



# Entwurf und Implementierung einer grafischen Benutzungsschnittstelle für Fluglotsen als Komponente eines Flugsicherungssimulationssystems

Diplomarbeit  
am Fachgebiet Flugführung und Luftverkehr

an der  
Technischen Universität Berlin

vorgelegt von cand. inf. Arne Burmeister  
Berlin, August 1996

---

**Aufgabensteller:**

- Prof. Dr.-Ing. M. Fricke
- Prof. Dr.-Ing. Stefan Jähnichen

**Betreuung:**

- Dipl.-Ing. Alexander Schmid

Diese Arbeit wurde im Rahmen des DFG-Projektes

*„Anthropotechnische Untersuchung  
der Mensch-Maschine-Schnittstelle  
in einem kooperativen Air Traffic Management“*

am Fachgebiet Flugführung und Luftverkehr der TU Berlin mit freundlicher Unterstützung der Deutschen Flugsicherung GmbH, RK Berlin durchgeführt.

**Projektmitglieder**

- Dipl.-Ing. Alexander Schmid
- Dipl.-Psych. Genia Grundmann
- Kai-Uwe Nielsen

---

# Inhalt

Abkürzungen .....	vii
<b>1 Einleitung</b>	
1.1 Thema .....	1
Einführung.....	1
Aufgabenstellung.....	1
1.2 Benutzungsschnittstellengestaltung.....	2
Ziele .....	2
Methoden .....	3
1.3 Vorgaben .....	3
Das Flugsicherungssimulationssystem.....	4
Die Entwicklungsumgebung.....	4
<b>2 Modellierung und Entwurf</b>	
2.1 Vorbereitungen .....	7
Existierende Systeme.....	7
Benutzerbefragung.....	9
2.2 Eine Styleguide.....	11
Vorlagen .....	12
Fensterklassen.....	13
Bedien- und Anzeigeelemente.....	14
2.3 Modularisierung .....	15
Kriterien.....	15
Client- und Frontendmodule .....	16
Allgemeine Module.....	17
2.4 Schnittstellen.....	17
Server-Client.....	18
Client-Frontend .....	19
Indirekte Verbindungen .....	19

---

2.5	Ein Client .....	20
	Die Clients des ATC-Systems.....	20
	Der CWP-Client .....	20
	Fensterlayout des CWP-Client.....	21
	Systemstatus ändern .....	23
	Voreinstellungen .....	23
2.6	Ein Frontend .....	24
	Das Radar-Frontend .....	24
	Fensterlayout des Radar-Frontend .....	25
	Funktionalität der Radardarstellung .....	26
2.7	Zusätzliche Frontends .....	29
	Elektronische Kontrollstreifen.....	29
	Verhandlungsführung.....	30
3	Implementierung	
3.1	Grundlagen.....	33
	Globale Variablen.....	33
	Umgebungsvariablen.....	33
	Dynamisches Nachladen .....	34
	Fensterklassen .....	34
3.2	Allgemeine Module .....	35
	Datenbankzugriff.....	35
	Interprozeßkommunikation .....	36
	Voreinstellungen .....	37
	Benutzungsoberfläche.....	37
	Dialoge.....	38
	Farben.....	39
3.3	CWP-Client Module .....	39
	CWP .....	39
	Client .....	40

---

3.4	Radar-Frontend Module.....	41
	Radar.....	41
	map.....	41
	target.....	42
3.5	Handbuch.....	43
	Systemstart.....	43
	Arbeitsplatzübernahme .....	44
	Fensterverwaltung .....	44
	Bedienung des Radarfensters .....	45
	Ändern der Einstellungen.....	45
4	Abschluß	
4.1	Bewertung.....	47
	Erneute Befragung.....	47
	Ergebnisse.....	47
4.2	Ausblick.....	48
	Implementation weiterer Frontends .....	48
	Umstellung auf objektorientierte Entwicklung.....	48
	WIAS und Online-Hilfe.....	49
	Anhang	
A	Module.....	51
	Diskette.....	51
	Dateien.....	51
	Referenz .....	51
B	Quellen .....	52
	Entwicklungsumgebung.....	52
	Datenbank .....	52
	Abbildungen.....	52
	Literatur .....	53
	Glossar .....	55



---

# Abkürzungen

**A/C:** Aircraft

**ADS:** Automatic Dependant Surveillance

**ATC:** Air Traffic Control

**ATM:** Air Traffic Management

**ATS:** Air Traffic Services

**AW:** Airway

**BTX:** Bildschirmtext

**COPS:** Common Operational Performance Specification for  
Controller Working Position

**CTA:** Control Area

**CWP:** Controller Working Position

**DFS:** Deutsche Flugsicherung GmbH

**DIN:** Deutsches Institut für Normung

**DLR:** Deutsche Forschungsanstalt für Luft- und Raumfahrt

**FIR:** Flight Information Region

**FL:** Flightlevel

**FMS:** Flight Management System

**FTP:** File Transfer Protocol

**GUI:** Graphical User Interface

**HCI:** Human Computer Interaction

**HTML:** Hypertext Markup Language

**HTTP:** Hypertext Transfer Protocol

**ILR:** Institut für Luft- und Raumfahrt an der TU Berlin

**IP:** Internet Protocol

**IPC:** Inter-Process-Communication

**ISO:** International Standards Organisation

**KATI:** Kooperatives ATM am ILR

**MMS:** Mensch-Maschine System

**NFS:** Network File System

---

**ODID: Operational Display and Input Development Group,  
EUROCONTROL**

**QNH: Luftdruck zur Kalibrierung der Höhenmesser**

**RK: Regionalkontrollstelle der Flugsicherung**

**SSR: Secondary Surveillance Radar**

**Tcl: Tool Command Language**

**TCP: Transmission Control Protocol**

**Tk: Tcl GUI-Toolkit**

**TRA: Temporary Reserved Airspace**

**UIR: Upper FIR**

**UTA: Upper CTA**

**W3C: WorldWideWeb Consortium**

**WIAS: Wetterdaten- und Informationsanzeigesystem**

**WIMP: Windows, Icons, Mouse and Pointer**

**WP: Waypoint**

**WWW: World Wide Web**

**ZFB: Zentrum für Flugsimulation Berlin GmbH**

**ZKSD: Zentraler Kontrollstreifendruck**



# 1 Einleitung

## 1.1 Thema

### Einführung

Zunehmende Automatisierung sowie moderne Rechner- und Darstellungstechnologien werden die Arbeitsplätze von Fluglotsen in Zukunft stark verändern. Eine wichtige Neuerung mit Einfluß auf die Gestaltung zukünftiger Lotsenarbeitsplätze wird die Einführung einer digitalen Datenverbindung zwischen Flugzeugen und Bodensystemen sein. Im Rahmen eines Forschungsvorhabens an der TU Berlin soll ein Simulationssystem für Lotsenarbeitsplätze im zukünftigen Air Traffic Management entwickelt und die Mensch-Maschine Schnittstelle für den Fluglotsen untersucht werden.

Das Simulationssystem für zukünftige Fluglotsenarbeitsplätze basiert auf einem Netzwerk mit Luftverkehrssimulation und Datensätzen zur Luftraumstruktur und zum ATS-Streckennetz in der Bundesrepublik Deutschland, Simulationssystemen von Flight Management Computern von Luftfahrzeugen. Die Mensch-Maschine Schnittstelle für die Fluglotsen stellt im Simulationssystem eine der Hauptfunktionen für weiterführende Untersuchungen dar. Als Neuheit gegenüber heute existierenden Fluglotsenarbeitsplätzen sollen grafische Hilfsmittel zur Bord/Boden Kommunikation via DataLink in die Benutzungsoberfläche integriert werden.

### Aufgabenstellung

Im Rahmen der Diplomarbeit soll, aufbauend auf den bisherigen Ergebnissen der anthropotechnischen Untersuchungen zur Fluglotsentätigkeit von DFS, DLR, EUROCONTROL und Untersuchungen an der TU Berlin, eine Mensch-Maschine Schnittstelle für Exekutiv- und Koordinationslotsen in Form einer grafischen Benutzungsoberfläche mit Hilfe von Tcl/Tk unter X-Windows geschaffen werden. Die Basis der Oberfläche soll eine synthetische Radardatendarstellung sowie eine elektronische Darstellung der Kontrollstreifeninformationen enthalten. Der Aufbau ist modular zu gestalten, um eine einfache Modifizierbarkeit zu gewährleisten. Maßgebend für die Entwicklung sind die Ergebnisse aus ODID I-IV sowie die Spezifikationen aus COPS.

Folgende Arbeitsschritte sind vorzunehmen:

- Definition und Spezifikation der Schnittstellen der Benutzungsoberfläche zu Funktionalität und Datenbank des Simulationssystems in Zusammenarbeit mit den Projektmitarbeitern

- Entwicklung eines modularen Programmsystems für die Benutzungsoberfläche des Fluglotsenarbeitsplatzes
- Entwicklung einer elektronischen Darstellung von Kontrollstreifen
- Einbindung der Kommunikation Lotse/Pilot über DataLink

Es sollen dabei die folgenden Umsetzungsaspekte beachtet werden:

- Textuelle und/oder grafische Darstellung;
- Gestaltung der Symbole;
- Sicherstellung maximaler Differenzierbarkeit der Symbole;
- Bildschirmaufteilung/Plazierung;
- Farbcodierung;
- Ausschluß von Homonymen und Synonymen;
- Bedienung über geeignete Eingabemedien (Tastatur, Maus)

Abschließend soll im Rahmen der Diplomarbeit eine Bewertung der Benutzungsoberfläche mit Hilfe aktiver Fluglotsen hinsichtlich Anordnung und Farbgestaltung der einzelnen Komponenten durchgeführt werden.

## 1.2 Benutzungsschnittstellengestaltung

### Ziele

Jedes System ist für eine Aufgabe nur dann von Nutzen, wenn es die benötigte Funktionalität besitzt und diese dem Benutzer auf eine für ihn verständliche Art zugänglich macht [No89]. Um dieses Ziel zu erreichen ist eine benutzer- und aufgabenorientierte Systemgestaltung notwendig.

Der Computer ist ein überaus universelles Werkzeug, so daß jede Software als eigenständiges System angesehen werden kann. Entsprechend der Mensch-Maschine-Schnittstelle besitzt jede Software eine Benutzungsschnittstelle, über die der Mensch Informationen angezeigt bekommt und Eingaben und Kommandos tätigt.

Die Benutzungsschnittstellengestaltung ist als Teil der Software-Ergonomie ein extrem interdisziplinäres Gebiet der Informatik. Sie basiert auf Kenntnissen der Softwaretechnik, Psychologie und Pädagogik, Arbeits- und Organisationswissenschaften, Linguistik, Kommunikations- und Medienwissenschaften sowie Grafik- und Produktdesign[Maa93].

Diese Erkenntnisse der beteiligten Gebiete führen zu den inzwischen auch genormten Gestaltungsgrundsätzen [ISO9241-10, DIN66234]:

1. Aufgabenangemessenheit
2. Selbstbeschreibungsfähigkeit
3. Steuerbarkeit
4. Erwartungskonformität
5. Fehlerrobustheit
6. Individualisierbarkeit
7. Lernförderlichkeit

Ted Nelson [Ne87 Seite 25] faßte die Ziele der Benutzungsschnittstellengestaltung kurz und prägnant zusammen:

*„Using a computer should always be easier than not using a computer.“*

## Methoden

Die genormten Gestaltungsgrundsätze geben dem Entwickler noch keine Methoden an die Hand, um benutzer- und aufgabenorientierte Software zu erstellen. Eine Hilfe sind die folgenden Prinzipien guten Designs [No89, Sh92]:

- Dinge sichtbar machen
- ein gutes konzeptionelles Modell liefern
- ein natürliches/intuitives Mapping benutzen
- ausreichendes Feedback geben

Teils durch die Aufgabenstellung, teils durch eine Featuritis des Entwicklers oder gar der Benutzer/Auftraggeber wird Software oft übermäßig komplex. Komplexität steht aber gerade im Gegensatz zur Benutzbarkeit. Einen Ausweg aus diesem Dilemma bietet die Verwendung von Metaphern [Eri90].

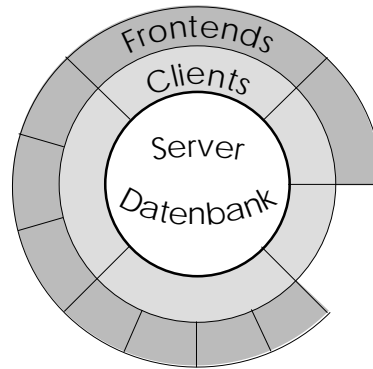
Metaphern machen ein System für den Benutzer überschaubar und vorhersehbar. Sie unterstützen somit die Gestaltungsgrundsätze 2–4. Eine eventuelle Einschränkung der Funktionalität muß dabei allerdings manchmal in Kauf genommen werden.

## 1.3 Vorgaben

Durch die Einbettung dieser Arbeit in das DFG Projekt *„Anthropotechnische Untersuchung der Mensch-Maschine-Schnittstelle in einem kooperativen Air Traffic Management“* müssen grundlegende Entwurfsentscheidungen mit den Projektmitgliedern abgestimmt werden. Dazu gehören insbesondere der Aufbau des Flugsicherungssimulationssystems und die Wahl der Entwicklungsumgebung.

## Das Flugsicherungssimulationssystem

Das Flugsicherungssimulationssystem ist als Client-Server System in einem TCP/IP-Netzwerk konzipiert. Der Aufbau ist hierarchisch wie in Abbildung 1-1 dargestellt.



1-1: Der Aufbau des ATC-Simulationssystems

Das System besteht aus einem Server, der das Radar simuliert. Mit ihm kommunizieren die Clients über eine netzwerkfähige IPC. In einer zentralen Datenbank werden statischen Informationen über Lufträume und Wegpunkte abgelegt. Die Datenbank muß über das Netzwerk auch von den Clients und Frontends erreichbar sein.

Jeder Client stellt einen Arbeitsplatz dar. Im Rahmen des Simulationssystems können noch zusätzliche Clients für die FMS-Simulation, einen Instructor, der die Aufgaben nicht realer Piloten und Lotsen übernimmt, und Flugsimulatoren existieren.

Für jeden Client können ein oder mehrere Frontends existieren, die die Benutzungsschnittstelle des Clients sind und Informationen darstellen und Eingaben ermöglichen. Jedes Frontend ist genau einem Client zugeordnet und läuft als eigenes Programm auf dem selben Rechner wie der Client.

## Die Entwicklungsumgebung

Entsprechend der Aufgabenstellung auf Seite 1 soll die Entwicklung mit Hilfe von Tcl/Tk unter X-Windows erfolgen. Diese Vorgabe impliziert eine Entwicklung unter UNIX. Damit kann bei der Entwicklung auf ein ausgereiftes, netzwerkfähiges Multiuser-/Multitaskingsystem [Bu96] zurückgegriffen werden.

Tcl [Ou95] ist ein Interpreter, der für die Softwareentwicklung nach dem Prinzip des Rapid-Prototyping gut geeignet ist. Die imperative Programmiersprache bietet mächtige Kommandos und eine klare Syntax. Durch die Erweiterung Tk lassen sich auch grafische Benutzungsschnittstellen auf einfache Weise erstellen. Der Quelltext von Tcl/Tk ist durch die Mächtigkeit der Befehle sehr kompakt und bleibt somit bei einer diszi-

plinierten Programmierung übersichtlich und relativ gut wartbar. Mit den Versionen Tcl 7.5 und Tk 4.1 sind insbesondere die Netzwerkfähigkeiten erweitert und die Portierung auf andere Systeme vorangetrieben worden.

Als Interpreter benötigt Tcl bei komplexeren Anwendungen wie dieser Arbeit eine erhebliche Rechenleistung und ausreichende Speicherkapazität. Am Institut für Luft- und Raumfahrt standen für die Entwicklung zwei IBM RS6000 zur Verfügung. Zeitweise wurden auch SUN Workstations des Fachbereichs Informatik benutzt.

Die synthetische Radardatendarstellung erfordert qualitativ hochwertige Bildschirme mit einer hochauflösenden Farbdarstellung. Dabei ist weniger die Farbtiefe als die Anzahl der Bildpunkte wichtig.



## 2 Modellierung und Entwurf

### 2.1 Vorbereitungen

#### Existierende Systeme

Vor der Entwicklung eines neuen Systems sollte der bisherige Stand festgehalten und eine Übersicht vorhandener Lösungen und Standards zusammengestellt werden. Dies hilft, gute vorhandene Ansätze zu übernehmen und existierende Fehler zu vermeiden.

Die DFS setzt in der Regionalkontrollstelle Berlin momentan das DERD-XL System ein. In Abbildung 2-1 ist ein typischer Lotsenarbeitsplatz zu sehen. Der Radarbildschirm ist dabei ein reines Überwachungsgerät.



2-1: Lotsenarbeitsplatz des DERD-XL Systems

Direkt am Bildschirm kann nur die Darstellung konfiguriert werden. Dabei muß oft auf eine Kombination aus Rollkugel- und Tastaturbedienung zurückgegriffen werden. Die Anweisung erteilen die Lotsen über Funk. Weitergehende Informationen sind über das WIAS zu beziehen, die über Nummerncodes aufgerufen werden.

Das System wurde im Rahmen einer Fluglotsenbefragung ausführlich untersucht, siehe Abschnitt „Benutzerbefragung“ auf Seite 10.

EUROCONTROL entwickelt im Rahmen der ODID-Gruppe einen Prototypen für zukünftige Lotsenarbeitsplätze [ODID94]. ODID IV ist — entsprechend COPS, siehe unten — ein fensterorientiertes System auf der Basis von X-Windows. Die Bedienung erfolgt ausschließlich mit Hilfe einer Maus. In Abbildung 2-2 auf Seite 8 ist ein Bildschirmfoto des Systems zu sehen.



2-2: Ein ODID IV Bildschirm

Das ODID-System verwendet eine Vielzahl von Fenstern. Neben dem Radarfenster und einer Knopfleiste zu dessen Konfiguration sind dies Fenster für Flugverlauf und -profil, Konfliktvergrößerung und -erkennung, Einflug- und Landelisten sowie ein- und ausgehende Nachrichten. Neben üblichen Möglichkeiten wie Kartenauswahl und der Einstellung von Darstellungsbereich und -größe bietet ODID IV automatisches Labelshifting, skalierbare Geschwindigkeitsvektoren, dynamische<sup>1</sup> Radarlabel, Unterstützung bei der Konflikterkennung und anderes mehr.

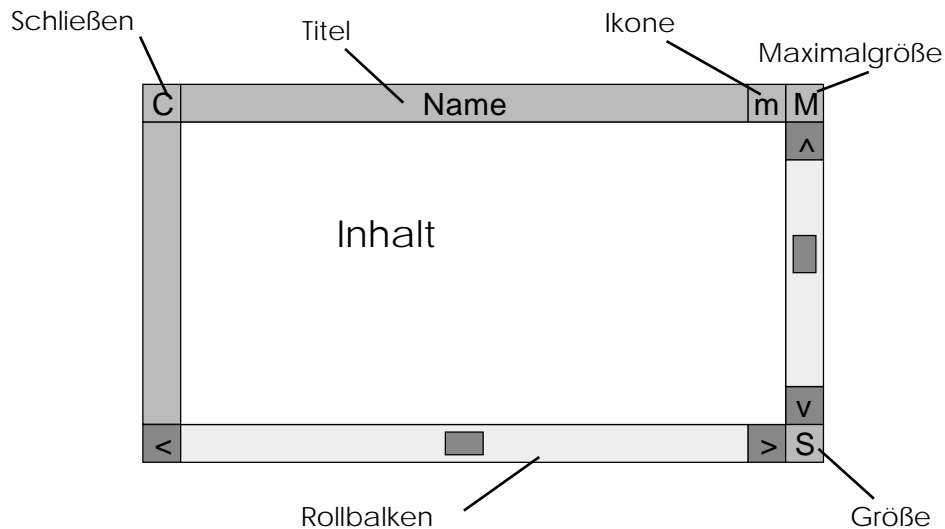
Des weiteren gibt EUROCONTROL mit COPS eine Richtlinie für zukünftige Systeme der Flugsicherung heraus [COPS91]. Neben vielen Hardwarevoraussetzungen, die für das Projekt und diese Arbeit wenig interessant sind, enthält die Spezifikation einige grundlegende Aussagen zur Entwicklungsumgebung, zur Gestaltung des Fenstersystems und zur Realisierung der synthetischen Radardatendarstellung.

COPS gibt als Entwicklungsumgebung ein UNIX/X-Windows System vor, wie es auch in diesem Projekt verwendet wird. Als Programmiersprache wird Ada empfohlen. Da diese Programmiersprache übermäßig komplex und die damit erstellten Programme entsprechend fehlerträchtig

1 Die Label ändern ihre Informationsmenge auf Grund von Bedarfsregeln. Dadurch werden die Label nicht unnötig groß, das Wechseln der Informationsinhalte und -positionen kann aber auch leicht zu Verwirrung führen!

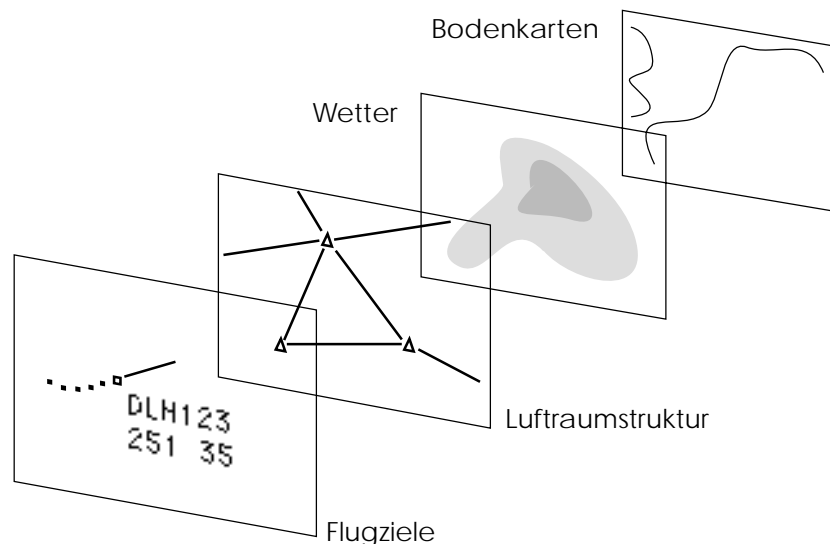


tig sind<sup>2</sup>, wird hier durch die Verwendung von Tcl/Tk von COPS abgewichen.



2-3: COPS-Fensterlayout

Für die Benutzungsschnittstellengestaltung gibt COPS ein WIMP-System vor, wie es auch X-Windows ist. Dazu wird ein Layout für die Gestaltung der Fensterrahmen angegeben. Wie in Abbildung 2-3 zu sehen, ist dieses jedoch mit keinem existierenden Standard konform, sondern eher eine Mischung aus vielen existierenden Systemen.



2-4: Das COPS-Folienmodell

<sup>2</sup> In [DI88] ist unter dem Eintrag *Ada* zu lesen: „Bekannte Informatiker warnen daher vor der Anwendung von *Ada* insbesondere in Bereichen, bei denen die Sicherheit im Vordergrund steht.“ Diese Aussage disqualifiziert den Einsatz von *Ada* in der Flugsicherung und führt die COPS-Aussage „*Ada is the future programming language*“ ad absurdum.

Für die synthetische Radardatendarstellung wird ein Folien-Modell nahegelegt, was intuitiv verständlich und eine sinnvolle Überlagerung der Informationen bietet. Dieses Modell wird in Abbildung 2–4 auf Seite 9 wiedergegeben.

## Benutzerbefragung

Zu Beginn des Projektes erfolgte eine Befragung aktiver Fluglotsen der DFS in der RK Berlin-Tempelhof. Dazu erstellte die Psychologin Genia Grundmann in Zusammenarbeit mit Beate Ulbrich vom Projekt „*Untersuchung zum Einsatz von ADS in der Flugsicherung unter Berücksichtigung operativer und kognitiver Aspekte der Fluglotsentätigkeit*“ einen Fragebogen zur Bewertung des DERD-XL Systems aus Sicht der Lotsen. Die Auswertung [GrUI96] erbrachte für das weitere Projekt wichtige Systemmängel und Verbesserungsvorschläge.

Die Anordnung der Bedienelemente des DERD-XL Arbeitsplatzes ist zu nahe an der Armablage, was leicht zu unbeabsichtigten Fehlbedienungen führt — siehe auch Abbildung 2–1 auf Seite 7. Insgesamt besteht der Arbeitsplatz aus einer übermäßig großen Anzahl von Bedienelementen.

Die Tastatur wurde kritisiert, da zum einen unbenutzte Tasten vorhanden sind, zum anderen wünschenswerte Funktionen nicht realisiert sind. Die Beschriftung der Tasten besteht aus zum Teil schwer erlernbaren Abkürzungen. Die Bedienung erscheint kompliziert, da sehr häufig zwischen Tastatur und Rollkugel gewechselt werden muß [DERD94].

Beim der synthetischen Radardatendarstellung bemängelten die Lotsen besonders das ungenügende Labelshifting und die fehlende Unterstützung beim Bestimmen von Entfernungen zwischen Bildschirmobjekten. Die vorhandenen Möglichkeiten der Karteneinblendungen sind ausreichend. Durch die einheitliche Farbgebung führt die Einblendung mehrerer Karten allerdings zu einer unübersichtlichen Darstellung.

Der Umgang mit den Kontrollstreifen wurde als problematisch erkannt. In Zeiten hohen Verkehrsaufkommens werden die handschriftlichen Eintragungen leicht unleserlich — siehe auch Abbildung 2–22 auf Seite 29.

Besonders schlecht schnitt das WIAS in der Bewertung der Lotsen ab. Das System wurde in Funktionalität, Informationsumfang, Bedienung und Darstellung als mangelhaft bewertet. Die reine Textdarstellung ist in Gliederung und Farbgebung oft unübersichtlich. Die BTX-ähnliche Bedienung über die Eingabe von Seitennummern ist nicht intuitiv und kaum zu merken. Die vorhandenen Informationen sind oft nicht vollständig oder veraltet. Als Effekt sehen viele Lotsen von der Benutzung des WIAS weitgehend ab.

Das neue System sollte in Bezug auf Bedienung, Funktions- und Informationsumfang optimiert werden. Dabei muß die Integration des Bord/Boden-DataLinks berücksichtigt werden. Als Ziel soll die Belastung der

Fluglotsen verringert und so neue Flugsicherungskapazitäten gewonnen werden. Dazu wird folgendes für das neue System gefordert:

- Integration elektronischer Kontrollstreifen
- Integration/Neuentwurf des WIAS
- Erweiterung des Labelshifting
- Entfernungsbestimmung
- Überarbeitung der Farbgestaltung
- Überprüfung der Labelinhalte
- Weitergehende Darstellung von Flugplan bzw. -verlauf

Im Verlauf der Befragung stellten sich auch Befürchtungen der Fluglotsen für die Zukunft heraus, die zu drei weiteren Forderungen führen:

1. Erhalt des Sprechfunks, da diese direkte Verbindung zwischen Lotse und Pilot nicht nur in Notsituationen als unersetzlich angesehen wird. Außerdem werden auch indirekt Informationen vermittelt, beispielsweise über Streßsituationen im Cockpit oder Sprachkenntnisse der Crew.
2. Die Kommunikation über DataLink benötigt ein ausreichendes Feedback, damit sie für den Lotsen nachvollziehbar bleibt.
3. Eine Unterstützung der Konflikterkennung ist durchaus erwünscht, jedoch keine automatische Lösung. Eventuell sind Lösungsvorschläge für die Lotsen akzeptabel.

## 2.2 Eine Styleguide

Ziel des Projektes ist ein integriertes System aus synthetischer Radar-darstellung, elektronischen Kontrollstreifen und WIAS. Ein solches System ist extrem komplex und stellt hohe Ansprüche an die Gestaltung der Benutzungsschnittstelle, um benutzbar und damit nützlich zu bleiben.

Die Bedienung soll zum überwiegenden Teil über direkte Manipulation<sup>3</sup> von Bildschirmobjekten erfolgen. Der Lotse soll bei Routineaufgaben entlastet werden. Dazu müssen die zur Überwachung des Luftraumes notwendigen Bedienungen über ein Eingabemedium realisiert werden. Die Integration aller Anzeige und Bedienelemente kann den Lotsen in seiner Tätigkeit entlasten, da zusammengehörige Informationen auch im Kontext dargestellt und eventuell direkt manipuliert werden.

---

3 Schon Shneiderman gibt die Flugsicherung als typisches Anwendungsgebiet der direkten Manipulation an [Sh92 Abschnitt 2.4.3]! Auch ODID IV arbeitet mit direkter Manipulation.

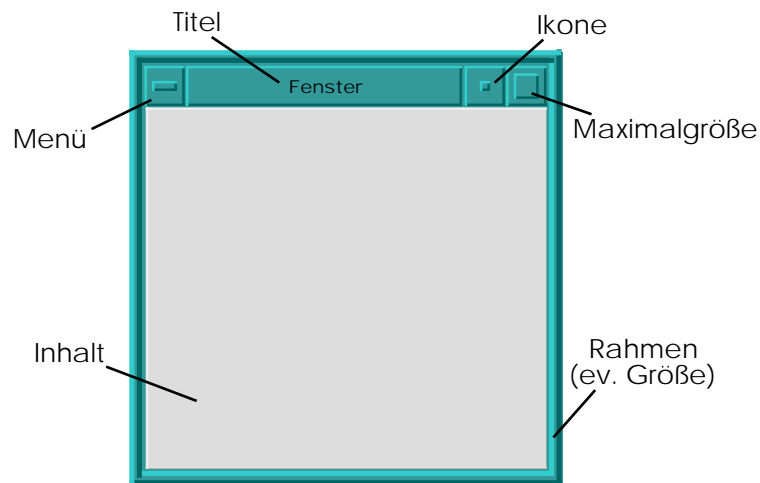
## Vorlagen

Eine Styleguide enthält genaue Angaben über das Aussehen, Verhalten und den Einsatz von Elementen der grafischen Benutzungsoberfläche. Es ist wichtig dies einheitlich für ein System festzulegen, damit alle Applikationen ein einheitliches „Look & Feel“ besitzen. Nur so erscheint das System dem Benutzer konsistent und vorhersehbar — entsprechend dem Gestaltungsgrundsatz der Erwartungskonformität.

Für das X-Window System existiert leider keine einheitliche Styleguide und auch keine einheitlichen Anzeige- und Bedienelemente. Statt dessen existieren eine Vielzahl von verschiedenen Gestaltungsrichtlinien. Eine sehr verbreitete ist die *OSF/Motif Styleguide* [MSG92]. Andere Systeme besitzen normalerweise nur eine Styleguide, die im Idealfall den gesamten Bereich von den Grundlagen der GUI-Gestaltung bis zu detaillierten Anweisungen und Beispielen zur Anwendung abdecken wie die *Macintosh Human Interface Guidelines* [MHIG92].

Für das ATC-System wird die Motif-Styleguide als Vorbild genommen. In Details und besonderen Fällen weicht das System aus folgenden Gründen ab:

1. Tcl hält sich nicht vollständig an die Motif Styleguide.
2. Die Motif Styleguide deckt Elemente wie Knopfleisten nicht ab.
3. Für das ATC-System werden teilweise Fensterlayouts benötigt, die nicht in den Motif-Fensterklassen enthalten sind.



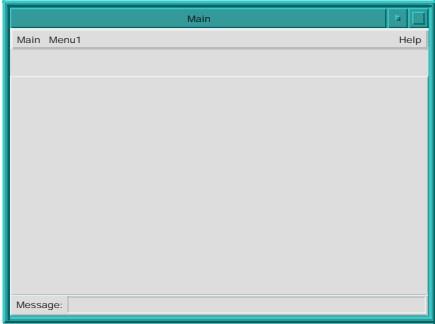


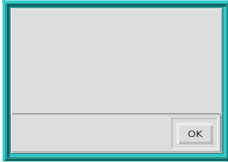
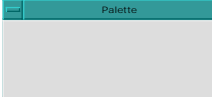
2–5: mwm Fensterdekoration

## Fensterklassen

Passend zu der Gestaltungsrichtlinie muß für ein X-Window System auch ein Window-Manager gewählt werden. Der Motif Window-Manager `mwm` entspricht diesem erwartungsgemäß. Darauf aufbauend können Fensterklassen für die benötigten Einsätze mit ihren Inhalten und De-

korationen definiert werden. Abbildung 2–5 auf Seite 12 zeigt die möglichen Dekorationen des `mwm` mit ihren Funktionen.

### 2–6: Fensterklassen

Klasse	Dekoration	Einsatz/Beispiel
Main		ständige/langfristige Anzeigen Radar
Dialog		Einstellungen Voreinstellungen
ModalDialog		Zusatzeingaben zu Kommandos Arbeitsplatzübergabe
Alert		Meldungen, Nachfragen Fehlermeldung
Palette		ständig im Vordergrund liegende Zusätze A/C-Information

Für das ATC-System werden typische Fenster wie Hauptfenster, Dialoge und Meldungsboxen benötigt. In wenigen Fällen werden modale Dialoge und Paletten eingesetzt. Da jede dieser Fensterklassen unterschiedliche Bedieneigenschaften besitzt, benötigt sie zur klaren Unterscheidung eine eindeutige Dekoration. In Tabelle 2–6 auf Seite 13 sind die Klassen mit ihren Dekorationen, Einsatzgebieten und einem typischen Aufbau des Fensterinhaltes aufgeführt.

## Bedien- und Anzeigeelemente

Tk stellt eine große Menge von einfachen Bedien- und Anzeigeelementen für die Gestaltung grafischer Benutzungsoberflächen in Form von Widgets zur Verfügung. Aussehen und Verhalten der Widgets ist an Motif angelehnt, jedoch sind nicht alle Motif-Elemente verfügbar und teilweise in ihrem Aussehen stark konfigurierbar. Diese unter X11 übliche übermäßige Konfigurierbarkeit führt sehr leicht zu einer inkonsistenten Gestaltung.

Aus diesem Grunde ist eine objektorientierte Programmierung für die Gestaltung von Benutzungsschnittstellen die beste Wahl. Mit der Konstruktion von Klassen können sehr leicht die Elemente der Benutzungsoberfläche entworfen und automatisch konsistent gehalten werden. Durch die Vererbung würden auch spezialisierte Elemente die Eigenschaften von der Oberklasse erben und ein einheitliches „Look & Feel“ vermitteln.

Leider steht diese Möglichkeit hier nicht zur Verfügung. Die Erprobung einer Erweiterung von Tcl um objektorientierte Sprachelemente war zwar erfolgreich, wurde jedoch aus Performancegründen verworfen. Für eine Weiterentwicklung in einem weniger engen Zeitrahmen wäre eine Umsetzung dieser Erweiterung von Prozeduren in C-basierte Tcl-Kommandos sicherlich sinnvoll. Auch ein Einsatz der existierenden Erweiterung [incr Tcl] ist denkbar.

Als Kompromiß werden in Tabelle 2–7 die in dem System verwendeten Anzeige- und Bedienelemente mit ihrer Widgetensprechung und den geforderten Optionen aufgeführt.

2–7: Anzeige- und Bedienelemente

Anzeige-/ Bedienelement	Tk-Widget	Tk-Optionen	Einsatz
Fläche	frame	border 1, relief raised	Grundfläche, die den Hauptbereich des Fensters ausfüllt und andere Elemente beinhaltet, kann mehrfach übereinander vorkommen
Rahmen	frame	border 2, relief groove	Rahmen um mehrere zusammengehörige Elemente
Bezeichner	label	relief flat, anchor east	Titel links von anderen Anzeigeelementen
Anzeige	label	relief flat	Anzeige von veränderlichen jedoch hier nicht zu beeinflussenden Informationen
	frame	border 2, relief groove	
Knopf	button	border 2, relief raised	zu Ausführen von Kommandos
	frame	border 1, relief sunken	Hervorhebung des Standardknopfes

## 2-7: Anzeige- und Bedienelemente

Anzeige-/ Bedienelement	Tk-Widget	Tk-Optionen	Einsatz
Menübalken	frame	border 2, relief raised	Balken oben über die ganze Breite des Fensters zur Aufnahme von Pull-Down Menüs
Menütitel	menubutton	relief flat	Titel eines Pull-Down Menüs im Menübalken
Knopfleiste	frame	border 2, relief flat	Balken unterhalb des Menübalkens zur Aufnahme von direkt zugänglichen Bedienelementen
Eingabefeld	entry	border 2, relief sunken	Feld zur Texteingabe — sollte möglichst wenig benutzt werden, da Maus Haupteingabegerät
Ankreuzknopf	checkboxbutton		zum Ein-/Ausschalten einer Wahlmöglichkeit
Auswahlknopf	radiobutton		zur Auswahl einer aus 2 bis ca. 7 festen Wahlmöglichkeiten
Menüknopf	menubutton	indicator, border 2 relief raised	Knopf mit Auswahl-Menü, das bei Drücken des Knopfes erscheint — für mehr als ca. 5 oder veränderlich viele Wahlmöglichkeiten statt mehrerer Auswahlknöpfe

## 2.3 Modularisierung

Um die Entwicklung komplexer Software mit imperativen Programmiersprachen zu vereinfachen ist die Modularisierung ein wichtiges Hilfsmittel. Mit ihrer Hilfe bleiben auch umfangreiche Softwareprojekte übersichtlich und damit test- und wartbar.

### Kriterien

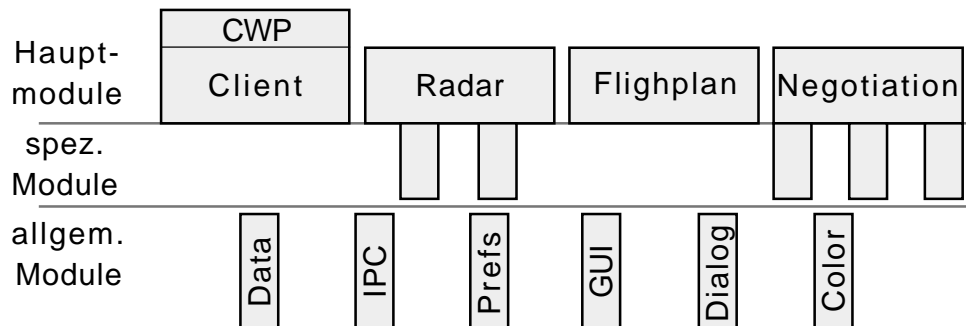
Für die Modularisierung können verschiedene Kriterien herangezogen werden. Typische Kriterien sind

- logische Zusammengehörigkeit,
- bearbeitete Datenstrukturen und
- Häufigkeit/Zeitpunkt des Bedarfs.

Oft ist eine Kombination oder gemischte Verwendung dieser Kriterien sinnvoll. Für das ATC-System wird dies auch angewendet. Die Modula-

risierung ist hierarchisch in drei Ebenen aufgebaut wie in Abbildung 2–8 dargestellt.

1. Die Hauptmodule stellen die ausführbaren Programme, also Client und Frontends dar.
2. Spezielle Module enthalten ausgelagerte Teile einzelner Programme.
3. In allgemeinen Modulen sind Teile zusammengefaßt, die von mehreren Programmen benötigt werden.



2–8: Modularisierung des ATC-Systems

### Client- und Frontendmodule

Der Client ist in zwei Hauptmodule aufgeteilt. Das Modul Client enthält dabei den eigentlichen Programmcode und wird vom Modul CWP aus eingebunden. In CWP werden die Pfade zum Nechladen und zur Ausführung der Frontends gesetzt. Damit sind diese nicht nur für den Client definiert, sondern werden durch die Vererbung der Umgebungsvariablen unter UNIX auch an vom Client gestartete Frontends weitergegeben.

Ein Hauptmodul enthält typischerweise Prozeduren wie Open, Close, Load, Save und Quit. Diese können auch von außerhalb des Programmes aufgerufen werden. Beim Client erfolgt dies beim Start automatisch durch CWP, bei den Frontends kann Open durch den Client aufgerufen werden.

Das Radar-Frontend besteht neben dem Hauptmodul aus einem Modul zur Darstellung der Karten und einem für die Flugziele. Ein Modul für das Lesen und Schreiben der Kartendatenbank ist als allgemeines Modul einzuordnen, da es auch von Programmen zur Wartung der Datenbank genutzt würde.

Bei den elektronischen Kontrollstreifen kann die Streifendarstellung und -bearbeitung von dem Hauptmodul getrennt werden.



Aus dem Verhandlungsführungs-Frontend können spezielle Module zur Verhandlungsverwaltung, zur Darstellung von Flugverläufen und -profilen ausgelagert werden.

## Allgemeine Module

Neben dem schon angesprochenen allgemeinen Modul zum Zugriff auf die Datenbank existieren noch weitere von mehreren Programmen genutzte Module.

Alle zur Interprozeßkommunikation benötigten Routinen können in ein separates Modul ausgelagert werden. Ebenso die Unterprogramme zum Lesen und Schreiben der Voreinstellungen.

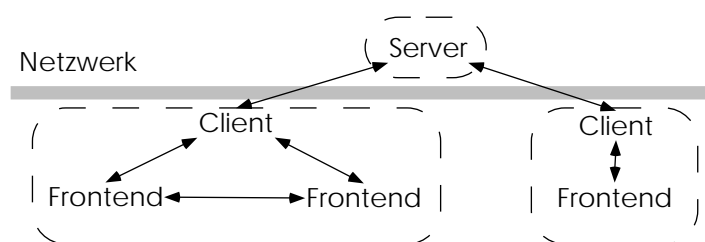
Für eine konsistente Gestaltung der Benutzungsschnittstelle ist es sinnvoll, Routinen zur Erstellung von Anzeige- und Bedienelementen zentral zu verwalten. Darüber hinaus können auch größere Einheiten wie Dialogfenster und Meldungsboxen einheitlich erzeugt werden.

Besondere Dialoge wie die Farbauswahl sollten auch als eigenständiges Modul existieren. Hier können auch Routinen zur Umrechnung zwischen verschiedenen Farbmodellen abgelegt sein.

## 2.4 Schnittstellen

Der Aufbau des ATC-Systems aus mehreren parallel arbeitenden Programmen bedingt eine Interprozeßkommunikation mit einer Vielzahl möglicher Verbindungen. Diese Verbindungen stellen Schnittstellen dar und müssen in Bezug auf Realisierung, Protokoll und Befehlsumfang festgelegt werden.

Um die Kommunikation übersichtlich und kontrollierbar zu halten, werden viele der möglichen Verbindungen nicht direkt realisiert, sondern durch indirekte Verbindungen über den Client oder Server ersetzt. Daraus ergibt sich das Kommunikationsmodell in Abbildung 2-9.



2-9: Kommunikationsmodell des ATC-Systems

## Server-Client

Die Kommunikation zwischen Client und Server muß entsprechend dem Abschnitt „Das Flugsicherungssimulationssystem“ auf Seite 4

netzwerkfähig sein. Dazu bietet sich eine IPC über Stream-Sockets mit Internet-Adressierung an. Diese wird von Tcl auch durch das Kommando `socket` unterstützt.

Für den Verbindungsaufbau wird von der Client-Seite dann der Name des Rechners, auf dem der Server läuft, und die Nummer des Ports, über den der Server erreichbar ist, benötigt. Für das Simulationssystem am ILR wurden diese wie folgt gewählt:

```
visual.fb12.tu-berlin.de:2704
```

Um den Server auf einfache Weise testen und warten zu können und Kodierungsunterschiede zwischen verschiedenen Rechnerarchitekturen zu umgehen, wird für die IPC ein Klartext-Protokoll verwendet. Ein solches rein auf ASCII basierendes Protokoll ist im Internet sehr üblich<sup>4</sup> und ermöglicht einen einfachen Verbindungsaufbau zum Server mit einem Telnet-Client. Die textuelle Codierung ist zwar mit einem erheblichen Overhead verbunden, bei den in dem System zu erwartenden Datenaustausch scheint dies aber im Verhältnis zu dem Gewinn an Kontrollierbarkeit und einfacheren Tcl-Schnittstellen unproblematisch.

Über die Client-Server Schnittstelle müssen sich die Clients identifizieren, Radardaten empfangen und Nachrichten austauschen. Die dazu nötigen Kommandos sind in Tabelle 2–10 aufgeführt. Dem Protokoll sind spätere Erweiterungen leicht hinzuzufügen.

2–10: IPC-Kommandos Client-Server

Kommando	von		Parameter	Bedeutung
	S	C		
LOG_IN		■	Arbeitsplatzkennung, Clienttyp	Anmeldung beim Server
TIME	■		Systemzeit	Synchronisation der Uhr
QUIT		■		schließt die Verbindung
RAD_UPD	■		START/END	der Beginn/das Ende eines Radarumlaufes
RAD_DAT	■		Callsign, Longitude, Latitude, Flightlevel, Airspeed, ...	Radardaten eines Flugzieles
RAD_CLP		■	Long./Lat.-Rechteck	Client setzt einen Ausschnitt für an ihn zu sendende Flugziele
MSG	■		Absender, Text	Nachricht
		■	Adressat, Text	

<sup>4</sup> Viele wichtige Internet-Protokolle sind Klartext-Protokolle wie Telnet, FTP, SMTP und HTTP — siehe auch [ScBoGeKa94].

## 2-10: IPC-Kommandos Client-Server

Kommando	von		Parameter	Bedeutung
	S	C		
FPL_GET		■	Callsign	Flugplananforderung
FPL_PUT	■		Callsign, WP, Zeit, ...	Flugplanänderung

## Client-Frontend

Ein Client und die dazugehörigen Frontends sind mit einem X-Server verbunden. Dadurch lassen sie sich mit der Tk eigenen IPC des `send`-Kommandos verbinden. Diese ist über X-Events realisiert und ermöglicht das Versenden beliebiger Tcl-Kommandos an jeden Tcl/Tk-Interpreter, der mit dem selben X-Server verbunden ist.

Auf dieser Basis besteht die Client-Frontend Schnittstelle nur noch aus der Vereinbarung gemeinsam benutzbarer Prozeduren. Theoretisch könnten beliebige Kommandos wie auch `set` oder gar `rename` gesendet werden. Dies würde jedoch leicht zu unbeabsichtigten Seiteneffekten führen. Aus diesem Grunde wird die Schnittstelle auf einige wenige festgelegte Prozeduraufrufe eingeschränkt, die in Tabelle 2-11 wiedergegeben sind.

## 2-11: IPC-Kommandos Client-Frontend

Kommando	Parameter	Bedeutung
Open		öffnet ein neues Hauptfenster
Close	Fenster	schließt ein Hauptfenster
Load		Voreinstellungen öffnen
Save		Voreinstellungen sichern
Preferences	(Liste mit Voreinstellungen)	Voreinstellungen als Liste zurückgeben/übernehmen
Quit	(sofort?)	Applikation beenden

Auch diese Schnittstelle ist im Einzelfall sehr leicht zu erweitern, indem zusätzliche Kommandos als extern aufrufbar vereinbart werden.

## Indirekte Verbindungen

Andere Schnittstellen als die in den beiden vorherigen Abschnitten beschriebenen werden im ATC-System nur indirekt verwirklicht. So findet die Kommunikation zwischen zwei Clients über den Server und die IPC zwischen zwei Frontends über den Client statt.

## 2.5 Ein Client

### Die Clients des ATC-Systems

Im Rahmen des Simulationssystems können verschiedene Arten von Clients vorhanden sein. Im Rahmen dieser Arbeit wird jedoch nur der Client des Lotsenarbeitsplatzes betrachtet. Andere Clients werden im Rahmen des Projektes teils von anderen Mitgliedern realisiert, teils nur angedacht.

Der Server benötigt eine FMS-Simulation, die aus Flugplänen virtuelle Flugprofile erstellt. Diese Simulation wurde schon zu einem früheren Zeitpunkt unter DOS erstellt und für dieses Projekt nach UNIX portiert. Für den Einsatz als Client mußte die Datei-Schnittstelle durch eine Socket-Schnittstelle ersetzt werden.

Im Rahmen dieser Arbeit wird ein CWP-Client entworfen und implementiert. Er stellt den gesamten Arbeitsplatz eines Fluglotsen im System dar. Von dieser Art Client können im ATC-System beliebig viele vorhanden sein. An jeden dieser Clients können Frontends wie die Radardatendarstellung angekoppelt werden.

Für das zukünftige Simulationssystem ist ein Instructor-Client vorgesehen. Über ihn kann ein Überwacher die Reaktionen ein oder mehrerer Piloten und Losten benachbarter Sektoren simulieren und das Verhalten des Servers beeinflussen.

Als weitere Möglichkeit bestände die Einbindung eines Flugsimulators. Zur Zeit wird der A340-Simulator des ZFB über eine Firewall mit dem Netzwerk des ILR verbunden. Dieser könnte dann über einen speziellen Client mit dem ATC-System verbunden werden.

### Der CWP-Client

Der Client für den Lotsenarbeitsplatz hat mehrere Funktionen innerhalb des Systems:

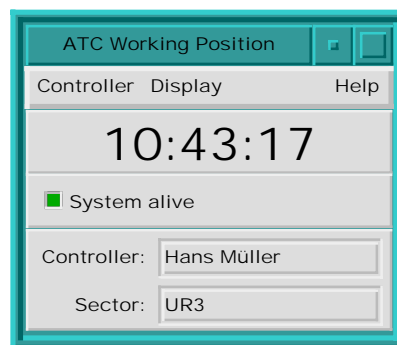
**Multiplexer für die Kommunikation:** Der Client baut eine Socket-Verbindung zum Server auf. Kommandos, die über diese Verbindung gelesen werden, müssen an entsprechende Frontends weitergeleitet werden.

**Identifikation des Arbeitsplatzes:** Für den Arbeitsplatz muß ein Sektor eingestellt werden können. Bei der Übernahme des Arbeitsplatzes muß sich der neue Lotse identifizieren, damit das System dessen Voreinstellungen benutzen kann.

**Verwaltung der Voreinstellungen:** Die Voreinstellungen für den Sektor (z.B. Höhenband) und des Lotsen (z.B. Farbwahl und Fensterplatzierung) sollten zentral verwaltet werden, da dies für den Lotsen, der das System als Ganzes sieht, einfacher zu handhaben ist.

**Kontrolle der Frontends:** Vom Client aus müssen die zu einem Arbeitsplatz gehörenden Frontends gestartet, beendet, gesichert und zurückgesetzt werden können.

**Anzeige des Systemstatus:** Während des Entwurfs stellte sich heraus, daß es wenig sinnvoll ist, für den Systemstatus ein eigenes Frontend zu erstellen. Diese Anzeigen sollten in den Client integriert werden, da dieser für den Lotsen das System darstellt.



2-12: Der CWP-Client

## Fensterlayout des CWP-Client

Aus dieser Funktionalität entsteht der in Abbildung 2-12 gezeigte Entwurf der Benutzungsoberfläche des Clients. Das Fenster der Klasse Main enthält von oben nach unten folgende Teile:

**Menübalken:** Neben dem Hilfe-Menü sind zwei Menüs enthalten, die mit ihren enthaltenen Kommandos in Tabelle 2-13 aufgeführt sind. Das erste Menü enthält übergreifende Kommandos, die das gesamte System beeinflussen, das zweite eine Liste aller Frontends, die darüber aktiviert werden können.

2-13: Menüs des Clients

Menü	Beschreibung	Verfügbar
<b>Controller</b>		
System ...	aktiviert/deaktiviert das System über die Server-Verbindung	immer
Take Over ...	Übernahme des Arbeitsplatzes	System aktiv
Preferences ...	Voreinstellungen des Lotsen	System aktiv
Quit	beendet das System (z.B. Wartung)	System down

## 2-13: Menüs des Clients

Menü	Beschreibung	Verfügbar
Display		
Radar	öffnet das Radar-Frontend	System aktiv
Flight Plan	öffnet das FlightPlan-Frontend	System aktiv
Negotiation	öffnet das Negotiation-Frontend	System aktiv

**Uhr:** Für die Lotsen ist die aktuelle Systemzeit eine wichtige Information, deshalb besonders groß. Auch bei einem Serverausfall sollte diese weiterlaufen.

**Systemstatus:** Das System kann sich in drei Stati befinden, die in Tabelle 2-14 aufgeführt sind. Durch die doppelte Codierung (Text/Farbe) wird eine verstärkte Aufmerksamkeit bei Statusänderungen erreicht — dies kann durch ein akustisches Signal noch verstärkt werden.

## 2-14: Systemstati

Status	Farbkennung	Bedeutung
down	rot	keine Verbindung zum Server
stalled	gelb	Verbindung vorhanden, aber seit mehr als 10 Sekunden keine Daten bekommen
alive	grün	mit aktivem Server verbunden

**Identifikation:** Die Arbeitsplatzidentifikation — Sektor und Lotse — wird ständig angezeigt. Eine Änderung dieser Informationen ist nur über das Menükommando „*Controller / Take Over ...*“ möglich.

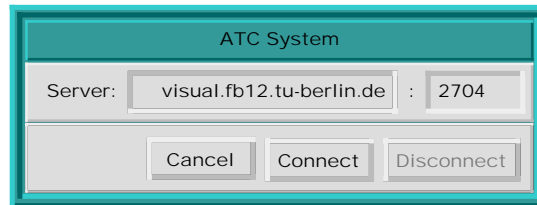
Daraufhin erscheint ein modaler Dialog wie in Abbildung 2-15 zu sehen. Dabei sind keine textuellen Eingaben nötig. Statt dessen bietet das System jeweils eine Liste der möglichen Sektoren und zugelassenen Lotsen an. Nur wenn der ausgewählte Lotse für den eingestellten Sektor zugelassen ist, wird der *OK*-Knopf freigegeben.



2-15: Dialog zur Arbeitsplatzübernahme

## Systemstatus ändern

Um den Systemstatus zu ändern, wird das Menükommando „*Controller / System ...*“ aufgerufen. Daraufhin erscheint ein modaler Dialog wie in Abbildung 2–16. Hier kann die Verbindung zu einem Server angegeben und aufgebaut oder eine existierende Verbindung unterbrochen werden.



2–16: Dialog zum System

## Voreinstellungen

Die letzte Aufgabe des Clients ist die Verwaltung der Voreinstellungen. Das System sollte für jede Arbeitsplatzidentifikation — bestehend aus Sektor und Lotse — einen eigenen Satz Voreinstellungen sichern, so daß jeder Lotse seine Arbeit wieder mit dem Bildschirmaufbau weiterführen kann, mit dem er zuletzt bei dem selben Sektor aufgehört hat. Dabei gibt es zwei Kategorien von Voreinstellungen.

**Lotssenspezifisch:** Darstellungsoptionen wie die Farbwahl

**Sektorspezifisch:** Höhenband, Fensterpositionen und -größen

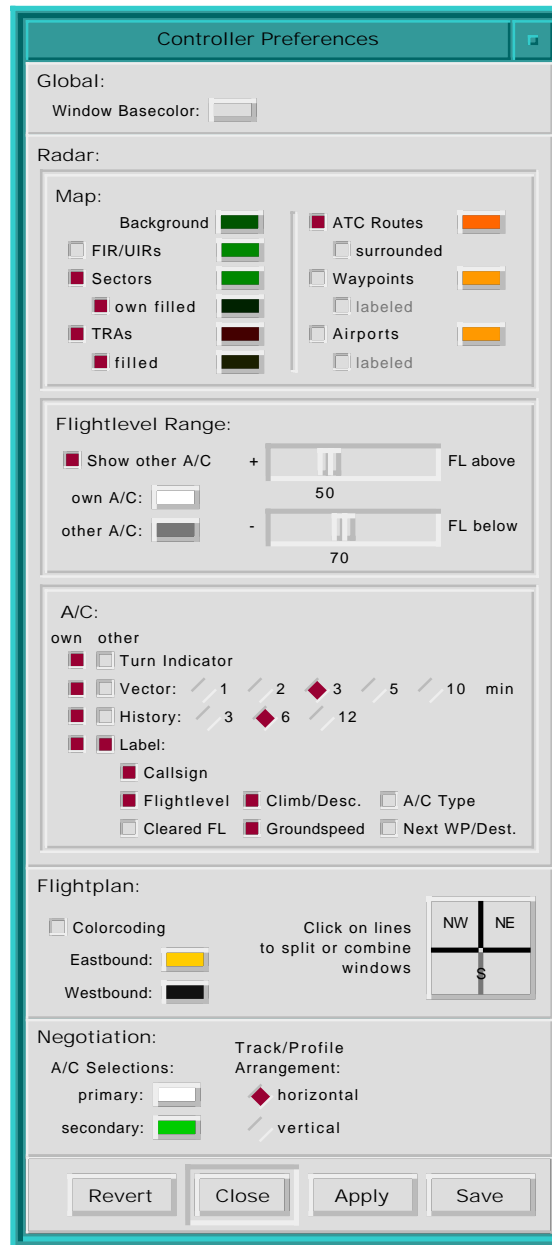
Sind keine lotssenspezifischen Einstellungen vorhanden, so werden diese von einem Systemstandard übernommen. Von den sektorspezifischen Einstellungen müssen für jeden Sektor eigene Systemvorgaben vorhanden sein.

2–17: Funktionen im Dialog Voreinstellungen

Knopf	Funktion
Revert	verwirft die Einstellungen und übernimmt die zuletzt gesicherten
Close	verwirft die Einstellungen und schließt das Fenster
Apply	wendet die Einstellungen an
Save	wendet die Einstellungen an und sichert diese

Die lotssenspezifischen Voreinstellungen werden über einen einheitlichen gemeinsamen Dialog verwaltet. Ein erster Entwurf ist die Abbildung 2–18 auf Seite 24. Die Funktionen der Knöpfe ist in Tabelle 2–17 wiedergegeben. Darin sind Einstellungen zur Farbwahl, Höhenbanderweiterung und Flugzieldarstellung im Radar, Zusammenfassung und Farbcodierung der Flugplanfenster und Farbcodierung der

Verhandlungsführung enthalten. Fensterpositionen und -größen werden durch *Save* ebenfalls gesichert und durch *Revert* verändert.



2-18: Dialog für Voreinstellungen

## 2.6 Ein Frontend

### Das Radar-Frontend

Die Radardarstellung ist eine Blick-Metapher —die Sicht des Lotsen von oberhalb des überwachten Luftraumes auf die Flugziele, Luftraumstruktur, das Wetter und den Boden.



Die synthetische Radardatendarstellung ist für den Fluglotsen zweifelsohne das wichtigste Arbeitsmittel bei der Überwachung des Luftraumes. Die grafische Darstellung der aktuellen Verkehrssituation ist ein unerläßlicher Bestandteil des Fluglotsenarbeitsplatzes und seit jeher Mittelpunkt der Lotsenarbeit.

Dieses Ergebnis brachte auch die Befragung [GrUl96] in der Vorbereitung des Projektes. Aus diesem Grunde muß das neue System auch eine Radardarstellung besitzen und diese als erste Grundlage entworfen und implementiert werden. Dabei soll dieses wichtige Hilfsmittel mit den Möglichkeiten des aktuellen Entwicklungsstandes der EDV optimiert werden. Dem Lotsen ist eine große Freiheit in Art und Menge der Informationsdarstellung zu gewähren, damit dieser seine Arbeit nach seinen individuellen Vorlieben gestalten und so effektiv durchführen kann.

## Fensterlayout des Radar-Frontend

Eine fensterorientierte Radardatendarstellung bietet neue Möglichkeiten für den Lotsen. Es können mehrere Radarfenster über- oder nebeneinander oder überlappend dargestellt werden. Trotzdem sollte ein Radar das Hauptfenster bleiben und die Mehrfachdarstellung nur in Sonderfällen, wie einer Ausschnittsvergrößerung auf Wunsch des Lotsen, hinzugefügt werden.

Sofern mehrere Radarfenster vorhanden sind, sollten diese alle die selben Einstellungen bezüglich Farbwahl und dargestellter Objekte haben, da verschiedene Einstellungen sowohl schwer zu handhaben als auch verwirrend für den Benutzer wären.

**Radardatendarstellung:** Die Radardatendarstellung in einem fensterorientierten System ist in Position und Größe veränderbar, dies entspricht auch den Vorstellungen von EUROCONTROL [COPS91]. Daraus ergibt sich zwingend die Verwendung von Rollbalken wie es bei der Fensterdarstellung von Objekten mit von der Fenstergröße abweichenden Ausmaßen üblich ist. Die Steuerung des Vergrößerungsfaktors sollte ebenfalls direkt erreichbar nahe den Rollbalken plaziert werden, da diese in der Anwendung zusammengehören.

**Menübalken:** Für selten benutzte Kommandos werden Menüs benutzt. Der Menübalken bekommt neben dem Hilfe-Menü zwei Menüs, die in Tabelle 2-19 auf Seite 26 aufgelistet werden. Das erste bezieht sich auf die Fensterverwaltung, das zweite auf die Kartendarstellung.

**Knopfleiste:** Für kurzfristige Änderungen der Flugzieldarstellung im Radar soll eine Knopfleiste vorhanden sein, die schnellen Zugriff auf die Einstellungen für Höhenband, Vektoren und Vergangenheitssymbole bietet. Außerdem muß eine Möglichkeit

## 2-19: Menüs des Radar-Frontends

Menü	Beschreibung	Verfügbar
<b>Radar</b>		
Open	öffnet ein weiteres Radarfenster	immer
Raise	zeigt ein Menü mit offenen Radarfenstern und holt das angewählte nach vorne	mehr als ein Radar
Close	zeigt ein Menü mit offenen Radarfenstern und schließt* das angewählte	mehr als ein Radar
Print ...	druckt ein Radarfenster aus	immer
Quit	schließt das Radar-Frontend	System down

<b>Map</b>		
◆ Upper	wählt den oberen Luftraum	immer
Lower	wählt den unteren Luftraum	immer
Border	Darstellung von Küstenlinien	immer
Highway	Darstellung von Autobahnen	immer
FIR	Darstellung von FIR/UIRs	immer
■ CTA	Darstellung von CTA/UTAs	immer
■ TRA	Darstellung von TRAs	immer
■ WP	Darstellung von WPs	immer
Barrier	Darstellung von Hindernissen	immer
■ Route	Darstellung von Luftstraßen	immer
■ Airport	Darstellung von Flughäfen	immer

\* Da die zusätzlichen Radarfenster weder Menü noch Knopfleiste haben, ist dies die einzige Möglichkeit des Schließens.

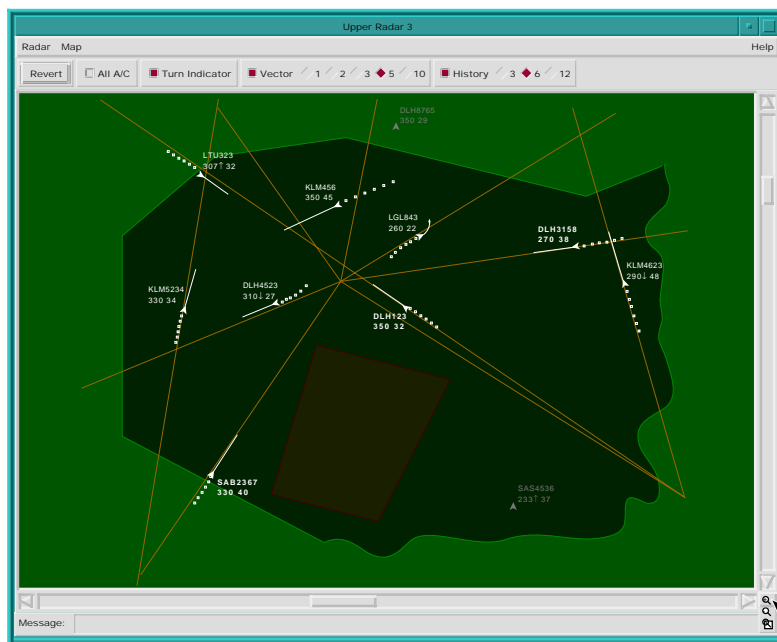
vorhanden sein, diese Einstellungen ebenso schnell wieder rückgängig zu machen, da die Lotsen solche Darstellungsänderungen oft nur kurzzeitig wünschen.

**Meldungszeile:** Für die Benachrichtigung der Lotsen über eingehende Verhandlungen und Meldungen ist ein Nachrichtenstreifen im Radar nötig, damit diese nicht übersehen werden, wenn die entsprechenden Fenster durch das Radar verdeckt sind. Eine Position am unteren Rand bietet sich dafür an. Zusammen ergibt sich ein Fenster wie in Abbildung 2-23 auf Seite 30.

## Funktionalität der Radardarstellung

Für die Überwachungsaufgaben des Lotsen stellt die Radardarstellung eine umfangreiche Funktionalität über direkte Manipulation zur Verfügung. Als Grundsätze der Funktionsbelegung gelten:

- Die Mausknöpfe sind in der Reihenfolge Links, Rechts, Mitte häufig/leicht bis selten/schwierig zu bedienen.



## 2-20: Radar-Frontend

- Der linke Mausknopf sollte nicht zu folgenschweren Aktionen führen können, statt dessen sollte dafür eher ein weniger leicht/häufig zu bedienender Knopf gewählt werden.
- Die Belegung der Mausknöpfe sollte einem einheitlichen erkennbaren Schema folgen.
- Doppelklick<sup>5</sup> sollte nur als Abkürzung für auch anderweitig erreichbare Funktionen eingesetzt werden. Ein Doppelklick sollte nur für den linken Mausknopf berücksichtigt werden.

## 2-21: Belegung der Mausknöpfe im Radar

	Links	Mitte	Rechts
allgemein	Auswahl von Objekten	objektspezifische Bearbeitungsfunktionen	Auswahlmenü mit Funktionen zu dem aktuellen Objekt

<sup>5</sup> Drei- oder gar weitere Mehrfachklicks verbieten sich auf Grund der Benutzbarkeit gänzlich.

## 2–21: Belegung der Mausknöpfe im Radar

		Links	Mitte	Rechts
A/C	Symbol	hebt das Flugziel hervor, Auswahl gilt auch für andere Frontends, eintretende Flugziele werden übernommen	Labelshifting	Menü zur Einblendung von Flugverlauf, Flugplanfenster, Informationsfenster
	Vektor		Kursänderung	
	Label		Höhenänderung	
Flughafen				Menü zum Aufruf von WIAS-Daten*

\* siehe Abschnitt 4.2

Die Bedienung mit der Maus ist entsprechend Tabelle 2–21 gestaltet. Ein Radarfenster mit mehreren durch den Lotsen eingeblendeten Zusätzen und Fenstern mit elektronischen Kontrollstreifen ist in Abbildung 2–23 auf Seite 30 zu sehen.

Die Integration eines DataLink in das ATC-System ermöglicht es dem Lotsen mehr Informationen über einzelne Flugzeuge zu erhalten, den Funkverkehr von trivialen Routineaufgaben zu befreien und einfache Anweisungen direkt zu geben. Die in der Radardatendarstellung zu integrierenden DataLink-Funktionen sind:

- Halbautomatische *An- und Abmeldung* mit automatischer Frequenzzuweisung bei Übernahme durch den Lotsen. Eintretende Flugzeuge werden blinkend dargestellt und per Masklick übernommen. Dadurch bekommen sie per DataLink die Frequenz des Lotsen übermittelt.
- *Flugverlaufsdarstellung* für ausgewählte Flugziele basierend auf deren FMS-Daten. Um die Verbindung zu den anderen im Radar enthaltenen Informationen nicht zu verlieren und die Fensteranzahl nicht unnötig zu erhöhen wird der Flugverlauf im Gegensatz zu ODID nicht in extra Fenster ausgelagert.
- aktuelle *Informationen zum Flugzeug* bezüglich Performance und Zustand
- einfache *Direktanweisungen* wie Höhen- und Kursänderungen, insbesondere für zeitkritische Notfallsituationen

Durch die zugänglichen FMS-Daten läßt sich eine Konflikterkennung realisieren. Mit diesem speziellen Gebiet und der Unterstützung der Lotsenarbeit bei der Konfliktlösung beschäftigt sich das Projekt „*Untersuchung zum Einsatz von ADS in der Flugsicherung unter Berücksichtigung operativer und kognitiver Aspekte der Fluglotsentätigkeit*“.

## 2.7 Zusätzliche Frontends

Für den CWP-Client sind zwei weitere Frontends vorgesehen, die im Groben schon hier entworfen werden. Das sind die zu integrierenden Kontrollstreifen und eine Verhandlungsführung über DataLink.

Insbesondere die Verhandlungsführung ist eine völlige Neuentwicklung, für die es kaum Ideen und gar Vorlagen gab. Der dargestellte Entwurf ist aus langen Diskussionen im Projekt hervorgegangen und auch in diesem Stadium nur eine augenblickliche Version vor weiteren Durchläufen des Rapid-Prototyping in Absprache mit Fluglotsen.

### Elektronische Kontrollstreifen

Obwohl für die momentane Form der Lotsenarbeit unersetzlich, sind die Kontrollstreifen — dargestellt in Abbildung 2-22 — ein großes Problem, sowohl für die Lotsen — siehe Abschnitt „Benutzerbefragung“ auf Seite 10 —, als auch für die Systementwicklung.

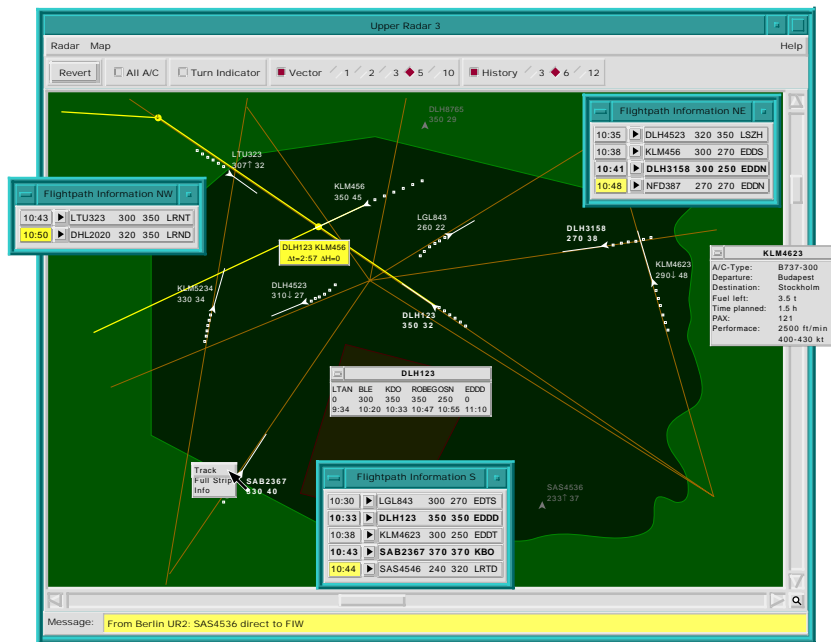


2-22: Kontrollstreifen

Kontrollstreifen in herkömmlicher Form erfordern einen hohen Aufwand für die Aktualisierung und Verteilung. Es muß sichergestellt werden, daß die gedruckten Informationen immer aktuell und bei allen betroffenen Lotsen vorhanden sind.

Diese Probleme könnten durch elektronische Kontrollstreifen besser gelöst werden als durch den Kontrollstreifendruck in heutigen Flugsicherungszentralen wie dies auch in ODID ermittelt wurde. Auf lange Sicht würde der ZKSD damit entfallen. Elektronische Kontrollstreifen könnten von System automatisch — auch in mehrfacher Ausfertigung — an betroffene Lotsen verteilt werden und könnten bei Änderungen immer auf dem neuesten Stand gehalten werden.

Einen Entwurf für elektronische Kontrollstreifen ist in Abbildung 2-23 auf Seite 30 integriert. Dort sind die Kontrollstreifen in Fenstern eines Flugplan-Frontends nach Eintrittsregionen geordnet. Die Pfeile stellen Erweiterungsknöpfe dar, mit denen zu einer vollständigen Streifendarstellung gewechselt werden kann.



## 2–23: Radarfenster mit Zusätzen

### Verhandlungsführung

Ein neuer Bestandteil des Lotsenarbeitsplatzes ist die Verhandlungsführung direkt über den Bildschirmarbeitsplatz. Diese wird erst durch Einführung eines digitalen Bord/Boden-DataLinks möglich und ist somit bisher ohne Beispiel.

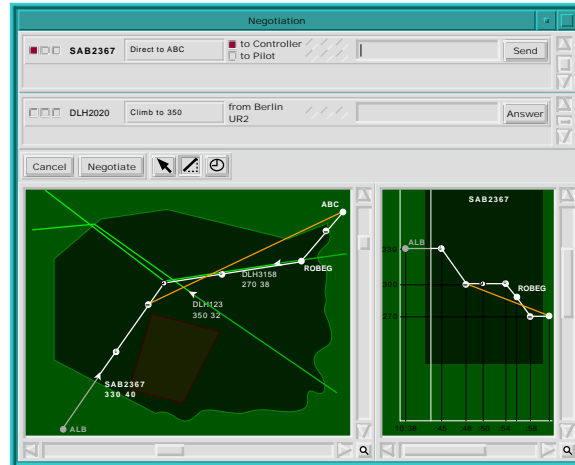
Für den Entwurf sind einige Einschränkungen und Vorgaben zu beachten:

- Die elektronische Verhandlungsführung kann hauptsächlich zur Ersetzung von einfachen Standardtätigkeiten genutzt werden. Für Sonderfälle, Notsituationen und als Sicherheit muß der Funk auch weiterhin erhalten bleiben! Dies ergibt sich nicht nur aus der Lotsenbefragung, sondern hängt auch mit den Einschränkungen eines automatischen Systems bezüglich der Möglichkeiten zu Flugverlaufsanweisungen zusammen. Eine umfassende Lösung würde hier schnell zu komplex werden.
- Die Flugverlaufsänderung muß mit dem Piloten und/oder den betroffenen Nachbarlotsen verhandelt werden. Dabei müssen alle beteiligten möglichst wenig belastet werden und dürfen trotzdem nicht das Gefühl bekommen, übergangen zu werden.
- Es erweist sich als sehr schwierig, einen vierdimensionalen Flugverlauf auf einem zweidimensionalen Arbeitsplatz nicht nur übersichtlich darzustellen, sondern auch noch einfach veränderbar zu gestalten.

Um dieses überaus komplexe und innovative Problem anzugehen, wurde ein dreiteiliges Verhandlungsführungsfenster entworfen. Für die Lösung sind auch zwei oder drei getrennte Fenster denkbar.

1. Eine Verhandlungsverwaltung, in der alle laufenden Verhandlungen repräsentiert sind und zur Beantwortung und Bearbeitung ausgewählt werden können. Dazu werden zwei getrennte Listen für eingehende und selbst initiierte Verhandlungen geführt.
2. Eine vereinfachte Radardatendarstellung, die nur ausgewählte Flugziele mit ihren Flugverläufen enthält. Hier können Flugverlaufsänderungen in einer horizontalen Ebene vorgenommen werden.
3. Ein Flugprofil, in dem die Höhe über der Zeit dargestellt wird. Hier werden Höhenänderungen und Zeitvorgaben eingestellt.

Dieser erste Entwurf benutzt eine wegpunktorientierte Herangehensweise. Als Alternative wäre auch eine tube<sup>6</sup>-orientierte Lösung möglich, die jedoch noch stärker von der heutigen Arbeitsweise der Lotsen abweicht. Während der Vorüberlegungen hat sich herausgestellt, daß eine Mischung der beiden Verfahren wenig sinnvoll weil zu komplex ist. Um die Umstellung für die Lotsen zu vereinfachen, wird deshalb zunächst ein wegpunktorientierter Ansatz verfolgt.



2–24: Verhandlungsführungs-Frontend

Der Entwurf ist in Abbildung 2–24 zu sehen. Die Verhandlungsverwaltung enthält dabei für jede laufende Verhandlung einen Streifen, der folgende Elemente enthält:

<sup>6</sup> Statt den Luftraum durch Luftstraßen zwischen Wegpunkten zu gliedern, ließe sich auch eine Unterteilung in vierdimensionale Abschnitte entlang der Flugverläufe denken. Dieser Ansatz wird auch von anderen Projekten untersucht.

**Priorität:** durch Anwählen priorisiert der Lotse diese Verhandlung und hebt sie in der Liste des Verhandlungspartners hervor

**Callsign:** das Flugzeug, um das es in der Verhandlung geht

**Kurzbeschreibung:** die Art der Flugverlaufsänderung — diese automatisch erzeugte Meldung erscheint mit dem Absender und dem Callsign auch als Nachricht in der Mitteilungszeile des Radarfensters

**Absender/Adressaten:** Auswahl/Angabe des/der Verhandlungspartner(s)

**Verhandlungsstatus:** Anzeige/Auswahl einer Reaktion durch Auswahl eines Auswahlknopfes: Ablehnung (rot, links), Neuanschlag (gelb, mittig) oder Annahme (grün, rechts)

**Kommentar:** Möglichkeit der Verhandlung einen persönlichen Kommentar mitzugeben

**Kommandoknopf:** Initiieren/Beantworten der Verhandlung entsprechend der Einstellungen



## 3 Implementierung

### 3.1 Grundlagen

#### Globale Variablen

Das ATC-System benutzt einheitlich — auch bei der Existenz mehrerer paralleler Prozesse<sup>1</sup> — eine kleine Anzahl von globalen Feld-Variablen.

**preference:** enthält die Voreinstellungen. Diese Daten werden über das Voreinstellungsmodul gelesen und gesichert und von sehr vielen Kommandos benutzt.

**map:** enthält die für die Radardatendarstellung der Karten notwendigen Koordinaten. Diese Daten werden über das Datenbankmodul gelesen und vom Modul `map` benutzt.

**data:** enthält alle übrigen globalen Daten. Diese Daten sind teilweise Abweichungen der Voreinstellungen unter den gleichen Einträgen. Diese Daten sind immer aktueller als die Voreinstellungen. Es existieren aber auch Einträge, für die es keine Voreinstellungen gibt.

**info:** enthält zusätzliche Informationen für die Radardarstellung, in `map` sind immer die dazugehörigen Koordinaten enthalten, in `info` aber nur teilweise Zusatzinformationen. Diese Daten werden über das Datenbankmodul gelesen und vom Modul `map` benutzt.

**tmp:** enthält Werte die nur kurzzeitig global gespeichert werden müssen. Dieses Feld sollte nur sehr wenige Einträge enthalten.

Die Einträge der Felder sind verschieden stark hierarchisch untergliedert<sup>2</sup>. Die Untergliederung erfolgt immer mit Kommata.

#### Umgebungsvariablen

Die Umgebungsvariablen sind unter Tcl in der globalen Variable `env` enthalten, die ein Feld mit je einem Eintrag für jede Umgebungsvariable ist. Die für das ATC-System wichtigen Umgebungsvariablen sind:

- 
- 1 Auch wenn die globalen Variablen in allen Prozessen gleich heißen, habe sie doch jeweils unterschiedliche Einträge und Werte! Sie sind nur vom Aufbau der Einträge und in ihrem Sinn identisch.
  - 2 In Tcl ist dies problemlos möglich, da das Feld nur eindimensional ist, die Indizes aber beliebige Formen annehmen können — dadurch sehen die Felder oft nur mehrdimensional aus, sind es aber nicht wirklich.

**ATC\_PATH:** Verzeichnis der Module — ungesetzt wird das aktuelle Arbeitsverzeichnis angenommen

**ATC\_DATA:** Verzeichnis der Datenbank, der Piktogramme und anderer statischer Daten — ohne Angabe wird hierfür das Verzeichnis `data` in `ATC_PATH` angenommen

**ATC\_PREFS:** Verzeichnis der Voreinstellungen — ungesetzt wird `.atc` im aktuellen Arbeitsverzeichnis angenommen

Die Umgebungsvariablen werden durch das Betriebssystem an vom Client erzeugte Prozesse vererbt und besitzen so in den Frontends automatisch die richtigen Werte.

## Dynamisches Nachladen

Über das Tcl-Autoloading kann ein dynamisches Nachladen von Modulen realisiert werden. Wird ein Kommando nicht gefunden, so durchsucht Tcl alle in der globalen Variable `auto_path` enthaltenen Verzeichnisse. Aus dem ersten Verzeichnis, dessen Index<sup>3</sup> das gesuchte Kommando enthält, wird das entsprechende Modul hinzugeladen.

Das Tcl-Autoloading wird so eingestellt, daß zuerst das Verzeichnis in `ATC_PATH` nach unbekanntem Kommandos durchsucht wird.

Leider werden unbenutzte Module oder Kommandos nicht wieder aus dem Interpreter entfernt<sup>4</sup>, um den Speicherbedarf zu reduzieren und den Interpreter so zu entlasten. Bei einem objektorientierten Entwicklungssystem wäre dies leicht möglich, indem Klassen ohne Instanzen und Unterklassen sich selbst löschen.

## Fensterklassen

Die Fensterklassen mit ihren jeweiligen Dekorationen müssen entsprechend Tabelle 2–6 auf Seite 13 dem Window-Manager bekanntgemacht werden. Dazu werden diese in die X-Resource-Datenbank eingetragen. Diese Einträge haben die Form

```
MWM*Fensterklasse.clientDecoration: Dekoration ...
```

Für die Hauptfenster der Programme ist dies aufwendiger, da die Fensterklasse nach dem Erzeugen nicht mehr geändert werden kann und das Hauptfenster schon beim Start des Tcl/Tk-Interpreters mit dem Programmnamen als Klasse erzeugt wird. Darum müssen für `CWP`, `Radar`, `Flightplan` und `Negotiation` identische Einträge entsprechend der Fensterklasse `Main` erzeugt werden. Die einfachste Art diese Vereinbarungen zu installieren, ist die Ergänzung der Datei `~/.Xdefaults`.

<sup>3</sup> Der Index `tclIndex` für das Autoloading darf nur über Dateien mit der Endung `.tcl` erzeugt werden! Dazu wird das Kommando `auto_mkindex` verwendet.

<sup>4</sup> Tcl kann mit dem Kommando `rename` auch Prozeduren entfernen.

## 3.2 Allgemeine Module

### Datenbankzugriff

Die Datenbank wird der Einfachheit halber als über NFS erreichbare Textdatei realisiert. Dadurch ist die Datenbank leicht zu warten, da sie notfalls mit jedem Texteditor und universellen UNIX-Dienstprogrammen [Bu96] bearbeitet werden kann. Durch die Verwendung des NFS ist der Zugriff über ein Netzwerk von jedem Client und Frontend möglich, ohne die Datenkonsistenz durch Redundanz zu gefährden.

Der Zugriff des Systems auf die Kartendaten erfolgt nur lesend, die Wartung der Datenbank erfolgt durch separate Programme. Außerdem kann der gesamte Datenbestand der benötigten Karten eingelesen und im Arbeitsspeicher gehalten werden, so daß die Performance des Datenbankzugriffes weniger relevant ist.

Für die ASCII-Daten, im folgenden SMD<sup>5</sup> genannt, muß ein einheitliches Format festgelegt werden, das es den zugreifenden Programmen ermöglicht, Art und Größe eines Datensatzes leicht festzustellen. Dazu werden die Datensätze so codiert, daß jeweils eine Zeile einen kompletten Datensatz enthält und das erste Wort die Art des Datensatzes angibt.

#### 3-1: Datenformat der Kartendatenbank

Kürzel	Parameter	Beschreibung
WP	Name, Long., Lat., Typ	Wegpunkt
AP	Name, Long., Lat., Typ, (Bahn, ...)	Flughafen, Landeplatz
BP	Long., Lat., Typ, Höhe	Hindernis
FIR	Name, Polygon	Fluginformationsgebiet
UIR		obere FIR
CTA	Name, Polygon	Kontrollsektor
UTA		obere CTA
TRA	Name, Höhenband, Polygon	zeitw. reservierter Luftraum
R		Flugbeschränkungsgebiet
D		Gefahrengebiet
P		Sperrgebiet
AW	Name, WP, (Höhenband, Richtung,	Luftstraße
UW	WP), ...	oberer AW

Teilweise sind für gleiche Datenformate unterschiedliche Kürzel vergeben, um die Daten nach Kategorien zu unterteilen. Über die Kategorien werden Farbgebung und Ein-/Ausblendung der Karten in der Radardarstellung bestimmt.

<sup>5</sup> SMD: Simple Map Data — einfache Kartendaten

Das Datenbankmodul stellt zwei Kommandos zum Lesen und Schreiben der Daten zur Verfügung. Die Daten werden in zwei globalen Feldern abgelegt. In `map` sind die Daten in für die Darstellung konvertierter Form vorhanden, damit die Darstellungsroutinen effizient darauf zugreifen können. In `info` sind über die reinen Darstellungskordinaten hinausgehende Informationen enthalten, die für spätere Anwendungen wichtig sein könnten.

**ReadSMD:** liest die Kartendaten aus einer SMD-Datei in die Felder ein und konvertiert dabei die Kartendarstellungsdaten in Koordinaten der Darstellung

**WriteSMD:** schreibt die konvertierten Kartendaten aus den Feldern in eine SMD-Datei

Wird keine Kartendatei angegeben, so benutzen die Kommandos die Datei in `preference(file,maps)` im Verzeichnis `env(ATC_DATA)`.

## Interprozeßkommunikation

Die Interprozeßkommunikation kann entweder über Netzwerk/Sockets oder lokal erfolgen. Die von Client und Server benutzten `socket`<sup>6</sup>-basierten Kommunikationskommandos sind

**NetServer:** erzeugt die Server-Seite eines Sockets, benötigt Rechnername, Portnummer und Namen der Prozedur, die die eingehenden Kommandos behandelt.

**NetClient:** öffnet eine Verbindung zu einem Server mit dem angegebenen Rechnernamen/Portnummer und leitet eingehende Kommandos an die angegebene Behandlungsprozedur weiter.

**ipc\_accept:** wird von `NetServer` an `socket` zum Aufbau neuer Verbindung angegeben, richtet die Behandlung über `ipc_receive` ein.

**ipc\_receive:** wird von `NetClient` und `ipc_accept` an `file-event` zur Behandlung eingehender Daten weitergegeben, ruft die an `NetClient` bzw. `NetServer` gegebene Behandlungsprozedur auf.

Die Kommandos für die lokale Kommunikation basieren auf dem Tk-Kommando `send` und bieten folgende Funktionalität:

**LocalStart:** startet die angegebene Applikation — sofern sie noch nicht aktiv ist — und fügt sie der Liste der aktiven hinzu

**LocalHide:** entfernt die angegebene Applikation aus der Liste der aktiven — wird normalerweise von der jeweiligen Applikation gesendet, um weitere Kommandos zu unterbinden

---

<sup>6</sup> Das Tcl-Kommando `socket` existiert erst ab Version 7.5!

**LocalReset:** baut die Liste der aktiven Applikationen mit Hilfe des Kommandos `winfo interps neu` auf

**LocalSend:** sendet ein Kommando an eine aktive Applikation — ist die Applikation nicht aktiv, wird nicht getan

**LocalCheck:** überprüft, ob die angegebene Applikation aktiv ist

Das `ipc`-Modul benutzt ohne Angabe von Server/Portnummer die Einstellungen aus `preference(server,host)` und `preference(server,port)`.

Die bekannten Applikationen sind in der Liste `preference(appl,all)` enthalten, davon sind die in `data(appl,active)` gerade aktiv.

## Voreinstellungen

Dieses Modul enthält Prozeduren zur Verwaltung der Voreinstellungen verschiedener Benutzer zu verschiedenen Sektoren. Ohne Zuhilfenahme einer Datenbank mit aufwändigen Bearbeitungsfunktionen kann dies nur über mehrere Dateien realisiert werden. Die Dateien sind im Verzeichnis aus `env(ATC_PREFS)` enthalten.

**LoadPrefs:** liest die Voreinstellungen für einen Sektor von einem Lotsen aus Dateien — ist keine lotsenspezifische Voreinstellung für den Sektor vorhanden, so wird der Sektorstandard des Systems geladen

**SavePrefs:** sichert die Voreinstellungen in eine Textdatei

**EvalPrefs:** bekommt eine Liste von Einstellung/Wert-Tupeln und legt diese in dem globalen Feld `preference` ab — wird unter anderem von `LoadPrefs` aufgerufen, eventuell auch nach Abschluß des Voreinstellungsdialoges

Das Modul `prefs` erzeugt die globale Variable `preference` und benutzt `env` und `data`. Sofern nicht anders angegeben, ist der Lotse in `data(controller)` und der Sektor in `data(sector)` enthalten.

## Benutzungsoberfläche

In diesem Modul werden Kommandos zur Erzeugung von einheitlichen Widgets/Widgetgruppen entsprechend der Styleguide bereitgestellt, sofern diese nicht durch einfache Optionen — siehe Tabelle 2–7 auf Seite 14 — mit Tk-Kommandos erreicht werden können.

Zur Vereinfachung der Programmierung werden auch komplexere Widgetgruppen erzeugt, die mehrfach verwendet werden können.

**defaultbutton:** erzeugt einen Knopf mit den angegebenen `button-`Optionen und einem Rahmen, der ihn als Standardknopf ausweist. Der Knopf wird als das angegebene Widget erzeugt, der Rahmen bekommt als letzte Namenskomponente `default`.

**buttonbar:** erzeugt einen Rahmen mit rechtsseitig ausgerichteten Knöpfen. Die Knöpfe haben die angegebenen Texte und werden von null durchnummeriert. Beim Drücken eines der Knöpfe wird `tkPriv(button)` auf eine Liste von übergeordnetem Widget und Knopfnummer gesetzt.

**imagearea:** erzeugt ein Canvas-Widget (`.area`) in Kombination mit horizontalem (`.x`) und vertikalem (`.y`) Rollbalken, sowie einem Vergrößerungsknopf (`.z`) in der Ecke.

Jedem dieser Kommandos muß ein Widgetname angegeben werden, als das das (übergeordnete) Widget erzeugt wird. Der Name des packbaren Widgets wird wieder zurückgegeben. Das Modul `gui` wird unter anderem von `dialog` und dem Hauptmodul `Radar` benutzt.

## Dialoge

Das Modul `dialog` enthält Kommandos, die Erzeugung und Umgang mit Dialogfenstern vereinfachen.

**Dialog:** erzeugt ein Fenster der angegebenen Klasse gefüllt mit einer Fläche und einer Knopfleiste mit den angegebenen Knöpfen

**Alert:** erzeugt einen modalen Dialog der Klasse `Alert` mit dem angegebenen Piktogramm, Text und Knöpfen

**Message:** erzeugt einen Dialog der Klasse `Dialog` mit dem angegebenen Text und Knopf

**doModalDialog:** führt den angegebenen Dialog als modalen Dialog, entfernt den Dialog und gibt die Nummer des gedrückten Knopfes zurück

**doDialog:** führt den angegebenen Dialog nicht modal, wartet bis zu einem gedrückten Knopf aus diesem Dialog und gibt die Nummer zurück

**center:** zentriert das erste Fenster über dem zweiten — ist kein zweites angegeben, so wird absolut zentriert

Dieses Modul wird von den Hauptmodulen und dem Farbmodul verwendet. Modale Dialoge sollten über dem dazugehörigen Hauptfenster zentriert werden, Fehlermeldungen eventuell auch absolut. Andere Dialoge sollten ihre Position behalten<sup>7</sup> und immer wieder an der selben Stelle auftauchen.

## Farben

Das Farbmodul `color` stellt alle zur Farbbearbeitung nötigen Kommandos zur Verfügung. Als Farbmodell für das ATC-System wurde das HSV-Modell gewählt, da hiermit für den Benutzer die Farbwahl einfacher zu handhaben ist [FoDa90] und Farbtönen wie heller, dunkler oder blässer einfach automatisch zu erzeugen sind.

### 3–2: Das HSV-Farbmodell

Komponente	Wertebereich	Bedeutung
Hue	[ 0 .. 360 [	Der Farbton als Winkel im Farbkreis von Rot (0°) über Gelb (60°), Grün (120°), Türkis (180°), Blau (240°), Violett (300°) bis Rot (360°) angegeben.
Saturation	[ 0 .. 1000 ]	Die Farbsättigung in Promille bestimmt die Ausdrucksstärke und Auffälligkeit der Farbe. Graustufen haben die Sättigung 0‰ — dann ist der Farbton irrelevant.
Value	[ 0 .. 1000 ]	Die Helligkeit in Promille reicht von Schwarz (0‰) bis Weiß (1000‰) — bei diesen Randwerten haben Ton und Sättigung keine Bedeutung mehr.

Die Farbbeschreibungskomponenten haben die in Tabelle 3–2 aufgeführten Bedeutungen und Wertebereiche. Da das System die Farben prinzipiell nur im HSV-Modell speichert und eine Konvertierung nur für die Übergabe der Farben als Tk-Optionen in der X11-Schreibweise benötigt wird, kommt das Modul mit den beiden folgenden Kommandos aus:

**HSVtoRGB:** errechnet aus Hue, Saturation und Value die Rot-, Grün- und Blauanteile der Farbe und gibt diese als X11-Farbcode zurück — der Algorithmus entstammt [FoDa90 Fig. 13.34]

**Color:** öffnet einen Dialog zur Farbbearbeitung, dazu müssen eine Bezeichnung/ein Titel für die zu wählende Farbe und deren initiale HSV-Werte übergeben werden, der Rückgabewert ist eine Liste mit den neuen HSV-Werten oder eine leere Zeichenkette bei Abbruch der Bearbeitung.

## 3.3 CWP-Client Module

### CWP

Dieses Startmodul ist das einzige Modul der Implementierung, das direkt vom Benutzer aufzurufen ist. Es liest dabei das Client-Modul ein,

<sup>7</sup> Die Fensterpositionen werden in `preference(geometry, fenster)` gespeichert, dies gilt insbesondere auch für die Hauptfenster.

von dem aus die Hauptmodule der Frontends gestartet und über Autoloading spezielle und allgemeine Module nachgeladen werden.

Zunächst werden die Umgebungsvariablen auf Standardwerte gesetzt, sofern sie nicht bereits definiert sind.

Ist beim Start ein Argument angegeben worden, so wird aus diesem Rechnername und Portnummer des Server gelesen und in `data(server,host)` und `data(server,port)` vermerkt.

Zum Schluß wird das Modul `Client` hinzugeladen und durch den Aufruf des dort enthaltenen Kommandos `Open` der `Systemdialog` geöffnet.

## Client

Das Hauptmodul `Client` enthält alle Kommandos des CWP-Clients abgesehen von den aus allgemeinen Modulen importierten.

Wie alle Hauptmodule unterbricht `Client` zunächst die lokale IPC, initialisiert das Autoloading, lädt die Voreinstellungen, öffnet das Hauptfenster durch Aufruf von `New` und läßt die lokale IPC wieder zu.

**New:** erzeugt das Hauptfenster der Applikation — wird automatisch am Ende des Moduleinlesens aufgerufen.

**Open:** öffnet den System-Dialog, um die Verbindung zum Server angeben zu lassen und herzustellen — wird vom Menükommando *Controller / System ...* aufgerufen. Die Verbindung zum Server wird mit Hilfe des Kommandos `NetClient` aus dem Modul `ipc` aufgebaut.

**Close:** schließt die Verbindung zum Server.

**Quit:** beendet den Client und alle Frontends, wenn der Benutzer durch einen Nachfragedialog zustimmt

**Load:** liest die Voreinstellungen von Datei ein — wird vor `New` und durch `Preference` aufgerufen

**Save:** sichert die Voreinstellungen in eine Datei — wird von `Preference` aufgerufen, anschließend sollte `Apply` ausgeführt werden

**Apply:** wendet die Voreinstellungen an — wird von `Preference` aufgerufen, dabei müssen auch Frontends benachrichtigt und eventuell gestartet oder beendet werden.

**Change:** öffnet den Übernahmedialog — wird vom Menükommando *Controller / Take Over ...* aufgerufen, die Auswahlménüs müssen mit Listen aus der Zulassungsdatenbank<sup>8</sup> `controller` in dem Verzeichnis `env(ATC_DATA)` gefüllt werden.



**Preference:** öffnet den Voreinstellungsdialog — wird vom Menükommando *Controller / Preferences ...* aufgerufen

**Frontend:** startet oder öffnet das angegebene Frontend — ist es bereits aktiv, wird `Open` gesendet, sonst wird es durch `exec` im Hintergrund gestartet

**MenuAdjust:** sperrt oder gibt Menüeinträge entsprechend der Serververbindung frei

Die meisten Kommandos werden durch die in `New` erzeugten Menüs aufgerufen. `Open` wird außerdem von `CWP` nach dem Nachladen aufgerufen. Über das *Display*-Menü werden auch die Frontends gestartet.

In `data(server, file)` ist die Dateivariablen der Socketverbindung gespeichert. Ohne Verbindung ist dieser Feldeintrag leer. Die Menüs müssen entsprechend dem Vorhandensein einer Verbindung durch `MenuAdjust` angepaßt werden.

## 3.4 Radar-Frontend Module

### Radar

Das Hauptmodul `Radar` wird vom Client durch Aufruf des Menükommandos *Display / Radar* gestartet. Das Frontend ist ein eigenständiges Programm, das über das IPC-Modul mit dem Client kommuniziert.

**New:** erzeugt das Radar-Hauptfenster unter Verwendung von `Open`, dessen Ergebnisfenster um weitere Widgets für Menüzeile, Knopfleiste und Nachrichtenzeile ergänzt wird

**Open:** öffnet ein Fenster mit einer Radardarstellung, es können Werte für Positionierung und Vergrößerung der Darstellung angegeben werden

**Close:** schließt das angegebene Radarfenster

**Quit:** Beendet das Frontend nach Sicherung der Einstellungen über den Client

**MenuAdjust:** sperrt oder gibt die Menüeinträge *Radar / Raise* und *Radar / Close* entsprechend den vorhandenen Radarfenstern frei

**Print:** öffnet den Druckdialog, in dem Radarfenster, Drucker und Druckoptionen gewählt werden können — bei Bestätigung wird der Canvas-Inhalt als PostScript gedruckt

---

8 Die Datenbank enthält für jeden Lotsen eine Zeile, die den Lotsennamen, einen Doppelpunkt und eine durch Kommata getrennte Liste von Sektoren enthält.

## map

Das Modul `map` enthält die für Kartenein- und -ausblendungen nötigen Kommandos. Dazu benötigt es in dem globalen Feld `map` die Koordinaten der Kartenelemente, wie sie durch das Datenbankmodul `data` erzeugt werden. Für die Bearbeitung ist die Tagvergabe aus Tabelle 3–3 eine wichtige Grundlage.

**SetRegion:** ändert oder erneuert die Kartendarstellung im angegebenen Radarfenster wie angegeben für den oberen oder unteren Luftraum unter Benutzung von `SetMap`

**CreateMap:** erzeugt ein Kartenelement mit der angegebenen Klasse im angegebenen Radarfenster aus den Koordinaten in der globalen Variablen `map`

**SetMap:** modifiziert die Kartendarstellung in dem angegebenen Radarfenster entsprechend einer angegebenen Liste von Tupeln aus Kartenobjektklasse und Boolean — das Boolean bestimmt, ob die Klasse hinterher dargestellt sein soll, dabei wird berücksichtigt, ob die Objektklasse schon darstellt ist und die Farben werden über `AdjustColor` angepaßt

**AdjustColor:** ändert die Farben einer Klasse von Kartenobjekten in dem angegebenen Radar entsprechend den Einstellungen in `preference`

## target

Die Flugziele in der Radardarstellung werden durch Kommandos im Modul `target` erzeugt, manipuliert und gelöscht. Für diese Aufgaben ist die Vergabe der Tags im Canvas der Radardatendarstellung besonders wichtig. Das Schema der Tagvergabe ist in Tabelle 3–3 wiedergegeben.

3–3: Tagvergabe im Radar-Canvas

Pos.	Tag	Aufbau	Bedeutung	Beispiel
0	Objekt	Klasse, Kennung, Typ	ein Objekt (eindeutig)	Label von Flugziel DLH123
1	Klasse		alle Objekte einer Unterklasse	alle FIRs
2	Unterklasse	Klasse, Typ	alle Objekte einer Klasse mit dem selben Typ	alle Labels
3	Gruppe	Klasse, Kennung	eine Gruppe von zusammengehörigen Objekten	Flugziel DLH123
...			zusätzliche Tags in unbestimmter Reihenfolge und Vorkommen nach Bedarf	Auswahl, Aktuelles Objekt, Flugzielart

**Labelshifting:** rotiert das Label des angegebenen Flugzieles im Uhrzeigersinn eine Position weiter — dabei können die Positionen NO, O, SO, SW, S, NW angenommen werden

**TargetSelect:** wechselt die Anwahl des Flugzieles — diese Anwahl muß eventuell anderen Frontends über den Client bekannt gemacht werden

**Update:** ist der Beginn oder das Ende eines Radarumlaufes (wird vom Client aufgerufen) — zu Beginn wird die History-Zählung weitergeführt und die älteste Position gelöscht, am Ende des Umlaufes wird die Zeichenordnung der Klassen korrigiert, nicht aktualisierte Flugziele entfernt und eine neue Liste von zu aktualisierenden Flugzielen in `data(targets)` erstellt; dabei werden die Farben der Flugziele angepaßt.

**Target:** erzeugt oder erneuert das angegebene Flugziel mit den angegebenen Daten (wird vom Client aufgerufen) — dabei wird entsprechend des Sektor-Höhenbandes das Flugziel in Haupt oder Nebenziel eingeordnet, nach Bedarf ein Label mit `CreateLabel` erstellt; ist das Flugziel im darzustellenden Höhenband, so wird für alle Radarfenster der Vergrößerungsfaktor und damit die Position errechnet, ein Icon gewählt und eventuell die Vektorendposition berechnet; dann wird das Flugziel entweder erzeugt oder verschoben/verändert; zuletzt wird das Flugziel in jedem Fall aus der Liste der zu aktualisierenden Ziele entfernt, damit es nicht am Ende des Umlaufes gelöscht wird

**CreateLabel:** erzeugt ein — eventuell mehrzeiliges — Label für das angegebene Callsign mit den angegebenen Daten entsprechend den Einstellungen in `preference`

## 3.5 Handbuch

### Systemstart

Die Installation des Programmcodes ist in Anhang A beschrieben. Vor dem Aufruf des CWP-Clients sollte ein Server innerhalb des Netzwerkes laufen, so daß der Client eine Verbindung zu ihm aufbauen kann.

Der Start des Clients erfolgt durch den Aufruf des Kommandos CWP. Dabei brauchen normalerweise keine Argumente angegeben werden. Es ist jedoch möglich als Argument eine Server-Beschreibung anzugeben. Der Aufruf sieht also wie folgt aus:

```
CWP [server.domain:port]
```

Liegen die Module des ATC-Systems nicht im aktuellen Arbeitsverzeichnis, so kann über die Umgebungsvariable `ATC_PATH` das Verzeichnis der

Module vorgegeben werden. Die Umgebungsvariable `ATC_DATA` gibt das Verzeichnis der Datenbank und anderer Hilfsdateien an, `ATC_PREFS` das Verzeichnis der Voreinstellungen.

Nach dem Start des Systems wird nach dem Hauptfenster des Systems automatisch der Dialog zur Systemverbindung geöffnet. Darin sind Name und Portnummer des Server entsprechend der letzten Verbindung oder des angegebenen Argumentes vorbesetzt, so daß durch Drücken des *Connect*-Knopfes die Verbindung zum Server hergestellt werden kann. Wird statt dessen *Cancel* gedrückt, so wird zunächst keine Verbindung aufgebaut.

Um die Verbindung zu einem späteren Zeitpunkt zu unterbrechen, aufzunehmen oder zu ändern, kann der System-Dialog über das Menükommando *Controller / System ...* aufgerufen werden.

### Arbeitsplatzübernahme

Um den Arbeitsplatz beim System für einen Sektor anzumelden, muß das Menükommando *Controller / Take Over ...* aufgerufen werden. Ohne diese Anmeldung sendet der Server auch bei bestehender Verbindung und geöffneten Frontends keine Radardaten. Das Menükommando ist nur bei bestehender Server-Verbindung anwählbar.

Für die Arbeitsplatzübernahme wird ein Dialog geöffnet, in dem sich zwei Menüknöpfe für Sektor- und Lotsenauswahl befinden. Durch Anklicken der Knöpfe erscheinen Menüs mit den dem System bekannten Sektoren und Lotsen, aus denen je einer ausgewählt werden muß. Wenn der ausgewählte Lotse für den angezeigten Sektor zugelassen ist, so ist der *OK*-Knopf anwählbar, ansonsten ist er hell gezeichnet.

Durch Anklicken von *OK* übernimmt der ausgewählte Lotse den angezeigten Sektor, durch Anklicken von *Cancel* wird die Auswahl nicht berücksichtigt und die Arbeitsplatzzuständigkeit bleibt bestehen wie gehabt.

Nach einer Übernahme setzt das System die Positionen und Größen der Fenster entsprechend der Voreinstellung des Lotsen für diesen Sektor neu. Gegebenenfalls werden nicht gewünschte Fenster geschlossen oder gewünschte aber nicht vorhandene neu geöffnet. Anschließend werden die Farbwahl und weitere Einstellungen angepaßt, so daß der Arbeitsplatz erscheint, wie der Lotse ihn zuletzt verlassen hat.

### Fensterverwaltung

Nach der Arbeitsplatzübernahme sind zumindest das Fenster *ATC Working Position* und ein Radarfenster zu sehen. Diese Fenster können durch Ziehen des Titels beliebig in ihrer Position verändert werden. Durch Anklicken des Fenstertitels kann das Fenster nach vorne geholt werden, falls es teilweise verdeckt ist.

Das Radarfenster kann durch ziehen an den Ecken oder Seiten des Rahmens in der Größe verändert werden. Das Systemfenster kann nur in der Breite verändert werden. Wird der Maximalgrößenknopf gedrückt, so wird das Fenster auf die gesamte Bildschirmgröße vergrößert — bzw. nur auf die Bildschirmbreite. Durch nochmaliges Drücken wird es wieder auf die vorherige Größe reduziert. Sollte der Bildschirm einmal zu unübersichtlich werden, können manche Fenster durch Drücken des Ikonenknopfes zeitweise auf ein Piktogramm reduziert werden, das jederzeit durch einen Doppelklick darauf wieder zur vorherigen Größe und Position zurückkehrt.

## Bedienung des Radarfensters

Der dargestellte Ausschnitt des Radarfensters kann frei gewählt werden. Dazu können die Rollbalken durch Ziehen des Reglers oder Drücken der Pfeile verschoben werden, wodurch sich auch der Ausschnitt in der Radardatendarstellung daneben ändert. Über den Vergrößerungsknopf an der rechten unteren Ecke der Darstellung kann durch Klicken mit der linken Maustaste die Darstellung vergrößert werden, mit der rechten verkleinert und mit der mittleren kann eine Ausschnittsvergrößerung gewählt werden.

Um eine Ausschnittsvergrößerung zu wählen, so muß mit dem Mauszeiger, der dann als Plus erscheint, in der Radardarstellung ein Rechteck aufgezogen werden. Dieses wird anschließend in einem neuen Radarfenster um den Faktor Zwei vergrößert dargestellt. Die Ausschnittsvergrößerung ist ein eigenständiges Fenster, das in Größe und Position wie das erste Radarfenster verändert werden kann. Ihm fehlen allerdings Menüzeile, Knopfleiste und Nachrichtenzeile.

Die Darstellung beinhaltet in allen Radarfenstern immer die gleichen Objekte, nur der Ausschnitt und die Vergrößerung können unterschiedlich sein. Über das Menü *Map* und die Knopfleiste im ersten Radarfenster wird die Darstellung aller Radarfenster beeinflusst — ebenso durch ein *Apply* oder *Save* im Voreinstellungsdialog.

## Ändern der Einstellungen

Sämtliche Einstellungen des Systems können über den Voreinstellungsdialog getätigt werden. Dieser wird über das Menükommando *Controller / Preferences ...* geöffnet. Hier können diverse Einstellungen an- oder ausgewählt werden.

Über die Farbknöpfe wird der Farbwahldialog geöffnet, der die Veränderung der Parameter Farbton, Sättigung und Helligkeit für die jeweilige Farbe ermöglicht. Die neue Farbe kann dabei direkt angesehen werden. Erst beim Drücken von *OK* wird der Farbwahldialog geschlossen und die Farbe in den Einstellungsdialog übernommen. Beim Drücken von *Cancel* wird die neue Farbe verworfen.

Der Voreinstellungsdialog besitzt vier Knöpfe am unteren Rand. Mit *Revert* werden alle Änderungen verworfen und die gesicherten Voreinstellungen erscheinen wieder. Dabei werden auch die Fensterpositionen und -größen wieder auf die ursprünglichen Werte gesetzt. Mit *Close* wird der Dialog mit den momentan in dem System vorhandenen Einstellungen geschlossen. Durch Drücken von *Apply* werden die gewählten Darstellungsoptionen im System angewendet. Mit *Save* werden sie angewendet und als Voreinstellungen gesichert, so daß *Revert* anschließend immer wieder auf diese Einstellungen zurücksetzt.

Der Aufruf von *Save* sichert auch die aktuelle Auswahl von Hauptfenstern inklusive ihrer jeweiligen Position und Größe.

## 4 Abschluß

### 4.1 Bewertung

#### Erneute Befragung

Um während der Entwicklung eine Rückmeldung zu dem bisherigen Entwurf zu erhalten und die Ergebnisse in die nächste Phase des Rapid-Prototyping einbeziehen zu können, wurde eine erneute Befragung von Fluglotsen der RK Berlin durchgeführt.

Dazu fertigte die Projekt-Psychologin einen neuen Fragebogen an, der sechzehn freiwillig teilnehmenden Lotsen vorgelegt wurde, während ihnen der Entwurf an Hand von Farbbildern erläutert wurde.

Nach der Vorstellung jedes Bildes wurde dazu direkt der Fragebogen ausgefüllt. Am Ende des Interviews wurde noch um allgemeine Kritik und Anregungen gebeten.

#### Ergebnisse

Die Ergebnisse sind für einen ersten Entwurf überaus zufriedenstellend. Die Beziehung der Fenster untereinander, die Plazierung der Informationsdarstellung, die Aufteilung der Menüs wurden gut bewertet. Die Erlernbarkeit, die Handlungsabfolgen, und der Gesamteindruck waren zwar nur befriedigend, die Bewertung an Hand statischer Bilder ist zu diesen Aspekten aber besonders schwierig.

Die Arbeitsplatzübernahme und der Voreinstellungsdialog ist insgesamt gut bewertet worden, hier scheint es wenig zu beanstanden zu geben und die Lösung scheint im Sinne der Lotsen zu sein. Das System sollte zusätzlich immer das örtliche QNH anzeigen.

Die Radardatendarstellung benötigt eine Maßstabsanzeige und die Historyauswahl sollte andere Werte besitzen. Die grundsätzliche Darstellung ist aber als sinnvoll angesehen worden. Den Lotsen fehlte teilweise der Steig- bzw. Sinkgradient. Die Konfliktdarstellung an Flugverlaufsschnittpunkten benötigt neben der Zeitdifferenz noch eine Minimalentfernung.

Die Flugplanfenster bedürfen noch einer Überarbeitung. Die Auswahl und Aufteilung der ständig und über den Klappfeil zusätzlich darstellbaren Informationen muß noch einmal genauer untersucht werden, da die Einflughöhe von den Lotsen als weniger wichtig eingestuft wurde als beispielsweise die Einflugrichtung. Die Sortierung der Streifen ist ebenfalls noch nicht endgültig. Zusätzliche Farbcodierungen für die Streifen wurden ebenfalls angedacht. Das Callsign sollte immer ganz links erscheinen.

Die Aufteilung der Informationen in Flugplan- und Informationsfenster muß ebenfalls noch einmal untersucht werden. Vielleicht kann durch eine Umverteilung der Informationen in Kombination mit dem Flugplan-Frontend auf eines der beiden Fenster ganz verzichtet werden.

Die direkte Manipulation wurde gut angenommen, die Bedienung bot jedoch noch einige Probleme. So fehlten eine Anzeige des gewählten Kurses bei der Vektormanipulation, eine Möglichkeit, Steig- und Sinkgradienten vorzugeben und Ausweichmanöver direkt einzugeben, da meist keine dauerhafte Kursänderung, sondern nur eine zeitweilige angewiesen wird.

Die Nachrichtenzeile sollte eine Möglichkeit zur direkten Reaktion bieten. Auch die Art der Nachricht (Anfrage, Info, ...) sollte klarer erkennbar sein.

Wie erwartet, brachte die Verhandlungsführung die meisten Probleme, da diese ja von Grund auf neu konzipiert wurde. Für ein hohes Verkehrsaufkommen wären Sammelanweisungen für mehrere Flugzeuge sinnvoll. Das Flugprofil könnte auch mehrere Flugzeuge enthalten<sup>1</sup>. Eine differenzierte Priorisierung wird nicht als notwendig erachtet.

## 4.2 Ausblick

### Implementation weiterer Frontends

Um ein sinnvoll einsetzbares und evaluierbares System zu erhalten müssen sicherlich weitere Frontends implementiert werden. Insbesondere die Flugplanfenster sind für eine vollständige Überwachungsfunktion wichtig.

Für die DataLink-Integration muß auch die Verhandlungsführung als Prototyp vorhanden sein. Diese wird aber sicherlich noch mehrere Versionen durchlaufen, bis ein gutes Ergebnis erreicht wird.

Außerdem ist eine automatische Konflikterkennung zu erweitern und mit automatischen Konfliktlösungsvorschlägen wie aus dem Projekt *„Untersuchung zum Einsatz von ADS in der Flugsicherung unter Berücksichtigung operativer und kognitiver Aspekte der Fluglotsentätigkeit“* zu kombinieren.

### Umstellung auf objektorientierte Entwicklung

Wie schon im Entwurf begründet, wäre eine objektorientierte Entwicklung für das System weitaus sinnvoller. Ein solcher Entwurf ist im Anfangsstadium meist aufwendiger, zahlt sich aber mit fortschreitender

---

<sup>1</sup> Dies wurde aber beim Entwurf auch als problematisch angesehen, da es so leicht zu Pseudokonflikterkennung kommt.



Entwicklungszeit aus. Aus vielerlei Gründen, insbesondere der verfügbaren Zeit und einem zu späten Entdecken von [incr Tcl] konnte in dieser Arbeit davon kein Gebrauch mehr gemacht werden.

Für eine weitere Entwicklung sollte aber trotzdem überlegt werden, ob ein Umstieg nicht immer noch sinnvoll wäre. So könnte eine ATC-Klassenbibliothek<sup>2</sup> erstellt werden, mit deren Hilfe auch weitere Projekte mit enormer Arbeitersparnis realisiert werden könnten.

## WIAS und Online-Hilfe

In einer Weiterentwicklung sollte das System um die Informationsmöglichkeiten des WIAS und eine Online-Hilfe ergänzt werden. Dazu bietet sich eine Realisierung beider Teile als Hypertextsystem an. Der Quasi-Standard für solche Systeme ist HTML, das Datenformat des WorldWideWeb [HTML].

Das Hilfe- und Informationssystem bestände dann aus einem Browser-Frontend und zwei umfangreichen Hypertexten.

Für zukünftige international vernetzte ATC-Systeme wäre dies auch eine sehr sinnvolle Lösung, da so die WIAS-Daten nur jeweils einmal weltweit vorhanden sein müßten und trotzdem jeder Lotse auf alle weltweiten Informationen zugreifen könnte. Dazu müßte jedoch entweder ein eigenes weltweites Netzwerk aufgebaut oder das existierende Internet genutzt werden.

---

<sup>2</sup> Die ODS-Toolbox stellt eine solche Bibliothek dar. Leider wird diese von EUROCONTROL auch für Forschungszwecke nicht im Quellcode herausgegeben.



# Anhang

## A Module

### Diskette

Der Programmcode der Implementierung ist dieser Arbeit als Diskette beigelegt. Die Diskette ist aus Kompatibilitätsgründen im DOS-Format und enthält zwei Dateien:

**LIESMICH.TXT:** Ein kurzer Begleittext mit Angaben zu Inhalt, Rechten und Format der Datei `CWP.GZ`.

**CWP.GZ:** Ein mit GNU-Zip komprimiertes tar-Archiv. Unter UNIX kann das Dekomprimieren durch Aufruf des Kommandos `gunzip cwp.gz` erfolgen und ergibt die Datei `CWP.tar`. Das Archiv kann durch `tar -xf CWP.tar` zerlegt werden.

### Dateien

Durch das oben beschriebene Dearchivieren entstehen die folgenden Dateien des ATC-Systems:

```
CWP
Client
Radar
map.tcl
target.tcl
data.tcl
ipc.tcl
prefs.tcl
gui.tcl
dialog.tcl
color.tcl
tclIndex
.atc
data
```

### Referenz

Der Disketteninhalt ist auch unter <ftp://c3po.cs.tu-berlin.de/Dokumente/pub/Archiv/Diplom/> zu beziehen.

Bei Problemen oder für Fragen ist der Autor über E-Mail unter <mailto:arne@cs.tu-berlin.de> zu erreichen.

## B Quellen

### Entwicklungsumgebung

Die Implementierung fand mit Hilfe von Tcl/Tk statt. Die Programmiersprache und viele damit/dafür entwickelte Erweiterungen und Applikationen sind frei kopierbar und für die wichtigsten Plattformen wie UNIX, MacOS und MS-Windows verfügbar.

Informationen zu Tcl sind bei *Sun Laboratories* im WWW unter

<http://www.sunlabs.com/research/tcl/>

im Usenet in

`news:comp.lang.tcl`

oder [Ou95] zu finden.

Das ATC-System wurde mit Tcl 7.5 und Tk 4.1 erstellt. Diese sind über anonymous-FTP unter

<ftp://ftp.ibp.fr/pub/tcl/>

<ftp://ftp.cs.tu-berlin.de/pub/tcl/>

zu beziehen.

### Datenbank

Die Kartendaten wurden zum überwiegenden Teil mit Hilfe elektronischer Texterkennung aus [AIP] übernommen. Leider wurden dem Projekt bis heute keine Luftraumdaten in elektronischer Form zugänglich gemacht. Ähnliche Probleme gibt es bei den topografischen Daten.

### Abbildungen

Sofern nicht im folgenden anders angegeben sind alle Photos und Grafiken der Abbildungen und Tabellen vom Autor selbst erstellt.

- Das Titelbild und die Photos in den Abbildungen 2-1 und 2-22 entstanden mit freundlicher Genehmigung in der RK Berlin der DFS.
- Abbildung 2-2 ist [ODID94] entnommen.
- Die Abbildungen 2-3 und 2-4 sind nach Vorlagen aus [COPS91] gezeichnet.

---

# Literatur

- [AIP] **„Luftfahrthandbuch, Deutschland“**, Deutsche Flugsicherungs GmbH, im Änderungsdienst
- [Bu96] **Arne Burmeister: „UNIXeasy: Der Einstieg in UNIX“**, Carl Hanser Verlag 1996
- [COPS91] **COPS Working Group: „Common Operational Performance Specification for Controller Working Position“**, Report Version 4, EURO-CONTROL 1991
- [DERD94] **DFS: „DERD-XL Handbuch“**, DFS 1994
- [DI88] **Lektorat des B.I. Wissenschaftsverlages (Hrsg.): „Duden Informatik“**, Dudenverlag 1988
- [DIN66234] **DIN 66234**, Deutsches Institut für Normung 1980-1988 in Anlehnung an [ISO9241]
- [Eri90] **Thomas D. Erickson: „Working with Interface Metaphors“** in [La90]
- [FoDa90] **James D. Foley, Andries van Dam, Steven K. Feiner und John F Huges: „Computer Graphics: principles and practice“**, 2. Ausgabe, Addison-Wesley Publishing Company 1990
- [HTML] **W3C: „HTML“**, <http://www.w3.org/pub/WWW/MarkUp/html-spec/>
- [ISO9241] **ISO 9241**, International Standards Organisation 1993
- [GrUl96] **Genia Grundmann und Beate Ulbrich: „Auswertung der Lotsenbefragung: Zusammenfassende Interpretation“**, KATI-Bericht, TU Berlin 1996
- [La90] **Brenda Laurel (Hrsg.): „The Art of Human-Computer Interface Design“**, Addison-Wesley Publishing Company 1990
- [Maa93] **Susanne Maaß: „Software-Ergonomie: Benutzer- und aufgabenorientierte Systemgestaltung“**, Informatik Spektrum 16/1993 Seite 191-205
- [Me93] **Heinrich Mensen: „Moderne Flugsicherung: Organisation, Verfahren, Technik“**, 2. Auflage, Springer-Verlag 1993

- 
- [MHIC92] **Apple Computer Inc.:** „**Macintosh Human Interface Guidelines**“, Addison-Wesley Publishing Company 1992
- [MSG92] **OSF:** „**OSF/Motif Style Guide**“, Revision 1.2, Open Software Foundation 1992
- [Ne87] **Ted Nelson:** „**Computer Lib / Dream Machines**“, Redmond 1987
- [No89] **Donald A. Norman:** „**Dinge des Alltags: gutes Design und Psychologie für Gebrauchsgegenstände**“, Campus Verlag 1989
- [ODID94] **EUROCONTROL:** „**ODID IV**“, EEC Report 269/94, EUROCONTROL 1994
- [Ou95] **John K. Ousterhout:** „**Tcl und Tk: Entwicklung grafischer Benutzerschnittstellen für das X Window System**“, Addison-Wesley Publishing Company 1995
- [ScBoGeKa94] **Martin Scheller, Klaus-Peter Boden, Andreas Geenen und Joachim Kampermann:** „**Internet: Werkzeuge und Dienste**“, Springer-Verlag 1994
- [Sh92] **Ben Shneiderman:** „**Designing the User Interface: strategies for effective human-computer interaction**“, 2. Ausgabe, Addison-Wesley Publishing Company 1992

---

# Glossar

**Autoload:** Tcl's Autoloading ist ein automatischer Such- und Nachlade-Mechanismus für Unterprogramme.

**Benutzungsschnittstelle:** der Teil eines Softwaresystems, mit dem der Benutzer direkt durch Ein- und Ausgaben konfrontiert wird

**Browser:** (engl. *to browse*) schmökern, auch Programm zur Betrachtung von Hypertexten

**EUROCONTROL:** Europäische Organisation für Flugsicherung

**Ergonomie:** die Wissenschaft von den Leistungsmöglichkeiten und -grenzen des arbeitenden Menschen sowie der besten wechselseitigen Anpassung zwischen dem Menschen und seinen Arbeitsbedingungen

**Event:** (engl.) Ereignis, auch eine asynchrone IPC-Form

**Featuritis:** ein „krankhafter“ Zwang, möglichs viele — nicht immer sinnvolle — Features in ein System zu integrieren

**Feedback:** eine Rückmeldung an den Benutzer — er/sie sollte immer wissen, was das System zu einem Zeitpunkt tut und möglichst auch wie lange es noch damit beschäftigt sein wird

**Firewall:** ein Gateway, das aus Sicherheitsgründen verschiedene Dienste blockiert oder nur eingeschränkt zuläßt

**Gateway:** ein Rechner, der verschiedene Teilnetzwerke verbindet

**Icon:** (engl.) Ikone, Piktogramm, Sinnbild

**Komplexität:** steigt exponentiell mit dem Funktionsumfang — die Benutzbarkeit sinkt aber mit steigender Komplexität

**konzeptionelles Modell:** das Modell des Designers bestimmt das initiale Systembild; durch die Arbeit mit dem System entwickelt der Benutzer ein eigenes Modell — je besser das konzeptionelle Modell, um so weniger weicht das Benutzermodell ab

**kooperativ:** Form des Multitasking, bei der die CPU-Zeit nur durch Abgabe der Kontrolle durch den aktiven Prozeß verteilt wird

**Labelshifting:** (engl.) Änderung der Label-Position eines Bildschirmobjektes

**Mapping:** Abbildung zwischen Funktion und Interface — natürliches Mapping ist die Nutzung äußerlicher Analogien und kultureller Standards

---

**Multitasking:** Eigenschaft eines Betriebssystems, mehrere Prozesse quasi-parallel auszuführen — Unterscheidung zwischen kooperativ und preemtiv

**Multiuser:** Eigenschaft eines Betriebssystems, mehrere Benutzer zu unterscheiden und ihre Daten gegeneinander zu schützen

**Nützlichkeit:** Abdeckung des Sachproblems durch den Funktionsumfang unter Berücksichtigung der Benutzbarkeit

**preemtiv:** Form des Multitasking, bei der das Betriebssystem die CPU-Zeit gleichmäßig auf die laufenden Prozesse verteilt

**Rapid-Prototyping:** Software-Entwicklung durch frühzeitige Erstellung von Prototypen und deren fortlaufende Weiterentwicklung

**single-inheritance:** Einfach-Vererbung: Vererbungsart in der objektorientierten Programmierung — jede Klasse hat genau eine Oberklasse

**Socket:** Realisierung einer netzwerkfähigen IPC-Variante

**Styleguide:** Gestaltungs-Richtlinie

**Tube:** (engl. Schlauch) Unterteilung des Luftraumes in schlauchähnliche Kontrollabschnitte, in denen sich je ein Flugzeug befinden kann

**Widget:** Tk-Bezeichnung eines Elementes der grafischen Benutzungsschnittstelle

**WIMP:** Windows, Icons, Mice and Pointer