Hochschule für angewandte Wissenschaften
FACHBEREICH ELEKTROTECHNIK UND INFORMATIK
BERLINER TOR 3
D - 20099 HAMBURG

Studienarbeit

Partizipative, evolutionäre Entwicklung im Robot-Art Umfeld

von Nils Dröge

Betreut durch Prof. Dr. Kai von Luck

7. Mai 2002

Inhaltsverzeichnis

1 Einleitung **2**
1.1 Motivation . . . 3
1.2 Aufgabe . . . 4
1.2.1 Agenten . . . 4
1.2.2 Robot – Art . . . 4

2 Vorgaben **6**
2.1 Teamumgebung . . . 6
2.2 Laborumgebung . . . 6
2.2.1 F18-Ctrl . . . 7
2.2.2 6.270 . . . 7
2.2.3 Plattform . . . 8

3 Entwicklungs–Modell **9**
3.1 Voraussetzungen . . . 9
3.2 Traditionelle Vorgehensmodelle . . . 9
3.2.1 Wasserfallmodell . . . 10
3.2.2 Spiralmodell . . . 10
3.3 eXtreme Programming (XP) . . . 11
3.4 Evolutionäre Entwicklung . . . 13
3.4.1 STEPS . . . 13
3.5 Konsequenz . . . 14

4 Durchführung **17**
4.1 Asimov: der Bumper muss an das 6.270 . . . 18
4.2 Der Motorport an das F18-Ctrl . . . 20
4.3 Ereignisgesteuert im Weltmodell . . . 22
4.4 Notwendige low level Sensoren . . . 23
4.5 Barke oder Satellit . . . 24
4.6 Realisierte Systeme . . . 25

5 Aussicht und Kritik **26**

A Konzeption und Skizzen von F18 **28**

1 Einleitung

Der erste Kontakt mit der Künstlergruppe F18 kam im Roboterlabor der Fachhochschule Hamburg zustande. Die bisherigen Konstruktionen von F18 hatten ein fest vorgegebenes Verhalten. Für die Spielshow Playground in Zürich wollte F18 nun mobile interaktive Roboter bauen und suchte dafür Informatik-Studenten, die bereit waren mit zu helfen.

Die Vorgaben über die Hardware waren nach den ersten Gesprächen wie folgt grob definiert:

- Es sollen drei Roboter werden. Sie sollen ca. 1 mal 1 Meter groß sein.
- Die Plattform hat zwei Antriebsräder mit der Achse unterhalb des Schwerpunktes und zwei Stützräder. Einer der Roboter soll eventuell einen Pedalantrieb haben.
- Die Roboter haben eine Kamera, ein Mikrofon, sowie weitere benötigte Sensoren.

Eine Serverstation sorgt für die Verwaltung der Daten:

- Die Roboter sollen sich mit einer Serverstation „unterhalten“.
- Die Serverstation bereitet die gesendeten Daten auf, um sie auf einer Lampenwand anzuzeigen.

Und schließlich die Verhaltensweise der Roboter:

- Die Roboter sollen auf Kinder und andere Roboter reagieren. Wenn keine Kinder in Sensorenreichweite sind oder kein Interesse zeigen, sollen die Roboter Formationen fahren, tanzen, fangen spielen oder ähnliches.

Die ursprüngliche Spezifikation ist im Anhang. Darin wird auch dargestellt, wie allgemein der erste Ansatz war.

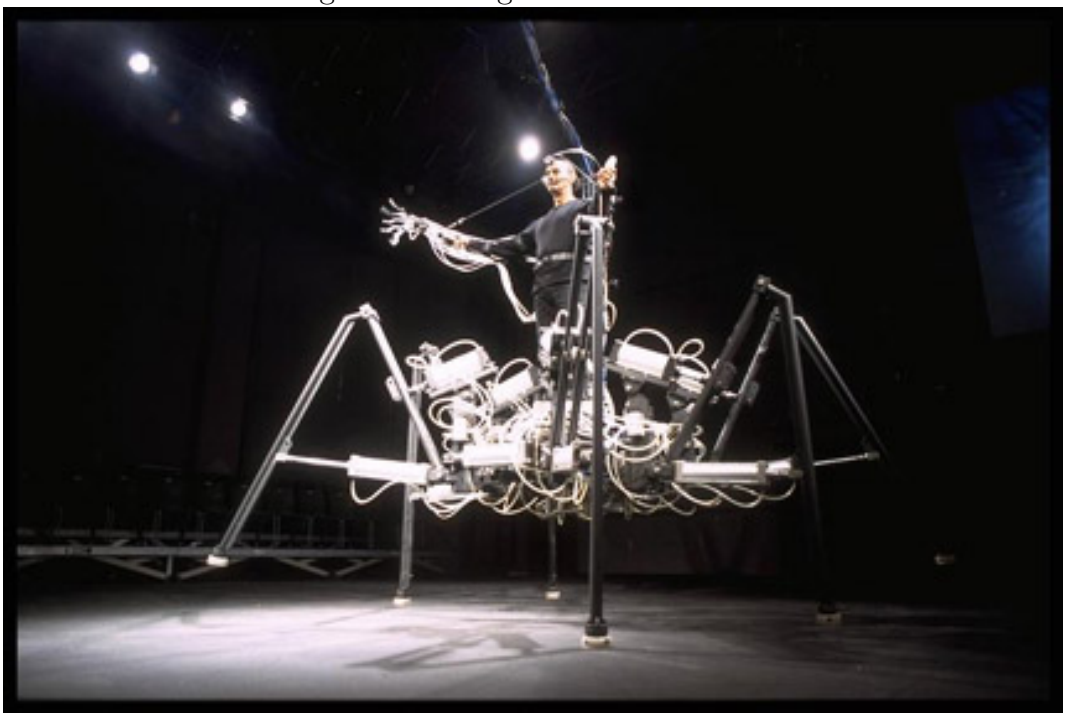
Um das Verständnis für den weiteren Text zu erhöhen, werden hier ein paar Begriffe eingeführt. Zum Teil wurden diese Begriffe durch die Künstlergruppe F18 geprägt. Die Gruppe bewegt sich in einem anderen Umfeld, in dem die technische Seite einer Konstruktion eine untergeordnete Funktion hat und somit Aspekte

aus einer anderen Sichtweise benannt werden. Der gesamte Aufbau, d.h. die Roboter, die Serverstation, das Spielfeld und das Verhalten der einzelnen Elemente, wird Installation genannt. Die Plattform beschreibt den Bewegungsapparat der Roboter. Mit Leinwand, Lampenwand oder Display ist immer dasselbe Element gemeint: Ein 4 mal 3 Meter großes Display aus Glühlampen.

1.1 Motivation

Neugierig geworden auf ein elektronisches Rieseninsekt, entschlossen wir uns zu einem Besuch der Gruppe F18. Die Konstruktion, die sie uns zeigten, war hinsichtlich der Größe und mechanisch sehr beeindruckend, nur leider hatte dieser Käfer so gar nichts von einem autonomen Roboter. Das angeschlossene Laptop hatte die Aufgabe, Tastendrücke in eine Motorsteuerung zu übersetzen. Eine zielgerichtete oder ungerichtete Bewegung ohne den Menschen war mit diesem Programm nicht möglich. In einer anschließenden Diskussion gaben wir unsere Zusage, bei dem Projekt Playground 2000 mitzuwirken. Das Team setzte sich aus 6 Mitgliedern der Gruppe F18 und uns drei Studenten, Rainer Balzerowski, Arne Wischmann und mir, Nils Dröge, zusammen.

Abbildung 1.1: 6-beinige Laufmaschine - exoskeleton



1.2 Aufgabe

1.2.1 Agenten

Im Zusammenhang mit dem Begriff Roboter trifft man häufig auf den Begriff Agenten. Beide Worte beschreiben einen sehr weiten Bereich. Unter Robotern versteht man alles vom Automaten über den Industrieroboter bis zur Menschmaschine. Unter einem Agenten versteht man im deutschsprachigen Bereich meist einen Geheim-Agenten wie James Bond. Aber in der Informatik wird dieser Begriff schon lange für ein 'selbständig im Auftrag handelndes System' benutzt. Die mobilen Roboter, die wir erstellen wollen, sind also Agenten im Sinne des englischen Ausdrucks Agent.

Rainer Balzerowski schrieb in seiner Diplomarbeit [9]: „Roboter: eine Wortschöpfung des tschechischen Malers und Schriftstellers Josef Capek. Allerdings war es das Theaterstück 'R.U.R. (Rossum's Universal Robots)' seines jüngeren Bruders Karel, ebenfalls Schriftsteller, das dieses Wort seit 1921 weltweit populär machte. Das tschechische Wort 'robota' bedeutet in etwa 'Zwangsarbeiter'."

1.2.2 Robot – Art

Die Roboter sollten auf dem Playground 2000 mit Menschen, insbesondere mit Kindern interagieren. Sie hatten kein konkretes Ziel zu erfüllen, sondern die Aufgabe war, Kunst zu sein. Kunst hat kein primäres Ziel, sie soll den Geist anregen, Freude hervorrufen oder Entsetzen. Roboter haben dagegen normalerweise ein Ziel, das heißt sie sollen eine Aufgabe möglichst zuverlässig erledigen.

Abbildung 1.2: Roboter umringt von Kindern

Wenn nun diese sich widersprechenden Anzätze aufeinandertreffen, kann man versuchen, diese in Einklang zu bringen, oder man geht neue Wege. Wir haben die Begriffe neu gemischt und haben daraus unser Vorgehen bestimmt.

Kunst hat viel mit erfahren zu tun. Wir sind also zu den Ursprüngen der Robotronik zurückgekehrt, um beim Erstellen der Roboter zu lernen und die

neuen Erfahrungen wieder einfließen zu lassen. Das hat in gewisser Weise etwas mit spielerischem Lernen zu tun, also sowohl mit dem Begriff Playground, als auch mit dem Studieren.

Chico MacMurtrie vom Amorphic Robot Works [20], einer Künstlergruppe in San Francisco, beschreibt seine Arbeit so: „The work is an ongoing endeavor to uncover the primacy of movement and sound. Each machine is inspired or influenced both by modern society and what I physically experience and sense. The whole of this input informs my ideas and work.“.

Wir wollten keine Roboter erschaffen, die eine nutzbringende Handlung ausführen, sondern beim Entwickeln und Bauen der Roboter Erfahrungen sammeln, aber auch ein Erlebnis für die Zuschauer erzeugen. „Intelligente Roboter“ stellen für die meisten Menschen eine ungeheure Faszination dar. Das Erforschen der Verhaltensweise - auch weniger komplexer - selbständig handelnder mobiler Maschinen ruft bei Kindern, aber auch Erwachsenen eine grosse Neugier hervor. Diese Neugier, zu erforschen was heute mit einfachen Mitteln möglich ist, hat uns zu diesem Projekt gebracht.

Konkret haben wir das System in vielen gemeinsamen Gesprächen mit den Künstlern der F18 entwickelt. Wobei wir versucht haben, eine beratende Rolle einzunehmen, d.h. wir haben Vorschläge gemacht, Anregungen gegeben, Schwierigkeiten und Probleme aufgezeigt und zusammen mit den daraus resultierenden Ideen der Künstler das Gesamtmodell entwickelt.

2 Vorgaben

2.1 Teamumgebung

Das gesamte Team zur Realisierung der Roboter und der Gesamtinstallation bestand aus 6 Mitgliedern der Künstlergruppe F18 [1] und drei Studenten der HaW Hamburg. Die Gruppe F18 hatte Erfahrung mit mechanischen Großkonstruktionen und einfachen autonomen, mobilen Automaten. Wir, die Informatik-Studenten, hatten demgegenüber durch das Roboterlab [7] Erfahrung mit kleinen, mobilen Robotern. Die Konstruktion der Roboter übernahm F18 und die Programmierung sollten wir übernehmen. Bei der Diskussion über die Schnittstellen bekam man Einblicke in den Arbeitsbereich der anderen Teams.

Für uns war es das Ziel, neben lauffähigen Robotern neue Erfahrungen mit großen Robotern zu machen. Außerdem interssierte uns die Art der Zusammenarbeit mit den Künstlern. Die Gruppe F18 hoffte, neues Knowhow und Hilfe bei der Realisierung zu finden.

2.2 Laborumgebung

Die Hauptbestandteile der Installation waren die drei Roboter. Daneben gab es eine Serverstation und eine Projektionswand. Die Serverstation sah aus wie ein umgedrehter Pilz mit vier Sitzen. Alle drei Roboter kommunizierten über Funk mit dem Server. Außerdem steuerte sie die Projektionswand, die aus fünfzig mal hundert Glühlampen aufgebaut war.

Die Roboter können sich nur über die Serverstation miteinander verständigen, eine direkte Kommunikation ist nicht vorgesehen. Alle drei Roboter sind im Prinzip gleich aufgebaut, die Unterschiede liegen im Bereich der optischen Aufbauten, zudem ist einer der Roboter mit einem Monitor ausgestattet. Da sich mit den Aufbauten auch ein unterschiedliches Gewicht und damit unterschiedliche Fahreigenschaften ergeben, musste dieses mit berücksichtigt werden.

Die „Intelligenz“ der Roboter ist auf mehrere Kontroller verteilt - zum einen gibt es das 6.270 Board, das ausschließlich für die Steuerung des Roboters zuständig ist, die Werte der Sensoren analysiert und den neuen Kurs berechnet. Zum anderen hat jeder Roboter zwei Atmel-Kontroller zur Steuerung und zum Auslesen der Sensoren sowie das Betreiben der Funkstrecke.

Abbildung 2.1: Serverstation

Die Kontroller kommunizieren über ein serielles ASCII Protokoll (s.a. Studienarbeit von Arne Wischmann [8]).

2.2.1 F18-Ctrl

Die Rechnerkarte F18-Ctrl wurde ursprünglich von Lars Vaupel gebaut, um bei vorherigen Projekten die Installationen (so werden bei F18 die Konstruktionen genannt) über einen PC mit der Tastatur zu steuern.

Die Karte setzt den Mikrocontroller Atmel 90S8515 [2] ein und hat eine Größe von 8 * 10 cm (1/2 Euroformat). Auf dem Board sind zwei RS232-Schnittstellen, wobei sie bei einer der Schnittstellen per Hardware von dem Atmel unterstützt wird; die andere benötigt ein Softwaremodul.

Da die Karte kein Display hat und der Kontroller in Assembler programmiert wurde, ist die Entwicklung einer komplexen Steuerung schwierig.

2.2.2 6.270

Das Kontrollerbord 6.270 [3] wurde am Massachusetts Institute of Technology, MIT entwickelt. Die Philosophie des 6.270 ist „to learn by having fun“. Es wurde

Abbildung 2.2: Projektionswand

am MIT für einen Contest benutzt, in dem die konkurrierenden Teams einen Roboter aus Lego bauen müssen. Dieser Roboter wird von dem 6.270 gesteuert, bei den Sensoren handelt es sich meist um IR-Dioden zur Entfernungsmessung.

2.2.3 Plattform

Die Plattform besteht aus 2 Antriebsrädern über dem Schwerpunkt - jedes Rad wird mit einem Dunker-Elektromotor, mit einer Leistung von Watt angetrieben - und zwei Stützrädern. Ein Ultraschallsensor von Polaroid scannt die Umgebung. Durch einen Elektromotor ist ein 180°- Schwenk möglich. Außderdem gibt es zwei Bumper mit je zwei Kontaktsensoren, die um den gesamten Roboter herumführen.

Abbildung 2.3: Zwei Roboter

3 Entwicklungs–Modell

3.1 Voraussetzungen

Die Benutzung eines Modells soll dafür sorgen, dass das Projekt zum festgelegten Zeitpunkt vollständig den Vorgaben entsprechend fertiggestellt wird. Ein gutes, geeignetes Vorgehensmodell ist nicht zum Selbstzweck da, sondern erleichtert die einzelnen Schritte. Mittlerweile gibt es eine ganze Reihe von Entwicklungsmodellen, angefangen vom Wasserfall- und Spiralmodell über iterative Modelle bis zum 'eXtreme Programming', welche alle ihre Vor- und Nachteile haben.

Für die Entscheidung, welches Modell wir nehmen, haben wir uns erst einmal die Vorgaben aufgeschrieben:

- Das Ergebnis oder Teile daraus soll über das Projekt hinaus wiederverwendbar sein.
- Es gibt eine harte Zeitlinie, zu der das Projekt abgeschlossen ist.
- Es gibt keine festen Vorgaben zum Verhalten der Roboter, im Gegenteil, es soll im Laufe des Projektes einzelne Verhaltensatome entwickelt und verfeinert werden.

3.2 Traditionelle Vorgehensmodelle

Die Systementwicklung läßt sich grob in vier Bereiche, auch Phasen genannt, aufteilen:

1. Analyse
2. Design
3. Implementierung/Programmierung/Testen
4. Abnahme/Installation/Integration

Jede dieser Tätigkeiten wird bewusst oder unbewusst bei jeder Entwicklung durchgeführt. Die Zeit, die man für die einzelnen Tätigkeiten benötigt, ist verschieden. Bei Entscheidungen im täglichen Leben verwischen die einzelnen Phasen, bzw. es wird wahllos zwischen ihnen gesprungen. Einzelne Abschnitte werden garnicht oder nur sehr kurz durchlaufen. Das kann dazu führen, dass die Entwicklung nie fertig wird, weil immer neue Ziele definiert werden. Oder kleine Anpassungen rufen große Änderungen hervor, weil sie vorher nicht bedacht werden. Um das zu vermeiden, wurden schon sehr früh in der Informatik Vorgehensmodelle entwickelt. Eines der ersten und bekanntesten ist das Wasserfallmodell.

3.2.1 Wasserfallmodell

Im Wasserfallmodell werden die einzelnen Phasen strikt getrennt, ein Sprung ist nur zu der darüber- bzw. darunterliegenden Phase möglich. Jede Phase wird nur einmal durchlaufen. Durch die Analyse und das anschließende Design wird verhindert, dass Änderungen am Entwurf unter Umständen große Veränderungen bei der Implementierung mit entsprechenden zeitlichen Verlusten zur Folge haben. Das Wasserfallmodell und das folgende Spiralmodell werden ausführlich von Jörg Raasch 1993 [4] vorgestellt. Das Wasserfallmodell ist nicht mehr Gegenstand der allgemeinen Betrachtung, es wird nur noch gelegendlich angewendet, und zwar dann, wenn es schon zu Beginn einer Entwicklung sehr klare Vorgaben gibt, und wenn zudem alle Schwierigkeiten geklärt sind, z.B. beim Entwickeln einer neuen Programmversion.

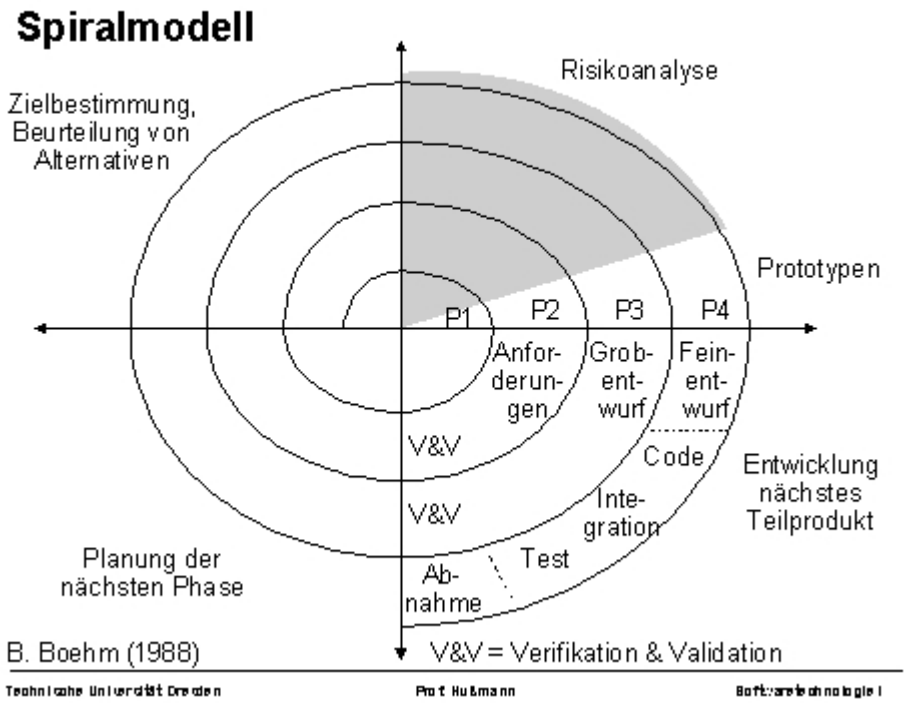
Im Projekt bei F18 standen zu Beginn noch nicht alle Randbedingungen zur Verfügung, somit ließ sich auch keine vollständige Analyse durchführen. Wir brauchten ein flexibleres Modell, in dem die Entwicklung der einzelnen Komponenten iterativ abgenommen werden konnten. Das führte uns zum Spiralmodell.

3.2.2 Spiralmodell

Vor dem Spiralmodell bestand das Problem aller bisherigen Modelle darin, dass die Projektplanung unzuverlässig und die Kontrolle des Projektfortschrittes mangelhaft waren. Um diese Schwächen auszugleichen, entwickelte man das Spiralmodell. Es erlaubt die freie problembezogene Kombination aller bereits existierenden Ansätze unter ständiger Kontrolle des Managements. Das Spiralmodell faßt den Entwicklungsprozeß als iterativen Prozeß auf, wobei jeder Zyklus folgende Aktivitäten enthält:

- Festlegung von Zielen, Alternativen und Rahmenbedingungen
- Evaluierung der Alternativen, Erkennen und Reduzieren von Risiken
- Realisierung und Überprüfung des Zwischenproduktes und Planung der Projektfortsetzung.

Abbildung 3.1: Spiralmodell nach Böhm; Quelle: Prof Hußmann, TU Dresden



- Am Ende jeder Windung steht ein Review, in dem der Projektfortschritt bewertet wird. Zudem wird der Projektfortschritt geplant und verabschiedet.

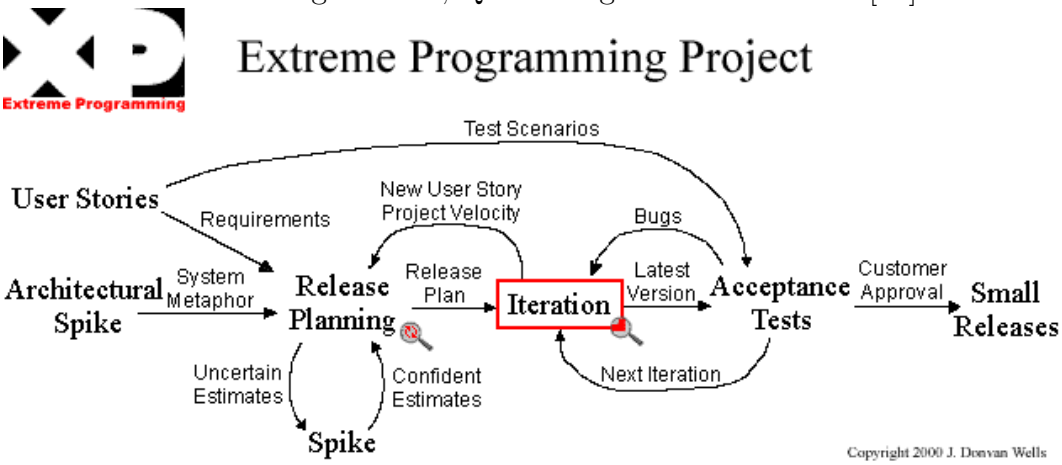
3.3 eXtreme Programming (XP)

'Extreme programming' kommt ursprünglich von Kent Beck [10]. Aber auch eine Diskussion im WikiWikiWeb [12] hat XP stark geprägt. XP beschreibt 12 Kern-Praktiken, die ausführlicheren Überblick wird in John Brewer's 'XP FAQ' [11] beschrieben. Diese werde ich im folgenden weiter einschränken:

- User-Stories
- zweier Teams
- (UnitTests, Functional Tests)
- Kommunikation: zwischen allen Beteiligten und möglichst unmittelbar
- Einfachheit: DTSTTCPW=Do the simplest possible thing that could possible work, YAGNI=You ain't gonna need it, OAOO=Once And Only Once

- Refaktorierung
- Feedback: Möglichst schneller Erfahrungsaustausch mit Anwendern, um Probleme so früh wie möglich kennenzulernen
- Mut: Code ändern, wenn es nötig ist, auch fundamental.

Abbildung 3.2: XP; Quelle: A gentle introduction [14]



Als erstes werden Testroutinen entwickelt, welche das Ergebnis auf Fehler kontrollieren und gleichzeitig das Ziel definieren. Es werden Zweier-Teams gebildet, die jeweils zusammen im ständigen Dialog abwechselnd programmieren, das heißt, einer schreibt, der zweite schaut zu und überprüft das Geschriebene. Das gesamte Team entwickelt zusammen, der Source-Code darf von jedem verändert werden, und es gibt keine Rechte am eigenen Code.

'Extreme programming' (XP) ist ein leichtgewichtiger Softwareentwicklungsprozess für kleine Teams. Der Prozess ermöglicht es, langlebige Software zu erstellen und während der Entwicklung auf vage und sich rasch ändernde Anforderungen zu reagieren. XP-Projekte schaffen ab Tag eins „Geschäftswert“ für den Kunden und lassen sich fortlaufend und außergewöhnlich stark durch den Kunden steuern.

XP geht davon aus, dass von den vier „magischen“ Plangrößen - nämlich Zeit, Umfang, Ressourcen und Qualität - höchstens drei gewählt werden können. Die vierte Größe ergibt sich dann zwangsweise. Setzt man Zeit, Ressourcen und Qualität fest, so kann nur eine ganz bestimmte Menge an Features in das Programm eingebaut werden.

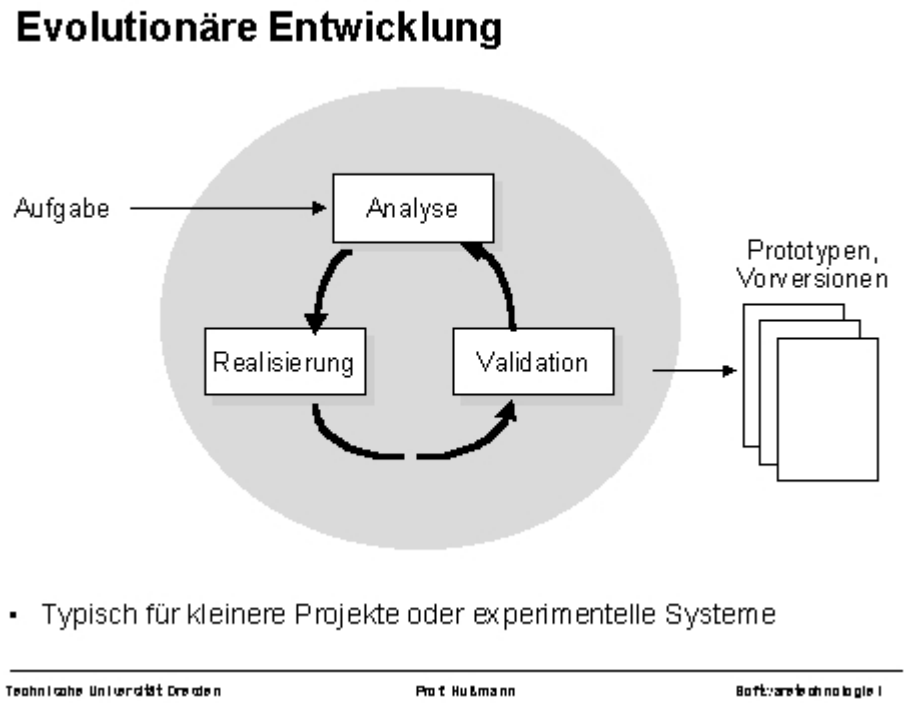
Es werden 12 Praktiken (Practices) eingesetzt, die sich synergetisch ergänzen, und durch die XP zu einer sehr konkreten und leicht erlernbaren Methode wird.

3.4 Evolutionäre Entwicklung

Die evolutionäre Softwareentwicklung hat auch Phasen, die nacheinander durchlaufen werden. Ähnlich wie beim Spiralmodell wird nach der letzten Phase wieder mit der ersten Phase fortgefahren, wenn das Projekt nicht abschlossen ist. Man spricht hier von einem Zyklus. Nach jedem durchlaufenen Zyklus kommt eine neue Version der Software heraus. Entscheidend ist, dass die Versionen nach einem Zyklus eine lauffähige Version des Produktes darstellen. In jeder Iteration wird das Programm erweitert, oder Teile werden verbessert.

Das Spiralmodell überprüft in jedem Zyklus die Ziele, um Maßnahmen ergreifen zu können, wenn ein Projekt nicht nach Plan verläuft. Dagegen können beim evolutionären Modell in einem Zyklus die Spezifikation erweitert oder verändert werden.

Abbildung 3.3: Evolutionäres Entwicklungsmodell; Quelle: Prof Hußmann, TU Dresden



3.4.1 STEPS

Von Christiane Floyd wurde 1992 mit STEPS (Software Technik für Evolutionäre, Partizipative Systemgestaltung) in ihrer Veröffentlichung „STEPS - Pro-

jekthandbuch“ [5], ein evolutionäres Entwicklungsmodell vorgestellt. Das Handbuch beschreibt Vorgehensweise und Techniken für den Entwicklungsprozess.

Wie bei 'eXtreme Programming' ist die Kommunikation mit dem Kunden sehr wichtig, sie findet während des gesamten Projektes bis zum Abschluß statt. Der Kunde ist nicht passiv, sondern ihm werden Rollen zugeteilt, die über die Entwicklung der Software hinaus gehen. Er ist verantwortlich für die Koordination und Bereitstellung der Ressourcen im Betrieb.

In jedem Zyklus wird mittels Partizipation der neue Projektschritt erarbeitet. Eine zentrale Komponente der Systementwicklung stellt Prototypen dar. Diese helfen bei der Evaluierung der Ziele. Floyd schreibt dazu: „Bezüglich der Funktionswahl gibt es die zwei Grundausprägungen 'horizontales Prototyping', bei dem alle anvisierten Systemfunktionen in vereinfachter Form realisiert werden, und 'vertikales Prototyping', bei dem nur ein Teil der Systemfunktionen in der geplanten Form realisiert werden. Häufig wird eine Mischung dieser beiden Konzepte verwendet.“ Floyd unterscheidet dabei zwischen drei Varianten.

- Exploratives Prototyping: Diese Prototypen werden benutzt, um die Funktionalität zu überprüfen. Sie sollen für den Benutzer die Anforderungen und sowie die Realisierbarkeit klären.

- Experimentelles Prototyping: Diese Prototypen dienen dem Nachweis von Lösungsvorschlägen. Kritische Ansätze können evaluiert werden.

- Evolutionäres Prototyping: Der evolutionäre Prototyp unterscheidet sich von den vorherigen in der Lebenszeit. Während explorative und experimentelle Prototypen Wegwerfprodukte darstellen, stellt der evolutionäre Prototyp eine Version des Produktes dar, welches iterativ erweitert wird.

STEPS ist ein zyklisches Projektmodell. In jedem Zyklus wird zusammen mit Benutzern an der Software gearbeitet, um Fehler im Design und in der Ablaufsteuerung aufzudecken und zu vermeiden. Bereits während des Herstellungsprozesses innerhalb eines Zyklus wird das Umfeld, in dem die Software eingesetzt wird, mit Hilfe der Partizipation auf den Einsatz der neuen Software-Version vorbereitet. Am Ende dieses Entwicklungsprozesses steht eine neue Version der Software zur Verfügung. Im darauf folgenden Einsatz, in dem die Benutzer die neue Version verwenden, werden neue Anforderungen und Fehler notiert, die zu einem neuen Zyklus im übergeordneten Softwareentwicklungsprozeß führen. Die verschieden Prototypen helfen bei Partizipation, überprüfen die Spezifikation und bieten schnelle Ergebnisse, welche die Motivation steigern.

3.5 Konsequenz

Weil es nicht möglich war, am Anfang eine vollständige Analyse durchzuführen, fiel das Wasserfallmodell und die klassische Variante das Spiralmodelles sofort

heraus. Der erste Gedanke war, ein iteratives Verfahren zu benutzen. Bei unserer Recherche fanden wir die evolutionäre Systementwicklung von C. Floyd. Letztendlich liess sich keines der erwähnten Entwicklungsmodelle bis ins Kleinste durchsetzen, dass von uns benutzte Entwicklungsmodell kommt der evolutionären Entwicklung am nächsten.

Als erstes haben wir uns zusammengesetzt, um die äußeren Randbedingungen aufzuschreiben: Abmaße des Feldes, Anzahl der Roboter, Aufgaben der Roboter und ähnliches. Diese galten auch gleichzeitig als grobe Definition der Aufgabe. Danach verteilten wir die Kompetenzen und definierten die Schnittstellen. Dieser gesamte erste Prozess ging über mehrere Wochen, in denen auch Details zu einzelnen Komponenten diskutiert wurden. Wir trafen uns in der Anfangszeit einmal wöchentlich. Bei diesen Treffen wurden Vorschläge und Ansätze diskutiert, welche innerhalb der Woche in den Gruppen ausgearbeitet oder weiter analysiert wurden.

Wie bei allen Projekten gab es auch bei diesem nur begrenzte Geldmittel und eine harte Zeitlinie. Bei vielen Projekten ist es sonst üblich, den verabredeten Abgabetermin (mehrfach) zu verschieben. Diese Möglichkeit hatten wir nicht, sonst hätte es nichts auf dem „Playground 2000“ zu zeigen gegeben. Der enge Zeitplan war das bestimmende Element für den Entwicklungsprozess. Wir mussten Aufgaben, die eigentlich voneinander abhängig waren, parallel abarbeiten, und bei der Gesamtlösung mußten wir mit berücksichtigen, dass einige Teillösungen nicht realisierbar sind, ob aus Zeitmangel oder aus technischen Gründen.

Wenn die Zeit knapp wird, hilft es oft, die benötigten Komponenten dazuzukaufen. Doch auch hier hatten wir ein strenges Budget. Das vorhandene Geld ließ es zudem nur zu, low level Sensoren zu benutzen.

Wenn die beiden wichtigsten Ressourcen knapp sind, hilft nur Kreativität weiter, oder ein Herabsetzen der Anforderungen und persönlicher Einsatz.

Schon die ersten Gespräche zeigten, dass wir bei der Zusammenarbeit mit den Künstlern nicht konsequent die in der Literatur und im Studium beschriebenen Entwicklungsmethoden und Diagramme benutzen konnten, um zu kommunizieren, außer bei Diskussionen innerhalb der HaW-Beteiligten.

Die grobe Aufteilung der Aufgaben war schon von Beginn an klar. Um im Zeitraum fertig zu werden, schien es notwendig, zuerst weiter die Kompetenzen abzuklären und danach möglichst alle zu erledigenden Arbeiten zu definieren. Man sollte eher benennen sagen, da auch dies nur in einem lockeren Gespräch geschah. Die Kernaussagen wurden festgehalten zur späteren Überprüfung. Das detaillierte Aufschreiben hätte zuviel Zeit beansprucht und erschien uns in diesem Rahmen nicht nötig zu sein. Die Anzahl der Aufgaben war überschaubar, und es wurden auch gleich die Rollen verteilt. Einige der aufgezählten Tätigkeiten wurden entweder gar nicht oder unvollständig erledigt; sie fielen aber nicht ins Gewicht, denn sie konnten zum Teil nicht aus Zeitgründen durchgeführt werden, oder sie haben das Projekt nicht vereitelt. Mit anderen Worten: Mit einer dokumentierten Beschreibung der Aufgaben, mit Definitionen der Rollen, Aktivi-

täten und Methoden wäre das Projekt sozusagen „gestorben“. Aus den genannten Gründen zeigen die folgenden Angaben nur relevante Punkte, die eine Aufzeichnung benötigten oder im Gedächtnis geblieben sind.

Um dem Projekt zu einem erfolgreichen Zieleinlauf zu verhelfen, haben wir uns zu folgenden Maßnahmen entschlossen:

- Die Gruppe F18 stellte für uns den Kunden dar, mit dem wir das Produkt erarbeiten.
- Das Projekt wird in Teile zerlegt, welche weitgehend unabhängig voneinander bearbeitet werden können.
- Es wurden Teams gebildet, welche sich um die Bearbeitung der Teilprojekte kümmerten. Die Teamzusammensetzung konnte sich dabei ändern. Ein Mitglied konnte in mehreren Teams arbeiten. Da die zu bearbeitenden Aufgaben ein sehr weites Spektrum an Fähigkeiten erforderten, vom Schweißen über Starkstromelektrik bis zur Programmierung, war diese Flexibilität notwendig.
- Es gab wöchentliche Treffen, bei diesen Sitzungen analysierten wir Probleme und Ziele. Die Spezifikation wurde erarbeitet und erweitert. Außerdem wurden die nächsten Schritte für die Teams festlegt.

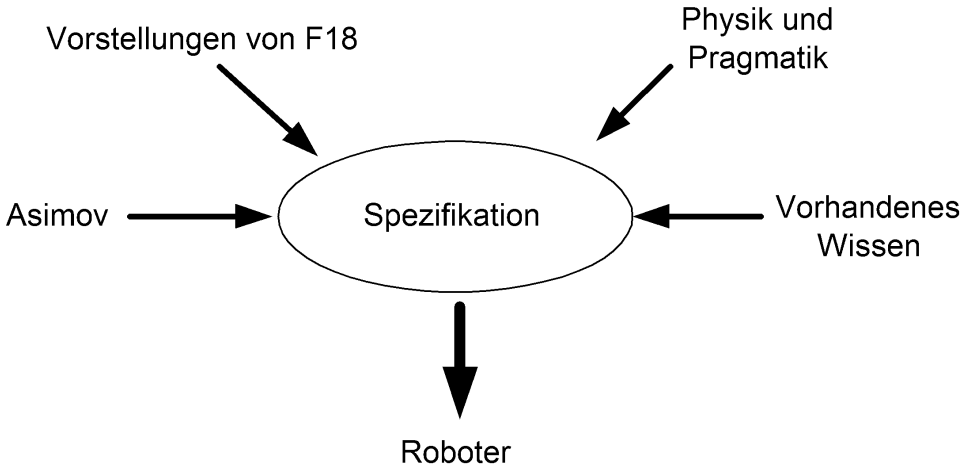
Zu Beginn des Projektes gab es nur einige allgemein formulierte Zielsetzungen, deshalb brauchten wir ein Entwicklungsmodell, welches es uns ermöglichte, die Spezifikation Stück für Stück zu erweitern. 'Extreme programming' war uns zum Zeitpunkt der Entstehung nicht so geläufig, zudem gab es wichtige Kriterien, die wir nicht erfüllen konnten. Die Techniken von XP liessen sich zudem nicht flexibel verändern. Um ein Projekt erfolgreich mit 'extreme programming' durchzuführen,war es aber notwendig, alle Kernpraktiken durchzuführen. Wir brauchten ein iteratives Modell, welches nicht auf Software beschränkt war.

Den größten gemeinsamen Nenner fanden wir im partizipativen, evolutionären Entwicklungsmodell STEPS. Es bot die Möglichkeit, die Spezifikation zyklisch zu erweitern. Zudem wurde mit den Prototypen eine mächtige Technik geboten, die Entwicklung iterativ zu erweitern, oder mögliche Teillösungen zu evaluieren.

4 Durchführung

Zu Anfang gab es viele Ideen, wie die Roboter sein sollten. Dabei waren die Vorstellungen zum Teil recht unterschiedlich, insbesondere die der Künstlergruppe F18 und die der HaW-Beteiligten. Dazu kamen noch triviale Nebenaspekte, zum Beispiel, dass ein Roboter niemanden verletzen oder gefährden sollte . Es galt als erstes, aus den vielen Vorgaben eine Spezifikation zu erarbeiten, die in der kurzen Zeit zu realisieren war.

Abbildung 4.1: Einflüsse in die Spezifikation



Es gibt ein sogenanntes magisches Dreieck, mit den Eckpunkten „Qualität, Kosten und Termin“. Es zeigt sehr schön, dass man nicht bei allen Faktoren maximieren kann. Wir hatten nun einen fixen Abgabetermin und ein begrenztes Budget, die Qualität durfte aber nicht leiden. Wir nutzten unser Wissen und Entwicklungen von früheren Arbeiten, um Kosten und Zeit zu sparen.

Viele Punkte zur Realisierung der Vorgaben waren noch ungeklärt, wir konnten also nicht vorbestimmen, wieviel Zeit wir brauchen würden. Um dennoch ein zuverlässiges System zu entwickeln, das möglichst viele Eigenschaften enthält, teilten wir die Funktionalität auf - mit einem Kern und Erweiterungen. Die Kernfunktionalität stellt die Grundfunktionen dar: Fahren, Kollisionen vermeiden. Technisch half uns die Subsumption Architiktur von Brooks [6], welche auch in der Studienarbeit von Arne Wischmann [8] beschrieben wird.

Eine Vorgabe war, daß sich der Roboter innerhalb eines Bereiches aufhalten soll, der nicht durch Hindernisse abgegrenzt ist. Es gab mehrere Möglichkeiten, dies zu realisieren. Diese Funktion stellt aus diesem Grund eine Erweiterung dar. Wenn, wie weiter unten beschrieben, ein Weltmodell zum Erfüllen dieser Aufgabe benutzt wird, dann muß der Roboter eine gewisse Planung seiner Handlung vornehmen, das heißt, er arbeitet deliberativ. Die Kernfunktion mit dem Ausweichen von Hindernissen arbeitet reaktiv.

Dieser modulare Aufbau mit einer Kernfunktionalität ermöglichte uns, das Gesamtsystem stabil zu machen. Diejenigen Teile, die sich aus Zeit-, Kosten- oder technischen Gründen nicht realisieren liessen, beeinträchtigen die Systemfunktionalität nicht. Das hybride Verhalten mit den gegensätzlichen Ansätzen: Reaktiv (reagieren auf die Umwelt) und deliberativ (Planen), hat den Vorteil, dass man den Kern sehr einfach halten kann. Das Reagieren auf Ereignisse ist eine sehr simple Strategie. Und die Erweiterungen können dem Roboter recht komplexe Fähigkeiten geben, siehe auch [16].

4.1 Asimov: der Bumper muss an das 6.270

Auch wenn wir bei der Überlegung der einzusetzenden Sensoren nicht an Asimov'sche Robotergesetze [15] dachten, führte uns der gesunde Menschenverstand zu den gleichen Vorüberlegungen: Wir gingen davon aus, dass bei dem „Playground 2000“ viele Kinder und Jugendliche auf dem Gelände sind. Es handelt sich schließlich um eine Spielemesse, und Kinder haben meistens, durch Filme gefördert, keine Angst vor den Robotern. Es gibt in der Filmwelt die bösen und die guten Roboter, und unsere Roboter sollten gute darstellen. Dazu gehört, dass sie keine schnellen Bewegungen machen, die bedrohlich wirken. Kinder können sich nicht vorstellen, dass ihnen der gute Roboter etwas antun könnte, auch nicht unbeabsichtigt. Und das sich der Roboter nicht selbst zerstört, war ohnehin voraussgesetzt. Er hat es auch nur einmal ernsthaft versucht, als einer der Roboter die Serverstation attackierte.

Also überlegten wir uns Sicherheitsmaßnahmen, die auch dann greifen würden, wenn das Programm oder ein Task abstürzt. Wir diskutierten darüber, welche Konsequenzen ein Notausschalter hätte, der die Stromversorgung unterbricht und wie wir den Bumper behandeln.

Ein gut sicht- und erreichbarer Notausschalter hat in dem oben beschriebenen Umfeld mehr Schaden als Nutzen. Die Versuchung, den Taster zu betätigen, erschien sehr groß. Das Trennen der Stromversorgung bedeutet aber einen manuellen Eingriff. Die Roboter könnten mehr „aus“ als „an“ sein. Auf der anderen Seite sind die Roboter sehr flink. Bis jemand den Schalter entdeckt und ausgelöst hat, ist es eventuell zu spät. Deshalb konzentrierten wir uns auf den Bumper als letzte Sicherheit.

Jeder Roboter sollte rundherum mit Bumpern bestückt sein, da auch ein Rück-

wärtsfahren möglich sein sollte, oder weil Richtungsbewegung zu seitlichen Kontakten führen konnten. Herausgekommen sind vorne und hinten je ein Bumper mit zwei Triggerpunkten. Die Bumper sind wie ein Halbkreis geformt und bilden zusammen einen Kreis um den Roboter herum. Die nächste Frage war, ob die Bumper direkt den Strom der Motoren oder des ganzen Systems ausschalten sollten. Wenn die Bumper alles ausschalten, ergeben sich die gleichen negativen Konsequenzen wie beim Notausschalter. Wenn wir die Motoren ausschalten, ist eine Motorbremsung nicht möglich, und die Trägheit der Robotermasse sorgt weiter für Schub. Eine zusätzliche Schaltung erhöht die Komplexität. Außerdem sollte es dann möglich sein, dass die Motoren durch die Kontroller wieder eingeschaltet werden können. Wenn aber die Motoren dann doch durch einen Kontroller gesteuert werden, warum dann nicht gleich auch die Bumper mit den Kontrollern bewerten? Im Roboterlabor der HaW Hamburg hatten wir Erfahrung mit nebenläufigen Prozessen auf dem 6.270 gemacht, die wichtige Aufgaben steuern. Zudem hatten wir nicht einmal erlebt, dass das Betriebssystem auf dem 6.270 abtürzte, sei es durch Tasks ausgelöst oder durch Eingangssignale. Das einzige, was den Kontroller in einen unkontrollierten Zustand brachte, war eine zu geringe Spannung. Aus diesen Gründen entschieden wir uns für folgende eine Lösung:

Abbildung 4.2: Roboter mit rundherum befestigten Bumpern

Alle Bumper werden direkt an das 6.270 angeschlossen. Auf dem Board läuft ein Hauptprozess, von diesem werden alle anderen gesteuert. Die Prozesse werden von diesem gestartet und können bei Bedarf gestoppt werden. Die einzige Aufgabe, die der Hauptprozess hat, ist: In einer Endlosschleife die Bumper-Eingänge zu überprüfen und bei Kontakt zuerst die Motoren anzuhalten. Danach werden alle anderen Prozesse in den Ausgangszustand, durch Stoppen und erneutes Starten, zurückversetzt. Das Betriebssystem des 6.270 benutzt ein Zeitscheibenverfahren, um zwischen den Prozessen zu wechseln. Dadurch ist sichergestellt, dass der Hauptprozess auch seine Aufgabe erfüllen kann. Auch wenn Isaac Asimov bei der Formulierung der Robotergesetze weitaus komplexere und leistungsfähigere Roboter im Sinn hatte, ist es sehr wichtig, sich auch bei „dummen Automaten“ in der Entwurfsphase Gedanken über die Sicherheit zu machen. Dieses Konzept

lässt unter der Berücksichtung, dass das Betriebssystem zuverlässig arbeitet, eine sehr genaue Beurteilung über die Wirksamkeit zu. Dennoch bleibt eine Unsicherheit übrig, zwar werden die Befehle zum Steuern der Motoren vorhersagbar abgesetzt, ob sie aber durchgeführt werden, muss gesondert betrachtet werden, da die Motoren von einem anderen Kontroller gesteuert werden.

Die drei Asimov'schen Robotergesetze (frei übersetzt)

1. Ein Roboter darf keinen Menschen verletzen oder durch Untätigkeit zu Schaden kommen lassen.

2. Ein Roboter muss den Befehlen eines Menschen gehorchen, es sei denn, solche Befehle stehen im Widerspruch zum ersten Gesetz.

3. Ein Roboter muss seine eigene Existenz schützen, solange dieser Schutz nicht dem ersten oder zweiten Gesetz wiederspricht.

4.2 Der Motorport an das F18-Ctrl

Wie die Motoren angesteuert werden, war schnell entschieden. Zwar wäre es aus Sicherheitsgründen am besten gewesen, die Motoren direkt an das 6.270 anzuschließen, aber gute Gründe sprachen dagegen: Die Motorports auf dem 6.270 waren für kleine Elektromotoren ausgelegt, der max. Strom bei zwei parallel geschalteten H-Brücken ist 1600 mA. Das heißt, es muss eine neue Leistungsschaltung entworfen werden. In einem Gespräch berichtete uns Lars, dass er in früheren Projekten schon eine Schaltung benutzt habe, die einen Atmel-Kontroller benutzte. Die Spannungspeeks beim Ein- und Ausschalten habe er durch die Benutzung von Relais kompensiert. Da wir genug mit anderen Aufgaben zu tun hatten, entschieden wir uns auch hier, auf bewährte Technik zurückzugreifen. Ein zweiter Kontroller war im System!

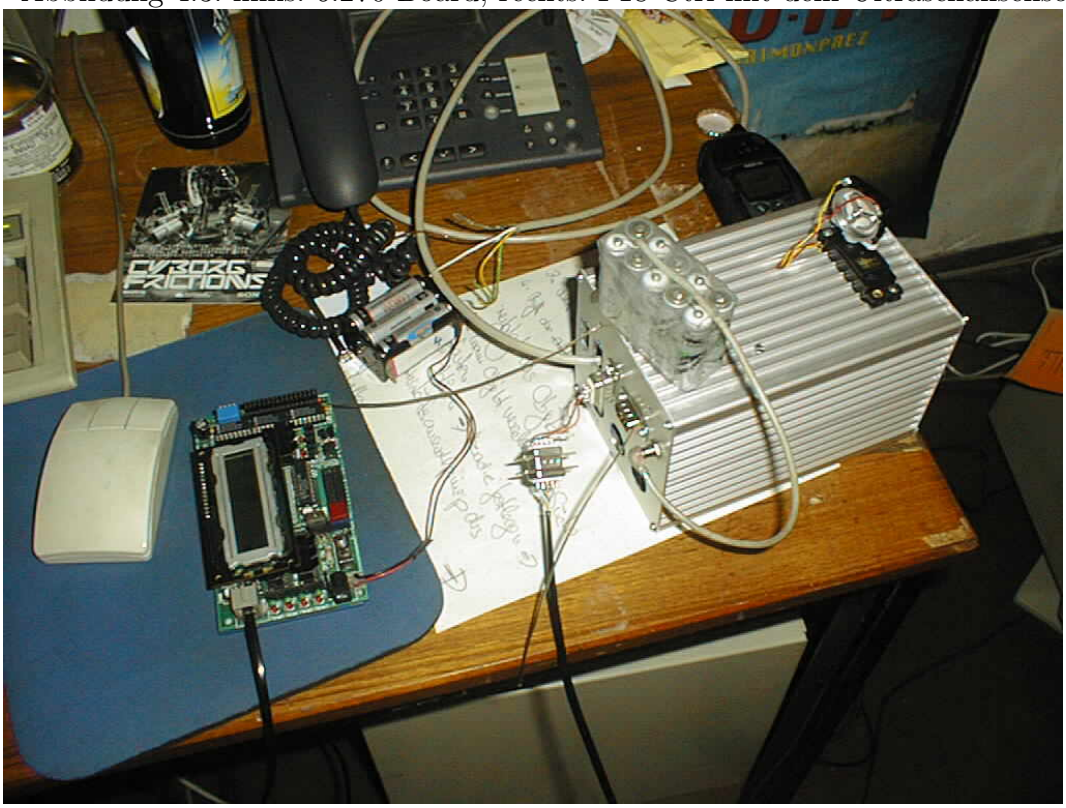
Nun mussten die beiden Kontroller gekoppelt werden. Dazu gab es zwei Möglichkeiten, zum ersten über eine serielle Verbindung, die bei bei den Kontrollern vorhanden und erprobt war. Zum zweiten hatten beide Prozessoren min. ein SPI-Register. Diese sind genau für die Kommunikation zweier Kontroller synchrone serielle Schnittstellen konzipiert, siehe dazu die Studienarbeit von Arne Wischmann [8].

Es gab drei Fragen zu beantworten:

1. Sind diese Schnittstellen kompatibel, vielleicht sogar genormt?

2. Beim 68HC11 sind die Ports mehrfach belegt; ist der Port, an dem das Register hängt auf dem 6.270-Board noch frei oder gibt es Konflikte mit System- und Betriebssystemfunktionen?

Abbildung 4.3: links: 6.270-Board, rechts: F18-Ctrl mit dem Ultraschallsensor



3. Wie schreibt man den Assemblercode und bindet ihn in das Betriebssystem ein?

Nachdem wir nicht eine der Fragen nach kurzen Recherchen beantworten konnten, entschieden wir uns für die sichere, aber langsame Methode mit der seriellen Schnittstelle.

Beide Kontrollerboards kommunizierten bereits seriell mit 9600 Baud mit dem PC. Jetzt mussten sie nur noch zusammenfinden. Der Atmel-Kontroller wird vom PC bereits mit Steuerungsbefehlen versorgt, wie wir es mit dem 6.270 vorhatten. Deshalb war das F18-Ctrl für uns die Referenz und wir mussten nur die Software des 6.270 anpassen.

Da der 68HC11 über zwei serielle Schnittstellen verfügte, überlegten wir, die Zweite für die Kommunikation mit dem F18-Ctrl zu benutzten. Leider waren die Aus- und Eingänge auf dem Board schon belegt. In vorherigen Untersuchungen hatten wir herausgefunden, dass die Betriebssystemfunktionen, die serielle Schnittstellen betreffen, Probleme bereiten. Sie entstanden dadurch, dass das Betriebssystem die serielle Schnittstelle überwacht. Über ein Hardwareregister lässt sich diese Überwachung ein- und ausschalten. Wir schrieben das Programm so, dass sich die Kommunikation mit dem PC und dem F18-Ctrl über DIP-Switches

umschalten liess. Mit einem Spike[1] überprüften wir, ob der Datentransfer funktionierte. Während der Übertragung verschwanden Zeichen. Das war auch nicht weiter verwunderlich, da der interpretierte Programmcode zu langsam war und kein Hardwarehandshake auf dem 6.270 vorgesehen war.

Den, in Assembler geschriebenen, seriellen Treiber des Betriebssystems wollten wir erst einmal nicht verändern, um nicht in den Vorarbeiten zu scheitern. Stattdessen nahmen wir den Vorschlag von Lars an, ein einfaches Softwarehandshake zu realisieren: Der 6.270 sendet so schnell er kann, und in Empfangsrichtung sendet er ein Leerzeichen, wenn er bereit ist. Später stellte sich heraus, dass diese Entscheidung die Möglichkeiten des 6.270 sehr stark einschränkt, da die Kommunikation einen nicht unerheblichen Aufwand darstellt, der die wertvollen Ressourcen, Speicher und Rechenleistung erheblich verringert.

4.3 Ereignisgesteuert im Weltmodell

Die Roboter sollten sich in einem festen Spielfeld aufhalten. Innerhalb dieses Bereichs sollten sie auf Personen reagieren und wenn niemand da war, eine Anzahl von Einzel- oder Gruppenfiguren aufführen.

Natürliche Bewegungsabläufe nachzuahmen heißt, die in der Natur vorkommenden Bewegungen bzw. Handlungen von Menschen - und allgemein von Lebewesen - von den Grundtatsachen her zu begreifen.

Der Mensch ist in der Lage, seine Handlung in Gedanken durchzuspielen. Dies geschieht zum Teil unbewusst für kleinere Abläufe, wie z.B. einen Gegenstand nehmen, oder auch lange Abläufe, wie z.B. an einen anderen Ort fahren. Die Voraussetzung dafür ist, dass der Mensch ein Model seiner Umgebung, ein sogenanntes Weltmodell hat, in dem er zukünftige Handlungen ablaufen lassen kann, ohne dass diese wirklich vollzogen werden.

Wenn sich Menschen immer in einem solchen Abbild der Realität bewegen, liegt der Gedanke nahe, Roboter müssten sich ebenfalls mit Hilfe eines Weltmodells bewegen.

Es gibt in der Natur jedoch mehr Lebewesen, die ohne ein Modell auskommen, z.B. Einzeller oder Bakterien. Tiere reagieren auf ihre Umwelt in der Weise, dass ein Ereignis in Abhängigkeit vom Zustand eine Reaktion auslöst. Ein Zustand ist z.B.: Der Magen ist nicht gefüllt; das Ereignis: Nahrung ist in der Nähe; und es folgt als Reaktion: Das Tier schwimmt in Richtung der Nahrung. Dieses Verhalten gilt, sofern es nicht geplant wird, als ereignisgesteuert.

Alle höheren Lebewesen haben ein Weltmodell. Es bringt den Vorteil, dass sich Ereignisse planen lassen.

Für die Bewegungen eines Roboters durchaus wichtig sind Situationen wie z.B., dass er irgendwo gegen etwas prallt, wenn er geradeaus fährt. Das muss berück-

[1] Ein Wegwerfprogramm zur Überprüfung einer Funktionalität oder Algorithmus

sichtigt werden. Der Nachteil ist allerdings, das Planen benötigt Ressourcen[2], und je genauer das Ergebnis sein soll, desto komplexer werden die Modelle und desto mehr Ressourcen benötigt man. Da wir mit dem 6.270 kein sehr leistungsfähiges System hatten, konnten wir solche komplexen Situationen nicht im voraus abbilden; es blieb nur, den Großteil ereignisgesteuert zu behandeln. Auch so lässt sich verhindern, gegen etwas zu prallen. Ist in Fahrtrichtung ein Hindernis, dann stoppt der Roboter oder ändert die Richtung.

Bei der Diskussion über die eingesetzten Sensoren kam die Frage auf, wie der Roboter Menschen erkennt. Die Antwort: Er erkennt sie an der Bewegung. Daraus folgt, das wir uns die Vergangenheit eines Objektes merken müssen, um einen Bewegungsvektor zu bekommen. Nun kann man für jedes Objekt ein oder mehrere Positionen speichern oder man numeriert die Objekte und schreibt diese Identifikationsnummer in eine Karte, in das Weltmodell also.

Wir sind dem Gedanken, alles in ein Modell zu schreiben, weiter nachgegangen. In der Literatur [17] findet man neben der einfachen Matrix auch Formen, in denen wichtige Bereiche feiner strukturiert werden. Die andere Möglichkeit ist, Objekte mit Vektoren zu markieren. Und natürlich mit mischbaren Systemen. Für welches Modell man sich entscheidet, hängt vom Einsatzzweck des Roboters ab - bei dem einen lassen sich Fahrwege besser berechnen, bei anderen lassen sich besser Änderungen der realen Welt einpflegen.

Wichtig für uns war, dass das Modell nicht viel kosten darf. Wir brauchten keine große Genauigkeit. Das Spielfeld war rechteckig, ideal für eine Pixelmatrix. Wenn wir alle festen Objekte schon vorher in das Modell eintrugen, dann ist alles andere ein Mensch oder Roboter. Die Position der anderen Roboter bekamen wir von der Serverstation. Wir mussten uns nur noch über die Genauigkeit der Matrix einigen, und die war abhängig von der Genauigkeit der Positionsbestimmung.

Um die beiden Vorgaben erfüllen zu können, der Roboter solle erstens einen bestimmten Bereich nicht verlassen und zweitens auf Passanten in diesem Bereich reagieren, brauchten wir zwei Arten von Sensoren. Zum einen Nahbereichssensoren, denn diese scannen alles in ihrer Reichweite. Und außerdem brauchten wir andere Sensoren, die in Lage sind, eine relative oder absolute Position zu bestimmen. Aus der Position können wir ermitteln, ob sich der Roboter innerhalb oder außerhalb des Spielfeldes befindet. Oder wir benötigen Sensoren, die ein Verlassen des Bereichs direkt melden, z.B. über Induktionsspulen.

4.4 Notwendige low level Sensoren

Das Auswerten eines Videobildes zur Identifikation von Objekten oder gar Bewegung ist eine rechenintensive Aufgabe. Um Hindernisse zu erkennen, reichen einfachere Sensoren. Eine bewährte Technik ist das Sonar - man sendet einen

[2]Rechenleistung und Speicher

Ultraschallimpuls und misst die Zeit, bis man das erste Echo empfängt. Daraus errechnet man die Entfernung.

Mit Infrarot kann man auch Entfernungen bestimmen. Erste Möglichkeit: Man sendet infrarotes Licht aus, und misst gleichzeitig, wie viel infrarotes Licht man empfängt[3]. Je heller, desto näher ist das Objekt. Leider reflektiert nicht jeder Gegenstand das Licht gleich gut, das führt zu einer Ungenauigkeit. Das größere Problem ist aber der Umstand, dass die Roboter im Freien agieren. Durch die Sonne können sich Gegenstände aufheizen und werden zur Infrarotquelle.

Die andere Form der Lichtsensoren, der Sharp [19], arbeitet genauer. Hier wird mit Hilfe einer Optik der Einfallswinkel gemessen: Je größer der Winkel, desto geringer der Abstand [18]. Aber auch hier bleibt das Problem mit den sonnenaufgeheizten Infrarotquellen.

Die Roboter sind ca. 1 Meter im Durchmesser, Infrarotsensoren arbeiten bis 1,5 Meter. Die Ultraschallsensoren fangen erst bei einem halben Meter an. Das waren die Gründe, warum wir ausschließlich Ultraschallsensoren eingesetzt haben.

Wir benutzten dazu einen Sensor von Polaroid, der auch schon bei anderen Robotern erfolgreich eingesetzt wurde.

4.5 Barke oder Satellit

Nachdem wir mittels Ultraschall Objekte im Sensorenbereich erfassen konnten, war der Roboter in der Lage, Kollisionen zu verhindern. Außerdem konnte auf die erfassten Objekte reagiert werden. Eine einfache Interaktion, wie z.B. verfolgen oder flüchten, war somit möglich.

Nun musste noch sichergestellt werden, dass der Roboter das Spielfeld nicht verlässt.

Das lässt sich durch einen virtuellen Zaun erreichen. Um das Feld werden Induktionsschleifen verlegt. Beim Überfahren bekommt der Roboter einen Impuls, der zu einer sofortigen Umkehr führen muss. Es muss dabei sichergestellt werden, dass der Roboter innerhalb der Induktionsschleifen wenden kann, damit er nicht darüber hinausfährt, dann dreht und beim Zurückfahren erneut ein Signal bekommt.

Es war nicht sichergestellt, dass wir eine solche Induktionsschleife verlegen durften. Zudem wollten wir das System auch für andere Auftritte benutzen, und nicht überall läßt sich etwas vergraben. Also brauchten wir eine andere Lösung.

Um die eigene Position zu bestimmen, bieten sich zwei Systeme an. Aus der Schifffahrt kennt man die Kreuzpeilung. Damit läßt sich die relative Position zu zwei feststehenden Referenzpunkten ermitteln. Aufgrund der oben erwähnten Einschränkungen bezüglich der Sonne kommt nur eine Peilung mit Funk in Frage.

[3] Die Zeitverzögerung kann aufgrund der geringen Entfernung wegen der Lichtgeschwindigkeit vernachlässigt werden

Die zweite Möglichkeit ist ein Satelitensystem. Für GPS gibt es mittlerweile komplette Geräte, die einem die absolute Postion über die serielle Schnittstelle liefern. Um mit diesen Werten etwas anfangen zu können, brauchen wir ein Abmaß des Spielfeldes, in dem wir dann die eigene Position überprüfen können.

4.6 Realisierte Systeme

In der ersten formulierten Spezifikation von F18 sollten die Roboter über zahlreiche Sensoren verfügen: „Sie sind mit Sensoren ausgestattet, die der Erfassung von Bild, Klang, Wärme, Distanz und biometrischen Daten (z.B. Pulsfrequenz) dienen.“ Siehe Anhang. Wie schon erwähnt, konnte aus Kosten- und Zeitgründen nur ein Teil umgesetzt werden. Das Hauptziel waren funktionsfähige und unbeaufsichtigt handelnde mobile Roboter. Dieses Ziel haben wir erreicht. Die drei Roboter haben je einen Polaroid-Ultraschallsensor, die um 180° schwenkbar sind. Zur Sicherheit haben alle Roboter Bumper, welche die Motoren sofort blockieren, falls es doch zu einem Zusammenstoß kommen sollte. Zudem verfügen sie über GPS-Empfänger zum Ermitteln der aktuellen Position. Und ein Roboter verfügt über eine Kamera - die Bilder werden zur Serverstation gesendet.

Die gesamte Installation wurde auf dem Playground 2000 vorgeführt. Die Roboter funktionierten auch in der unbekannten Umgebung wie geplant und es wurde niemand verletzt. Leider funktionierte das GPS in der Schweiz nicht zuverlässig, so dass ein vollständig unbeaufsichtigter Betrieb nicht möglich war. Auch auf einem 'Tag der offenen Tür' der HaW Hamburg wurde ein Roboter im vielbenutzten Eingangsbereich vorgeführt. Auch in dieser neuen Umgebung arbeitete das System sicher und zuverlässig.

Die entwickelte Plattform läßt sich nun einfach erweitern: Der Server könnte eine Bildauswertung vornehmen. Die Roboter könnten mit zusätzlichen Sensoren ausgestattet werden, oder das Verhalten der Roboter kann durch die einfache Syntax der 6.270-Board verändert oder erweitert werden. Statt des GPS läßt sich ein an die Umgebung angepasstes System zur Ortsbestimmung einsetzen.

5 Aussicht und Kritik

Zum Abschluß des Projektes stellt man sich die Frage: Was ist gut gelaufen und was hätte man anders machen können? Am Anfang hatten wir das Problem, dass mit den Künstlern von F18 und den Studenten der HaW Hamburg unterschiedliche Ansätze der Problemlösung und andere Abstaktionen der Vorstellung über das Ergebnis aufeinander trafen. Es gab aber mehr Gemeinsamkeiten als Unterschiede. Neben der Leidenschaft für die Arbeit im Hinblick auf eine gemeinsame Vision, benutzen auch die Künstler der F18 ein strukturiertes Vorgehen und Methoden zur Erfüllung der Aufgaben. Zum Lösen von neuen Problemen braucht ein Ingenieur seinerseits künstlerische Fantasie und Kreativität.

Einen Großteil der Anfangszeit verwendeten wir darauf, eine „gemeinsame Sprache“ zu entwickeln. Wie schon erwähnt, steht bei Robot-Art die Audio-Visuelle Erfahrung im Vordergrund. Dieser Erfahrungsprozess setzt schon bei der Entwicklung ein, sodass dies auch beim Vorgehensmodell berücksichtig werden muß. Da auch während der Realisierung noch Ideen einfließen sollten, hat sich das evolutionäre Entwicklungsmodell bewährt. Die Prototypen helfen technische Detaillösungen zu überprüfen und unlösbare Probleme rechtzeitig zu erkennen. Durch den modularen Aufbau des System mit einem simplen Kern konnte der Ausfall des GPS in der Schweiz ohne große Änderungen an der Software kompensiert werden.

Die Techniken des Projektmanagments ließen sich aufgrund des Umfeldes, insbesondere wegen der vielen Unbekannten nicht einsetzten. Die Studienphase, welche normalerweise vorausgehend sein sollte, verlief parallel. Dadurch war es vielfach nicht möglich, einen festen Zeitplan für die Teilprojekte vorzusehen. Um nicht planlos vorzugehen, wurden innerhalb der Teilprojekte, welche von den Teams bearbeitet wurden, Artbeitspakete erstellt. Bei kritischen Teillösungen haben wir für die Studienphase Deadlines vorgeschrieben.

Um mehr Bewegungsvarianten für die einzelnen Roboter implementieren zu können, hätten wir zumindest eine fertige Plattform zu einem früheren Zeitpunkt haben müssen. Die großen Motoren erzeugen erwartungsgemäß eine große Störstrahlung, diese zu kompensieren ist bei einem kleinen Modell nicht möglich. So hätte man das GPS warscheinlich auch in der Schweiz benutzten können.

Im Großen und Ganzen kann ich unser Vorgehen empfehlen. Das fertige System stellt einen guten Ausgangsstatus dar, indem man jetzt noch zahlreiche Erweiterungen einbauen kann.

Während der vielen Wochen hatten wir zusammen viel Spaß, so dass die Arbeit und die investierte Freizeit in den Hintergrund geriet. Die Zusammenarbeit mit F18 hat mir neben den vielen amüsanten Augenblicken viele neue Erfahrungen gebracht, die ich nirgendwo anders hätte sammeln können. Besonders die Zeit mit meinen beiden Mitstreitern der HaW Hamburg möchte ich nicht missen. Einige Momente der arbeitsreichen Wochenenden und Treffen bei F18 bis teilweise spät Nachts fürten zuweilen Lachausbrüchen, die uns die Tränen in die Augen schießen ließen. An dieser Stelle bedanke ich mich bei meinem betreuenden Professor Kai von Luck und meiner Familie, insbesondere meiner Frau Andrea für die Geduld und Unterstützung.

A Konzeption und Skizzen von F18

Ikit ist eine Installation bestehend aus fünf Objekten, die drei Einheiten ergeben: drei mobile Funktions-Systeme MFS, eine stationäre Servereinheit SSE und ein grob auflösendes Rastermatrixdisplay RMD. Die mobilen Roboter agieren frei in einem begrenzten Terrain. Sie tauschen über ein Funknetz sowohl mit der Serverstation als auch untereinander Informationen aus. Ihnen sind unterschiedliche Aufgabenbereiche zugeordnet.

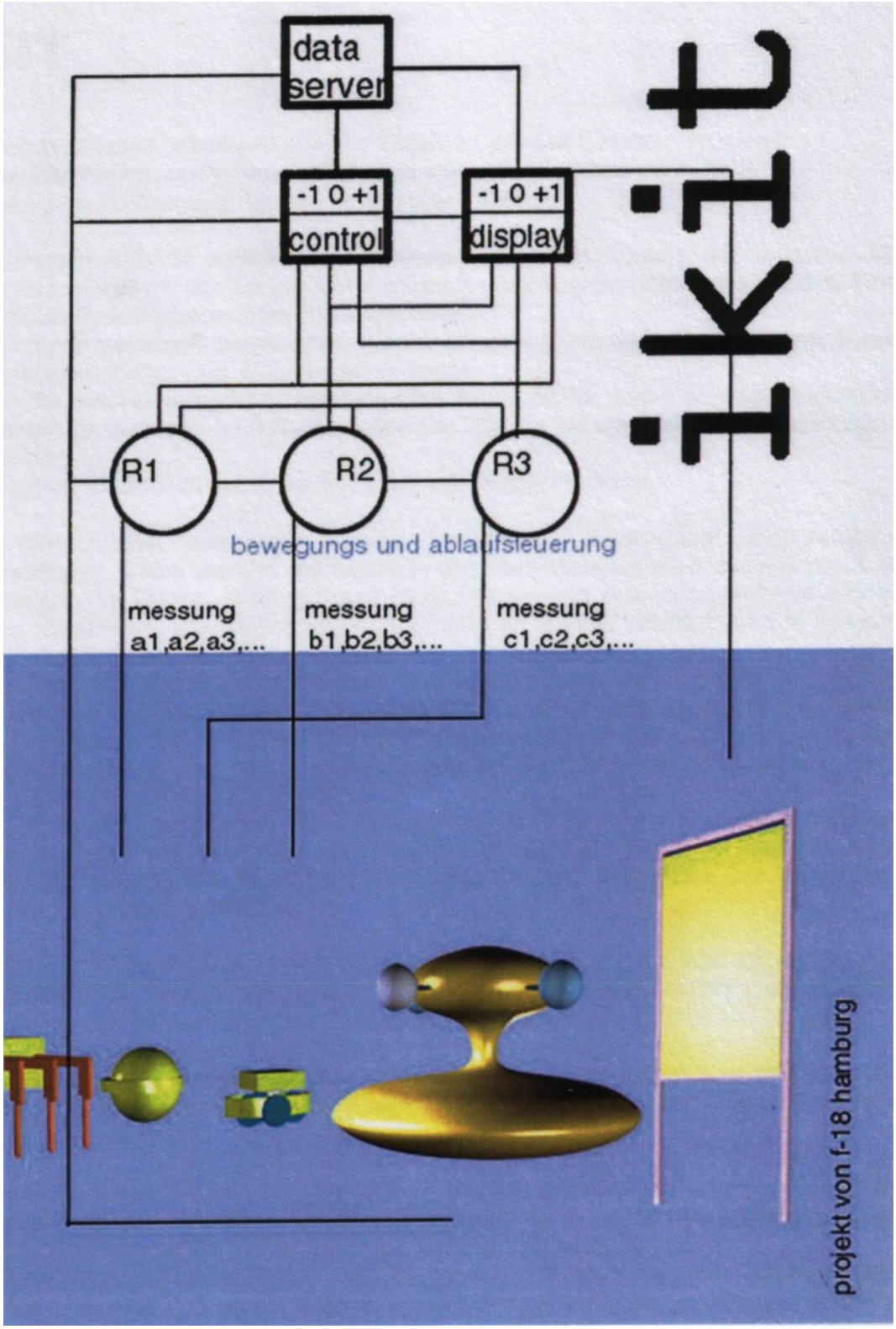
Sie sind mit Sensoren ausgestattet. Durch die Veränderung der Umgebung wird ihre Aufmerksamkeit und damit ihr Verhalten beeinflusst. Jedes der Objekte hat einen eigenen physiognomischen Charakter.

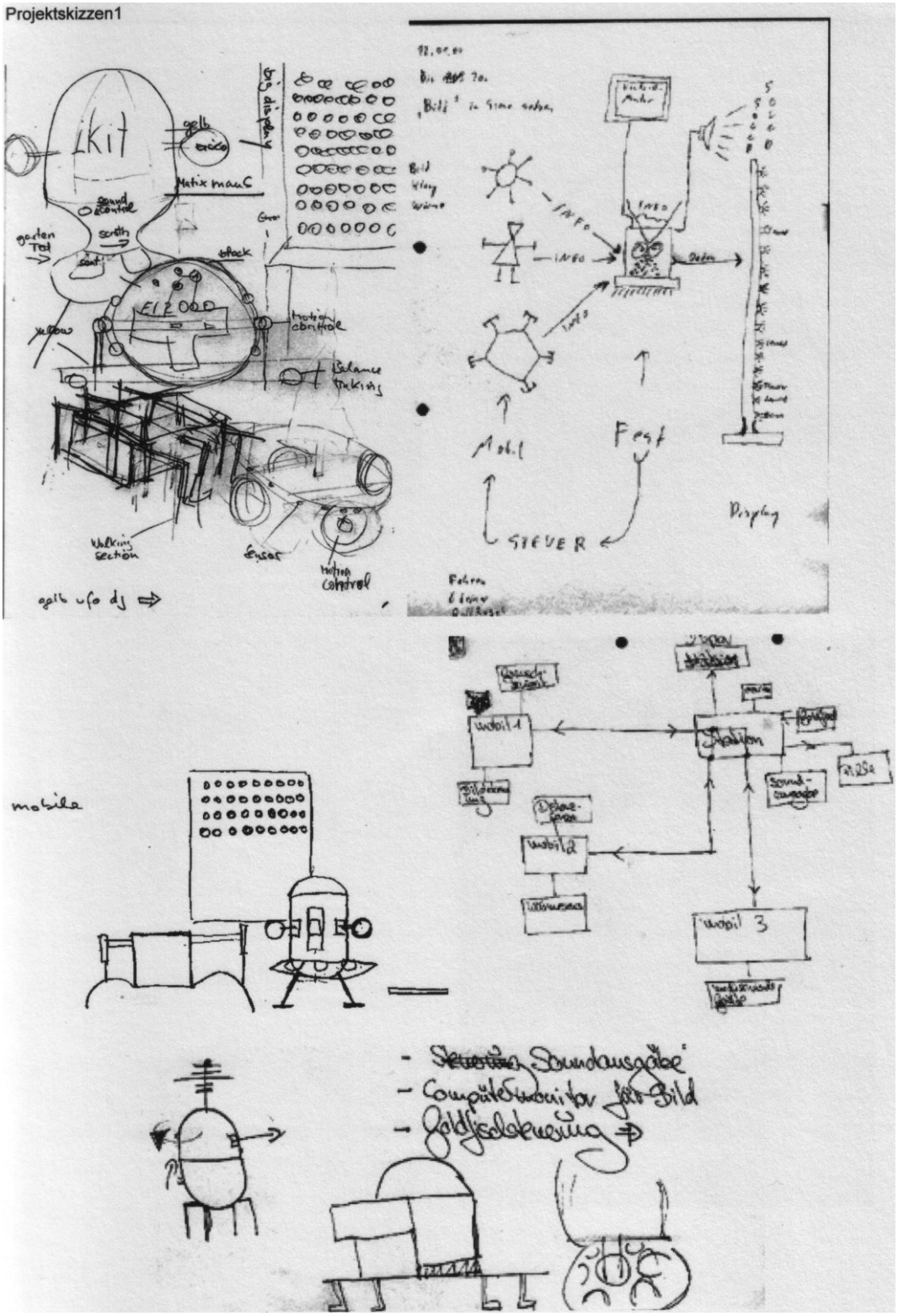
Die Serverstation wiederum betreibt Verwaltung, Auswertung und Ausgabe der Informationen. Diese von der Station empfangenen und ausgewerten Daten bestimmen sowohl das Verhalten der mobilen Roboter (z.B.Fokussierung auf Interaktionspartner), als auch das der Serviceelemente der SSE . Der Station dient ein ca. 3 x 2 Meter großer Zylinder mit Sitzgelegenheiten als Basis.

Das Display wird den Verlauf der Untersuchung darstellen. Es wird durch die Serverstation gesteuert. Die Darstellungsfläche des Displays beträgt ca. 2x4 Meter und ist mit rund 512 Glühbirnen bestückt.

Um die Robotikinstallation in ihrer Komplexität zu verwalten wird hier ein lokales Netzwerk eingesetzt. Das System wird durch die Inputs und rückgekoppelten Outputs gleichzeitig beeinflußt. Die unterschiedlichen Gewichte der Beeinflussung sind die Vorgaben des zu überprüfenden Verhaltens.

Die gesammelten Daten der Umgebung werden transformiert und setzen bestimmte Verknüpfungen zu einem neuen Bild zusammen. Dass ein grobauflösendes Display die Daten abarbeitet und sie hiermit ihrer Vernichtung zuführt ist gewollt. Die Bewegungs-, Verarbeitungs- und Ausgaberythmik steht dabei in Zusammenhang mit dem Verhalten des Publikums, welches die Funktionsweise des Systems beeinflusst.


data
server
-1 0 +1
control
-1 0 +1
display
R1
R2
R3
bewegungs und ablaufsteuerung
messung
a1,a2,a3,...
messung
b1,b2,b3,...
messung
c1,c2,c3,...
projekt von f-18 hamburg


Projektskizzen1
Mobil
Fest
SIEVER
Display
mobile

Literaturverzeichnis

[1] F18 Homepage: online. - URL http://www.f18.de/

[2] ATMEL: Microcontroller 90S8515; AVR 8-Bit RISC. online. - URL http://eu.atmel.com/

[3] MIT: 6.270 - Automous Robot Design Competition. online. - URL http://web.mit.edu/6.270/

[4] Jörg Raasch: Systementwicklung mit Structurierten Methoden. 1993

[5] C. Floyd: STEPS - Projekthandbuch. Universität Hamburg. 1992.

[6] Rodney A. Brooks: A Robust Layered Control System for a Mobile Robot, IEEE Journal of Robotics and Automation, Vol. 2, No. 1, März 1986, Seite 14 bis 23, online. URL - http://www.ai.mit.edu/people/brooks/papers/AIM-864.pdf

[7] Robolab der HAW Hamburg: online. URL - http://www.informatik.fh-hamburg.de/~robots/

[8] Arne Wischmann: Studienarbeit 2001, HaW Hamburg, online. URL - http://www.informatik.fh-hamburg.de/~lego/Projekte/Wischmann/studiarbeit.pdf

[9] Rainer Balzerowski: Diplomarbeit 2002, HaW Hamburg, online. URL - http://www.informatik.fh-hamburg.de/~lego/Projekte/Balzerowski/diplomarbeit-www.pdf

[10] Kent Beck: online. URL - http://c2.com/ppr/about/author/kent.html

[11] John Brewer: XP FAQ. online. - URL http://www.jera.com/techinfo/xpfaq.html

[12] WikiWikiWeb: Extreme Programming Roadmap. online. - URL http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap

[13] Frank Westphal: Extreme Programming. online. - URL http://www.frankwestphal.de/ExtremeProgramming.html

[14] J. Donovan Wells: A gentle introduction. online. - URL http://www.extremeprogramming.org/

[15] Isaac Asimov: Alle Robotergeschichten

[16] Ronald C. Arkin: Behavior-Based Robotics. 3. printing. Cambridge, Mass. [u.a.] : MIT Press, 2000. - ISBN 0-262-01165-4

[17] Joseph L. Jones and Anita M. Flynn: Mobile Robots - Inspiration to Implementation. Addison-Wesley, 1993. Deutsche Übersetzung: Mobile Roboter - Von der Idee zur Implementierung. 1996. - ISBN 3-89319-855-5 (Pp.)

[18] Jens Altenburg; Uwe Altenburg: Mobile Roboter - Vom einfachen Experiment zur künstlichen Intelligenz (2. Aufl.). 2000. - ISBN 3-446-21510-7

[19] Sharp: . online. - URL http://www.sharp.com

[20] Chico MacMurtrie: Amorphic Robot Works. online. - URL http://www.thetech.org/exhibits_events/online/robotics/robotart/chico.html