Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Diplomarbeit

Oliver Schmidt

Entwicklung eines Mehrbenutzer-Webservice am Beispiel eines Kamera-Servers für mobile Roboter

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Oliver Schmidt

Entwicklung eines Mehrbenutzer-Webservice am Beispiel eines Kamera-Servers für mobile Roboter

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Technische Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Kai von Luck
Zweitgutachter : Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 7. März 2006

Oliver Schmidt

Thema der Diplomarbeit

Entwicklung eines Mehrbenutzer-Webservice am Beispiel eines Kamera-Servers für mobile Roboter

Stichworte

Mobile Roboter, .NET-Framework, XML-Webservices, IT-Sicherheit, WSE-Framework.

Kurzzusammenfassung

Diese Diplomarbeit beschäftigt sich mit dem Design und der Realisierung eines Kamera-Services für mobile Roboter. Das System ist für einen Mehrbenutzer-Betrieb konzipiert, so dass mehrere Roboter gleichzeitig das System benutzen können. Für jeden Benutzer steht ein eigenes Benutzer-Konto zur Verfügung, auf dem Objekte angelegt und verwaltet werden können, welche vom System beobachtet werden sollen. Die Beschreibung erfolgt momentan über die Farbe der Objekte. Das System ist so gestaltet, dass es einfach um weitere Beschreibungsformen erweitert werden kann. Der Zugriff auf den Kamera-Service erfolgt über XML-Webservices, so dass Benutzer von verschiedenen Plattformen das System nutzen können. Zentraler Bestandteil des Kamera-Service ist eine Objekt-Datenbank, auf die die Benutzer über einen Webservice zugreifen können. Eine Applikation aktualisiert ständig die Positionen der gespeicherten Objekte mit Hilfe eines Kamera-Frameworks. Die Sicherheit des Systems wird über eine Variante des Kerberos-Protokolls gewährleistet, mit dem sowohl die Zugangskontrolle, als auch die Datenverschlüsselung vorgenommen wird.

Oliver Schmidt

Title of the paper

Development of a multiuser-webservice by example of a camera-server for mobile robots

Keywords

mobile robots, .NET-Framework, XML-Webservices, IT-Security, WSE-Framework.

Abstract

This thesis covers the design and the realization of a camera-service for mobile robots. The system is designed for a multiuser service. Thus several robots are able to use the system at the same time. Each user has his own account available. On each account objects can be created and managed, which should be monitored by the system. The description is temporary handled by the color of the objects. The construction of the system enables to add further descriptions. The access on the camera service is done through a XML-Webservice. Thus user can use the system from different platforms. Central component of the camera-service is an object-database, on which the the can access through a webservice. An application frequently updates the positions of the saved objects by using a camera-framework. The system security is guaranteed by a variation of the Kerberos-protocol. Which supervises the access and executes the data encryption.

Danksagung

An dieser Stelle möchte ich mich bei einigen Leuten bedanken, die mich während meines Studiums und speziell bei meiner Diplomarbeit unetrstützt haben.

Ich danke meinem Betreuer Kai von Luck für die fachliche und moralische Unterstüzung bei der Erstellung meiner Diplomarbeit. Ich danke meinen Eltern für die finanzielle Unterstützung während des Studiums. Meiner Schwester danke ich für das Korrektur-Lesen. Last but not least möchte ich mich bei meiner Freundin Andrea für ihre Geduld bedanken.

Vielen Dank euch allen!

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

In der heutigen Zeit verbreitet sich der Einsatz von Kameras zur Überwachung von Vorgängen immer mehr. Das liegt zum einen daran, dass die Kamera-Technologien immer billiger werden und zum anderen daran, dass die heutige IT-Technik, die Bildinformationen viel besser und schneller auswerten kann, als noch vor weniger Zeit. So werden z.B. heutzutage viele technische Produktions-Vorgänge mit Kameras überwacht, um eine bessere Qualität der Produkte erreichen zu können. Es können jedoch nicht nur technische Vorgänge, sondern auch Personen beobachtet werden. Es wäre z.B. denkbar, die Auslastung des öffentlichen Nahverkehrs beobachten zu lassen, um einen optimalen Einsatz der zur Verfügung stehenden Fahrzeuge zu berechnen. Es wird sicherlich eine sehr große Anzahl solcher Einsatzbereiche geben.

Ein weiterer Aspekt der modernen Datenverarbeitung ist die Datensicherheit. Diese spielt auch in vielen Kamera-Anwendungen eine große Rolle, vor allem wenn persönliche Daten erfasst werden. Dieses könnte z.B. der Fall sein, wenn eine automatische Anwesenheitsüberwachung am Arbeitsplatz entwickelt würde. Aber nicht nur die Geheimhaltung von sensiblen Daten ist wichtig, sondern auch, dass diese vor unbefugten Manipulationen geschützt werden. Dieses ist bei allen Anwendungsfällen wichtig, bei denen sehr großer Schaden, durch verfälschte Daten entstehen könnte.

Ein weiterer wichtiger Punkt, der in der heutigen Datenverarbeitung im allgemeinen und somit auch für eine Kamera-Anwendung von Bedeutung ist, ist die Unabhängigkeit von bestimmten Technologien. Ein modernes IT-System sollte aus voneinander möglichst unabhängigen Komponenten bestehen, um ein möglichst großes Einsatzgebiet abdecken zu können.

1.2 Worum geht es hier

Diese Diplomarbeit beschäftigt sich mit dem Design und der Implementierung eines Kamera-Webservice unter Berücksichtigung von sicherheits-technischen Aspekten.

Diese Arbeit baut auf meiner Studienarbeit über einen Kamera-Service auf (Schmidt, 2005, Schmidt). In der Studienarbeit wurde basierend auf der „MS .NET-Technologie“ ein Prototyp entwickelt, mit dem ein mobiler Client über WLAN bei einem Server die Positionen bestimmter Objekte abfragen kann. Diese Objekte bewegen sich auf einem abgegrenzten Feld (Roboter-Fußball-Spielfeld), welches von einer Kamera überwacht wird. Unter Verwendung des Kamera-Frameworks (Revout, 2003, Diplomarbeit Revout) werden mit entsprechenden Bildverarbeitungs-Algorithmen die Positionen der Objekte überwacht und können vom Client abgefragt werden. Dabei ist ein Prototyp entstanden, der diese Aufgabe prinzipiell erfüllt.

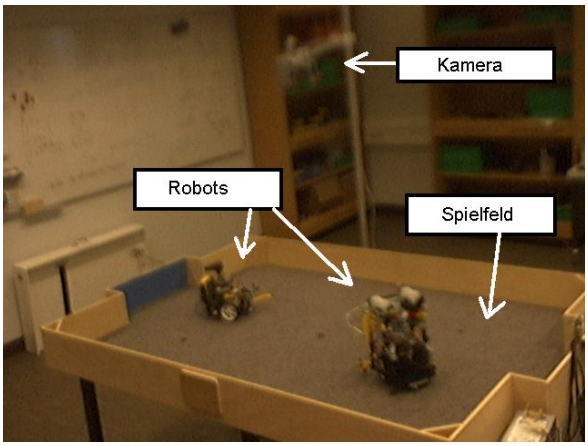

Abbildung 1.1: Kamera-Einsatz beim Robot-Fußball

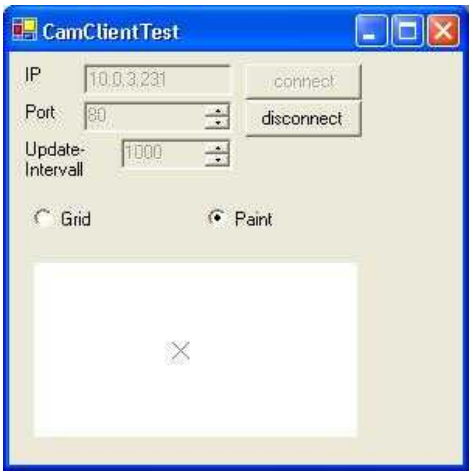

Abbildung 1.2: Benutzer-Schnittstelle beim mobilen Client

Ziel dieser Arbeit soll die Entwicklung eines Mehrbenutzer-Kamera-Service auf der Basis von Standard-Technologien sein. Dabei soll für den Service ein Sicherheitskonzept entwickelt werden, welches die grundlegenden Aspekte der IT-Sicherheit abdeckt.

1.3 Übersicht

Im Kapitel „Analyse“ werden die Anforderungen an das System mit Hilfe von Szenarios festgelegt. Im Kapitel „Design und Realisierung“ wird ein Gesamtsystem entworfen, welches die im Kapitel 2 gestellten Anforderungen erfüllen soll. Dazu wird das Gesamtsystem in einzelne Komponenten aufgeteilt und die entsprechenden Technologien diskutiert. Außerdem wird in diesem Kapitel gezeigt, was realisiert worden ist. Das Kapitel „Evaluation“ überprüft, ob die gesteckten Ziele realisiert worden sind und überprüft die Einsatzfähigkeit des Systems Anhand von einem Testaufbau. Das abschließende Kaptitel „Resümee“ zeigt einen Überblick über die erreichten Ziele und gibt einen Ausblick auf zukünftige Weiterentwicklungen.

1.4 Warenzeichen

Alle in dieser Arbeit verwendeten Firmen- und/oder Produktnamen sind Warenzeichen und/oder eingetragene Warenzeichen ihrer jeweiligen Hersteller in ihren Märkten und/oder Ländern.

2 Analyse

In diesem Kapitel wird zunächst das Umfeld des zu entwickelnden Systems beschrieben. Anschließend werden die Ziele festgelegt, die das System leisten soll.

2.1 Szenario

In diesem Abschnitt soll die Funktionsweise des Kamera-Service anhand eines Szenarios im allgemeinen und im speziellen Anwendungsfall „Robot-Fußball“ gezeigt werden.

2.1.1 Die Peripherie

Der „Robot-Fußball“ wird nach dem Vorbild des RoboCups[1] in dem Roboter-Labor der HAW Hamburg gespielt.

Roboterfußball wird als standardisiertes Problem benutzt, an dem sich Ergebnisse aus den verschiedenen Forschungsdisziplinen direkt vergleichen lassen. Der RoboCup bietet damit den Forschern die Möglichkeit, das Erreichte im direkten internationalen Vergleich zu testen und so gemeinsam Fortschritte in den jeweiligen Fachdisziplinen zu erzielen.

Gespielt wird dabei in mehreren Ligen, bei denen es verschiedene Regeln für die Größe der Spielfelder, Roboter und über die einzusetzenden Sensoren gibt. Auch die Anzahl der Roboter ist variabel. In den folgenden Abbildungen sind zwei Beispiele zu sehen.

[1] RoboCup ist eine internationale Initiative zur Förderung der Forschung und interdisziplinären Ausbildung in den Bereichen Künstliche Intelligenz und autonome mobile Roboter. Der Dachverband des RoboCup ist die (Federation, 2006, Robocup Federation). Sie veranstaltet seit acht Jahren jährliche RoboCup-Weltmeisterschaften.

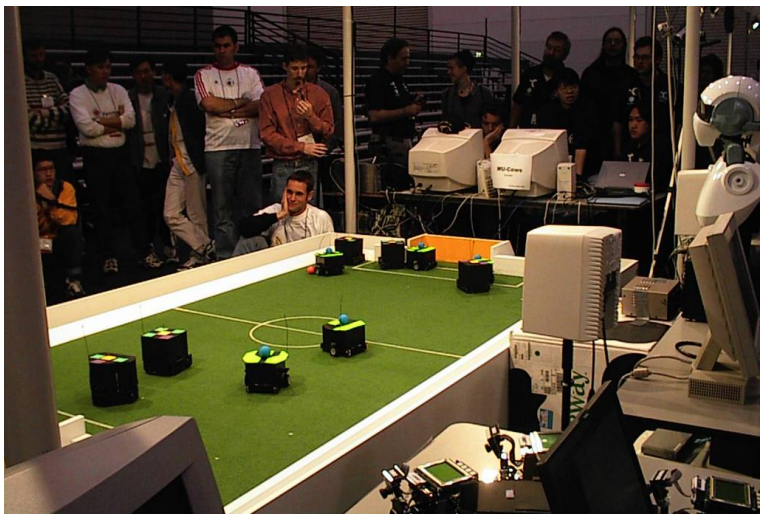

Abbildung 2.1: Spiel 4 gegen 4 mit Farbmarkierung

Hier ist ein Spiel der „Small-size League“ zu sehen, bei dem jeweils vier Roboter pro Mannschaft eingesetzt werden. Deutlich zu erkennen sind die Farbmarkierungen, die auf den Robotern angebracht wurden, um die Positionen der Roboter durch ein externes System überwachen zu können.

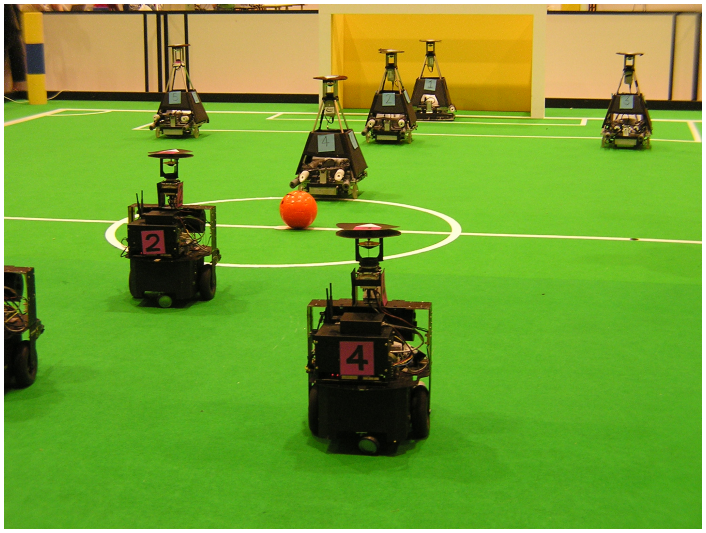

Abbildung 2.2: Spiel 4 gegen 4 mit On-Board-Sensorik

Im Gegensatz zum ersten Beispiel hier ein Spiel der „Middle-size Robot League“. Auch hier spielen jeweils vier Roboter pro Mannschaft, aber es gibt keine Farbmarkierungen. Statt dessen führen die Roboter diverse Sensoren mit sich. Sie verhalten sich also autonom.

2.1.2 Prinzipien der Sensorik

Prinzipiell sind für den Einsatz von Sensoren für die Überwachung von Objekten mehrere Perspektiven denkbar. Zum einen kann man die Objekte selbst mit Sensoren ausrüsten, dieses bezeichnet man im Bereich der Robotik als „On-Board-Sensorik“. Zu diesem Szenario gibt es im Robot-Labor der HAW-Hamburg bereits viele Arbeiten[2]. Die „On-Board-Sensorik“ hat aber den Nachteil, dass bei dieser Perspektive das „Blickfeld“ der Sensoren häufig durch andere Objekte oder durch beschränkte Sichtwinkel oder Reichweiten eingeschränkt wird. Im speziellen Anwendungsfall „Robot-Fußball“ könnte z.B. der Ball oder die Tore durch den gegnerischen Roboter verdeckt sein.

Abbildung 2.3: On-Board-Perspektive

Eine Alternative zur „On-Board-Sensorik“ ist die „Global-Map-Sensorik“ bei der die Objekte nicht die Sensoren mitführen, sondern eine externe Sensorik das gesamte Einsatzgebiet der Objekte von außen überwacht. Ein solches Szenario soll der Kamera-Service realisieren. Dazu wird eine Kamera so über die zu überwachende Fläche platziert, dass sie möglichst immer alle zu überwachenden Objekte im Blickfeld hat. Mit einer entsprechenden Bildverarbeitung werden dann aus dem Kamera-Bild die Positionen der überwachten Objekte berechnet. Der Kamera-Service soll es einem Client ermöglichen, die Positionen von einem oder mehreren Objekten beobachten zu lassen. Im Anwendungsfall „Robot-Fußball“ können

[2] Eine davon ist z.B. die Diplomarbeit von (Manger, 2004, Michael Manger), der eine Roboterplattform entwickelt hat, die über diverse On-Board-Sensoren verfügt

dann z.B. die Positionen des Balls, der Tore oder des eigenen, bzw. gegnerischen Roboters überwacht und angefordert werden.

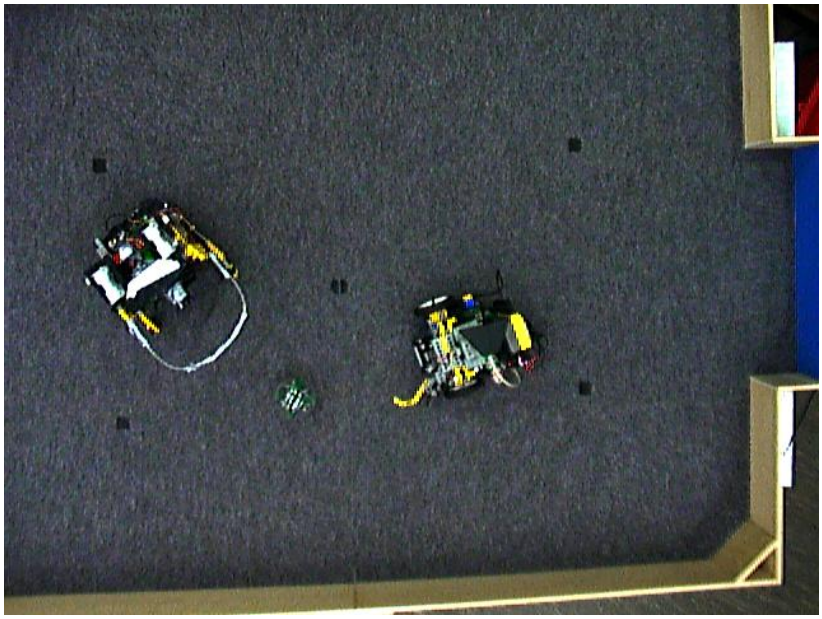

Abbildung 2.4: Global-Perspektive

2.2 Aktueller Entwicklungsstand

In meiner Studienarbeit wurde ein Prototyp entwickelt, der die grundsätzliche Funktion eines Kamera-Service für den Anwendungsfall „Robot-Fußball“ demonstriert. Das System, das in dieser Diplomarbeit entwickelt werden soll, wird auf Basis der Erkenntnisse, die durch den Prototypen gewonnen wurden, so entworfen werden, dass die festgestellten Schwachstellen behoben werden.

Beim Prototypen kommunizieren Client und Server über ein sehr einfaches, nicht standardisiertes Protokoll. Dieses soll durch ein standardisiertes Webservice Protokoll ersetzt werden. Das hat folgende Vorteile. Zum einen erhöht es die Erweiterbarkeit und zum anderen kann eine Spezifizierung der Schnittstelle mitgesendet werden.

Beim Prototypen wurde vorrausgesetzt, dass das System nur intern im Intranet des Robot-Lab eingesetzt wird. Daher enthält das System bisher keine Schutzmechanismen, die vor einem unbefugten Zugriff schützen. Es kann bisher jeder Client der Zugriff auf das Intranet hat unbeschränkt auf das System zugreifen. Da diese Arbeit aber auch exemplarisch für

Anwendungen außerhalb der HAW stehen soll, sollte jeder Client sich vor der Benutzung authentifizieren und so für ihn freigeschaltete Aufgaben autorisiert werden.

Ein weiterer Punkt ist die Einführung von Benutzerkonten. Darüber hinaus gibt es einige technische Aspekte die noch verbesserungswürdig sind. Dazu gehören das Kamerabild-Grabbing in der Bildverarbeitungssoftware und die Anbindung des Kamera-Service an die Bildverarbeitung. Auch das Zeitverhalten des Prototypen ist noch verbesserungswürdig, da bisher bei jeder Positionsabfrage, die Bildverarbeitungssoftware neu konfiguriert wurde.

2.3 Funktionalität des Systems

Ein oder mehrere Clients sollen gleichzeitig das System nutzen können. Jeder berechtigte Client, soll beim System ein oder mehrere Objekte registrieren können, deren Positionen dann laufend überwacht werden. Die Beschreibung der Objekte geschieht im einfachsten Fall durch die Farbe der Objekte. Wobei das System so aufgebaut werden sollte, dass eine Erweiterung, um alternative Beschreibungsformen, ohne großen Aufwand möglich ist. Außerdem soll jeder Client stets die Positionen seiner eigenen Objekte abfragen können. Zugriff auf ein Objekt soll grundsätzlich nur der Client haben, der dieses Objekt erstellt hat.

Die Funktionalität des Systems wird im folgendem Abschnitt in einzelne Anwendungsfälle aufgeteilt.

2.3.1 Anmeldung eines Clients am Server

Ein Client möchte auf das System zugreifen und meldet sich zunächst beim System an. Anschließend muss er sich authentifizieren. Der Server überprüft nach der Authentifizierung die Zugangsberechtigung des Clients und erteilt diesem bei positiver Überprüfung Zugang zum System.

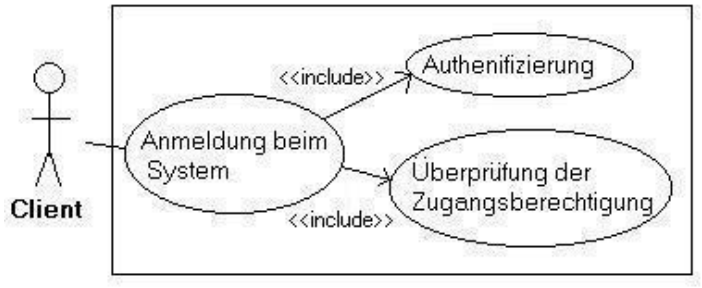

Abbildung 2.5: Anwendungsfall Anmeldung

2.3.2 Client fragt Benutzer-Konto ab

Ein Client hat sich erfolgreich beim System angemeldet und fordert beim Server eine Liste aller beobachteten Objekte an, auf die er Zugriffsrechte besitzt.

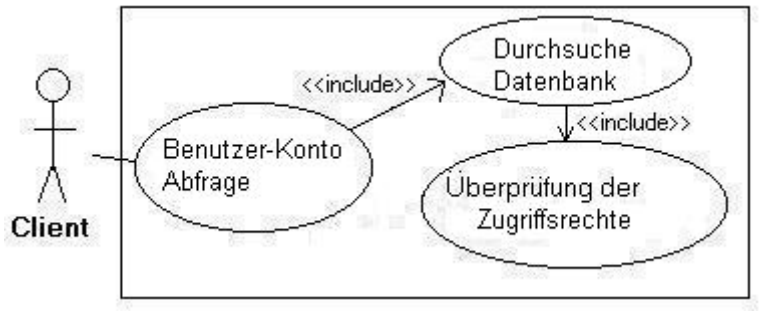

Abbildung 2.6: Client fragt Benutzer-Konto ab

2.3.3 Objekt-Modifikation

Ein Client möchte ein Objekt hinzufügen, löschen oder verändern. Dazu überprüft der Server, ob der Client die entsprechende Berechtigung hat und führt bei einer positiven Überprüfung die geforderte Modifizierung durch.

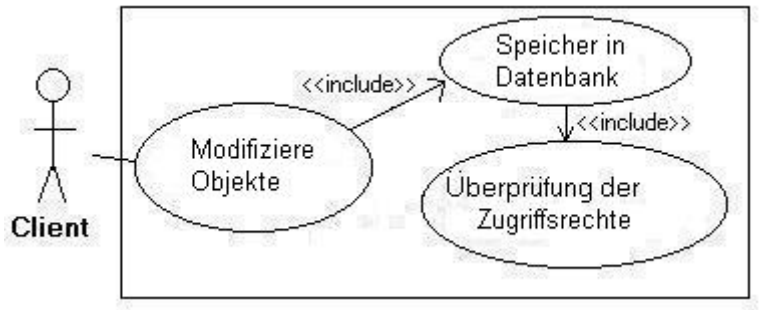

Abbildung 2.7: Anwendungsfall Objekt-Modifikation

2.3.4 Abfrage Objekt-Positionen

Ein Client fordert von allen Objekten auf die er Zugriff hat die aktuellen Positionsdaten an. Der Server überprüft die Berechtigung und sendet bei erfolgreicher Überprüfung die aktuellen Positionen zurück.

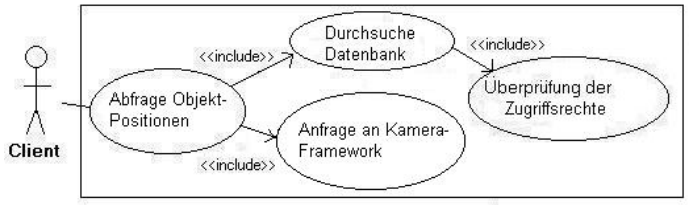

Abbildung 2.8: Anwendungsfall Abfrage Objekt-Positionen

2.4 Sicherheits-Aspekte

In diesem Abschnitt wird zunächst gezeigt, welche Aspekte im Bereich IT-Sicherheit zu beachten sind. Anschließend wird diskutiert, welche Rolle die einzelnen Aspekte beim Kamera-Service spielen.

2.4.1 Allgemeine Schutzziele der IT-Sicherheit

Im Bereich IT-Sicherheit gibt es vier klassische Schutzziele, die bei sicherheitskritischen Anwendungen eine Rolle spielen. Diese sind Vertraulichkeit, Integrität, Verbindlichkeit, und Verfügbarkeit. Das Schutzziel Vertraulichkeit soll sicherstellen, dass ein Unbefugter keinen Zugriff auf Daten oder Geräte erhalten kann, die schützenswert sind. Die Integrität soll sicherstellen, dass die Daten nicht unbefugt manipuliert oder gelöscht werden können. Der Aspekt Verbindlichkeit berücksichtigt Probleme, die vor allem für Rechtssicherheit wichtig sind. Dieses sind Unabstreitbarkeit, Abrechnungssicherheit und Übertragungssicherheit von Transaktionen. Der Teilbereich Verfügbarkeit beschäftigt sich mit allen Maßnahmen, die die ständige Verfügbarkeit des IT-Systems sicherstellen sollen. Dieses umfasst sowohl den Schutz vor beabsichtigten Angriffen auf die Verfügbarkeit, als auch die Funktionssicherheit des Systems an sich.

2.4.2 Schutzziel „Vertraulichkeit und Integrität“

Die Vertraulichkeit spielt in den meisten Anwendungsfällen eine große Rolle. Für den Kamera-Service im speziellen Anwendungsfall Robot-Fußball spielt die Vertraulichkeit die Rolle, dass bestimmte Positions-Daten für gegnerische Parteien unzugänglich sein sollen. Es sind auch andere Anwendungen für den Kamera-Service denkbar, wie z.B. das Beobachten von Personen. Für einen solchen Fall, wäre die Vertraulichkeit unbedingt sicherzustellen. Ebenso verhält es sich mit der Integrität. Man stelle sich vor, es wird ein technischer Prozess überwacht, bei dem in bestimmten Situationen entsprechende Aktionen gestartet werden sollen. Da diese Arbeit exemplarisch für diverse Anwendungsfälle stehen soll, werden die entsprechenden Maßnahmen in das System integriert.

Um die Vertraulichkeit sicher zu stellen, müssen zwei grundsätzliche Voraussetzungen erfüllt werden. Zum einen darf niemand, der nicht dazu berechtigt ist, Zugang zum System bekommen. Es muss also eine Zugangskontrolle installiert werden, die die Zugangsberechtigung eines Benutzers überprüft. Zum anderen darf ein potenzieller Angreifer keine Informationen über sicherheitskritische Daten oder Zugangsdaten erhalten können, indem er mitlauscht. Dieses verhindert man durch eine ausreichend sichere Verschlüsselung. Darüber hinaus ist es bei den meisten Anwendungen erforderlich die im System vorhandenen Daten oder Geräte mit entsprechenden Zugriffsrechten zu versehen, so dass nicht jeder Benutzer auf alle im System vorhandenen Daten und Geräte zugreifen kann, sondern nur auf die, die für ihn freigegeben sind.

Zur Sicherstellung der Integrität muss sichergestellt werden, dass die Daten weder im System, noch auf dem Kommunikationskanal unbemerkt manipuliert werden können.

2.4.3 Schutzziel „Verbindlichkeit“

Dieses Schutzziel spielt im Anwendungsfall Kamera-Service eine untergeordnete Rolle. Die Verbindlichkeit ist dann wichtig, wenn eine Transaktion rechtliche Konsequenzen nach sich zieht, wie z.B. ein Vertragsabschluss. Dieses ist beim Kamera-Service eher nicht der Fall, es sei denn man würde ihn z.B. als kostenpflichtigen Service anbieten.

2.4.4 Schutzziel „Verfügbarkeit“

Unter dem Begriff „Verfügbarkeit“ im Sinne der IT-Sicherheit verbergen sich verschiedene Aspekte. Einmal ist damit gemeint, dass das System vor bösartigen Angriffen auf die Funktionalität des Systems zu schützen ist. Dazu sind diverse Angriffsmöglichkeiten auszuschließen. Verbreitet sind in diesem Zusammenhang sogenannte „denial of service“-Angriffe. Hier-

bei handelt es sich um eine Überlastung eines Servers durch sehr viele gleichzeitig ankommende Anfragen von Clients. Ein wirklich wirksamer Schutz gegen solche Angriffe gibt es nicht. Es sollten jedoch einige Maßnahmen ergriffen werden, um solche Angriffe zu mindest zu erschweren. Dazu gehören z.B. Implementierungen von Timeouts, falls ein Client nicht mehr antwortet, oder das Sperren bestimmter IP-Adressen, wenn diese sich auffällig verhalten haben. Ein anderer Aspekt der auch unter den Begriff „Verfügbarkeit“ fällt, ist die Sicherheit vor technischen Problemen. Maßnahmen die in diesem Zusammenhang zu treffen sind, sind z.B. regelmäßige Datensicherung oder Vermeidung von „Single Points of Failure“ durch redundante Systeme. Bei der Entscheidung über solche Maßnahmen ist immer eine Risiko und Schadensanalyse zu erstellen. Desto höher das Risiko eines Fehlers und desto höher der mögliche Schaden, desto größer sollten die Sicherheitsmaßnahmen in diesem Punkt gestaltet werden. Beim Kamera-Service kommt es also wieder auf das jeweilige Einsatzgebiet an.

2.5 Zeitverhalten

In diesem Abschnitt wird untersucht, welches Zeitverhalten der Kamera-Service haben sollte, um für einen Einsatz im Anwendungsfall „Robot-Fußball“ geeignet zu sein. Außerdem wird gezeigt, welche Aufgaben des Systems besonders Zeitkritisch sind.

Beim Robot-Fußball bewegen sich die Roboter auf einer Fläche von 87x119 cm mit einer Geschwindigkeit von ca. 30 cm/s. Der Ball bewegt sich nicht wesentlich schneller. Bei der Größe der Roboter von ca. 15x15 cm und dem Durchmesser des Balles von ca. 8 cm wäre es sinnvoll, wenn man die Position eines Objektes auf 2-3 cm genau abfragen könnte. Daraus ergibt sich, dass beim Abfragen der Positionen eine Antwortzeit von ca. 100 ms erforderlich wäre, um die gewünschte Genauigkeit zu erreichen.

Die Antwortzeit wird bei diesem System von zwei Komponenten abhängig sein. Zum einen stellt der Kommunikationsweg zwischen Client und Kamera-Server eine Zeitverzögerung da. Vor allem, wenn man zur Kommunikation wegen Sicherheitsaspekten und Kompatibilität ein Protokoll verwendet, welches zu den eigentlichen Daten noch einige zusätzliche Daten überträgt. Zum anderen wird die Bilderkennungs-Software eine gewisse Zeit benötigen, um das Bild von der Kamera zu erhalten und auszuwerten. Die Zeiten, die der Kamera-Service zur Verwaltung der Objekte und der Benutzerkonten benötigt werden im Vergleich zu den beiden aufgezeigten zeitkritischen Vorgängen zu vernachlässigen sein.

Die Antwortzeiten in einem kabellosen Netzwerk liegen im Bereich von einigen Millisekunden. Auch die Übertragungsraten solcher Netzwerke von einigen Megabit pro Sekunde lassen den Einsatz von Protokollen mit größeren Datenmengen zu, ohne dabei für diese Anwendung ein zeitliches Problem zu verursachen.

Die Bilderkennungssoftware ist dagegen wesentlich zeitkritischer. Ilia Revout hat in seiner Diplomarbeit (Revout, 2003, Diplomarbeit Revout) mit seinem Kamera-Framework Bearbeitungszeiten von 0,26 s bei mittlerer Auflösung und der Kombination von sieben Bildfiltern ermittelt. Eine solche Zeit wäre also schon sehr kritisch, da sie eine Objektposition schon ca. 10 cm verfälschen könnte.

Eine gewonnene Erkenntnis aus diesem Abschnitt ist, dass der Kommunikationsweg aus zeitlicher Sicht weniger Problematisch ist, als die Bildauswertung. Zu viele registrierte Objekte können also zu unbrauchbaren Ergebnissen führen. Würde der Anwendungsfall viele mögliche Objekte erfordern, müsste die Bildauswertung auf mehrere PCs verteilt werden. Eine weitere Erkenntnis ist, dass sich eine höhere Auflösung des Kamera-Bildes auf die Auswertungszeit und somit negativ auf die Genauigkeit der zu ermittelnden Position auswirkt. Eine zu geringe Auflösung bringt aber ebenso eine Ungenauigkeit. Hier ist also jeweils ein Kompromiss zu suchen.

3 Design und Realisierung

Nachdem im vorherigen Kapitel die Anforderungen an das System festgelegt worden sind, wird in diesem Kapitel das System entworfen und die Technologien diskutiert, die im System eingesetzt werden. Dazu wird das Gesamtsystem zunächst relativ grob in einzelne Komponenten zerlegt, um eine Gesamtübersicht zu erhalten. Diese Komponenten werden in den darauf folgenden Abschnitten weiter zerlegt, so dass am Ende viele kleine Komponenten mit überschaubaren Aufgaben entstehen, die zusammen das komplexe Gesamtsystem bilden. Auf diese Weise können einzelne Komponenten eventuell auch für andere Aufgaben eingesetzt oder durch alternative Komponenten ersetzt werden.

3.1 Überblick auf das System

In der folgenden Grafik sieht man die groben Komponenten des Systems, die sich aus der Analyse ergeben haben. Wir haben einen mobilen Client, der über einen kabellosen Kommunikationskanal mit einer Komponente verbunden ist, die für die Verwaltung aller Objekte zuständig ist, die im Auftrag der Clients beobachtet werden sollen. Diese Komponente soll den Zugriff von Clients auf die Objekte und deren Positionsdaten überwachen. Außerdem soll diese Komponente über eine geeignete Schnittstelle mit einer Bilderkennungssoftware verbunden sein, um so ständig die Positionen der gespeicherten Objekte aktualisieren zu können. Um die geforderten Sicherheitsanforderungen erfüllen zu können ist ein Sicherheitskonzept erforderlich, welches sowohl die über den Kommunikationskanal übertragenen, sicherheitskritischen Daten verschlüsselt, als auch eine Zugriffskontrolle auf das System und die gespeicherten Daten herstellt.

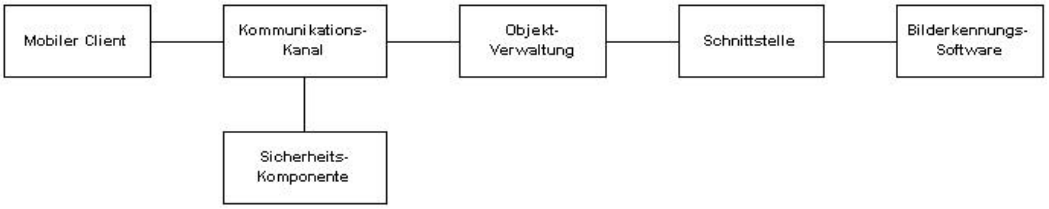

Abbildung 3.1: System-Überblick

3.2 Zusammenwirkung der Komponenten

In dem folgendem Sequenzdiagramm wird dargestellt, wie die einzelnen Komponenten des Systems zusammenwirken sollen. Dazu wird hier exemplarisch der Anwendungsfall „Client fordert Objekt-Positionen an“ dargestellt. Eine mobile Client-Anwendung stellt über ein Webservice-Protokoll eine Positions-Anfrage an die Objektverwaltung und übergibt dabei die eigene User-Identität und ein Sicherheits-Token, mit denen der Benutzer sich authentifiziert. Diese Anforderung wird vom WS-Server der Objektverwaltung bearbeitet. Dazu wird zunächst die Authentizität des Benutzers mit Hilfe einer Sicherheitskomponente überprüft, die im Abschnitt 3.8 vorgestellt wird. Bei einer negativen Überprüfung wird nur eine entsprechende Fehlermeldung zurückgesendet. Bei einer positiven Überprüfung werden die Benutzer-Rechte überprüft und die Anforderung an das „DB-Interface“ weitergegeben. Dort werden aus der Objekt-Datenbank, alle Objekte, auf die der entsprechende User zugreifen darf, mit deren zuletzt aktualisierter Position gelesen und über den WS-Server an den Client zurückgesendet.

Nebenläufig zu diesem Abfragemechanismus arbeitet der „Position-Watcher“. Dieser fragt laufend die Objekt-Beschreibungen aller in der Objekt-Datenbank gespeicherten Objekte ab und ermittelt mit Hilfe des Kamera-Frameworks die aktuellen Positionen. Diese werden dann anschließend in der Objekt-Datenbank aktualisiert.

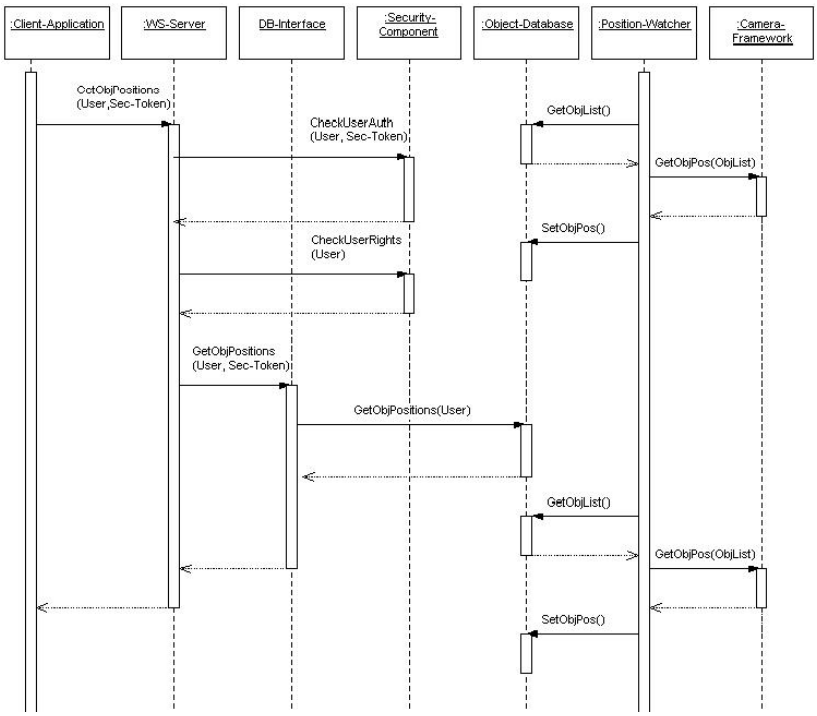

Abbildung 3.2: Zusammenwirkung der Komponenten

3.3 Kommunikationskanal

Die in Kapitel 2 herausgearbeiteten Anforderungen an die Kommunikation zwischen Client und Server waren eine kabellose, abhörsichere, plattformunabhängige und zeitnahe Verbindung (Antwortzeit < 0,1s).

3.3.1 Hardware

Die heutzutage standardmäßig genutzten kabellosen Netzwerkverbindungen sind WLan und Bluetooth. Beide erfüllen vom Zeitverhalten her den geforderten Ansprüchen. Die gestellten Sicherheitsanforderungen werden bei beiden Technologien nicht automatisch gewährleistet, sondern müssen durch entsprechende zusätzliche Protokolle auf der Software-Ebene erreicht werden. Prinzipiell eignen sich also beide Technologien.

3.3.2 Technologie-Auswahl

Das Übertragungs-Protokoll sollte eine möglichst lose Kopplung zwischen Client und dem Kamera-Server darstellen. Da im Robot-Labor oder in einer anderen späteren Anwendung des Kamera-Services die verschiedensten Clients auf allen möglichen Plattformen denkbar sind, sollte das Übertragungs-Protokoll eine möglichst große Unabhängigkeit von eingesetzter Hard- und Software bieten.

Es gibt diverse Middleware-Konzepte, die eine Kopplung zwischen verteilten Software-Komponenten herstellen. Zu nennen wären hier z.B. CORBA, JINI und DCOM. Jini ist eine Middleware, die auf Java aufbaut. Java gewährleistet zwar eine Plattformunabhängigkeit, trotzdem ist man hier aber zumindest auf eine Programmiersprache festgelegt. DCOM hingegen ist eine Entwicklung von Microsoft und wird nicht von allen Plattformen unterstützt. CORBA ist zwar vom Prinzip her Plattform-Unabhängig und unterstützt auch die meisten Programmiersprachen, hat aber den Nachteil, dass der Einsatz dieser Technik einen verhältnismäßig hohen Wartungsaufwand erfordert und das ein offener Port erforderlich ist. Dieses stellt natürlich ein Sicherheitsrisiko, weshalb der Einsatz von CORBA über eine Firewall nicht funktioniert. Hierzu gibt es auch einen ausführlichen Vergleich zwischen den Konzepten CORBA, DCOM und JavaRMI von (Raj, 1998, Gopalan Suresh Raj)

Ein relativ neuer Standard zum Verbinden von verteilten Software-Komponenten sind die XML-Webservices. Dieser Standard wurde entwickelt, um die Nachteile der zuvor vorhandenen Middleware-Konzepte auszuschalten. Die Webservice-Protokolle sind gültige XML-Dokumente und werden über das HTTP-Protokoll übertragen. Es werden also im Gegensatz

zu den zuvor beschriebenen Ansätzen nur standardisierte und offenliegende Protokolle verwendet.

Bei einem Webservice unterscheidet man zwei Typen von Protokollen. Zum einen gibt es das „SOAP-Protokoll“[1], welches zur Übertragung von Nachrichten zwischen Kommunikationspartnern dient. Das SOAP-Protokoll besteht aus zwei Subelementen, dem SOAP-Header und dem SOAP-Body. Der Header ist in der Spezifikation von Webservices optional für die Übertragung von Zusatzdaten, wie z.B. Benutzerinformationen oder Sicherheitselementen vorgesehen. Der SOAP-Body ist dagegen laut Spezifikation zwingend erforderlich und enthält einen speziellen Service-Aufruf. Ursprünglich wurde SOAP für Ein-Wege-Nachrichten konzipiert. In der Praxis wird SOAP jedoch meistens nach dem Muster von RPC-Aufrufen verwendet, so dass eine Nachricht zum Aufruf einer entfernten Methode versendet und vom Web Service eine Antwort-Nachricht zurückgesendet wird. Auf diese Weise realisiert man mit einem Webservice eine übliche Zwei-Wege-Kommunikation. Anzumerken ist das Webservices keine Objektreferenzen auflösen, so wie es verteilte Middleware-Architektur (z.B. CORBA, DCOM) tun. Die Bindung an bestimmte Ziel-Objekte, muss also entweder vom Entwickler oder der eingesetzten Entwicklungsumgebung vorgenommen werden.

Ein weiteres Protokoll, dass eine wichtige Rolle bei der Webservice-Technologie spielt, ist das „WSDL-Protokoll“[2]. Dieses liefert einem anfragenden Client wichtige Metadaten. Es liefert z.B. Informationen darüber, wie der SOAP-Body aufgebaut sein muss und welche Parameter in welcher Reihenfolge übergeben werden müssen.

Auch wenn mit dem Einsatz der Webservice-Technologie der Nachteil der fehlenden Objektauflösung entsteht, überwiegen die entstehenden Vorteile. Zum einen die völlige Unabhängigkeit von Plattformen und Programmiersprachen, die durch die Verwendung des XML-Formates zur Datenübertragung entstehen. Zum anderen bringt die Nutzung des HTTP-Protokolls keine Probleme, wenn die Komponenten über Firewall-Grenzen hinaus verteilt sind. Darüber hinaus bieten Webservice den Vorteil, dass es Zusatzprotokolle gibt, die für die Sicherheit der Daten sorgen können.

Eine ausführlichere Einführung in Webservices und einen detaillierteren Vergleich mit CORBA und JavaRMI kann hier nachgelesen werden(Okutan und Schnaufer, 2004).

3.3.3 Realisierung des Kommunikationskanals

Um die Kommunikation auf Basis des Webservice-Protokolls zu realisieren, bietet die Entwicklungsumgebung „MS Visual Studio 2005“ dem Entwickler eine gute Unterstützung. Sowohl das WDSL-Protokoll, als auch das SOAP-Protokoll werden von der Entwicklungsum-

[1] SOAP ist die Abbkürzung von „Simple Object Access Protocols“
[2] WSDL steht für „Web Service Description Language“

gebung automatisch generiert. Auch die Zwei-Wege-Kommunikation wird durch die Umgebung realisiert. Der Entwickler kann den Web-Service, wie einen statischen Methoden-Aufruf betrachten. Allerdings erfolgt die Übertragung der Daten standardmäßig im Klartext, daher müssen zusätzliche Maßnahmen ergriffen werden, um die Datensicherheit zu gewährleisten. Für die Realisierung der Sicherheits-Protokolle stehen zusätzliche Komponenten zur Verfügung, die im Abschnitt 3.8.4 vorgestellt werden.

3.4 Mobiler Client

3.4.1 Design

Der mobile Client besteht aus einer Schnittstelle, über den ein Robot auf einen Objekt-Buffer zugreifen kann. In diesem werden die Objekt-Positionen, die vom Kamera-Service angefordert wurden zwischengespeichert. So kann eine Robotersteuerung jeweils die zuletzt übermittelten Positionen abfragen, ohne durch eine zeitliche Verzögerung blockiert zu werden. Der „Object-Buffer“ fragt regelmäßig die Positionen der gespeicherten Objekte beim Kamera-Service ab. Dieses sollte dann in einem eigenen Thread ablaufen, damit die Objekt-Positionen jederzeit über die Robot-Schnittstelle abgefragt werden können. Zur Kommunikation mit dem Kamera-Service dient die Klasse „WS-Client“.

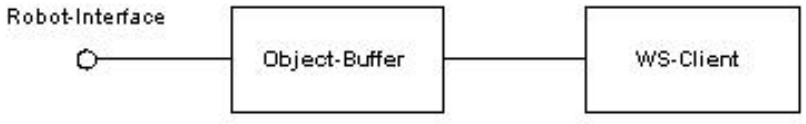

Abbildung 3.3: Entwurf des mobilen Clients

3.4.2 Hardware

Als mobiles Gerät steht im Robot-Labor ein „Pocket PC 2003“ zur Verfügung. Der „Pocket PC 2003“ verfügt sowohl über WLan, als auch über Bluetooth. Eine Schnittstelle zwischen dem Pocket-PC und dem im Robot-Labor als Roboter-Steuerung eingesetztem 6.270-Board vom MIT ist bereits entwickelt. Damit lässt sich die Kommunikation zwischen einer Roboter-Steuerung und dem Kamera-Service über den Pocket-PC kabellos realisieren.

Abbildung 3.4: Pocket-PC 2003

3.4.3 Software

Auf dem „Pocket PC 2003“ ist standardmäßig das Betriebssystem "MS Windows CE 4.2"vorhanden. Dieses erlaubt den Einsatz des „MS.NET-Frameworks“ in einer kompakten Version. Aber auch eine Java-VM ist für dieses Betriebssystem vorhanden. In beiden Programmierumgebungen sind gute Unterstützungen für Webservices vorhanden, so dass beide ohne Probleme eingesetzt werden können.

3.4.4 Realisierung des mobilen Clients

Das Design des mobilen Clients hat ergeben, dass die Klasse „Object-Buffer“ über die Klasse „WS-Client“ ständig die aktuellen Positionen der relevanten Objekte abfragt. Da das Ziel dieser Arbeit die Realisierung des Kamera-Services ist, wird die Roboter-Schnittstelle hier nicht realisiert. Wie man diese Anbindung realisiert, zeigt (Gerling, 2003, Mirco Gerling) in seiner Diplomarbeit.

Daher beschränkt sich die Realisierung des mobilen Clients an dieser Stelle auf eine Benutzeroberfläche, die mit Hilfe der Klasse „WS-Client“ alle Funktionen des Kamera-Service demonstrieren und testen kann. Dazu gehören die Authentifikation beim System, die Konfiguration der Objekte, sowie die Abfrage und Anzeige der Objekt-Positionen.

Diese Test-Umgebung könnte später bei einem realen Einsatz des Systems, zur Konfigurierung der Objekte weiter verwendet werden.

3.5 Objekt-Verwaltung

3.5.1 Design

Die Objekt-Verwaltung ist die zentrale Komponente des Kamera-Service. Diese verfügt über eine Webservice-Server-Komponente als Schnittstelle zum Mobilen Client. Diese Komponente entpackt die Anforderung des Clients und überprüft die Authenzität und die Rechte des Users. Anschließend gibt sie die Anforderung an das „DB-Interface“ weiter. Ein Benutzer kann alle Objekte in der Objekt-Datenbank, für die er Zugriffsrechte hat, abfragen oder editieren.

Die Komponente Position-Watcher hat die Aufgabe, ständig die Positionen aller in der Objekt-Datenbank gespeicherten Objekte zu aktualisieren. Dazu greift er über eine geeignete Schnittstelle auf das Kamera-Framework zu. Dieses wiederum kommuniziert mit der Kamera und filtert aus dem Kamera-Bild die Positionen der Objekte heraus.

Mit dieser Architektur kann ein Client niemals direkt auf das Kamera-Framework zugreifen. Es muss also nur sichergestellt werden, dass der Access-Controler nur berechtigte User auf die entsprechenden Objekte zugreifen lässt. Auf diese Weise erreicht man ein überschaubares Sicherheitskonzept. Außerdem sind die Vorgänge Client-Zugriff und Objekt-Überwachung logisch getrennt.

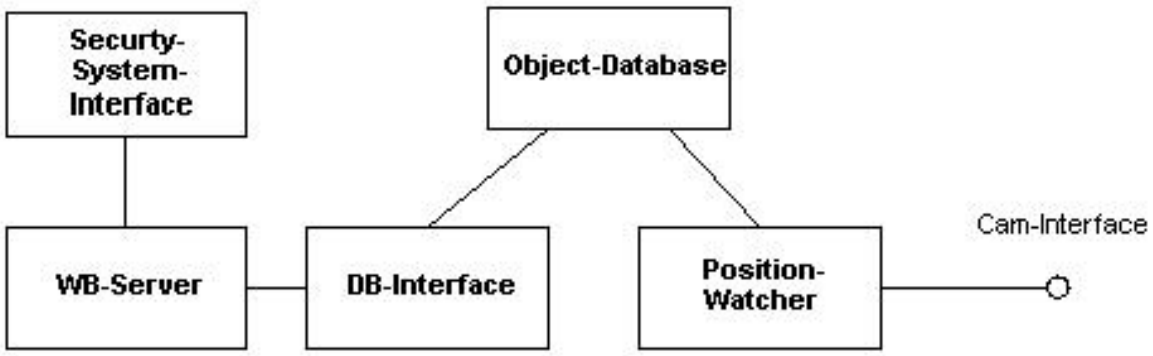

Abbildung 3.5: Entwurf der Objekt-Verwaltung

3.5.2 Object-Database

Der zentrale Bestandteil der Objekt-Verwaltung ist also die Objekt-Datenbank, auf die sowohl die Clients über die Benutzer-Schnittstelle, als auch der Position-Watcher zugreifen.

Die Datenbank sollte so gestaltet werden, dass das System jederzeit um weitere Möglichkeiten zur Objekt-Beschreibung erweitert werden kann, ohne das dazu die Struktur der Datenbank modifiziert werden muss. Dazu wird jedes Objekt durch einen Objekt-Typ und einen Parameter-String beschrieben. Daher gibt es eine Tabelle „Objekt-Types“ in der die verschiedenen Beschreibungstypen gespeichert werden. Zum Beispiel könnte man die Objekt-Beschreibungstypen „RGB-Format“ und „HSV-Format“ in dieser Tabelle speichern. Die Deutung der einzelnen Parameter wird erst im Position-Watcher vorgenommen, so dass bei einer Erweiterung der Objekt-Beschreibungs-Typen hier lediglich ein weiterer Eintrag in die Tabelle „Objekt-Types“ eingefügt werden muss.

Da jeder normale Benutzer nur Zugriff auf seine eigenen Objekte haben soll, muss jedem Objekt genau ein Benutzer zugeordnet werden. Dafür gibt es eine Tabelle „Users“ in der alle Benutzer eingetragen werden, die Objekte in der Datenbank anlegen.

Die Tabelle „Objects“ enthält alle Objekte, die beobachtet werden sollen. Jedem Objekt wird ein Benutzer, ein Objekt-Typ sowie ein Parameter-String zugeordnet. Diese drei Attribute bilden zusammen den Primary-Key der Tabelle. So ist es ausgeschlossen, dass ein Benutzer zweimal das gleiche Objekt speichert. Es können jedoch mehrere Benutzer das gleiche Objekt speichern. Dieses ist sinnvoll, da jeder Benutzer nur auf seine eigenen Objekte zugreifen kann. Es ist Aufgabe des Position-Watchers doppelt vorhandene Objekte zu erkennen und nur einmal an die Bildverarbeitung weiter zu leiten. Außerdem enthält diese Tabelle noch die Attribute „X-Pos“ und „Y-Pos“, damit der Position-Watcher die ermittelten Objekt-Positionen in die Datenbank schreiben kann. Das Attribut „Last-Update“ gibt an, wann die Position das letzte mal aktualisiert wurde. So ist es möglich, dass ein Benutzer erkennen kann, wie aktuell die Positionsdaten sind. Mit Hilfe des Attributs „Last-Request“ kann der Position-Watcher erkennen, wann der entsprechende Benutzer zuletzt auf das Objekt zugegriffen hat. Auf diese Weise kann das System sich auf die Beobachtung der Objekte konzentrieren, auf die innerhalb einer gewissen Zeitspanne zugegriffen worden ist. So ist ausgeschlossen, dass Objekte von inaktiven Benutzern, die Performance des Systems belasten.

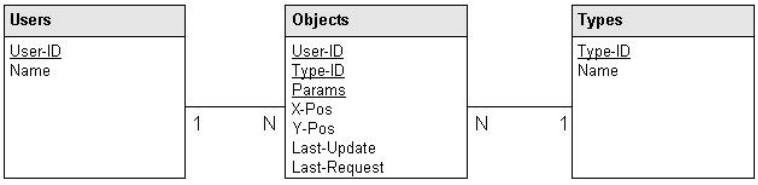

Abbildung 3.6: Struktur der Objekt-Datenbank

3.5.3 DB-Interface

Das Database-Interface ist eine Klasse, die alle Zugriffe auf die Datenbank kapseln soll. Dazu stellt sie nach außen die drei Methoden „connect“, „executeSQL“ und „disconnect“ zur Verfügung.

3.5.4 Security-System-Interface

Das Security-System-Interface stellt den Zugriff auf das Sicherheitssytem her und führt alle kryptographischen Funktionen durch, die das Sicheitskonzept 3.8 erfordert.

3.5.5 Position-Watcher

Der Position-Watcher hat zwei Aufgaben. Er muss die Bildverarbeitung mit den Daten der zu überwachenden Objekte konfigurieren und er muss die ermittelten Positionsdaten in die Datenbank schreiben.

Da die Konfigurierung der Bildverarbeitung einige Zeit in Anspruch nimmt, sollte sie nur dann erfolgen, wenn sie auch notwendig ist. Da er von der Datenbank keine direkte Benachrichtigung über eine Veränderung der Datensätze erhalten kann, müsste er eine Nachricht vom „DB-Interface“ bekommen, wenn dort eine Modifikation der Objekt-Daten vorgenommen wird. Da das „DB-Interface“ aber nicht in einem Prozess abläuft, sondern nur in einem temporären Webservice-Aufruf instanziert wird, ist eine Nachrichten-Übertragung an den Position-Watcher nur schwer zu realisieren. Alternativ kann der Position-Watcher ein Polling auf die Datenbank durchführen. Dieses nimmt zwar Prozessor-Ressourcen in Anspruch, ist aber unter Berücksichtigung der Erwartung, dass die Objekt-Daten relativ selten modifiziert werden, kein Problem. Eine Abtastrate von einer Sekunde ist völlig ausreichend, da bei jeder Veränderung des Objekt-Bestandes eine neue Konfigurierung der Bildverarbeitung vorgenommen werden muss, die ebenfalls eine Zeit benötigt, die in diesem Bereich liegt.

Die Aktualisierung der Objekt-Positionen sollte in Realzeit erfolgen, da die Objekte sich relativ schnell auf der zu beobachtenen Fläche bewegen und die Positionsdaten sonst keinen praktischen Wert mehr hätten. Daher sollte diese Aktion durch ein Event von der Bildverarbeitung ausgelöst werden. Die Schnittstelle zwischen dem „Position-Watcher“ und der Bildverarbeitung wird im Abschnitt „Schnittstelle zwischen Objekt-Verwaltung und Kamera-Framework“ erarbeitet.

Die Funktionsweise des Position-Watchers wird in der folgenden Grafik veranschaulicht.

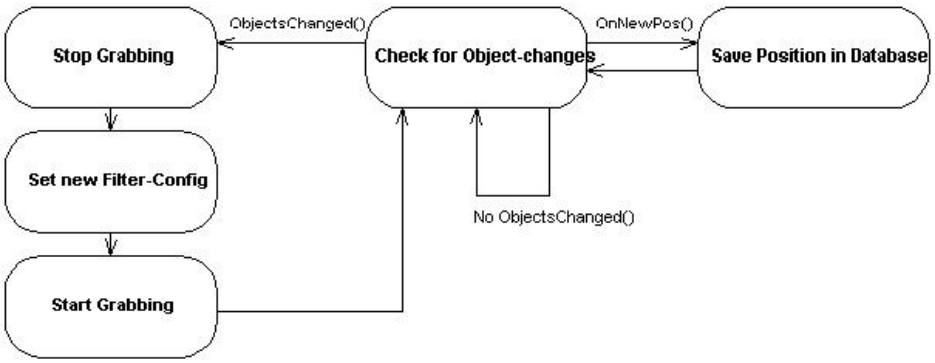

Abbildung 3.7: Funktionsweise des Position-Watchers

3.5.6 Realisierung

Die herausgearbeiteten Anforderungen an die Programmiersprache für die zentrale Objekt-Verwaltung sind Plattformunabhängigkeit, eine gute Unterstützung der Webservice-Technologie und eine einfache Datenbankanbindung.

Die Datenbankanbindung sollte mit nahezu jeder Programmierumgebung ohne größere Probleme zu realisieren sein. Die Plattformunabhängigkeit ist nur mit Umgebungen zu realisieren, die nicht direkt auf den Betriebssystem-APIs aufbauen, sondern noch eine weitere Abstraktionsebene zwischen Betriebssystem und Anwendung einsetzen. Daher kommen auch hier Java und das MS.NET-Framework in Frage, da diese durch die „Java virtuell Machine“, bzw. durch die „Common Language Runtime“ auf verschiedenen Plattformen an die jeweiligen Betriebssystem-APIs angebunden werden können. Klassen, die die Verwendung von Webservices unterstützen, sind ebenfalls in beiden Technologien vorhanden.

Entscheidet man sich für das MS.NET-Framework bietet sich der Einsatz des MS-SQL-Servers als Datenbank-System an, da sich so eine unkomplizierte Datenbank-Anbindung realisieren lässt. Wie man den Datenbank-Zugriff von einem .NET-Webservice realisieren kann beschreibt Scott Short in seinem Buch (Short, 2002, Webdienste mit dem .NET Framework entwickeln). Zum Thema Sicherheit weißt er vor allem auf drei Dinge hin. Der Zugriff auf die Verbindungs-Daten der Datenbank dürfen nur von dem Host des Webservice-Servers erreichbar sein. Dazu stellt das .NET-Framework die Methoden-Direktive „Internal“ zur Verfügung, die dieses sicherstellt. Es sollte nur ein Benutzer-Konto verwendet werden, dass nur die benötigten Berechtigungen hat, die unbedingt erforderlich sind. Außerdem warnt er vor unsicheren SQL-Zeichenfolgen. Dieses ist beim Kamera-Service ausgeschlossen, da hier die Clients nicht direkt SQL-Befehle ausführen lassen können, sondern nur vordefinierte Befehle mit eigenen Parametern aufrufen können.

3.6 Bilderkennung

3.6.1 Software

Die Anforderungen an die Bilderkennung sind eine möglichst schnelle Verarbeitung der Bilddaten zu Positionsdaten. Außerdem sollte die Position eine Genauigkeit von wenigen Zentimetern haben. Des weiteren müssen mehrere Objekte gleichzeitig beobachtet und somit aus dem Kamera-Bild extrahiert werden können. Die Bilderkennung sollte auch Objekte nach verschiedenen Beschreibungskriterien erkennen können.

In dieses Anforderungsprofil passt die Diplomarbeit von Ilia Revout(Revout, 2003, Diplomarbeit Revout) hinein. Er hat ein Framework entwickelt, mit dem man ein aktuelles Kamerabild mit Hilfe von diversen Filtern verarbeiten kann. Da die Filter untereinander kombiniert werden können, ist es z.B. möglich mehrere Farbschwerpunkt-Filter zu kombinieren, um Objekte, die durch ihre Farbe beschrieben wurden, zu beobachten. Aber auch andere Beschreibungen von Objekten, wie z.B. durch ihre Form wären mit dem Framework zu realisieren, indem man die entsprechenden Filter einsetzt.

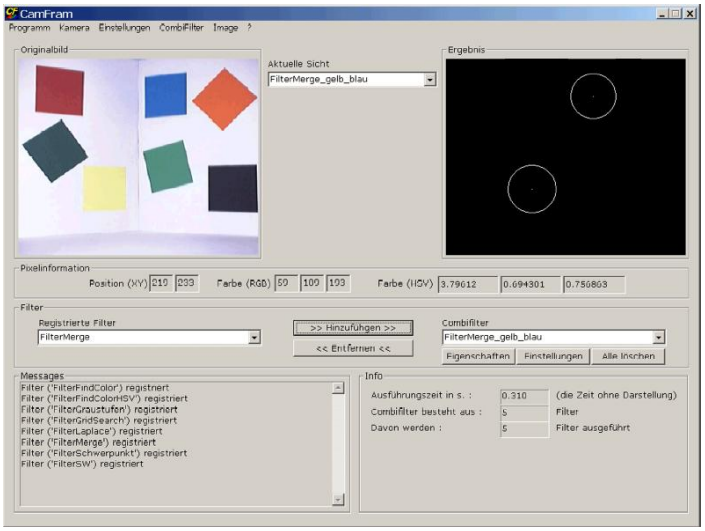

Abbildung 3.8: Kamera-Framework von Ilija Revout

Zum erforderlichen Zeitverhalten hat die Analyse gezeigt, dass die Geschwindigkeit der Bildauswertung für den Anwendungsfall Robot-Fußball gerade so ausreichen könnte, um akzeptable Anforderungszeiten zu erreichen. Bei Anwendungen, bei denen wesentlich mehr Objekte zu beobachten sind, müsste die Rechenleistung auf mehrere Rechner aufgeteilt werden. Da das Framework von Ilia Revout eine solche Verteilung nicht integriert hat, müsste dann die Objektverwaltung die Aufgabe übernehmen, mehrere Instanzen des Frameworks auf verteilten Rechnern anzusprechen.

3.6.2 Bildformat-Transformation

Da für die Kamera-Überwachung des Spielfeldes ein möglichst hochauflösendes und qualitativ hochwertiges Kamera-Bild erforderlich ist, sollte die hochwertigste im Labor vorhandene Kamera für den Kamera-Service eingesetzt werden. Dieses wäre z.B. eine Firewire-Kamera von Sony (DFW-VL 500). Diese Kamera liefert jedoch kein RGB-Format, sondern nur YUV-Formate. Das Kamera-Framework arbeitet jedoch bisher ausschließlich mit dem RGB-Format. Daher muss eine entsprechende Bildformat-Transformation durchgeführt werden.

Das YUV-Format beruht auf dem sogenannten YCbCr-Farbmodell, welches aus der analogen Farbfernsehtechnik (PAL) stammt. Hierbei werden zum einen die Helligkeit, man spricht auch von Luminanz (Y) und zum anderen der Farbanteil, welcher als Chrominanz (UV) bezeichnet wird, gespeichert. Die Chrominanz ist in zwei Komponenten aufgeteilt. Diese geben die jeweilige Differenz von Blau (U) und Rot (V) zur Luminanz an. Bei der Berechnung der Luminanz werden die einzelnen Farbwerte mit Faktoren versehen, welche die Unterschiede berücksichtigen, mit der das menschliche Auge die Farben wahrnehmen kann. Es handelt sich daher also um ein verlustbehaftetes Komprimierungsformat, ähnlich wie z.B. beim JPEG-Verfahren. Die Umrechnung vom YUV-Format ins RGB-Format erfolgt über folgende Formeln (Die Funktion „clip“ schneidet Werte die außerhalb von 0-255 liegen ab):

C = Y - 16; D = U-128; E = V - 128;

R = clip(round(1,164383 * C + 1,596027 * E));

G = clip(round(1,164383 * C - 0,391762 * D) - 0,812968 * E);

B = clip(round(1,164383 * C + 2,017232 * D));

Eine weitere Eigenschaft des YUV-Bildformats ist die Möglichkeit, die Abtastraten für die beiden Chrominanz-Kanäle niedriger zu wählen, als für den Luminanz-Kanal. Auf diese Weise werden zusätzlich Daten eingespart, ohne dass es für einen menschlichen Beobachter spürbar wird. Dieses liegt daran, dass sich bei bewegten Bildern die Helligkeitswerte nicht im gleichen Tempo ändern, wie die Farbwerte. Diese Methode bezeichnet man als „Chroma Subsampling“. Die Kamera „Sony DFW-VL 500“ liefert folgende drei Abtast-Verhältnisse: „4-4-4“ (Luminanz:Chrominanz = 1:1), „4:2:2“ (Luminanz:Chrominanz = 2:1) und „4:1:1“ (Luminanz:Chrominanz = 4:1). Wobei nur die Formate „YUV411“ und „YUV422“ in der höchsten Auflösung von 640x480 Pixeln zur Verfügung stehen. Da beim Einsatz im Kamera-Service sowohl die Luminanz, als auch die Chrominanz eine wichtige Rolle bei der Erkennung der zu beobachtenden Objekte spielen, ist das Format „YUV422“ für diesen Einsatz das geeignetere Format. Das „Chroma Subsampling“ muss bei der Konvertierung ins RGB-Format entsprechend berücksichtigt werden.

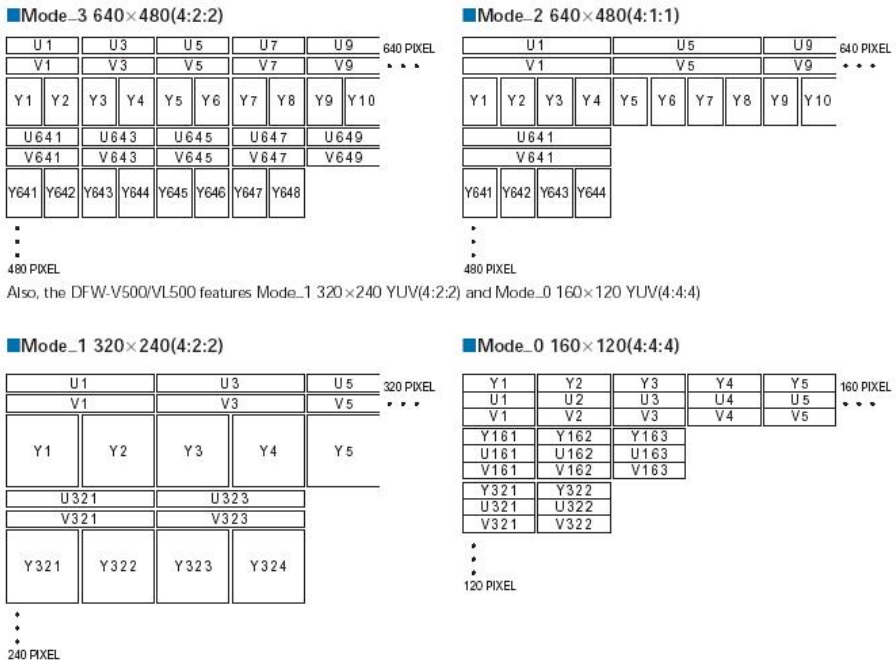

Abbildung 3.9: Formate der Kamera „Sony DFW-VL 500“

Im Kamera-Framework steht zur Interpretation der Kamera-Daten die Klasse „Image-Grabber“ zur Verfügung. Bisher ist nur eine Grabber-Klasse implementiert, die nur das RGB-Format akzeptiert. Es muss also eine neue Grabber-Klasse implementiert werden, die die Umrechnung vom „YUV-Format“ ins „RGB-Format“ vornimmt. Das Framework ist so entworfen worden, das die Grabber-Klasse in einer einzelnen DLL-Datei gekapselt ist und ein Benutzer diese Datei je nach Bedarf austauschen kann.

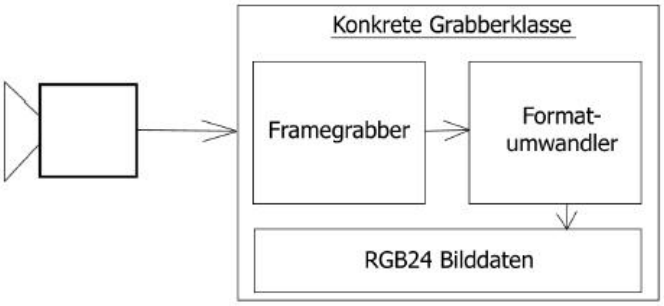

Abbildung 3.10: Grabber-Klasse

3.7 Schnittstelle zwischen Objekt-Verwaltung und Kamera-Framework

Die Anforderungen an die Schnittstelle zwischen der Objekt-Verwaltung und dem Kamera-Framework sind Abhängig davon, ob man eine oder mehrere Instanzen des Kamera-Frameworks einsetzen möchte. Bei mehreren Bilderkennungs-Instanzen sollten diese wegen der Rechenzeiten auf mehrere Rechner verteilt werden. In diesem Falle würde sich eine lose Kopplung zwischen der Objekt-Verwaltung und der Bilderkennung empfehlen, da eventuell unterschiedliche Bildverarbeitungssysteme angebunden werden könnten. Außerdem müsste in diesem Fall eine weitere Komponente eingeführt werden, die sich um die Verteilung der Bilderkennung kümmert.

Geht man aber nur von einer einzigen Bildverarbeitungseinheit aus, was für den Anwendungsfall Robot-Fußball laut Analyse ausreichen sollte, dann kann die Bindung zwischen Objekt-Verwaltung und der Bildverarbeitung auch fest gestaltet werden. Da die Rechenleistung der Objektverwaltung im Vergleich zur Bildverarbeitung relativ gering ist, können in diesem Fall beide Einheiten auf einem Rechner platziert werden. Zur Anbindung zwischen der auf MS.NET aufgebauten Objekt-Verwaltung und dem mit MFC-C++ entwickelten Kamera-Frameworks stehen folgende drei Modelle zur Verfügung, die bereits in meiner Studienarbeit (Schmidt, 2005, Studienarbeit Schmidt) in Kapitel 4.6 diskutiert wurden. Die Lösung Integration durch eine Konvertierung des MFC-Source-Codes zu .NET-Code kommt wegen eines zu hohen Programmieraufwandes nicht in Frage. Auch die Lösung durch Aufrufe der MFC-DLLs aus dem .NET-Code heraus ist nicht zu empfehlen, da nur prozedurale Aufrufe möglich sind und daher mit Wrapper-Klassen gearbeitet werden müsste, was wiederum einen hohen Aufwand erfordert. Daher kommt nur eine Lösung in Frage, bei der beide Einheiten als separate Prozesse laufen und eine Interprozesskommunikation zwischen diesen stattfindet. Bei dem verteilten Modell sollte man eine entsprechende Middleware einsetzen, bei dem einfachen Modell kann die Kommunikation auch relativ einfach über Shared-Memory'[3] oder über Windows-Messages'[4] erfolgen. Dabei haben die Messages den Vorteil, dass sie ein Event-Handling zulassen, während man bei der Shared-Memory-Lösung mit einem Polling arbeiten müsste. Daher ist die Message-Lösung besser geeignet, da sie schneller ist und Ressourcen schont.

Der Position-Watcher ist die Komponente der Objekt-Verwaltung, die die Kommunikation mit dem Kamera-Framework führt. Dazu schickt der Position-Watcher sequentiell eine Nachricht „WorkWithMe“ an das Kamera-Framework, mit der auch der Applikations-Name des

[3] Als Shared Memory bezeichnet man einen Speicher, der vom Betriebssystem verwaltet wird und mehreren Prozessen zugänglich ist. Wie man so etwas bei Windows realisiert wird hier (Todorovic, 1999) beschrieben

[4] Windows-Messages sind Nachrichten die Applikationen an andere Applikationen senden können. Wie man diese realisiert wird hier (Frenette, 1999) beschrieben

Position-Watchers übergeben wird. Wenn das Kamera-Framework nicht gestartet ist, geht die Nachricht ins Leere. Sonst empfängt ein Receiver die Nachricht und schickt ab diesem Zeitpunkt jede festgestellte Positionsänderung per Nachricht an den Position-Watcher. Stellt der Position-Watcher fest, dass der Bildfilter des Kamera-Frameworks neu gesetzt werden muss, erzeugt und speichert er die neue Filterdatei und schickt dem Kamera-Framework eine Nachricht, dass es den Filter neu laden soll. Auf diese Weise können der Position-Watcher und das Kamera-Framework unabhängig voneinander gestartet werden. Sobald beide gestartet sind, fangen sie automatisch an miteinander zu kommunizieren. In dem folgenden Sequenzdiagramm wird dieser Nachrichten-Austausch veranschaulicht.

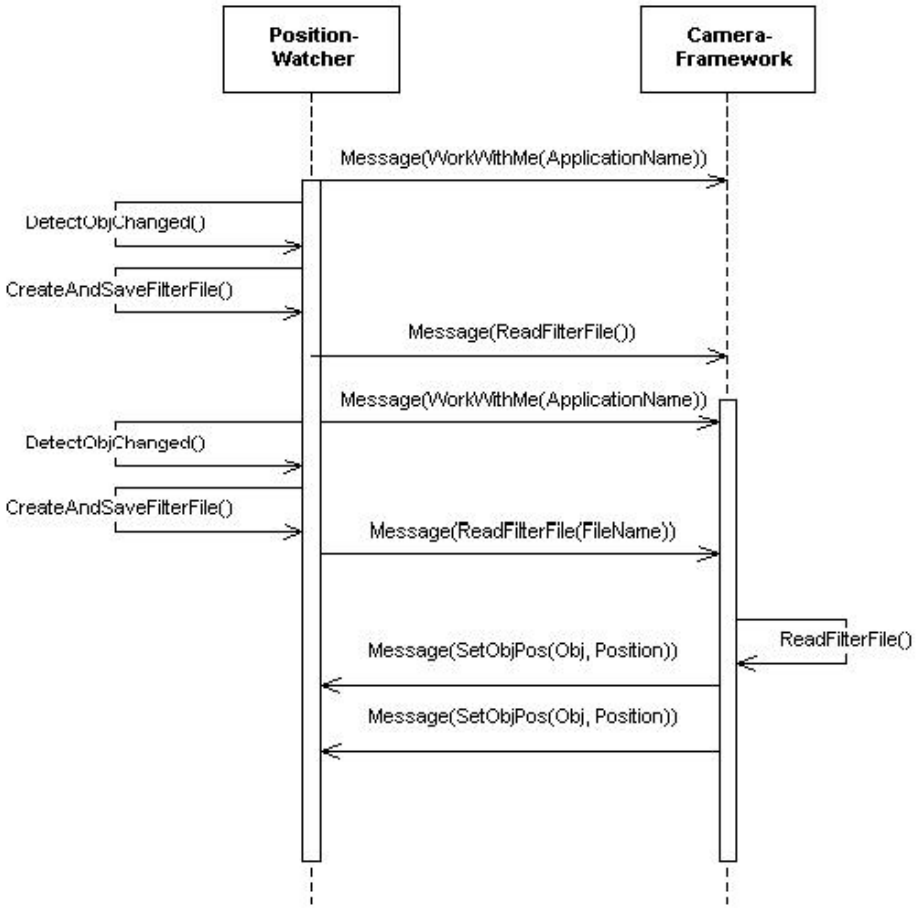

Abbildung 3.11: Kommunikation zwischen Objekt-Verwaltung und dem Kamera-Framework

3.8 Sicherheits-Konzept

3.8.1 Allgemein

Die Anforderungen an das Sicherheitskonzept, die die Analyse ergeben hat, sind Hauptsächlich die Vertraulichkeit und Integrität. Dazu müssen zwei Voraussetzungen erfüllt sein. Es dürfen nur autorisierte Personen auf das System zugreifen und dort nur Aktionen ausführen können, für die sie die Berechtigung haben. Dieses bedeutet für den Kamera-Service, dass eine Komponente benötigt wird, die eine sichere Authentifizierung der Clients vornehmen kann. Diese Komponente soll einem Client also ein Beglaubigungsschreiben ausstellen können, dass er auch wirklich derjenige ist, der er vorgibt zu sein.

Des weiteren muss sichergestellt sein, dass ein potenzieller Angreifer keine Informationen bekommen kann, mit denen er dem System eine Fremde Identität vortäuschen und so Zugang zum System erhalten kann. Zur um die Integrität zu gewährleisten, muss sichergestellt werden, dass die Daten weder im System, noch auf dem Kommunikationskanal unbemerkt manipuliert werden können. Dazu wird eine Verschlüsselungstechnik benötigt, die erstens von einem Angreifer nicht entschlüsselt und zweitens nicht durch Wiederverwendung der verschlüsselten Nachricht missbraucht werden kann. Um letzteres zu gewährleisten, muss die Verschlüsselungs-Technologie Zeit- und Transaktionsstempel verwenden.

Da die Kommunikation zwischen den Clients und dem Kamera-Service mit XML-Webservices realisiert wird, sollten die eingesetzten Sicherheitstechnologien gut in das Webservice-Protokoll integriert werden können.

Natürlich gibt es bei sicherheitskritischen Applikationen weitere Anforderungen, die über die hier behandelten Aspekte hinaus gehen. Diese Arbeit beschränkt sich aber auf diese Bereiche, da sie für den Anwendungszweck voll ausreichen. Die gesetzlichen Anforderungen, die für ein heutiges IT-System hinsichtlich der Sicherheit zu beachten sind, findet man beim (BSI, 2005, Bundesamt für Sicherheit in der Informationstechnik)

3.8.2 Technologie-Auswahl

Eine Standard-Technologie für die Zugangskontrolle zu einem verteilten System in einem offenen Netzwerk ist das Kerberos-Protokoll. Dieses funktioniert zusammengefasst nach folgendem Prinzip. Es gibt einen sogenannten Key Distribution Center (KDC), der an eine Datenbank mit allen zugangsberechtigten Clients angeschlossen ist. Bei diesem meldet sich ein Client an und bekommt ein Beglaubigungsschreiben, dass der Client auch der ist, der er vorgibt zu sein. Außerdem gibt es einen sogenannten Ticket Granting Service (TGS), bei diesem kann ein Client nach Vorlage seines Beglaubigungsschreiben, Zugangstickets für

alle Services anfordern, für die er Zugangsberechtigt ist. Der KDC und der TGS laufen üblicherweise auf dem gleichen Host. Die RFC-Spezifikation des Kerberos-Protokolls findet man hier (Kohl und Neuman, 1993, Kerberos-Spezifikation RFC 1510), eine kürzere übersichtliche Zusammenfassung gibt es von der TU Chemnitz (Mack, 2003, Kerberos-Übersicht).

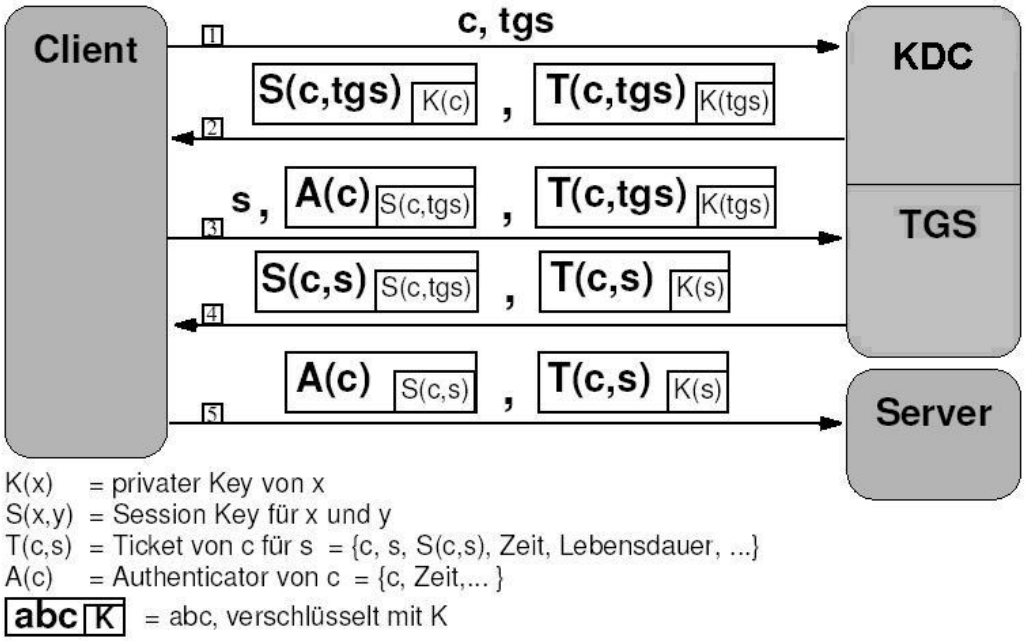

Abbildung 3.12: Kerberos-Protokoll

Das Kerberos-Protokoll ist hauptsächlich dazu entwickelt worden, um in einem offenen Netzwerk mit mehreren Servern, mit einer einzigen Anmeldung auszukommen. Beim Kamera-Service ist zwar zunächst nur ein Server vorgesehen, so dass der Einsatz einer single-sign-on-Lösung nicht unbedingt notwendig wäre und man auch eine zentrale Lösung einsetzten könnte. Der Einsatz des Kerberos-Protokolls bringt aber viele weitere Vorteile mit sich, die auch für den Kamera-Service interessant sind. Ein Vorteil ist, dass in vielen Netzwerken bereits ein Kerberos-System vorhanden ist, so dass man die Zugangskontrolle für den Kamera-Service sehr einfach in das vorhandene System integrieren kann. Auch eine Erweiterung, um weitere Server wäre damit schon vorbereitet. Desweiteren löst Kerberos viele sicherheitskritische Problemstellungen. Passwörter werden nie über das Netzwerk übertragen und können sofort nach dem Anmeldevorgang aus dem Arbeitsspeicher gelöscht werden. Sämtliche Tickets werden verschlüsselt. Die Gültigkeit der Tickets wird zeitlich beschränkt. Verwendete Sicherheitsmechanismen können zentral beeinflusst werden.

Mit Hilfe von Kerberos lässt sich also sicherstellen, dass nur berechtigte Benutzer auf den Kamera-Service zugreifen. Um aber auch sicherzustellen, dass ein Benutzer, der Zugriff auf den Kamera-Service hat, nur auf die Objekte zugreifen darf auf die er Zugriffsrechte

hat, muss die Objekt-Verwaltung zusätzlich bei jedem Objekt-Zugriff prüfen, ob der jeweilige Client zugriffsberechtigt ist.

Die Verwendung von Kerberos stellt aber auch Gefahren dar. Kerberos nimmt an, dass sichere Hosts auf einem unsicheren Netzwerk verwendet werden. Wenn jedoch ein Angreifer Zugriff auf den Rechner erlangt, auf dem der Kerberos-Service läuft, besteht die Gefahr, dass das gesamte Kerberos-System abgehört werden kann. Eine weitere Gefahr des Protokolls ist die Anfälligkeit für Replay-Angriffe. Die beiden Schutzmechanismen, die das Kerberos-Protokoll für diese Problematik anbietet, sind ein Zeitstempel'[5] und die Überprüfung der IP-Adresse des Benutzers. In dem Zeitraum bis zum Ablauf des Zeitstempels, gibt es aber keinen wirksamen Schutz, da ein Angreifer seine IP-Adresse fälschen kann. Dieses ist für den Kamera-Service, unter der Voraussetzung, dass die Daten nur verschlüsselt übertragen werden, relativ unproblematisch. Beim Abfragen der Objekte bekommt ein Angreifer nur verschlüsselte Daten zurück. Ein wiederholtes hinzufügen eines Objektes würde auch keinen Schaden verursachen, da jeder Benutzer ein bestimmtes Objekt nur einmal anlegen kann. Ein bereits gelöschtes Objekt kann nicht noch mal gelöscht werden. Darüber hinaus hat das Kerberos-Protokoll noch einige weitere Nachteile, die aber im Bezug auf den Kamera-Service nicht relevant sind und daher wird hier nur auf diese verwiesen, aber nicht weiter eingegangen. Genaueres dazu kann in Kapitel 5 der Seminararbeit von (Hoier, 2002, Niels Hoier) nachgelesen werden.

Zusätzlich zur Zugangs-Kontrolle durch Kerberos, muss die Kommunikation zwischen Client und Kamera-Service durch eine geeignete Verschlüsselung vor einem „Lausch“-Angriff geschützt werden. Dazu müssen beide Kommunikationspartner ihre Anforderungen, bzw. Antworten verschlüsseln. Zu diesem Zweck gibt es synchrone und asynchrone Verschlüsselungsverfahren. Synchrone Verfahren arbeiten schneller als asynchrone Verfahren, erfordern aber normalerweise vor der Kommunikation einen relativ komplizierten Schlüsselaustausch. Beim Kerberos-Protokoll wird standardmäßig ein synchrones Verfahren verwendet. Der Schlüsselaustausch entfällt hier, da der Client einen Sitzungsschlüssel vom TGS-Server erhält, der dem Server dann in dem verschlüsselten Server-Ticket übergeben wird.

Es gibt zwei verbreitete Algorithmen, die eine sichere synchrone Verschlüsselung gewährleisten. Diese sind „AES“ (Krautschick und Maybaum, 2005, AES) und „3DES“ (Castelino, 2003, 3DES). Beides sind Blockchiffre-Verfahren, die heute als sicher gelten. Der AES-Algorithmus arbeitet in der Software-Version wesentlich schneller als 3DES. 3DES lässt sich dafür besser in einer Hardware-Version umsetzen und hat daher auch seine Berechtigung. Für den Kamera-Service wird eine Verschlüsselung auf Software-Basis benötigt. Daher ist AES hier das effektivere Verfahren.

[5] Daher erfordert der Einsatz des Kerberos-Protokolls auch synchronisierte System-Uhren zwischen dem Kerberos-Server und den Servern.

In dem folgenden Sequenzdiagramm wird eine Client-Anforderung beim Kamera-Server aus sicherheitstechnischer Sicht dargestellt. Die eigentliche Funktionalität wird hier vereinfacht abgebildet, um das erarbeitete Sicherheitskonzept zu veranschaulichen.

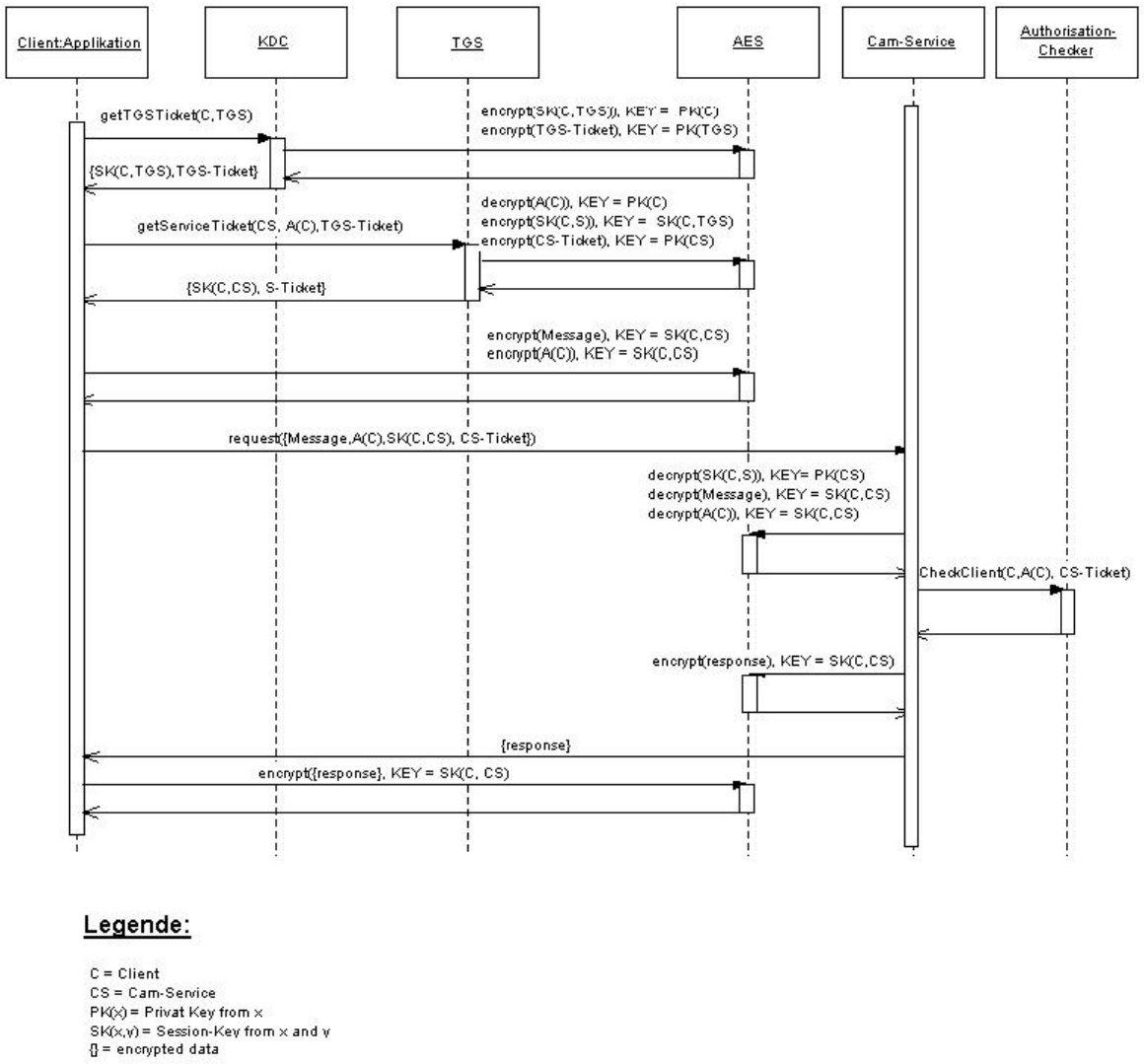

Abbildung 3.13: Sequenzdiagramm des Sicherheitskonzepts

Das Sequenzdiagramm verdeutlicht noch mal, dass der Datenaustausch zwischen Client und Kamera-Service mit diesem Konzept stets verschlüsselt abläuft. Da hier mehrere Schlüssel im System verteilt sind, soll die folgende Tabelle verdeutlichen, wem welche Schlüssel bekannt sind und wie die Schlüssel ausgetauscht werden.

Schlüssel	Art	wem bekannt	Bezugsquelle
PK(C)	private	Client und KDC	Wird aus dem Usernamen und dem Passwort generiert. Da der KDC das Passwort des Users kennt, können beide Seiten den Privat-Key des Clients berechnen. Es muss daher kein Passwort übertragen werden.
PK(TGS)	private	KDC und TGS	Erzeugt vom TGS, muss dem KDC zugänglich sein
PK(CS)	private	Cam-Service und TGS	Erzeugt vom Cam-Service, muss beim TGS eingetragen werden
SK(C, TGS)	sektion	Cam-Service und TGS	Erzeugt vom KDC, muss dem TGS zugänglich sein
SK(C, CS)	sektion	Cam-Service und Client	Erzeugt vom TGS, wird dem Client und dem Cam-Service verschlüsselt übertragen.

Tabelle 3.1: Schlüssel-Verteilung im Kerberos-System

3.8.3 Webservice-Sicherheit

Wenn man nach Sicherheitsmechanismen für Webservices recherchiert, findet man bei den meisten vorhandenen Quellen zunächst einen Hinweis auf eine SSL-Verschlüsselung. Dieses liegt wohl daran, dass SSL (Secure Socket Layer) das am weitesten verbreitete Protokoll zur Verschlüsselung von Daten im Internet ist. Zur Absicherung eines Webservice eignet sich das SSL-Protokoll aber nur bedingt. Da dieses Protokoll auf der Transport-Schicht arbeitet, ist es in der Lage, den Transportweg zwischen Absender und Empfänger abzusichern. Dieses nennt man Sicherheit einer Punkt zu Punkt Verbindung. Das zu übertragende Dokument selbst, wird dadurch aber nicht geschützt. Das SOAP-Protokoll kann nicht mit SSL digital signiert oder verschlüsselt werden, da sonst wichtige Routing-Informationen verloren gingen. Dazu kommt, dass SSL Informationen über die Identität eines Ausrufers, sowie die Gültigkeitsdauer einer Anwendersitzung nicht ausreichend ausdrücken kann. Daher ist eine Absicherung des Webservice über SSL, besonders bei einem mehrstufigen Webservice, als unsicher einzuschätzen (Ende zu Ende Verbindung). Wobei es nicht schadet SSL einzusetzen. Um die Webservice-Technologie sicher anzuwenden, sind aber weitere Schutzmaßnahmen erforderlich.

Der Standard „XML-Encryption“ wurde vom W3C entwickelt, um XML-Dokumente oder Dokumentblöcke verschlüsseln zu können. Der Standard erlaubt nicht nur das selektive Verschlüsseln einzelner Dokument-Teile, sondern lässt sich auch die Anwendung beliebiger kryptografischer Algorithmen zu. Der Empfänger einer solchen Nachricht kann anhand von

mitgelieferten Informationen erkennen, welche Teile des empfangenen Dokuments, mit welchen Algorithmen verschlüsselt wurden. Auf diese Weise kann er das Original-Dokument wieder herstellen, wenn er im Besitz eines gültigen Schlüssels ist. Mit „XML-Encription“ lässt sich also die Vertraulichkeit einer Webservice-Nachricht sicherstellen.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
  <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
     Type="http://www.w3.org/2001/04/xmlenc#Element">
     <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc">
       <IV>
          ZG9uJ3RTcHk=
       </IV>
     </EncryptionMethod>
     <CipherData>
       <CipherValue>
          sVjhJIW5QnIeY3brbpamNOcz/Ja+RmnG0pLo7vWnmTp+vpVs53c0Y
          VDeb4gmYEcOBTAe00S810cySJpKGkgbVksD9zo6U2LpS466KXIp5NDVRgJcHZnp8tr
          o5mb90g2gB56bw+IyskKh7QDMbvM7ACdT6SGauu0dSGIT3Q5kTT1qQWQ4easDZ1ShH
          pVYrBXPRI//3QGyjrnOOclh8T5eqRhRRAuwUc3A4RqaH7M6QTIb36vOTBRuFbfFESB
          w8648Xi8GlSpu39cVoMjEUTrldnJo1gpWdU5JWFf9RqzhmBQ=
       </CipherValue>
     </CipherData>
  </EncryptedData>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Abbildung 3.14: Beispiel: Verschlüsselte SOAP-Nachricht

Der Einsatz von „XML-Encryption“ stellt jedoch noch keine ausreichenden Informationen über die Identität eines Ausrufers zur Verfügung. Dafür gibt es einen weiteren vom W3C entwickelten Standard namens „XML-Signature“. Dieser definiert Regeln und die Syntax, um XML-Dokumente zu signieren. Dabei können auch einzelne Teile eines Dokumentes einzeln Signiert werden. Dieses ist eine sehr gute Eigenschaft für den Einsatz in einem Webservice, da auf diese Weise wichtige Routing-Informationen nicht zerstört werden. Ebenso wie bei „XML-Encryption“ beschreibt erlaubt auch diese Technik den Einsatz von beliebigen Signatur-Algorithmen. Auf diese Weise lassen sich Integrität und Verbindlichkeit von XML-Dokumenten sicherstellen.

Diese beiden vorgestellten XML-Sicherheit-Standards stellen die Grundlage für eine speziell für Web Services entwickelte Sicherheits-Technologie namens „Web Services Security“(WSS). WSS lässt analog zu XML-Encryption eine freie Auswahl der Algorithmen und Schlüsseltypen zu. Entwickelt wurde dieser Standard von der „Organization for the Advancement of Structured Information Standards“(OASIS). Diese Spezifikation ist von (Oasis, 2005, asis) ausgiebig beschrieben.

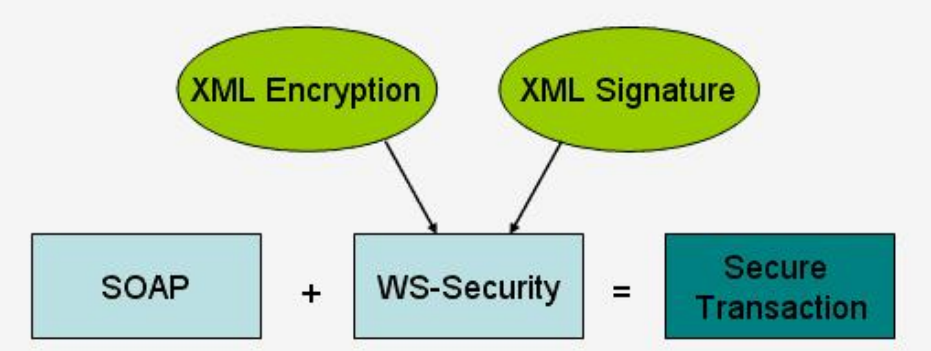

Abbildung 3.15: WS-Security

3.8.4 Webservice-Sicherheit bei .NET

Eine häufig beschriebene Möglichkeit einen Webservice auf .NET-Basis sicher zu machen, besteht darin die Schutzmechanismen des Microsoft Internet Information Services (IIS) zu nutzen. Auf diese Möglichkeit verweist z.B. „Christian Weyer“ in seinem Buch (Weyer, 2002, "'XML Web Service-Anwendungen mit MS.NET"'). Das IIS ist ein Windows-Spezifisches Konzept, welches stark mit dem Betriebssystem verwurzelt ist. Es stammt aus einer Zeit, in der es das Webservice-Konzept noch gar nicht gab und wurde entwickelt, um Sicherheit für Web-Anwendungen zu gewährleisten. So stehen hier auch die gängigen Sicherheitsmechanismen zur Verfügung. Im Vergleich zum WSS-Standard ist die Implementation jedoch verborgen, so dass jederzeit neue Sicherheitslücken auftreten können. Ein weiterer Nachteil ist, dass es eine Plattform-Spezifische Lösung ist. Der Vorteil von den IIS-Sicherheitsmechanismen ist der geringere Entwicklungsaufwand, da es in der ASP.NET-Bibliothek integriert ist. Die entsprechenden Methoden, um die vorgestellten Schutzziele zu erreichen, sind in beiden Konzepten vorhanden. Daher ist der Einsatz von WSS zu bevorzugen.

Um den WSS-Standard unter MS.NET einzusetzen gibt es das WSE-Framework (Ewald, 2004, "Web Services Enhancements"'). Dieses unterstützt unter anderem auch den Einsatz des Kerberos-Protokolls. Allerdings schreibt das Framework vor, dass ein Client sein Kerberos-Token ausschließlich beim Active-Directory des jeweiligen Domänen-Controllers anfordern kann. Dieser stellt dann ein Kerberos-Ticket für den Benutzer aus, der auf dem Rechner angemeldet ist, von dem die Anforderung gekommen ist. Diese Umsetzung ist normalerweise praktisch, da sie dem Entwickler viel Arbeit erspart. Für den Anwendungsfall „Kamera-Service“ ist diese Lösung aber nicht zu gebrauchen, weil die mobilen Clients in unserem Robot-Labor in keiner Windows-Domäne angemeldet sind.

Als Alternative bietet das WSE-Framework sogenannte „custom binary security token“ an. Mit diesen kann der Entwickler eigene Security-Token definieren. Um ein „custom binary

security token“ zu realisieren, muß ein Token-Manager entwickelt werden. Dort können die Eigenschaften definiert werden, die das Sicherheits-Token haben soll. Dazu gehören z.B. die Gültigkeitsdauer, der Schlüssel und der Client. Auch eine Verschlüsselung des Token ist möglich. Wie man mit dem WSE-Framework eigene Security-Token definieren kann, ist hier (Microsoft, 2005, "WSE - CustomSecurityToken"') beschrieben. Auf diese Weise ist es möglich, einen eigenen Kerberos-Server zu verwenden, der es erlaubt, dass auch Clients außerhalb einer Windows-Domäne zugreifen können. Dieser muß selbst entwickelt werden, damit die Kerberos-Tickets in Form von „custom binary security token“ vom WSE-Framework verwendet werden können.

Desweiteren unterstützt das WSE-Framework die Verwendung eigener Methoden um die SOAP-Nachrichten zu verschlüsseln. Dazu kann der Entwickler eigene Eingangs- und Ausgangs-Filter definieren, wie man das macht wird in diesem Artikel (Fitsner, 2005, "WSE - Security-Filter"') am Beispiel eines Username-Tokens erläutert, das Prinzip ist aber bei Custom-Tokens das gleiche. In den Ausgangsfiltern können Teile oder die gesamte Nachricht im SOAP-Body verschlüsselt werden. Informationen über die Verschlüsselung und das Sicherheits-Token können im SOAP-Header übertragen werden. Bei einer verschlüsselten Zwei-Wege-Kommunikation müssen also beim Client und beim Server jeweils ein Ausgangs- und ein Eingangsfilter entwickelt werden.

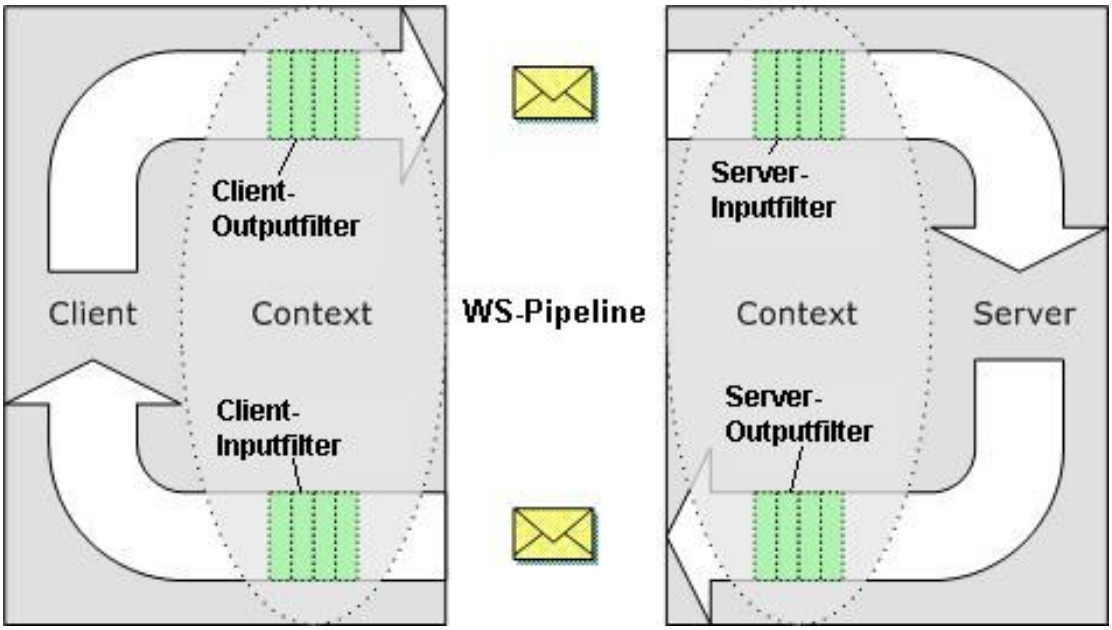

Abbildung 3.16: WSE-Security-Filter

Desweiteren bietet das WSE-Framework fertige Klassen zur Berechnung der bekannten Verschlüsselungs-Algorithmen und Schlüssel-Generierungen an.

Die folgende Grafik zeigt ein Beispiel für eine SOAP-Nachricht, die mit einem selbst verschlüsselten Custom-Binary-Token verschlüsselt ist. Ein solcher Token wäre in diesem Fall das Kerberos-Ticket für den Kamera-Service.

Abbildung 3.17: SOAP-Nachricht mit Custom-Binary-Token verschlüsselt

3.8.5 Design des Kerberos-Server

Da die Clients mit dem Kamera-Service über Webservices kommunizieren, bietet es sich an, auch die Clients mit dem Kerberos-Server über Webservices kommunizieren zu lassen. Dazu braucht der Kerberos-Server eine Webservice-Server-Komponente, die als Schnittstelle für die Clients dient. Das Kerberos-Protokoll lässt sich problemlos in SOAP-Nachrichten verpacken, da es jeweils nur eine Anfrage vorsieht und kein zustandsbehaftetes Handshake-Verfahren umgesetzt werden muß. Es müssen nur einige Schlüssel gespeichert werden. Diese können aber in der angebunden Datenbank abgelegt werden.

Da dieser Kerberos-Server eine spezielle Sicherheitslösung für die mobilen Clients in dem Robot-Labor der HAW-Hamburg realisieren soll, werden hier nicht alle Möglichkeiten, die das Kerberos-Protokoll liefert umgesetzt. Einige Features, wie z.B. das Zugreifen auf andere Kerberos-Realms[6] wird hier weggelassen. Der KDC-Server und der TGS-Server als eine Einheit zusammengefasst. Beide werden über den gleichen Webservice aufgerufen und haben auch eine gemeinsame Datenbank. Dadurch wird ein komplizierter Schlüssel-Austausch zwischen KDC und TGS überflüssig.

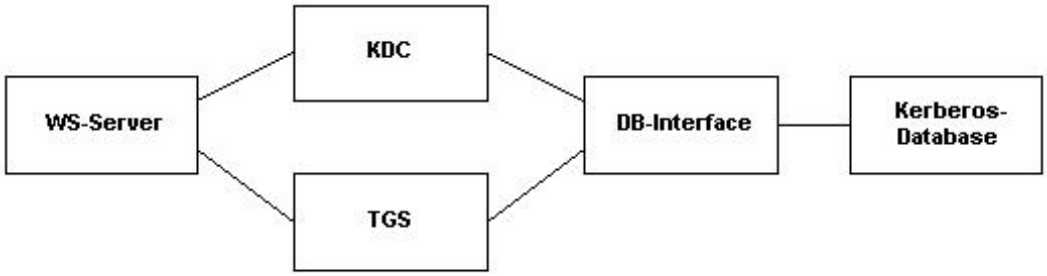

Abbildung 3.18: Aufbau des Kerberos-Servers

Die Kerberos-Datenbank muß eine Tabelle haben, in der alle zugelassenen User mit ihrem Namen, ihrem Passwort und ihrer Rolle eingetragen werden. Aus dem Namen und dem Passwort generiert der KDC dann den privaten Schlüssel des jeweiligen Clients. Jedem User wird eine Rolle zugeordnet. Für den Fall des Kamera-Services kommen z.B. die Rollen „user“ und „admin“ in Frage. Diese Rolle wird dem Kamera-Service über das Server-Ticket übertragen. Außerdem gibt es eine Tabelle, in der alle Schlüssel abgelegt werden, die dem Kerberos-Server bekannt sein müssen. Diese sind der private Schlüssel des TGS-Servers, die privaten Schlüssel der Ziel-Server und der Session-Key von Client und TGS-Server. Da alle diese Schlüssel aus Bytecode bestehen und in diesem Fall sogar alle AES-Schlüssel sind, können sie in einer Tabelle abgelegt werden. Die Identifikation kann über den Namen des jeweiligen Host in Kombination mit dessen Typ (Client, TGS, Service) erfolgen. Die

[6] Als „Realm“ bezeichnet man bei Kerberos eine Domäne

Session-Keys von Client und TGS-Server werden dann unter dem Namen des jeweiligen Clients abgelegt[7]. Außerdem kann jedem gespeicherten Schlüssel ein Verfalls-Zeitpunkt zugeordnet werden, um eine automatische Neuerzeugung der Schlüssel zu ermöglichen.

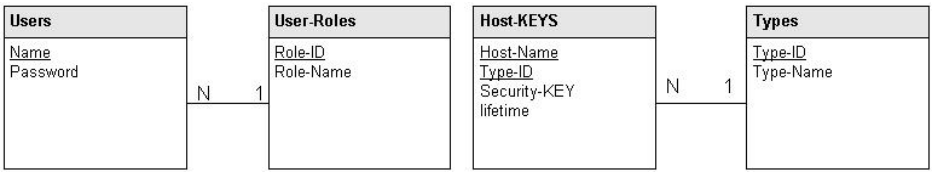

Abbildung 3.19: Aufbau der Kerberos-Datenbank

Die Funktionsweise der Kerberos-Komponente wird in dem folgenden Zustandsdiagramm veranschaulicht. Dabei ist zu erkennen, dass der TGS-Server zunächst einmal das TGS-Ticket'[8] entschlüsselt, um dann den Authenticator'[9] überprüfen zu können. Bei einem erfolgreichen Zugriff wird der Session-Key des jeweiligen Clients gelöscht, um einen zusätzlichen Schutz vor Mißbrauch zu erreichen.

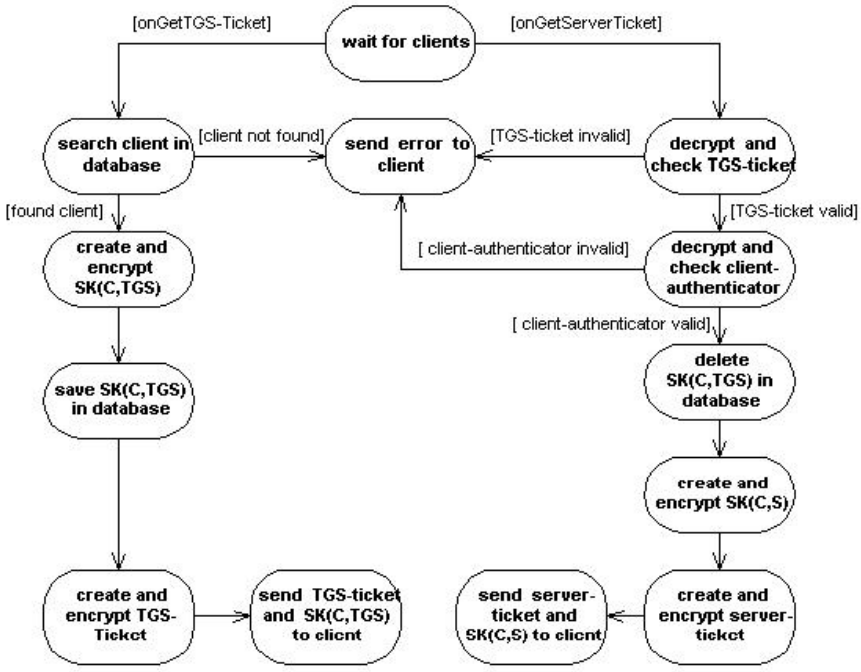

Abbildung 3.20: Funktionsweise des Kerberos-Servers

[7] deshalb ist die Identifikation über den Typ erforderlich, für den unwarscheinlichen Fall, dass ein Client genauso heißt, wie ein Server.

[8] Durch das TGS-Ticket Erhält der TGS-Server die Information, um welchen Client es sich handelt.

[9] Der Authenticator enthält den Zeitstempel und die IP-Adresse des Clients in verschlüsselter Form

4 Evaluation

In diesem Kapitel werden die Ergebnisse dieser Arbeit auf ihre praktischen Fähigkeiten hin überprüft. Dazu werden im wesentlichen die Aspekte, die sich in der Analyse herausgehoben haben untersucht. Diese sind die Funktionalität aus Sicht des Benutzers, das Zeitverhalten, die Erweiterbarkeit und die Sicherheit des Systems.

4.1 Die Funktionalität des Systems

An dieser Stelle wird überprüft, wie die Anwendungsfälle 2.3, die die Analyse ergeben hat, realisiert wurden und die Realisierung auf ihre praktische Eignung hin untersucht.

Der Anwendungsfall „Anmeldung eines Clients am Server“ sollte die Überprüfung der Authentizität und der Zugangsberechtigung des Users beinhalten. Realisiert wurde dieser Anmeldevorgang durch den Einsatz der Kerberos-Technologie. Dieses bedeutet in der Praxis, dass der Benutzer einen einmaligen Anmeldevorgang ausführen muß, um dann bei einer positiven Überprüfung uneingeschränkt das System nutzen zu können. Dieser Anwendungsfall ist also im Sinn der Analyse realisiert worden.

Die Anwendungsfälle „Client fragt Benutzerkonto ab“ und „Abfrage Objektpositionen“ wurden zu einem Anwendungsfall zusammengefasst. Der Client kann vom Kamera-Service eine Liste seiner eigenen Objekte anfordern. Diese Liste enthält die aktuellen Positionen der Objekte. Aufgrund der geringen Datenmenge, macht es in der Praxis keinen Sinn diese beiden Vorgänge von einander zu unterscheiden.

Der Anwendungsfall „Objekt-Modifikation“ umfaßt die drei Vorgänge „Hinzufügen“, „Ändern“ und „Löschen“. Realisiert wurden tatsächlich „Hinzufügen“ und „Löschen“. Der Vorgang „Ändern“ kann durch ein sequentielles Aufrufen von „Löschen“ und „Hinzufügen“ realisiert werden.

Damit der Kamera-Service vor der Ausführung der Anwendungsfälle, die Benutzerrechte und die Authentizität des Benutzers feststellen kann, muß der Benutzer nur sein Sicherheitstoken und einen Benutzer-Autenticator mitsenden.

Die Mehrbenutzer-Anforderung wurde durch den Einsatz, der dafür sehr gut geeigneten Webservice-Technologie erfüllt. Die dazu erforderliche nebenläufige Benutzer-Abwicklung wird vom .NET-System erzeugt.

4.2 Das Zeitverhalten

Das Zeitverhalten ist sehr entscheidend für den praktischen Nutzen des Systems. Beim Zeitverhalten gibt es zwei wesentliche Aspekte. Der eine ist die Frequenz mit der Objektpositionen übermittelt werden. Desto höher die Frequenz desto genauer kann der Weg eines bewegten Objektes verfolgt werden. Der zweite Aspekt ist die zeitliche Verzögerung, diese ist bei einem mobilen Roboter noch wesentlicher, da er sonst auf Änderungen in seiner Umwelt zu spät reagieren kann.

Um diese beiden zeitlichen Aspekte zu überprüfen, wird ein praktischer Versuch durchgeführt. Dieser soll zeigen, wie das System auf eine unterschiedliche Anzahl von Objekten, sowie auf verschiedene Geschwindigkeiten reagiert. Dazu werden an einem Roboter mehrere nebeneinander platzierte Farb-Markierungen angebracht. Anschließend wird er über eine Teststrecke fahren gelassen. Von einem Test-User werden die Positionen der Farbmarkierungen beobachtet. Die übermittelten Positionen werden dabei visualisert, aber ohne dass bei einer Positionsänderung die Anzeige gelöscht wird “[1]. Dadurch wird ein Punktmuster entstehen. An den Abständen der einzelnen Positionspunkten kann man die Frequenz erkennen, also die Genauigkeit mit der der Weg eines Objektes verfolgt werden kann. Um eine eventuelle zeitliche Verzögerung festzustellen, wird der Versuch sofort gestoppt, sobald der Roboter die Fläche vollständig durchfahren hat. An der Differenz zwischen der zuletzt übermittelten und der tatsächlichen Position kann dann eine zeitliche Verzögerung festgestellt werden.

Abbildung 4.1: Versuchsaufbau: Ein Roboter, der mit mehreren Farbpunkten markiert ist

[1] Wie sich bei der Durchführung dieses praktischen Tests herausgestellt hat, unterstützt das verwendete „.NET-CompactFramework Version 2.0“ die dazu notwendigen Grafikfunktionen nicht, so dass dieser Test nur von einem Desktop-PC durchgeführt werden kann, auf dem die Vollversion des .NET-Frameworks installiert ist. Somit kann dieser Test, eventuelle Verzögerungen durch die WLAN-Übertragung nicht berücksichtigen, sondern nur die Funktionalität ab der Webservice-Schnittstelle testen.

Die folgende Grafik soll veranschaulichen, wie die Ergebnisse zu interpretieren sind.

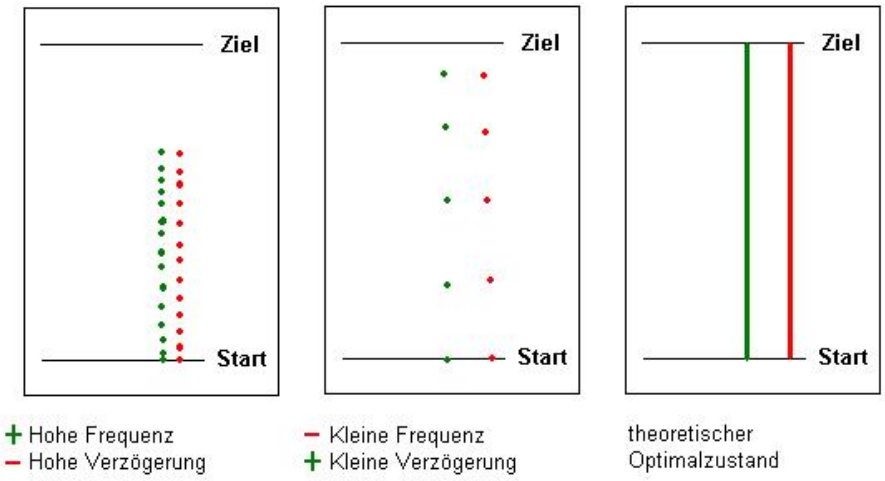

Abbildung 4.2: Interpretation der Ergebnisse

Hier wird das Versuchs-Ergebnis dargestellt.

Es ist eine deutliche Verschlechterung der Abtast-Frequenz bei steigender Anzahl der Objekte und steigender Geschwindigkeit zu beobachten. Diese Verschlechterung verläuft aber linear, so dass sie mit einer höheren Rechenleistung bewältigt werden könnte. Eine Verzögerung ist dagegen nicht festzustellen. Das Ergebnis entspricht damit der Erwartung, die sich in der Analyse 2.5 gebildet hat. Der Schwachpunkt des Systems in Bezug auf das Zeitverhalten, ist die lange Rechenzeit, die das Kamera-Framework bei der Berechnung komplexer Filter benötigt. Während bei nur einem Objekt und langsamer Geschwindigkeit, der Weg des Objektes nahezu optimal verfolgt wird, sind die Ergebnisse bei vier Objekten und großer Geschwindigkeit schon sehr ungenau. Die Zeit, die der Service benötigt, ist im Vergleich zu vernachlässigen, denn sonst wäre eine Verzögerung zu beobachten.

Eine weitere Erkenntnis, die bei diesem Versuch gewonnen wurde, ist das die Beschreibung der Objekte über RGB-Farbwerte zu keinen guten Ergebnissen führte. Schon geringe Schwankungen in den Lichtverhältnissen führen so zu einem unbrauchbaren Ergebnis. Wurden die Objekte über das HSV-Format beschrieben, fiel dieser Effekt sehr viel kleiner aus. Wie man in dem Versuch, der mit HSV-Beschreibungen durchgeführt wurde, sehen kann, sind einige Farben anfälliger für Lichtschwankungen, als andere. Grün“[2]hat sich als am unanfälligsten herausgestellt.

[2]Die Farben werden durch die Umrechnung HSV nach RGB nicht richtig dargestellt, da das Kamera-Framework den V-Anteil nicht berücksichtigt.

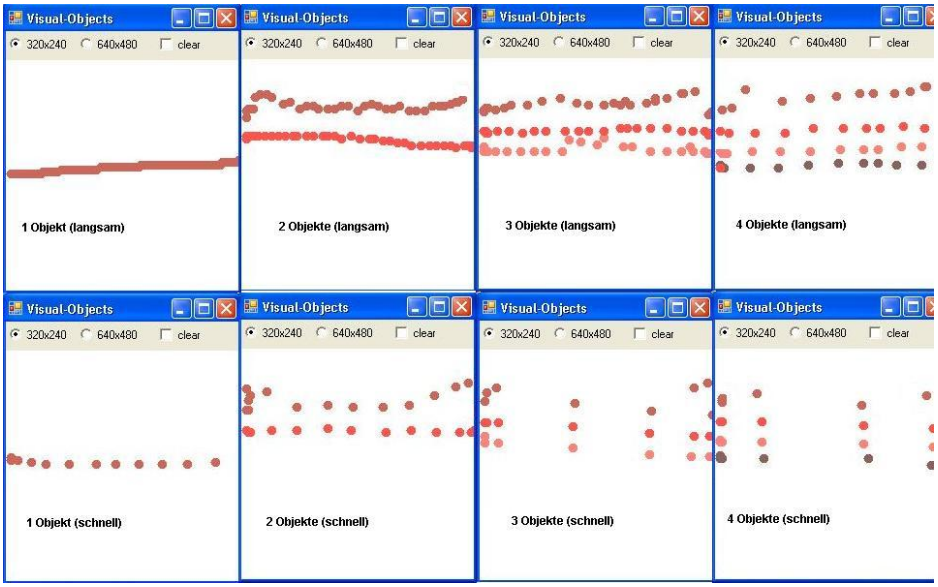

Abbildung 4.3: Ergebnisse der Untersuchung des Zeitverhaltens

4.3 Sicherheit

Die Haupt-Anforderungen die an das System, aus sicherheitstechnischer Sicht gestellt wurden, waren die Vertraulichkeit und Integrität. Das entstandene Sicherheitskonzept stellt dieses sicher, ohne dass es den Benutzer bei der Nutzung des Systems behindert. Es ist nur zu Beginn einer Sitzung eine Eingabe des Names und des Passwortes erforderlich. Auch die Zeit die für die Verschlüsselung aufgewendet wird, ist für den Benutzer kaum spürbar.

4.4 Erweiterbarkeit

Aufgrund der klaren Trennung der Aufgabenbereiche der einzelnen Komponenten kann das System ohne Probleme erweitert oder verändert werden. So wäre es möglich ein anderes Sicherheitskonzept oder ein anders Kamera-Framework zu verwenden. Durch die Webservice-Technologie können Benutzer von verschiedenen Plattformen aus das System benutzen. Auch die Erweiterbarkeit, um andere Objekt-Beschreibungsformen macht das System flexibel für andere Aufgabenfelder.

5 Resümee

5.1 Ergebnisse

Ziel dieser Arbeit war die Entwicklung eines Systems, das mobilen Robotern die Möglichkeit erschaffen sollte, Informationen über ihre Umwelt von außen erhalten zu können. Die Informationen sollten dabei vor fremden Zugriffen geschützt sein.

Dabei ist ein System entstanden, bei dem mobile Clients die Positionen von bewegten Objekten beobachten lassen können. Jeder Client verfügt über ein eigenes Benutzerkonto, auf dem er eigene Objekte erzeugen und definieren kann. Die Beschreibung der Objekte erfolgt momentan über deren Farbe. Das System ist so entworfen, dass es ohne großen Aufwand um andere Beschreibungsformen erweitert werden kann.

Das System ist auf mehreren Komponenten aufgebaut, deren Aufgabenbereiche deutlich voneinander getrennt sind, so dass die Austauschbarkeit einzelner Bestandteile gewährleistet ist.

Die Kommunikation zwischen den Clients und dem System läuft über standardisierte, plattformunabhängige Protokolle, so dass das System für alle denkbaren mobilen Roboter nutzbar ist.

Der Schutz der Informationen eines Clients erfolgt über ein Sicherheitskonzept, welches auf Standardtechniken beruht. Dieses umfasst Schutzmaßnahmen, die die Vertraulichkeit und Integrität sicher stellen.

5.2 Bewertung der Ergebnisse

Das entstandene System erfüllt die Anforderungen in Bezug auf die Aspekte Erweiterbarkeit und Funktionalität.

Die Mulituser-Fähigkeit ist nur teilweise erfüllt. Die verwendete Webservice-Technologie ist zwar sehr gut geeignet einen Mulituser-Betrieb durchzuführen. Eine große Schwäche des jetzigen Systems ist aber, dass der Kamera-Service nur dann brauchbare Ergebnisse liefert,

wenn die Anzahl der zu beobachtenden Objekte gering ist. Dieses hat zwei Ursachen. Zum einen läuft der Kamera-Service auf dem gleichen Host, wie das Kamera-Framework. Zum anderen ist nur eine Instanz des Kamera-Frameworks vorhanden, so dass die Bild-Filter und deren Anwendung um so komplexer werden, desto mehr Objekte beobachtet werden sollen. Da die Objektverwaltung im Vergleich zur Bildverarbeitung relativ wenig CPU-Ressourcen verbraucht, ist die Art der Bildverarbeitung das größte Problem des jetzigen Systems.

Ein weiterer Nachteil der jetzigen Lösung ist die begrenzte Fläche die durch den Kamera-Service abgedeckt werden kann. Die Kamera hat nur einen begrenzten Blickwinkel. Um die Fläche die beobachtet wird zu beeinflussen stehen die beiden Parameter Kamera-Zoom und Kamera-Positionierung zur Verfügung. Der Zoom der Kamera ist allerdings begrenzt. Genauso verhält es sich mit der Kamera-Positionierung. Zumindest in dem Robot-Labor ist die Position durch die Raumhöhe begrenzt. Dieses hat zur Folge das sich schon die verhältnismäßig kleine Fläche des Robot-Fussball-Feldes nicht ganz vollständig überwachen läßt. Außerdem hat eine Vergrößerung der fokusierten Fläche, eine schlechtere Bildauflösung zur Folge. Dieser könnte man zwar durch eine Erhöhung der Auflösung des Kamerabildes entgegen wirken, was aber wieder ein schlechteres Zeitverhalten zur Folge hätte.

Für Anwendungsbereiche, bei denen die registrierten Objekte häufig modifiziert werden, stellt möglicherweise die Tatsache, dass die Bildverarbeitung bei einer Modifikation kurz unterbrochen werden muß, ein Problem dar. Dieses ist in der Funktionsweise des verwendeten Kamera-Frameworks begründet.

Die wichtigsten Anforderungen an die Sicherheit, die Vertraulichkeit und Integrität sind erfüllt worden. Dagegen ist die Verfügbarkeit nur vor Störungen, von unautorisierten Benutzern geschützt. Wenn autorisierte Benutzer zu viele Objekte anlegen oder mit einer hohen Frequenz ihre Objekte modifizieren, ist die Verfügbarkeit gestört.

5.3 Ausblick

Die Bewertung der Ergebnisse hat gezeigt, dass das jetzige System, eine gute Lösung für den Zugriff eines Clients auf den Kamera-Service darstellt. Dagegen hat sich die Methodik mit der die Bildverarbeitung eingesetzt wird, als nur bedingt tauglich herausgestellt. An dieser Stelle sollte eine mögliche Weiterentwicklung des Kamera-Services ansetzen. Dazu werden hier zwei mögliche Modelle vorgestellt, die die bisherigen Schwachpunkte beheben könnten.

Um einen Vergleich der beiden Modelle, die im Folgenden vorgestellt werden, mit dem jetzigen System zu ermöglichen, veranschaulicht die folgende Grafik kurz das Prinzip des jetzigen Modells.

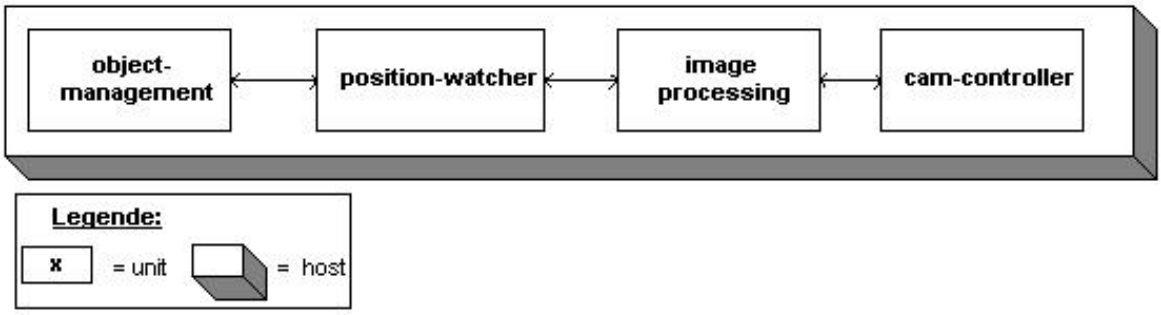

Abbildung 5.1: jetziges einfaches Modell der Bildverarbeitung

Es ist deutlich zu erkennen, dass nur eine Instanz der Bildverarbeitung vorhanden ist und diese auch noch auf dem gleichen Rechner ablaufen, wie die Objektverwaltung. Um die zwei Schwachpunkte „begrenzte Objektkapazitäten“ und „begrenzte zu überwachbare Fläche“ zu beheben, müssen Modelle entwickelt werden, die die Aufgaben auf mehrere Rechner verteilen. Die folgenden beiden Modelle, nutzen die bisher entstandene Infrastruktur und liefern problemspezifische, verteilte Lösungsansätze.

Zunächst wird ein Lösungsansatz vorgestellt, der das Problem „begrenzte Objektkapazitäten“ beheben soll. In diesem Modell ließt eine Verteilungseinheit die Objekte aus der Datenbank und verteilt sie auf mehrere Rechner, auf denen ausschließlich die Bildauswertung ausgeführt wird. Diese erhalten ihre Bilddaten von einem Rechner, der für das Grabben des Kamerabildes zuständig ist. Auf diese Weise würden die Prozesse, die viel CPU-Zeit benötigen, auf mehrere physikalische Einheiten verteilt. Dabei wäre es eventuell eine interessante Frage, wie man die Objekte am geschicktesten verteilt, damit die Bildverarbeitungs-Algorithmen am effektivsten ablaufen könnten. Ansonsten wäre dieses Modell aber relativ einfach umzusetzen. Es müsste aber überprüft werden, ob ein solches System aufgrund des zusätzlichen Datentransfers überhaupt effektiv arbeiten könnte.

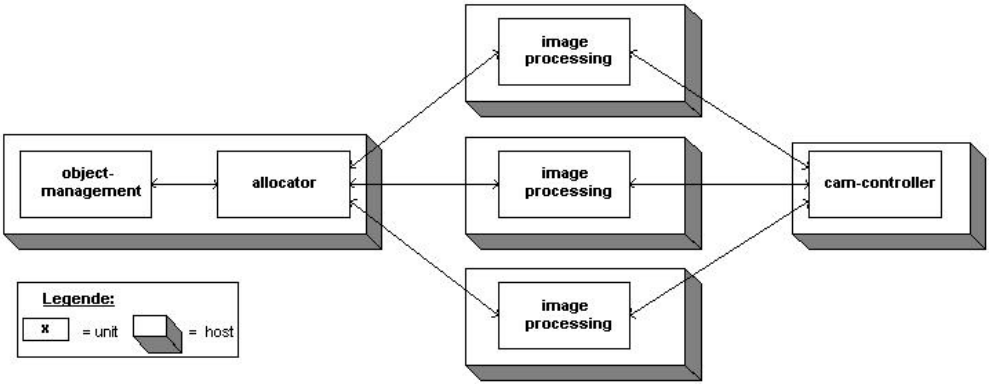

Abbildung 5.2: Verteilung der rechenintensiven Prozesse

Ein weitergehender Lösungsansatz, der vor allem auf das Problem „begrenzte zu überwachbare Fläche“ zugeschnitten ist, wird hier vorgestellt. Bei diesem Modell werden nicht nur die rechenintensiven Aufgaben verteilt, sondern auch die zu überwachende Fläche wird hier aufgeteilt. Dazu bekommt jede Bildverarbeitungseinheit eine eigene Kamera, die einen Teilabschnitt der gesamten Fläche überwacht. Die Herausforderung, die ein solches Modell an den Entwickler stellt, ist die Schwierigkeit der Ortung. Um ein solches System effektiv betreiben zu können, müßte ein Mechanismus entwickelt werden, der eine geschickte Objektübergabe zwischen den einzelnen Stationen ermöglicht. Auch die Zeitverzögerung bei einer Neukonfigurierung der Bildfilter wäre hier problematisch, da diese sehr häufig erforderlich wäre. Das Kamera-Framework müßte also entweder so verbessert werden, dass eine Online-Konfiguration möglich wäre oder es müßten redundante Kameraeinheiten verwendet werden.

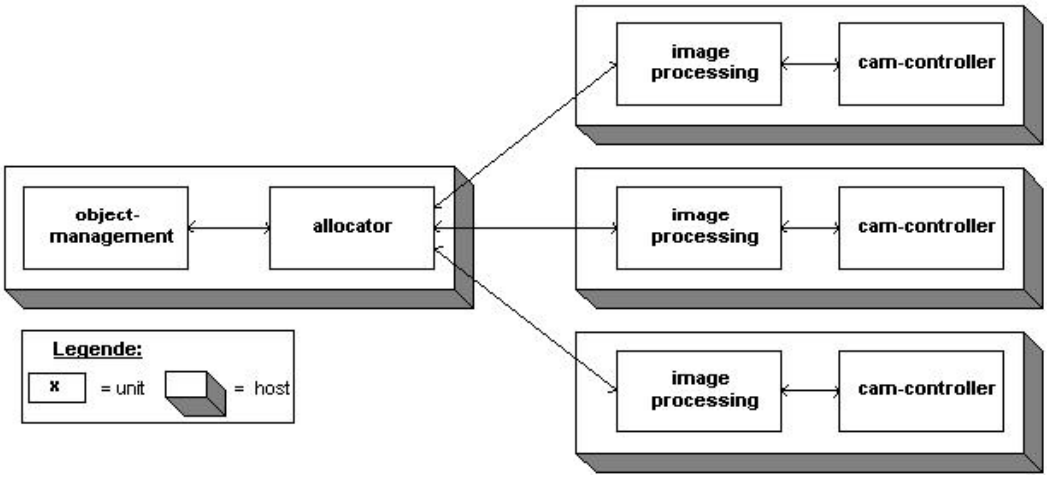

Abbildung 5.3: Aufteilung der Fläche auf mehrere Kameraeinheiten

Aber nicht nur eine alternative Software-Architektur kann die Performance der Bildverarbeitung steigern. Auch ein Einsatz von spezieller Hardware, die für eine schnelle Bildverarbeitung optimiert ist, kann zu einer erheblichen Verbesserung beitragen. Hier wäre z.B. der Einsatz eines Cell-Prozessors(Geschwind, 2006, Cell-Prozessor) geeignet. Dieser ist ein Prozessor, der zusammen von IBM, Sony und Toshiba entwickelt wurde. Er besteht aus acht einzelnen Prozessor-Einheiten und ist für eine parallelisierte Berechnung aufwendiger Rechenoperationen konzipiert worden. So soll er z.B. in der Playstation 3 zum Einsatz kommen, wo er vor allem aufwendige 3D-Grafik berechnen soll.

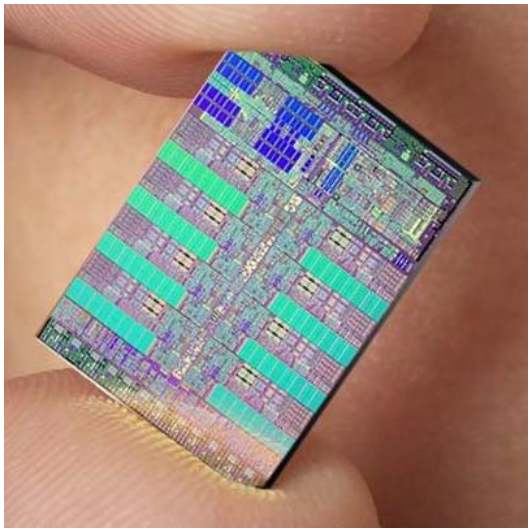

Abbildung 5.4: Cell-Ship

Neben dem Vorschlag für mögliche Erweiterungen, soll hier noch ein kleiner Ausblick auf mögliche weitere Anwendungsfälle des Systems gegeben werden. Dabei ist anzumerken, dass die meisten möglichen Anwendungsfälle, bei denen die Positionen von Objekten geortet werden sollen, heute eher durch kleine Funksender gelöst werden, da solche Systeme wesentlich einfacher zu realisieren sind. Diese lassen aber keine so genaue Ortung zu, wie optische Systeme.

Ein gutes Beispiel hierfür ist der reale Fußball. Dort kommt es manchmal auf wenige Zentimeter an, wenn es darum geht ob der Ball die Torlinie überschritten hat oder nicht. Dort wurden bereits Bälle mit entsprechendem Funk-Ships getestet. Das Ergebnis war unbrauchbar, da das System nicht genau genug gearbeitet hat. Daher werden nun optische Hilfsmittel diskutiert. Mit einem System, wie diesem hier, könnte der Schiedsrichter ein mobiles Gerät mit sich führen, welches ihm stets die Position des Balles innerhalb der letzten Sekunden anzeigt.

Ein weiteres Beispiel, bei dem es weniger auf die Genauigkeit, dafür um so mehr auf die Datensicherheit des Systems ankommt, wäre z.B. ein Krankenhaus. Hier könnte ein solches System dem Personal zur Verfügung gestellt werden. Das System könnte erkennen, bei welchem Patienten sich ein Mitarbeiter gerade aufhält und ihm so die entsprechenden Daten des Patienten übermitteln, z.B. welche Medikamente er einnehmen soll. Dabei könnte die Rollenverteilung so genutzt werden, dass einem Arzt mehr Informationen übermittelt werden, als einer Krankenschwester.

Zum Abschluß erfolgt noch eine kurze Einschätzung, welchen Nutzen diese Diplomarbeit für das Robot-Labor haben könnte. Zum einen hat sie die Möglichkeit erschaffen, die Roboter mit einer neuen Methode der Sensorik zu versehen, wodurch einige Probleme, die

bei der bisher verwendeten On-Board-Sesorik auftraten, behoben werden könnten. Es ein Instrument geschaffen worden, das den Robotern eine wesentlich bessere Orientierung in ihrer Umwelt ermöglicht. Dieses kann für zukünftige Arbeiten verwendet und je nach Anforderungs-Profil erweitert werden. Zum anderen wurde ein Sicherheitskonzept entwickelt, welches für mobile Einheiten konzipiert ist und auch für andere Anwendungen genutzt werden könnte.

Literaturverzeichnis

[BSI 2005] BSI: Datenschutz-Richtlinien. (2005), November. – URL http://www.bsi.de. – Zugriffsdatum: 2005-11-16

[Castelino 2003] CASTELINO, Kenneth: 3DES. (2003), Januar. – URL http://dnclab.berkeley.edu/~kenneth/courses/sims250/des.html. – Zugriffsdatum: 2006-02-16

[Ewald 2004] EWALD, Tim: Programmieren mit Web Services Enhancements 1.0 für Microsoft .NET. (2004), Juni. – URL http://www.microsoft.com/germany/msdn/library/xmlwebservices/ProgrammierenMitWebServicesEnhancements10FuerMicrosoftNET.mspx. – Zugriffsdatum: 2006-02-15

[Federation 2006] FEDERATION, Robocup: (2006), Januar. – URL http://www.robocup.org/. – Zugriffsdatum: 2006-01-15

[Fitsner 2005] FITSNER, Feodor: Custom WSE 3.0 Policy Assertions for Signing and Encrypting. (2005), November. – URL http://www.codeproject.com/soap/WSE30UsernameAssertion.asp. – Zugriffsdatum: 2006-02-15

[Frenette 1999] FRENETTE, Martin-Pierre: Sending a message to the Main Frame Window of Another Application. (1999), Dezember. – URL http://www.codeproject.com/threads/sendmsg.asp. – Zugriffsdatum: 2006-01-15

[Gerling 2003] GERLING, Mirco: PDA-gestützte Robotersteuerung mit funkbasierter Serveranbindung. (2003), Dezember. – URL http://users.informatik.haw-hamburg.de/~kvl/gerling/diplom.pdf. – Zugriffsdatum: 2005-02-01

[Geschwind 2006] GESCHWIND, Michael: The Cell architecture. (2006), Januar. – URL http://domino.research.ibm.com/comm/research.nsf/pages/r.arch.innovation.html. – Zugriffsdatum: 2006-03-05

[Hoier 2002] HOIER, Niels: Authentifikation mit Kerberos. (2002), Juni. – URL http://www.informatik.uni-hamburg.de/RZ/lehre/18.415/seminararbeit/5_Authentifikation.pdf. – Zugriffsdatum: 2005-11-23

[Kohl und Neuman 1993] KOHL, John ; NEUMAN, Clifford: The Kerberos Network Authentication Service (V5), RFC 1510. (1993), September. – URL http://www.ietf.org/rfc/rfc1510.txt. – Zugriffsdatum: 2005-11-16

[Krautschick und Maybaum 2005] KRAUTSCHICK, Dirk Krautschick D. ; MAYBAUM, Stefan S.: AES. (2005), Januar. – URL http://www.hoever.fh-aachen.de/WS0506/krypto/Referate/8_AES.pdf. – Zugriffsdatum: 2006-02-16

[Mack 2003] MACK, Dr. D.: Kerberos (Zusammenfassung). (2003), Oktober. – URL http://www-zeuthen.desy.de/afs-workshop-2003/kerberos.pdf. – Zugriffsdatum: 2006-02-13

[Manger 2004] MANGER, Michael: Design und Realisierung einer experimentellen Plattform für Roboterfußball. (2004), Mai. – URL http://users.informatik.haw-hamburg.de/~kvl/revout/diplomarbeit.pdf. – Zugriffsdatum: 2006-02-16

[Microsoft 2005] MICROSOFT: How to Create a Class Representing a Custom Binary Security Token. (2005), Januar. – URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/c95a067e-178c-4f9f-8617-41c5d86ae356.asp. – Zugriffsdatum: 2006-02-15

[Oasis 2005] OASIS: WSS-Spezifikationen. (2005), November. – URL http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss. – Zugriffsdatum: 2005-11-17

[Okutan und Schnaufer 2004] OKUTAN, Can ; SCHNAUFER, Sascha: Einführung in Web Services. (2004), Juni. – URL http://www.informatik.uni-mannheim.de/pi4/lectures/ss2004/seminar/vortraege/einfuehrung.pdf. – Zugriffsdatum: 2005-11-23

[Raj 1998] RAJ, Gopalan S.: A Detailed Comparison of CORBA, DCOM and Java/RMI. (1998), September. – URL http://my.execpc.com/~gopalan/misc/compare.html. – Zugriffsdatum: 2005-11-17

[Revout 2003] REVOUT, Ilia: Design und Realisierung eines Frameworks für Bildverarbeitung. (2003), Dezember. – URL http://users.informatik.haw-hamburg.de/~kvl/revout/diplomarbeit.pdf. – Zugriffsdatum: 2005-11-16

[Schmidt 2005] SCHMIDT, Oliver: Entwicklung und Aufbau eines Kamera-Service basiert auf .NET-Technologie. (2005), Februar. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/studien/schmidt.pdf. – Zugriffsdatum: 2005-11-16

[Short 2002] SHORT, Scott: *Webdienste mit dem .NET Framework entwickeln*. 1. Auflage. Microsoft-Press, 2002. – ISBN 3-86063-644-8

[Todorovic 1999] TODOROVIC, Zoran: Interprocess Communication using Shared Memory. (1999), Dezember. – URL http://www.codeproject.com/threads/sharedmemipc.asp. – Zugriffsdatum: 2006-01-15

[Weyer 2002] WEYER, Christian: *XML Web Service-Anwendungen mit Microsoft.NET*. 1. Auflage. addison-Wesley, 2002. – ISBN 3-8273-1891-2

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 6. März 2006	
Ort, Datum	Unterschrift