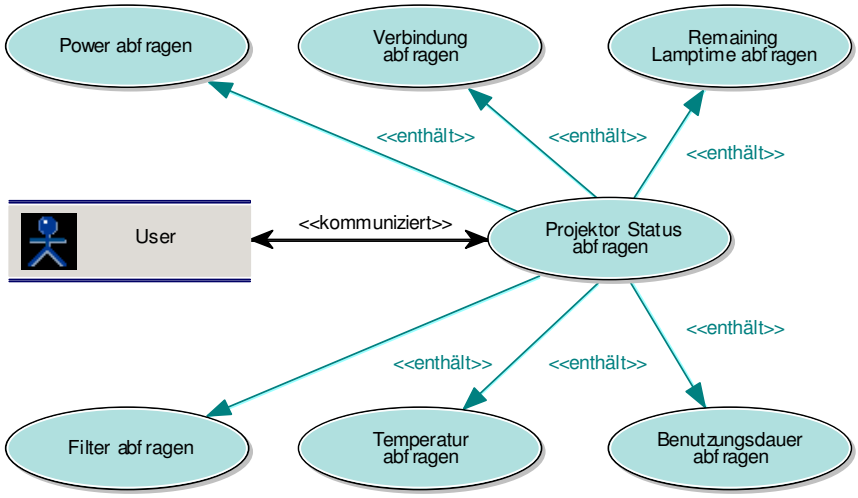


Bild 2.3: Status abfragen Use Case

2.2.4 Steuerung

Da sich Projektoren und Projektorgruppen hinsichtlich ihrer Steuerung ähnlich verhalten, ist die Steuerung bis auf die Anzahl der grafische Steuerelemente (Steuerwidgets) für beide gleich (Bild 2.4). Dazu in einem später Abschnitt noch detailierter. Bei diesen Use Cases sei noch anzumerken, dass bevor eine Steuerung möglich ist, der gewünschte Projektor bzw. die Gruppe auszuwählen ist.

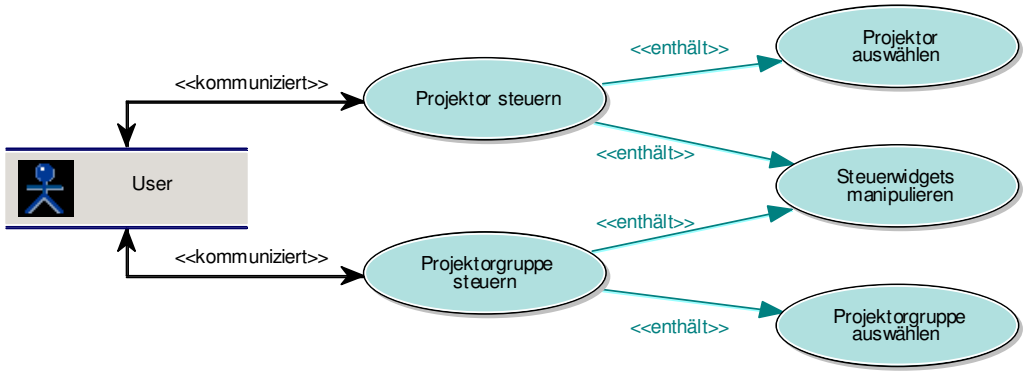


Bild 243: Steuerung der Projektoren Use Case

2.2.5 Einstellungen und Projektortypen

Der Nutzer muss in der Lage sein, mit Hilfe einer XML Treiber Datei, welche alle nötigen Infomationen über den neuen Projektortyp beinhaltet und später detailiert vorgestellt wird, neue Projektortypen anzulegen und diese auch bei Bedarf wieder zu löschen. Darüber hinaus kann er Systemeinstellungen ändern und nach Wunsch konfigurieren.

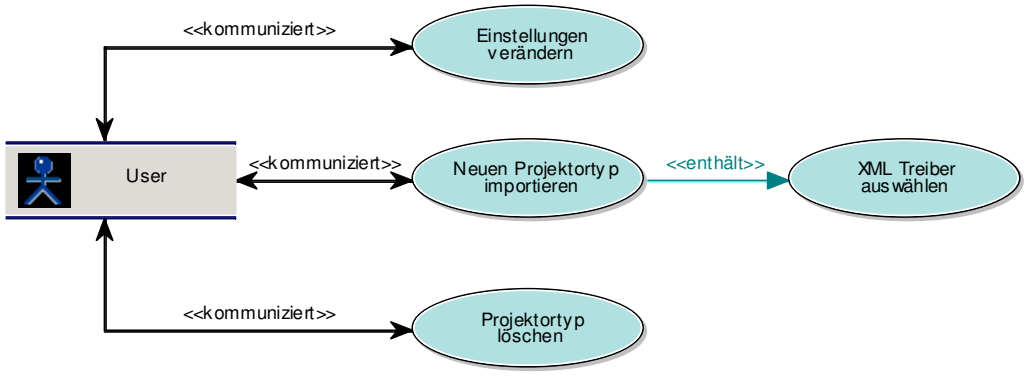


Bild 2.5: Einstellungen und Projektortypen Use Case

2.3 Modellierung der Projektoren

2.3.1 Einheitliche Abbildung

Schwerpunkt dieser Arbeit und gleichzeitig die größte Herausforderung bei der Projektormodellierung ist die Darstellung der unterschiedlichen Kommunikationsprotokolle. Dabei ist das **Hauptziel eine einheitliche Abbildung**, die es ermöglicht, alle von den Projektorherstellern eingesetzten proprietären Protokolle aufzunehmen. Für eine Programmierung, die hinsichtlich der angeschlossenen Geräte und deren Standards variabel sein soll, ist dies notwendig.

2.3.2 Funktionen und Funktionsklassen

Jeder Projektor hat eine bestimmte Funktionalität. Dazu ist einführend anzumerken, dass prinzipiell zwischen Funktionen und Kommandos bzw. Befehlen unterschieden werden muss. Während Funktionen wie z.B. Kontrast, Helligkeit oder Power die Fähigkeiten des Projektors beschreiben, bewirken Kommandos die Steuerung der Funktionen. Für eine computergestützte Verarbeitung ist es zusätzlich notwendig, die Arbeitsweise der verschiedenen Beamerfunktionen zu repräsentieren und somit den Kommandos ihre Semantik zu geben. Dabei ist es denkbar, die Handhabung der Kommandos in folgende Funktionstypen zu untergliedern:

- Funktionen der **Adjust Klasse** sind über Inkrementierung oder Dekrementierung, schrittweise Erhöhen bzw. Verringern, ihres Zustandes einstellbar. Die Befehlsstruktur sieht somit auch nur zwei verschiedene Parameter (+/-) vor. Diesem Typ können beispielsweise Helligkeit und Kontrast zugeordnet werden. Der Wertebereich von Adjust Funktionen ist durch das Minimum und das Maximum sowie durch eine optionale Schrittweite gekennzeichnet.

- Funktionen der **Value Klasse** besitzen den gleichen Wertebereich wie Adjust Funktion. Das heißt Wertemenge mit Minimum, Maximum und Schrittweite. Der Unterschied ist jedoch, dass mit Value Funktionen der Wert direkt gesetzt werden kann.

- Funktionen der **State Klasse** besitzen als Wertebereich mindestens zwei einstellbare Zustände. Zu diesen Funktionen zählen unter anderem Power, Source (Input) und Sprache. Diese Zustände sind direkt wählbar.

- Die **Execute Klasse**, als der simpelste Funktionstyp, beschreibt Fähigkeiten, welche keine Parameter benötigen sowie keine internen Zustände besitzen. Reset, Factory Reset und ähnliche einfache Funktionen sind diesem Typ zuzuordnen.

- Funktionen der **Query Klasse** zeichnen sich in ihrer Anwendung aus, indem sie keinerlei Übergabeparameter benötigen. Wie der Name schon sagt, senden Funktionen des Query Typs, wie zum Beispiel die Lampenbrenndauer, eine Anfrage und geben infolgedessen den Wert des angefragten Zustands zurück.

2.3.3 Funktionsumfang und Darstellung

Funktionen beschreiben die Fähigkeiten eines Projektors. Um die **Basic Functions** während der Softwareimplementierung zu identifizieren, ist es nötig, die Funktionen intern darzustellen. Dazu reicht es aus, eine Abkürzung für jede Funktion festzulegen. Grundsätzlich sind die Abkürzungen die standardisierten Beschreibungen, mit denen es möglich ist, eine Projektorfunktion beispielsweise direkt über die allgemeine Programmierschnittstelle API oder ein Terminalprogramm anzusprechen. Darüber hinaus wird dieser Beschreibungsstandard intern zur Abbildung der Funktionstypen genutzt. Folgende Zuordnung zeigt den unterstützten Funktionsumfang (Basic Functions) und die zugehörigen vereinheitlichten Abkürzungen. Zusätzlich weist die Abbildung eine mögliche Gruppierung und die ideale, aber nicht zwingende Klassifizierung der Fähigkeiten eines Projektors aus.

Funktion	Intern	Klasse	Gruppe	Beschreibung
Power	PWR	State	General	Ein- und Ausschalten
Reset	RST	Execute	General	Einstellungen zurücksetzen
Rear	REA	State	General	Rückprojektion
Ceiling	CEI	State	General	180° Bilddrehung
Source	SRC	State	General	Bildquelle wählen.
Source Search	SRS	Execute	General	Input Suche
Contrast	CON	Value	Screen	Kontrast
Brightness	BRT	Value	Screen	Helligkeit
Color Hue	HUE	Value	Screen	Rot und Grün Balance
Color Temperature	CLT	Value	Screen	Farbtemperatur
Sharpness	SHP	Value	Screen	Bildschärfe einstellen.
Keystone	KEY	Value	Screen	Trapezkorrektur einstellen.
Picture Mute	PMT	State	Screen	Bild abschalten.
Width	WID	Value	Screen	Bildbreite
Horizontal Position	HPO	Value	Screen	Bildposition horizontal
Vertical Position	VPO	Value	Screen	Bildposition vertikal
Magnify	MAG	Value	Screen	Vergrößerung einstellen.
Freeze	FRE	State	Screen	Standbild
Format	FOR	State	Screen	Seitenverhältnis auf 16:9
Volume	VOL	Value	Audio	Lautstärke einstellen.
Mute	MUT	State	Audio	Stummschalten.
Treble	TRE	Value	Audio	Höhen
Bass	BAS	Value	Audio	Tiefen
Loudness	LOU	State	Audio	Höhen u. Tiefenverbesserung
On Screen Display	OSD	State	Detail	On Screen Display
Language	LAN	State	Detail	Sprache wählen
Lamp Time	LMT	Query	Detail	Lampenbrenndauer
Remaining Lamp Time	RLT	Query	Detail	verbleibende Lampenbrenndauer
Unit On Time	UOT	Query	Detail	Betriebsstunden
Temperature	TEM	Query	Detail	Interne Temperatur
Status	STA	Query	Detail	Statusabfrage

2.3.4 Request und Response

Nachdem verdeutlicht wurde, dass Kommandos die Funktionen der Beamer steuern, ist ein weiterer Aspekt hinsichtlich des Datenaustauschs zwischen Software und Projektionsgerät zu betrachten. Da die Software auf einem Client/Server Modell basiert, werden die Daten, die zum Gerät hin gesendet werden, als Anfrage (engl. Request), die zurückkommenden als Antwort (engl. Response) bezeichnet.

Soll nun ein Kommando zum Projektor gesendet werden, wird der Befehl in das dem Projektor zugehörige Request Format gebracht. Dazu werden Steuerzeichen am Anfang (Header) und oder am Ende (Tail) hinzugefügt. Um die Antwort auswerten zu können, ist es umgekehrt natürlich auch erforderlich, die Response von den zusätzlichen Informationen zu befreien.

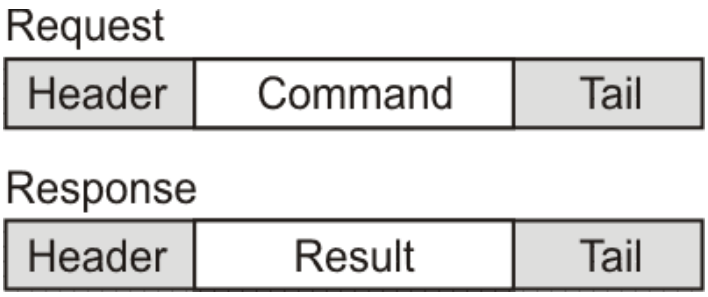


Bild 2.6 Aufbau Request und Response

2.3.5 Wertebereiche

Ein weiteres Problem von Projektoren ist der Wertebereich auf dem ein Steuerkommando operiert. Die hier festgelegten zwei verschiedenen Möglichkeiten einen Bereich auszudrücken, unterscheidet man ähnlich wie die Funktionsklassen. Die Parameter der Kommandos müssen, wenn vorhanden, einen der beiden folgenden Standards entsprechen.

- Der **Value Wertebereich** ist durch ein Minimum und ein Maximum sowie eine optionale Schrittweite gekennzeichnet und liegt im Bereich der natürlichen Zahlen. Die selbstverständliche Ordnungsrelation der einzelnen Werte bestimmt die Reihenfolge in welche diese zueinander stehen. Dies ist unter anderem eine Voraussetzung, dass eine Zustandsänderung mit den Operatoren +/- überhaupt durchführbar ist. Anzumerken sei noch, dass die Funktionsklassen Adjust und Value diesen Wertebereich implizieren.

- Der **State Wertebereich** besteht aus einer endlichen Anzahl fest definierter Zustände, die in keiner Relation zueinander stehen. Zusätzlich können sie direkt durch ein Kommando referenziert werden. Prinzipiell ist es möglich, einen Value Wertebereich in einen State Wertbereich umzuformen, indem alle möglichen Werte als State Zustände abgebildet werden. Die Ordnungsrelation geht dabei aber verloren. Auch hier wird dieser Wertebereich bei Verwendung des Funktionstyps State impliziert.

2.3.6 Verschiedene Ergebnisse

Nun, nachdem alle Probleme bezüglich Anfragen, Kommandos und Funktionen eines Beamers auf einen Nenner gebracht sind, ist das Augenmerk auf die Antworten, die der Projektor an die Software zurücksendet, zu richten. Wie bereits angeführt, basiert die Kommunikation auf dem Client/Server Modell, wobei in diesem Fall die Software den Client und der Beamer den Server darstellt. Die letzten Abschnitte befassen sich folglich mit allen notwendigen Vorkehrungen, einen gültigen Request an den Server zu schicken.

Wie sieht es mit der Response aus? Prinzipiell ähnlich wie die Anfrage kann man die Antworten betrachten. Um das Ergebnis (Result) auswerten zu können, muss, wie im Abschnitt „Request und Response“ aufgezeigt, die Antwort von Header und Tail befreit werden. Die so gewonnen Daten werden wiederum auch hier in die folgenden Klassen unterteilt:

- **Confirm Results** sind eine positive Bestätigung und können als erfolgreiche Ausführung eines Kommandos gedeutet werden. Sie enthalten keine weiteren Informationen.

- **Value Results** liefern den Wert, der zur angefragten Funktion korrespondiert. Dieses Ergebnis ist, genauso wie ein Confirm Result, eine positive Bestätigung und kann als erfolgreiche Ausführung des Kommandos gedeutet werden.

- **State Results** liefern einen definierten Zustand aus einer Zustandmenge. Meistens sind diese Zustände kodiert und müssen noch, in eine dem Nutzer verständliche Form, umgesetzt werden. Ein Beispiel hierfür wäre Temperatur, deren Zustände „Überhitzt“ bzw. „Normal“ seien könnten. Auch hier ist das Ergebnis positive Bestätigung.

- **Error Results** zeigen mit optionaler Zustandsliste Probleme bei der Ausführung des Kommandos an. Error Results sind im Gegensatz zu den anderen Ergebnisklassen funktionsunabhängig und können somit bei jeder Anfrage auftreten.

2.3.7 Licht im Dunkel der Klassen

Um mehr Licht in die Zusammenhänge der bisher vorgestellten Klassen zu bringen und die Dringlichkeit der Klassifizierung aufzuzeigen, soll hier noch mal ein kurzer Überblick gegeben werden. Im Allgemeinen hat jeder Projektor eine bestimmte Anzahl von Funktionen. Diesen Funktionen werden, je nach proprietären Kommunikationsprotokoll, Funktionsklassen (Adjust, State, Execute oder Query) zugeordnet die einen Wertebereich (value, state, keine) implizieren. Zusätzlich muss jede Funktion, durch die hier vorgestellten Typen ihre Antworten (confirm, value, state, error) spezifizieren. Im Endeffekt ermöglichen diese Klassifizierungen dem Programm alle Daten, die während der Arbeit mit Projektor ausgetauscht werden, zu analysieren und der Nutzerschnittstelle zur Verfügung zu stellen.

	Adjust	Value	State	Execute	Query
Wertebereich	value	value	state	-	-
Get Befehl	erforderlich	optional	optional	-	erforderlich
Set Befehl	erforderlich	erforderlich	erforderlich	erforderlich	-
Get Parameter	keine	keine	keine	-	keine
Set Parameter	inc(+) | dec(-)	value	state	keine	-
Get Result	value | error	value | error	state | error	-	value / state
Set Result	confirm (ok) | value | error	confirm (ok) | value | error	confirm (ok) | state | error	confirm (ok) | error	error

2.3.8 Darstellung im Nutzerinterface

Nachdem die Use Cases, die anfallenden Aufgaben, die in einem Management System behandelt werden sollten, definiert sind, ergibt sich eine Vorstellung wie das Nutzerinterface schematisch aussehen könnte. Dabei sollen Aufgaben, wie „Projektor hinzufügen“, „Einstellungen ändern“ und „Programm beenden“ in der Menüleiste ausgewählt werden. Wie in Bild 2.6 verdeutlicht, sind die Projektorgruppen am linken Fensterrand angeordnet und stellen, bei Auswahl, die ihnen zugeordneten Projektoren mit ihren Hauptinformationen im rechten mittleren Teilfenster dar. Wird nun in diesem Fenster ein Projektor selektiert, werden die ihm zur Verfügung stehenden Steuerelemente im Steuerpanel-Fenster (rechts unten) angezeigt. Diese klar strukturierte Oberfläche gewährleistet in wenigen Schritten die gewünschten Funktionen, in einem dem Nutzer logischen Gliederung, auszuwählen.

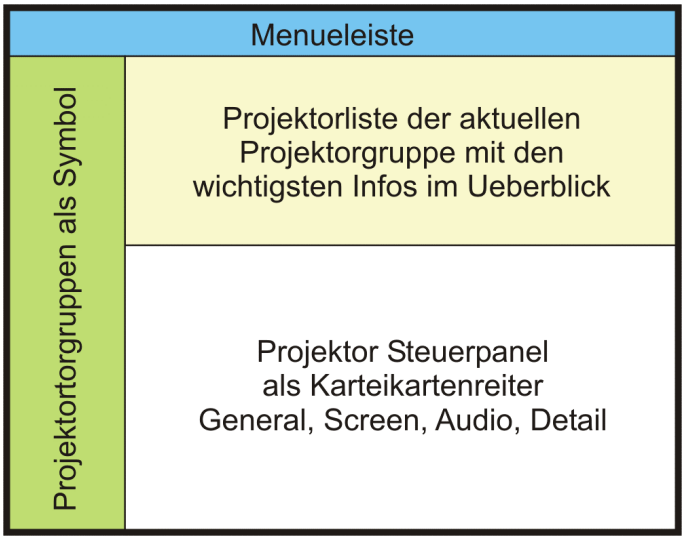


Bild 2.6: Entwurf eines Nutzerinterfaces

2.4 XML Treiber Philosophie

2.4.1 Sinn und Zweck

Eine Software für nur einen Projektortyp zu entwickeln wäre nicht sinnvoll, daher ist es erstens unabdinglich den Quellcode flexibel hinsichtlich verschiedener Geräte zu gestalten und zweitens Mittel und Wege zur Verfügung zu stellen, um nachträglich weitere Projektoren hinzuzufügen. Dieses Thema findet man weitestgehend in jedem Programm. Gelöst werden die Anforderungen durch so genannte Treiber, Codestücke die an passender Stelle ausgetauscht werden. Nun ist es nicht Ziel der Aufgabenstellung bei jedem neuen Gerät einen neuen Quellcode zu programmieren, der die neue Funktionsweise abbildet. Gesucht ist ein Weg, der mit möglichst einfachen Mitteln in kurzer Zeit einen neuen Beamertyp dem Programm vorstellt. Noch einmal, bei dem Thema Projektortreiber geht es um die Erfassung eines neuen Typs, nicht um das Hinzufügen eines tatsächlich existierenden Gerätes. Denn wenn der Software dieser neue Typ einmal bekannt ist, können danach ohne weiteren Aufwand unzählige bestehende Beamer konfiguriert werden.

Zurück zu der Frage nach simplen Wegen eines Treibers. Bei aufmerksamer Betrachtung der vorherigen Kapitel wird deutlich, dass alle Bemühungen, ein anpassungsfähiges Programm zu entwickeln, bereits getroffen wurden. Der hier propagierte interne Standard und die einheitliche Funktionsdarstellung sind die Grundpfeiler der universellen Einsetzbarkeit. Es ist somit die Aufgabe des Treibers, die Eigenarten des neuen Projektortyps in den softwareinternen Standard darzustellen. Hier gilt dies für das geräteabhängige Kommunikationsprotokoll. Noch einfacher ausgedrückt: findet man eine Möglichkeit zu Beschreibung des proprietären Standards, und zwar in einer Form, die das Programm kennt und versteht, kann man im Endeffekt weitere Projektortypen benutzen. Hier kommt die universelle, weit verbreitete Beschreibungssprache XML ins Spiel.

2.4.2 Warum XML?

Wie im letzten Abschnitt erwähnt, XML ist universell und weit verbreitet und somit jedermann zugänglich und bekannt. Zusätzliche Vorteile, die XML attraktiv als Beschreibungssprache für die proprietären Protokolle macht, sind die logische Bedeutung der Tags sowie die Speicherung von reinen Daten, in diesem Fall die Speicherung der proprietären Protokolldaten ohne Veränderung. Die Definition eigener Tags möglich, was somit die Anwendung der softwareinternen Semantik auf die Protokolle ermöglicht. Ein weiterer Aspekt, der auf den ersten Blick keinen weiteren Nutzen bringt, ist die flexible Darstellung von XML Dokumenten. Dies ermöglicht weiten Spielraum von einmal mit XML erfassten Daten. Zu guter letzt spricht schlicht die enge Bindung zwischen Python, der hier eingesetzten Programmiersprache, und XML dafür, diese Metasprache als einen Typus Treiber einzusetzen.

2.4.3 Ein Beispiel

Das nachfolgende Beispiel zeigt schemenhaft wie eine mögliche Konfigurationsdatei aussehen könnte. Dieser Ausschnitt eines XML Dokuments beschreibt Teile des Kommunikationsprotokolls eines fiktiven Projektortyps. Den Kern der Beschreibung bildet natürlich die Funktionalität des Beamers, gemäß den oben vorgestellten Funktionsklassen. Das Beispiel zeigt die Darstellung des wohl komplexesten Funktionstypen State.

```
01    <function id="SRC" type="State">
02      <name>Source</name>
03      <range type=”state”>
04        <state id="1">RGB1</state>
05        <state id="2">RGB2</state>
06        <state id="3">Video</state>
07        <state id="4">Viewer</state>
08        <state id="5">S-Video</state>
09        <state id="6">LAN</state>
10      </range>
11      <set>
12        <request>
13          <command parameter="state">input ####</command>
14          <parameter id="1">rgb1</parameter>
15          <parameter id="2">rgb2</parameter>
16          <parameter id="3">video1</parameter>
17          <parameter id="4">viewer1</parameter>
18          <parameter id="5">svideo1</parameter>
19          <parameter id="6">lan1</parameter>
20        </request>
21        <response>
22          <result answer="confirm">ok</result>
23        </response>
24      </set>
25      <get>
26        <request>
27          <command parameter="none">input</command>
28        </request>
29        <response>
30          <result answer="state">input:#####</result>
31          <answer id="1">computer1</answer>
32          <answer id="2">computer2</answer>
33          <answer id="3">video1</answer>
34          <answer id="4">viewer1</answer>
35          <answer id="5">svideo1</answer>
36          <answer id="6">lan1</answer>
37        </response>
38      </get>
39    </function>
```

2.4.4 Deklarationsregeln

Der im Beispiel dargestellt Teil eines XML Treibers, ist ein Vertreter einer typischen Projektor Funktionalität. Wichtig ist hier, dass die Funktion Source durch den Typ „State“ beschrieben wird. Dieser erlaubt die Definition von Zuständen. Zusätzlich ermöglicht er, Anfrageparameter sowie Antworten über das *id* Attribut auf den Status abzubilen. Grundlegend wird jede Funktionsdefinition in ein GET und SET untergliedert und dort jeweils mit REQUEST und RESPONSE beschrieben. Dabei wird eine RESPONSE durch ein COMMAND und eventuelle PARAMETER, ein REQUEST hingegen durch ein RESULT und mögliche ANSWERS beschrieben. Zur besseren Übersicht hier ein Auszug aus der **Document Type Definition DTD**. Auf eine komplette Definition, die alle Sonderfälle behandelt, soll hier hinsichtlich eines besseren Verständnisses verzichtet werden.

```
01   <!ELEMENT function  (name, (get | set)*>
02   <!ELEMENT name      (#PCDATA)>
03   <!ELEMENT set       (request, response)>
04   <!ELEMENT get       (request, response)>
05   <!ELEMENT request   (command, parameter?)>
06   <!ELEMENT response  (result, answer?)>
07   <!ELEMENT command   (#PCDATA)>
08   <!ELEMENT parameter (#PCDATA)>
09   <!ELEMENT result    (#PCDATA)>
10   <!ELEMENT answer    (#PCDATA)>
```

Ein weiteres Merkmal im XML Treiber ist die Möglichkeit mit **Wildcardzeichen #####** Parameterersetzungen zu kennzeichnen. Erscheint das Zeichen im COMMAND, sagt es aus, dass an dieser Stelle ein Ersatz stattfinden muss, um einen gültigen Befehl zu erlangen. Umgekehrt dient das Wildcardzeichen im RESULT um eine Antwort zu erlangen. Zur besseren Übersicht abschließend alle Deklarationsmöglichkeiten im Überblick.

- Funktionstyp ***function type***: Adjust | State | Value | Execute | Query
- Funktionsidentifikator ***function id***: Siehe Liste aller definierten Basic Functions
- Befehlsparameter ***command parameter***: state | value | none
- Antwortparameter ***answer parameter***: state | value | confirm

3 Implementierung

3.1 Python Klassen

3.1.1 Verbindung / Anschluss

Eine Verbindung, implementiert durch die *PrjConnection* Klasse ist eine abstrakte Sichtweise auf den realen Beameranschluss und dessen Kommunikationsregeln. Mit den Methoden der Klasse ist man in der Lage, ohne weitere Kenntnis über Protokoll oder Interface der wirklichen Verbindung, erstens zu prüfen ob der Projektor angeschlossen ist und zweitens Daten zum Projektor zu schicken und zu empfangen. Die Arbeitsweise ist relativ einfach und ist nachdem die Daten als String übergeben wurden, im Wesentlichen aus vier Schritten aufgebaut:

- **Request bilden**
  Daten werden von *PrjProtocol* von entsprechenden Header und Tail umhüllt.
- **Request senden**
  Es wird der Request mit Hilfe der *PrjInterface* Klasse an den Projektor gesendet.
- **Response empfangen**
  Sofort nach dem senden wird eine Response von der *PrjInterface* Klasse empfangen.
- **Response umwandeln**
  Um wieder Daten zu erhalten werden Header und Tail von der Response entfernt.

Die Klasse *PrjInterface* ist die generische Schnittstelle und ist in der Lage mehrere Arten von Interfaces in sich zu vereinen. Worauf ihr zur Laufzeit eine weitere Klasse zugeordnet wird, die die notwendigen Methoden zum Senden und Empfangen von Daten, je nach reeller Anschlussart, kennt. Die Connection Klasse startet automatisch bei ihrer Instanziierung einen Hintergrundthread der den **Verbindungsstatus** mit Hilfe der Methode *check()* fortlaufend aktualisert. In Bild 3.1. werden die Zusammenhänge der drei Klassen noch mal verdeutlicht.

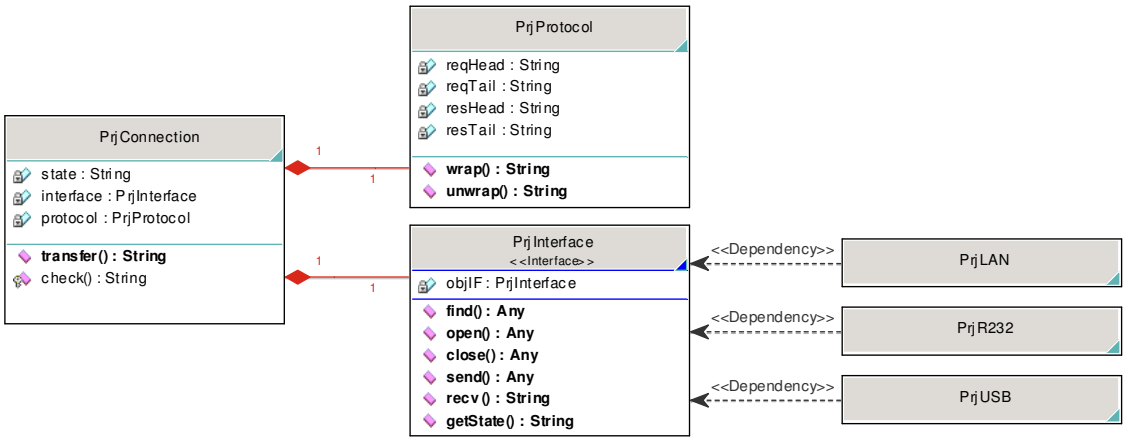


Bild 3.1: Klassendiagramm Verbindung

3.1.2 Projektorfunktionen

Wie bereits festgestellt, ist es möglich, jede noch erdenkliche Form von Projektorfunktionen mit ihren Steuerungskommandos einer der vorgestellten fünf Funktionsklassen zuzuordnen. Genau dies geschieht auch bei der Implementierung. Durch die Umsetzung der Funktionsklassen und ihrer Wertebereiche im Quellcode, erhält man, die in Bild 3.2 aufgezeigten, fünf instanzierbaren Klassen:

- *PrjFuncAdjust*
- *PrjFuncValue*
- *PrjFuncState*
- *PrjFuncExecut*
- *PrjFuncQuery*

Nach dem Programmstart wird für jede reale Projektorfunktionalität, entsprechend ihrer Klasse, ein Funktionsobjekt instanziert. Jedes einzelne Objekt repräsentiert folglich einen Teil der Steuerung des Projektors. Beispielsweise ist über die Methoden *get()* und *set()* der Klasse *PrjFuncState,* ein Zustand vom Beamer abfrag- und einstellbar. Steuerkommandos, Wertebereiche, mögliche Antworten, Errors usw., kurz das ganze proprietäre Verhalten des Projektor steckt Funktion für Funktion in diesen Objekten.

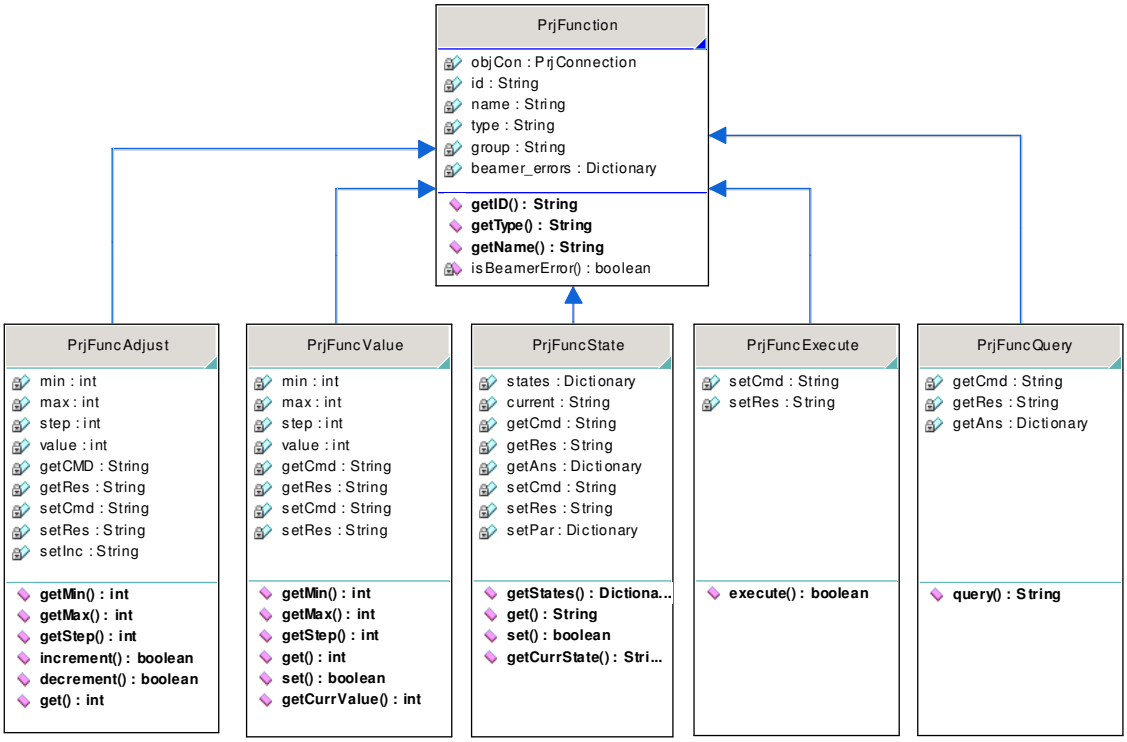


Bild 3.2: Klassendiagramm Verbindung

3.1.3 Der einzelne Projektor

Jeder, dem System hinzugefügte Projektor, wird als Objekt der Klasse *PrjProjector* behandelt und ist demzufolge die höchste Abstraktion des realen Gerätes. Um ein umfangreiches Management umzusetzen, ist das Projektorobjekt mit einer Vielzahl an Attributen und Metainformationen ausgestattet. Einen detaillierten Einblick soll folgende Auflistung geben.

- ***attrFile*** beinhaltet das zugehörige Treiberfile, welches den Projektortyp beschreibt
- ***attrPower*** repräsentiert den Betriebszustand des Projektors (Ein, Aus, Stand by)
- ***attrLamp*** ist die aktuelle Lampenbrenndauer in Stunden
- ***attrConnection*** zeigt den Verbindungszustand zum Projektor
- ***attrTemperature*** vertritt den Temperaturzustand des Gerätes
- ***attrState*** ist der allgemeine Zustand des Projektors, z.B. „Running“ oder „Fan Error“
- ***attrUsage*** beinhaltet die Gesamtnutzungsdauer in Stunden
- ***attrIPAddress*** speichert, wenn erforderlich, die aktuelle IP Adresse

- ***metaName*** ermöglicht die Angabe eines Names zur besser Identifizierung
- ***metaLocation*** lässt die Spezifizierung eines Raumes oder Ortes zu
- ***metaLamp*** sollte Zusatzinformationen zur Lampe beinhalten, z.B. Hersteller, Typ, Preis
- ***metaFan*** ist ähnlich wie bei metaLamp, beinhaltet zusätzliche Angaben zum Lüfter
- ***metaProjector*** gibt Auskunft über Eigenheiten/-arten des Projektors
- ***metaSupport*** zeigt an wo notwendige Hilfe bei Reparaturen bezogen werden kann
- ***metaResponse*** erfasst die Verantwortlichen, z.B. Hausmeister oder Rechenzentrum
- ***metaUndefined*** dient als zusätzliches Informationsfeld ohne vorherige Festlegung

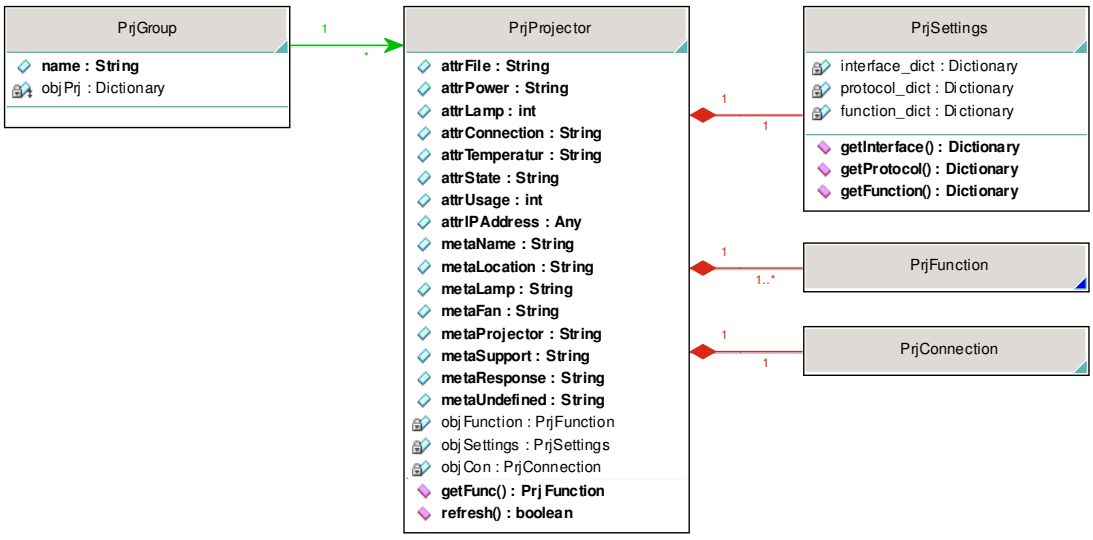


Bild 3.3: Klassendiagramm Verbindung

Die hier vorgestellte *PrjProjector* Klasse besitzt zusätzlich ein Liste der Beamer Funktionalität in Form der in Abschnitt 3.1.2 erläuterten Projektorfunktionen. Ferner noch repräsentiert ein Verbindungsobjekt den reellen Anschluss mit Protokoll und Schnittstelle. Das dem Projektor zugeordnete *PrjSettings* Objekt ist verantwortlich, das Treiberfile zu parsen und die notwendigen proprietären Informationen bereitzustellen. Zusammenfassend ist zu sagen, dass die *PrjProjector* Klasse und die ihr zugehörigen Objekte, Attribute und Metainformationen es erlauben einen reellen Projektor innovativ und, wie in der der Spezifikation gewünscht, flexibel abzubilden.

3.1.4 XML Parser

Der XML Parser ist in drei Stufen untergliedert. Als ersten Schritt parst die Klasse *TreeParser* die XML Treiber Datei und erstellt daraus einen **DOM Baum**. Die enthaltenen Daten werden normalisiert, das heißt „Whitespace“ wird entfernt und „New Line“ sowie „Carriage Return“ werden umgewandelt. Die nächste Aufgabe ist es, die Schnittstellen- und Funktionsinformationen aufzubereiten. Dabei werden sie durch den *Revealer* in das zur späteren Instanzierung der Funktionstyp-, Interface- und Protokollobjekt(e) benötige Format gebracht. Das Resultat des zweiten Schrittes sind ein Schnittstellen- (interface_dict) und ein Funktionsdictionary (function_dict), die die eben angeführten Informationen enthalten. Zum Schluss ist es nun möglich, diese beiden Datenstrukturen über die Klasse *PrjSettings* abzufragen und beim Erzeugen den oben angeführten Objekten mitzugeben.

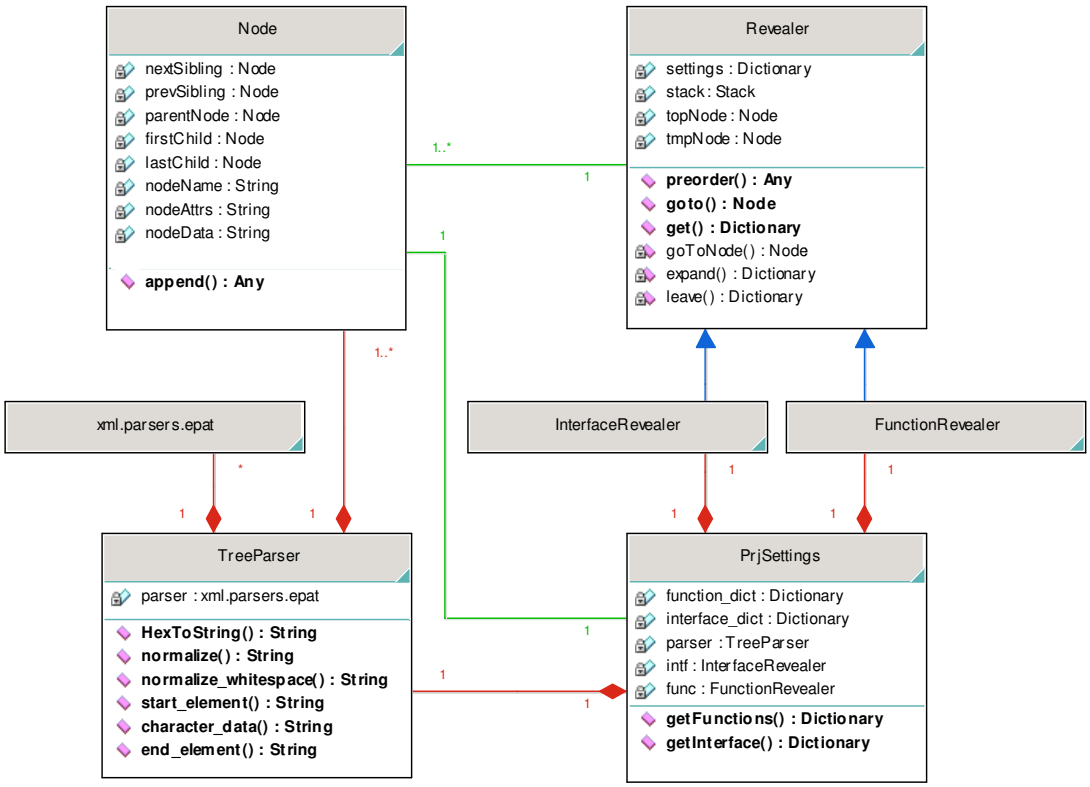


Bild 3.4 Klassendiagramm XML Parser

3.1.5 GUI Klassen

Wie aus dem vergangen Abschnitt bekannt ist, abstrahiert die Klasse *PrjProjector* die Funktionalität und die Verbindung eines real existierenden Projektionsgerätes. Um dieses Gerät mit Hilfe einer grafischen Oberfläche zu steuern, wird in der Klasse ***PrjControlPanel*** eine Reihe von so genannten Widget-Objekten, implementiert in der Klasse ***PrjWidget***, instanziert und als eine Steuerungseinheit angezeigt.

Jedes Widget, prinzipiell als grafisches Steuerelement zu verstehen, steht in Beziehung zu einer entsprechende Projektorfunktion (bekannt aus Abschnitt 3.1.2), die die Befehle an das Steuerelement in Projektorbefehle umsetzt und zum Gerät sendet. Alle Oberflächenelemente erben die zur Darstellung notwendigen Eigenschaften aus der Klasse *Tkinter.Frame*, welche unter Python zur Verfügung gestellt wird. Die differenten Widgets dienen zur Darstellung der unterschiedlichen Projektorfunktion. So sind beispielsweise den Projektorfunktionen *PrjFuncAdjust* und *PrjFuncValue* den Steuerelementen *PrjUpAndDownWidget* und *PrjScaleWidget* zuzuordnen.

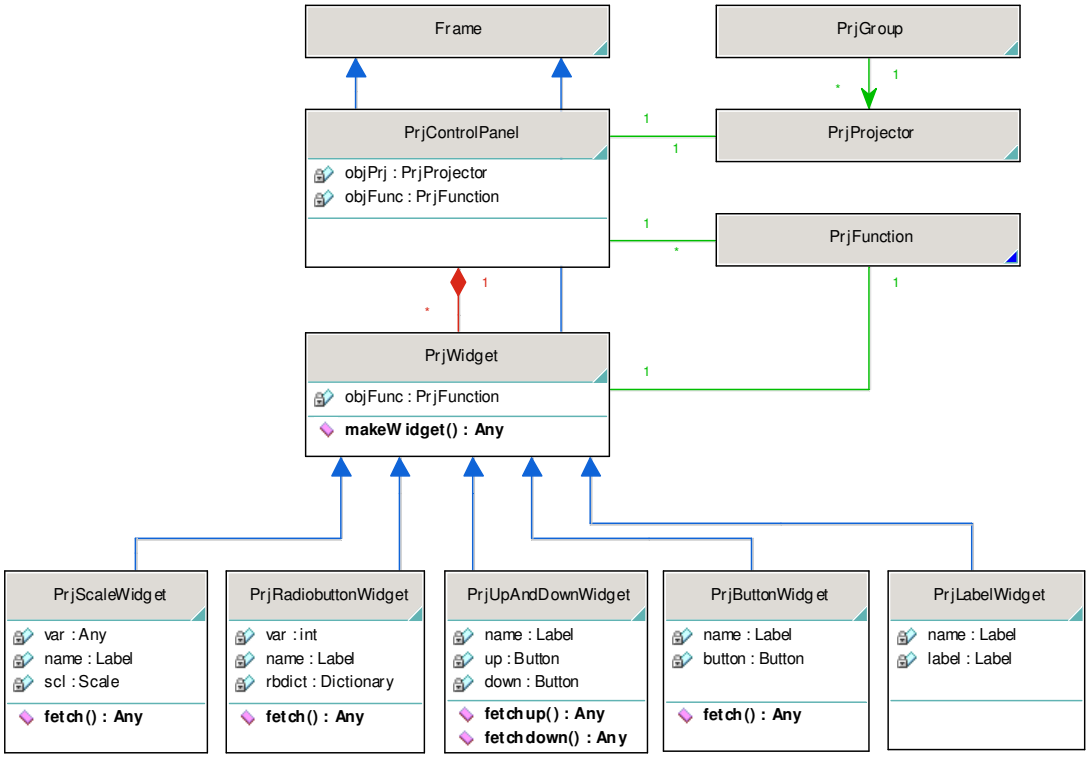


Bild 3.5: Klassendiagramm Verbindung

3.2 Nutzer Interface

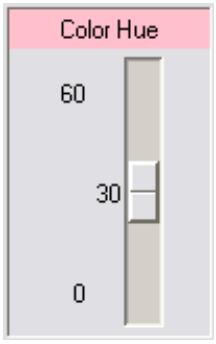
3.2.1 Python's Tkinter Module

Zu Implementierung des Nutzer Interfaces sei zu erwähnen, dass das ***Tkinter*** Module ein Quasi-Standard zur Umsetzung einer Grafischen Nutzeroberfläche mit der Programmiersprache Python ist. Es ist die Schnittstelle zu dem ***Tk* GUI Toolkit** von Scriptis, welches eine frühere Entwicklung der Sun Labs war. Die Tatsache, dass beide für die meisten Unix Plattformen, Windows und Macintosh Systeme erhältlich sind, favorisiert *Tkinter* zur Realisierung der Projektor-Management-GUI.

3.2.2 Steuerwidgets

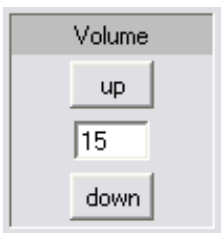
PrjRadiobuttonWidget

Dieses Steuerelement eignet sich zur Darstellung aller Projektorfunktionen, welche mehrere Zustände besitzen. Somit ist es die grafische Umsetzung der Funktionsklasse ***PrjFuncState***. Es kann einerseits mit Buttons (wie hier dargestellt) oder mit Radiobuttons angezeigt werden. Es ist weiterhin immer nur ein Zustand wählbar. Durch die Hervorhebung und die Anordnung gibt es einen schnellen Überblick über die Eigenschaften und den tatsächlich vorhandene Situation der Projektorfunktion.



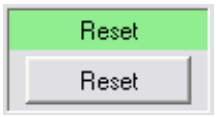
PrjScaleWidget

Wie deutlich erkennbar ist, werden die Werte, der hinterlegten Funktionalität ***PrjFuncValue***, durch eine Skale verändert. Dies gibt dem Nutzer die Möglichkeit auf einen Blick den aktuellen, den minimal und den maximal einstellbaren Wert abzulesen und die vorgegebene Schrittweite einzuhalten. Zusätzlich ist das Verhältnis des aktuellen Wertes leicht zu Minimum und Maximum einzuordnen. Diese schnelle und intuitive Relativierung hilft besonders bei Einstellungen des Projektionsbildes sowie bei Tonwerten.



PrjUpAndDownWidget

Erhöhen und Verringern eines Wertes, das Kennzeichen der Funktionsklasse ***PrjFuncAdjust*** spiegelt sich in diesem Widget wieder. Die Angabe des aktuellen Wertes ist optional und kann bei Bedarf ein- und ausgeblendet werden. Darüber hinaus ist auch bei diesem Steuerelement eine selbsterklärende Handhabung gewährleistet.



PrjButtonWidget

Dieses Steuerelement ist einfach und funktionell in seinem Aufbau..Die zugeordnete ***PrjFuncExecute*** Klasse bedingt dieses Design, da nur ein Steuerkommando abgesendet werden muss.

PrjLabelWidget

Noch einfacher ist dieses Widget aufgebaut, da es als einzige Aufgabe die Anzeige des durch die PrjFuncQuery Klasse in Erfahrung gebrachten Wertes oder Zustandes hat.

3.3 Testserver als Projektorersatz

3.3.1 Erfordernis

Bei der Softwareentwicklung gilt das Hauptaugenmerk der Steuerung von festinstallierten Beamern über ein Netzwerk, so genannte LANs. Projektoren mit einer Ethernet-Schnittstelle arbeiten nach dem bekannten Client-Server Prinzip und stellen einen internen Serverdienst bereit. Dieser ermöglicht es, Anfragen und Steuerungskommandos an den Beamer zu senden. Um keine Hardwareressourcen zu verschwenden bzw. keine Projektoren unnötig zu blockieren, genügt es für den Großteil der Softwareentwicklung einen Testserver zu programmieren, der den einfachen Dienst des Projektors nachbildet. Ähnlich eines vertikalen Prototyps soll diese Nachbildung nicht die komplette Funktionsbreite darstellen, sondern eine möglichst exakte Umsetzung konzentriert auf wenige Fähigkeiten bieten.

3.3.2 Protokoll und Funktionsweise

Der Testserver speichert die Zustände von POWER, SOURCE und BRIGHTNESS. Weiterhin erlaubt er LAMPTIME abzufragen und RESET durchzuführen. Das dazu verwendete Protokoll ist dem Kommunikationsstandard von *Infocus* angelehnt und wird in folgender Tabelle veranschaulicht.

Funktion	Typ	Get	Set
PWR	State	PWR;?	PWR;#
SRC	State	SRC;?	SRC;#
LMT	Query	INF,5	
BRT	Adjust	BRT;?	BRT;#
RES	Execute		RES;1

Die obige Protokolldarstellung entspricht der intern implementierten Standardisierung und wird im Abschnitt ***Softwareintern genutztes Protokoll (SGP)*** näher erläutert. Soweit zu den Anfragen (REQUEST) an den Server. Die Antworten (RESPONSE) sehen ähnlich einfach aus. Sie werden hinsichtlich der Anfrageergebnisse unterschieden. Erfolgreiche Requests werden mit „OK“, optio-

nal gefolgt von einem Wert, bestätigt. Hingegen werden missglückte Anfragen mit „ERROR“ beantwortet. Das Ende jeder Response wird mit einem Zeilenumbruch gekennzeichnet. Erwähnenswert ist noch, dass der Dienst des Testservers auf **Port 50007** bereitgestellt wird.

3.3.3 Programmierung

Für die objektorientierte Implementierung des Testservers, wahlweise hier in der Programmiersprache Python, werden drei Klassen entworfen, die hier kurz vorgestellt werden. Die Klasse ***InternalStates*** bildet die oben angesprochenen Zustände POWER, BRIGHTNESS, SOURCE und LAMPTIME ab. Jeder Zustand kann über eine entsprechende GET/SET Methode abgefragt bzw. verändert werden. Um den Kommunikationsstandard des Serverdienstes darzustellen, existiert die Klasse ***InternalProtokoll***. Sie wertet die vom Server empfangenen Zeichenfolgen nach einem bestimmten Schema aus und verändert bei Bedarf die inneren Zustände des Servers. Zum Schluss übernimmt die Klasse ***TestServer***, basierend auf einer Socketschnittstelle, alle auftretenden Aufgaben, die beim Datenaustausch über Netzwerke auftreten können.

3.4 Terminalprogramm

3.4.1 Allgemein

Um die Leistungsfähigkeit des Softwareentwurfs hinsichtlich der Projektor Funktionsklassen zu zeigen und ein erstes funktionsfähiges Programm zu entwickeln, ist ein Terminal ohne GUI implementiert und auf den Softwarekern (vorgestellt unter 3.1.1 bis 3.1.4) aufgebaut. Dadurch dient das Terminal unter anderem der Erprobung der XML Treiber Philosophie. Da die Handhabung nicht so intuitiv wie mit einer grafischen Nutzeroberfläche ist, werden in diesem Kapitel die grundlegenden Bedienungshandgriffe aufgelistet.

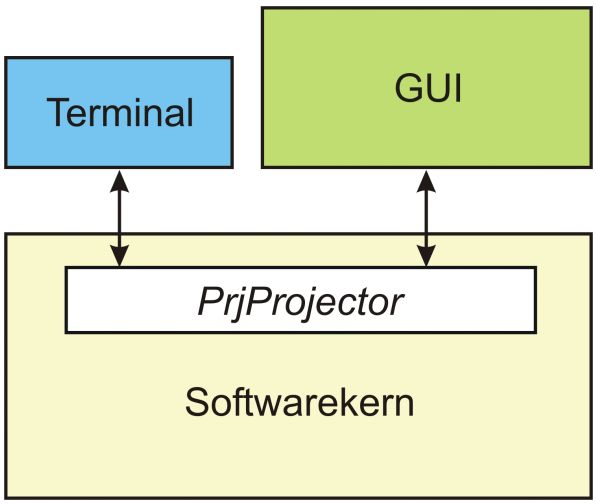


Bild 3.6: Schemadarstellung Projektor Software

3.4.2 Startparameter

Beim Starten des Terminalprogramms sind über eine Argumentenliste gewünschte Einstellungen vorzunehmen. Für eine erfolgreiche Startkonfiguration, ist es mindestens erforderlich Projektortyp (-t) und Projektoranschluss (-i) anzugeben. Wir das Programm für ein LAN Interface gestartet ist weiterhin eine IP-Adresse (-ip) zu wählen. Folgendes Beispiel sowie die Tabelle geben über die Handhabung der Startparameter detailliert Auskunft.

```
PrjTerminal.py –t NEC_LT245 –i LAN –ip 192.168.0.10
```

Argument	Beschreibung
h	Listet alle Argumente auf
f	Listet alle zur Verfügung stehenden Befehle auf
i	Spezifiziert die gewünschte Schnittstelle
ip	Gibt die IP-Adresse an
p	Gibt den Port an, erforderlich bei unbekannte Projektortypen
l	Listet alle Projektortypen auf
if	Listet alle unterstützten Schnittstellen
t	Spezifiziert einen Projektortyp
d	Zeigt eine kurze Beschreibung an
c	Listet maximale zur Verfügung stehende Projektor-Steuerbefehle
v	Listet Terminalversion

3.4.3 Befehle

Während des Betriebes können mit Hilfe folgender Befehle die beschriebenen Aktionen ausgeführt werden. Hierzu ist anzumerken, dass (bis auf **`ASCII`** und **`HEX`**) die aufgelisteten Befehle keine Projektorsteuerung vornehmen.

Befehl	Beschreibung
CMD	Zeigt eine Liste aller zur Verfügung stehenden Befehle an
FUNC	Zeigt eine Liste der aktuell zur Verfügung stehenden Projektorfunktionen
HELP	Zeigt eine Kurzbeschreibung an
CHECK	Überprüft Verbindungsstatus
ASCII	Sendet ASCII Zeichen zum Projektor, Beachte korrekte Einstellung von EOC
HEX	Sendet eine HEX Zeichenfolge zum Projektor z.B.HEX 02:00:00:00:00:02
EOC	Zeigt das "End of Command" Zeichen an, Setzt ein neues EOC
IP	Ändert die IP Adresse, auf der der Projektor gesucht wird
PORT	Ändert den Port, auf dem der Service gesucht wird
QUIT	Beendet das Programm

3.4.4 Funktionen

Das Kernstück des Projektorterminals bilden die Funktionen. Sie entsprechen in ihrer Gesamtheit der Funktionalität des durch die Startparameter selektierten Beamers. Funktionen repräsentieren die Steuerbefehle und können mit **`FUNC`** aufgelistet werden. Am Beispiel von Bildquelle „Source“, mit zugehörigen Kommando **`SRC`**, soll der grundlegende Umgang mit ihnen verdeutlicht werden.

Kommando	Beschreibung
`SRC`	Der Befehl ohne Parameter fragt den aktuellen Zustand ab
`SRC ?`	Der Befehl mit Fragezeichen gibt eine Kurzinfo über „Bildquelle“ sowie mögliche Parameter und deren Beschreibung
`SRC 1`	Setzt die Bildquelle auf den „Input 1“

4 Zukünftige Entwicklungsmöglichkeiten

4.1 Weiterentwicklung der Software

4.1.1 Softwarehosting Source-Forge

Zur Fortsetzung der hier gewonnenen Erkenntnisse und zur Weiterentwicklung der Software, ist unter dem weltweit größten Open Source Softwareentwicklungsportal SourceForge (www.source-forge.net) das Projekt „**Projector Management and Control**“ (pronetmanager) angelegt.

4.1.2 Befehlsfolgen und verbesserte HEX-Kommandos

Zur Umsetzung der in Absatz 1.3.2 angesprochenen Befehlsfolgen sind Erweiterungen der Funktionsklassen nötig. Dabei ist es erforderlich jeder auftretenden Befehlsverkettung eine Semantik zuzuordnen um die Abhängigkeit zum Ergebnis des vorherigen Steuerkommandos oder dessen Antwort vom Projektor darzustellen. Vorstellbar wäre eine **Handshake Attribut**, welches diese Zusammenhänge für jeden einzelnen Befehl umsetzt.

Weiterhin ist es sinnvoll, zur Darstellung komplexer hexadezimaler Befehle, im Entwurf Methoden zur Abbildung von Prüfsummen und Zahldarstellungen (big Endian / Little Endian) zu berücksichtigen.

4.2 Erweiterung des Funktionsumfangsang

4.2.1 XML Treiber Utility mit globaler Datenbank

Ein Tool zur Erstellung eines XML Treibers wäre eine sinnvolle Zusatzanwendung, die syntaktisch Fehler unmöglich machen würde. Nicht nur der gebotene Komfort und die Vereinfachung, die durch so ein Programm entsteht würde, wäre ein großer Vorteil, sondern auch die Möglichkeit zur Treibererstellung für unversierte Nutzer, würde zu einer größeren Akzeptanz und folglich zu einer schnellerer wachsenden Verbreitung dieses hier entworfenen Standards führen. Da, wie in dieser Arbeit aufgezeigt, letztendlich unter anderem die Menge der unterstützen Projektortypen über Erfolg einer Software entscheidet, ist dies eine fundamentale Erweiterung.

Eine Datenbank, die erfolgreich eingesetzte Treiber Files beinhaltet und jeder Distribution der Software zugänglich macht, würde zu einer schnellern Verbreitung neuer Projektortypen führen. Dabei wäre es durchaus denkbar, dass auch Endnutzer neue Treiber publizieren können und somit eine Vielzahl an Personen die Grundlage für ein universelles Projektor-Management schaffen.

4.2.2 Events und Mailfunktionalität

Ein Benachrichtigungsmechanismus, wie bei den meisten Programmen bereits vorhanden, unterstützt die Alarmierung bei eintretenden Problemfällen. Dazu zählen unter anderem, Überhitzung des Projektors, Defekt der Lampe oder des ganzen Gerätes, anstehende Wartungsarbeiten und

Diebstahl. Durch eine Mailfunktionalität könnte eine Bekanntgabe dieser oder ähnlicher Fälle über die Grenzen der Management-Software hinaus, den Nutzwert des Programms erheblich steigern. Somit könnte der Administrator, beispielsweise beim täglichen Abrufen seiner Mails zu Begin eines Arbeitstages, sofort Probleme erkennen. Weiterhin wäre es durchaus vorstellbar, dass auch der Hausmeister oder die Wartungsfirma per Email von anstehenden Aufgaben in Kenntnis gesetzt werden.

Grundvoraussetzung für eine ereignisbasierte Mailerweiterung, ist das Anlegen von Events im System. Diese können dann an Projektorfunktionen gebunden werden. Sinnvolle Erweiterungen wären ferner Eventprioritäten, Verantwortlichkeiten oder bestimmte Maßnahmen, die einem Ereignis zuzuordnen sind. All diese Zusätze erleichternden den Umgang mit Projektoren und leisten wichtige Dienste hinsichtlich eines erfolgreichen Managements.

4.2.3 SNMP Unterstützung

Die wohl wichtigste Erweiterbarkeit, um diese Software auch für professionellen Anwender attraktiv zu machen, ist eine SNMP Erweiterung. Vorstellbar wäre ein SNMP Daemon, verbunden mit dem Softwarekern der Projektor Management Software. Dieser bietet einer Netzwerkmanagementstation die Möglichkeit über das SNMP Protokoll Projektordaten aus einer MIB abzurufen. Eine konzeptueller Entwurf wird im folgenden Bild illustriert.

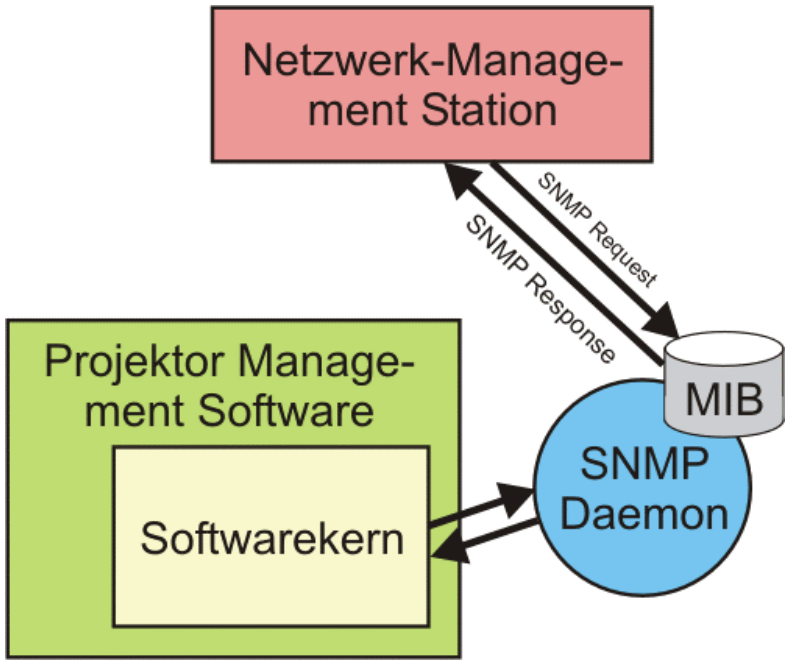


Bild 4.1 Zusammenspiel SNMP Daemon und Software

5 Quellenverzeichniss

[A01] Alex Martelli und David Ascher, *Python Cookbook*
1. Auflage, O'Reilly Verlag, 2002 Sebastopol

[B01] Benjamin Benz und Thorsten Thiele, *Fernkontrolle*
c´t 2004 Heft 14, Seite 214ff

[B02] Benjamin Benz und Thorsten Thiele, *Brücken bauen*
c´t 2004 Heft 13, Seite 200ff

[G01] Gaynor Redvers-Mutton/Paul Crocket, *Interaction Design*
John Wiley & Sons, 2002 Danvers

[G02] Günter Born, *Jetzt lerne ich XML*
1. Auflage, Markt + Technik Verlag, 2001 München

[H01] Hartmut König, *Protocol Engineering*
1. Auflage, B.G. Teubner Verlag, Wiesbaden 2003

[M01] Mark Lutz und David Ascher, *Einführung in Python*
1. Auflage, O`Reilly Verlag, 2000 Köln

[M02] Mark Lutz, *Programming Python*
1. Auflage, O'Reilly Verlag, 2001 Sebastopol

[M03] Martin v. Löwis und Nils Fischbeck, *Python 2*
2. Auflage, Addison-Wesley Verlag, 2001 München

[P01] Python – Website and mailsystem
http://www.python.org/

[S01] Steve Holden, *Python Web Programmierung*
1. Auflage, New Riders, 2002 Indianapolis

[S02] Ian Sommerville, *Software Engineering*
6. Auflage, Addison-Wesley Verlag, 2001 Essex

[W01] Wikipedia – Die frei Enzyklopädie: Projektor
http://de.wikipedia.org/wiki/Diaprojektion

[W02] Wikipedia – Die frei Enzyklopädie: SNMP
http://de.wikipedia.org/wiki/SNMP

[W03] Wikipedia – Die frei Enzyklopädie: Managment
http://de.wikipedia.org/wiki/Management