Projekt:	**Heizungssteuerung**
Teilprojekt:	**Steuerungssystem**
Dokumenttyp:	**Komponentenbeschreibung**
Dokumentname:	***sensuators.doc***
Verantwortlich:	**Team 2, Gruppe 2 (K. Kohler, Ch. Peper, M. Verlage)**
Benutzer:	**Team 2**
Beschreibung:	Beschreibung der Sensoren- und Aktuatorenschnittstelle

Vers#	***Datum***	***Erstellung (Beteiligung)***	***Bemerkung***
V 1	18.03.96	Christian Peper	Aus der alten Dokumentenmenge von K. Kohler, C. Peper, M. Verlage durch Einschränkung, Überarbeitung, Zusammenfassung und vieles mehr, s. Change Log

I Beschreibung der Schnittstelle *Kontrollsystem-Umwelt*

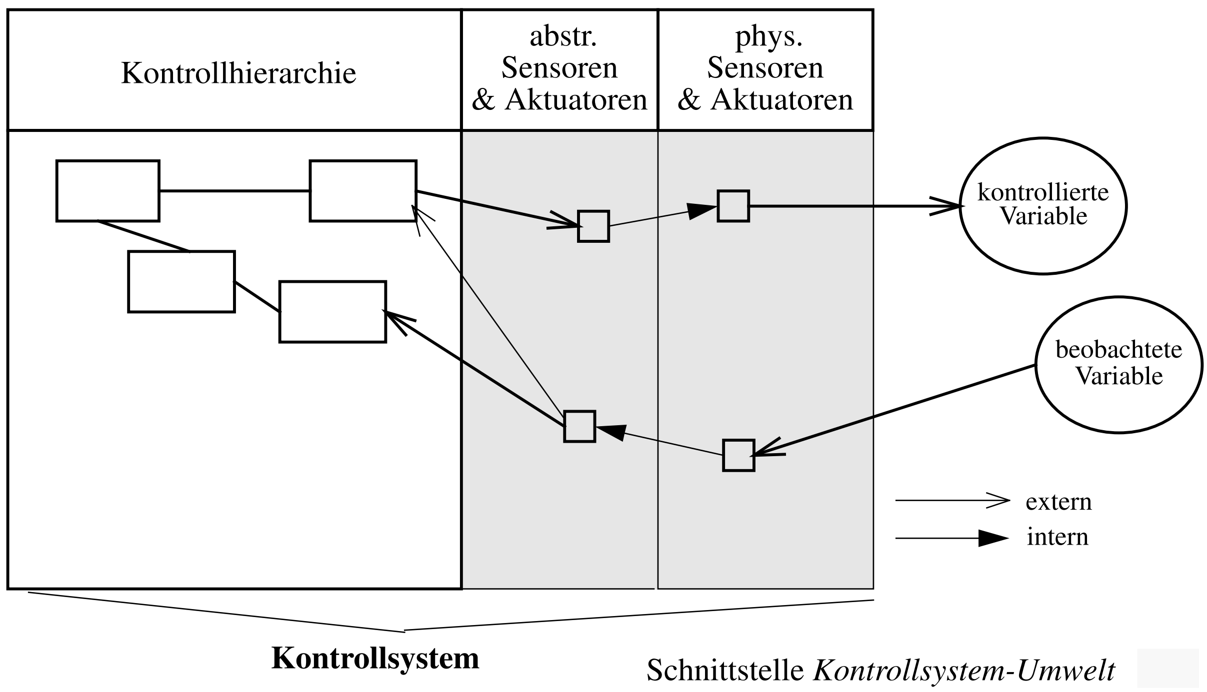
1 Einleitung

1.1 Übersicht

Das Kontrollsystem kann seine Umwelt beobachten und kontrollieren. Aufgrund vergangener oder aktueller Zustände der Umwelt (beobachtete Variablen) werden neue Soll-Werte für die kontrollierten Größen berechnet. Es wird ein Regelkreis zwischen beobachteten und kontrollierten Variablen hergestellt. Die die Regelung realisierenden Komponenten (Kontrollhierarchie) können jedoch nicht direkt die entsprechenden Größen (z.B. Jalousienöffnung) wahrnehmen oder beeinflussen. Es bedarf einer Schnittstelle zwischen Umwelt und Kontrollhierarchie. Im folgenden wird ein Überblick über die detaillierteren Beschreibungen der einzelnen Komponenten dieser Schnittstelle gegeben.

Die Schnittstelle besteht aus einem Hardware-Teil (physikalische Sensoren und Aktuatoren) und einem Software-Teil (abstrakte Sensoren und Aktuatoren). Die physikalischen Bauteile beobachten und beeinflussen die Umwelt direkt. Die abstrakten Software-Bausteine kapseln diese und bieten für die Software des Kontrollsystems eine komfortable Schnittstelle. Zwischen dem physikalischen und abstrakten Teil der Schnittstelle findet eine Kommunikation über Dienste statt.

Die nachfolgende Abbildung verdeutlicht die Beziehungen zwischen den Konzepten.



Darstellung der Beziehungen zwischen den Komponenten des Kontrollsystems

1.2 Aufgaben der Schnittstelle

Die Aufgaben dieser Schnittstelle sind:

- Der Kontrollhierarchie zuverlässige (d.h. in fest vorgegebenen Toleranzbereichen) Informationen über den Zustand der Umwelt mit einem angemessenen und vorhersagbaren Zeitverlust zu liefern.

- Zuverlässig (d.h. in fest vorgegebenen Toleranzbereichen) nach Vorgaben der Kontrollhierarchie für eine Veränderung der kontrollierten Variablen in angemessener und vorhersagbarer Zeit zu sorgen.

Dabei sollen die Eigenschaften der physikalischen Geräte, sofern möglich, verborgen werden bzw. verbessert werden (z.B. Steigerung der Zuverlässigkeit bei Meßwerten durch Mittelwertbildung).

1.3 Probleme

Die Zuverlässigkeit der Schnittstelle ist abhängig von der Zuverlässigkeit der physikalischen Bauteile. Es kann zu Situationen kommen, in denen von der Schnittstelle ein falsches Zustandsbild gegeben wird bzw. die von der Kontrollhierarchie gewünschten Aktionen nicht korrekt umgesetzt werden, ohne daß dieses von der Schnittstelle oder der Kontrollhierarchie bemerkt wird.

Zwischen dem Auftreten eines physikalischen Ereignisses oder dem Erreichen eines Zustands der Umwelt und dem Bekanntwerden in der Kontrollhierarchie kann es zu Verzögerungen kommen (Reaktionszeit + Umwandlungszeit + Übertragungszeit). Diese Verzögerung muß nicht für alle Variablen und Sensoren bzw. Aktuatoren gleich sein! Es kann zu Verzerrungen im Abbild der Umwelt kommen. Der repräsentierte Zustand der Umwelt im Kontrollsystems zu einem Zeitpunkt muß nicht notwendigerweise so in der Umwelt zu existiert haben. Im zukünftigen Anwendungsfeld der Sensoren werden dadurch aber keine Probleme verursacht.

1.4 Aufbau der Dokumentation

Die Dokumentation der Schnittstelle wurde bewußt so gestaltet, daß sie gemeinsame Informationen unterschiedlicher Varianten gruppiert. Das verfolgte Prinzip war, gemeinsame Informationen von Sensoren und Aktuatoren bzw. deren Unterklassen in eine gemeinsame Beschreibung zu packen. Dadurch soll die Wartbarkeit erhöht und generische Auslegungen der Komponenten ermöglicht werden.

Die Dokumentation der Sensoren und Aktuatoren besteht aus folgenden Teilen:

- *Schnittstelle*: (I) Gibt eine Einführung, beschreibt die Beschreibungsmittel sowie einige allgemeine Muster von Eigenschaften.
- *Umwelt-Modell*: (II) Beschreibung der Modellierung des Informationsflusses von der Umwelt in das Kontrollsystem.
- *Sensuatoren*: (III) Beschreibung der Eigenschaften, die Sensoren und Aktuatoren gemeinsam sind.
- *Sensoren*: (IV) Eine Beschreibung der Sensoren. Es wird nicht konkret auf einen bestimmten Sensortyp eingegangen, sondern ihre allg. Funktionsweise und Eigenschaften erläutert.
  - *Bereichssensoren*: (V) Es wird eine Unterklasse von Sensoren erläutert. Diese liefern einen Messwert aus einem Intervall.
  - *Binäre Sensoren*: (VI) Es wird eine Unterklasse von Sensoren erläutert, die nur binäre Werte (ein/aus, offen/zu, usw.) zurückliefern.
- *Aktuatoren*: (VII) Eine Beschreibung der allg. Funktionsweise und Eigenschaften von Aktuatoren.
- *Produkte:* (VIII) stellt den internen Aufbau von Sensoren bzw. Aktuatoren vor. Durch die Abhängigkeit von bestimmten physikalischen Sensor/Aktuator-Produkten werden hier einige Eigenschaften eingeschränkt und spezialisiert.

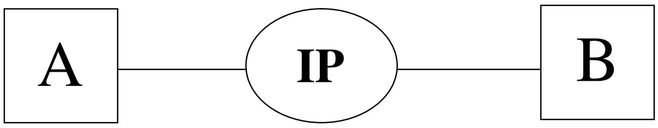
- *Instanzen:* (IX) enthält die Spezialinformationen von Sensoren/Aktuatoren, die sich aus deren Instanziierung gemäß der eingeschränkten Problembeschreibung ergeben.

2 Beschreibung von Architektur und Verhalten

Bei der Beschreibung der Sensoren und Aktuatoren werden immer architekturelle Rahmenbedingungen angegeben und anschließend die zur Verfügung gestellten Dienste mit ihren Eigenschaften beschrieben. Hier sollen erstmal die grundlegenden Begriffe eingeführt werden.

2.1 Architektur

Wir gehen davon aus, daß zwei miteinander kommunizierende Komponenten A und B über einen oder mehrere *Interaktionspunkte* (IP) miteinander verbunden sind. Es dürfen i.a. auch mehr als zwei Komponenten mit demselben IP verbunden sein.



2.2 Dienste

Die Darstellung der Dienste geht von folgenden Voraussetzungen aus:

Ein Dienst wird beschrieben durch eine Menge von möglichen Abfolgen von Interaktionen eines bestimmten Typs. Solche Interaktionstypen (=*Dienstelemente*) können formale Parameter enthalten. Typische Interaktionstypen von hier angegebenen Dienste sind z.B.:

2.2.1 Anfrage *(request)*

Die Anfrage dient der Übermittlung eines neuen Auftrags von einem Auftraggeber (Client) an einen Server.

2.2.2 Quittung *(receipt)*

Der Server kann einen solchen Auftrag durch Versenden einer Quittung an den Auftraggeber bestätigen.

2.2.3 Antwort *(response)*

Manche Aufträge erfordern das Versenden von Antworten durch den Server an den Auftraggeber.

2.2.4 Abbruch *(cancel)*

Ein momentan in Bearbeitung befindlicher Auftrag soll abgebrochen werden.

2.2.5 Fehler *(error)*

Falls vom Server durch geeignete Mechanismen (Selbstkontrolle) Fehler bei der Durchführung des Auftrags entdeckt werden, so wird der von diesem Dienstausfall betroffene Client davon unterrichtet.

2.3 Parameter

Es gibt bei der Beschreibung drei Sorten von Parametern. Zum einen gibt es Parameter, die für eine Instanz fest sind. Wir bezeichnen diese als *Instanziierungs-Parameter*. Außerdem gibt es Parameter, die bei der Initialisierung festgelegt werden. Sie können durch wiederholte Initialisierung neu festgelegt werden. Wir bezeichnen diese als *Initialisierungs-Parameter*. Und drittens gibt es die *Dienst-Parameter*, die zusammen mit der Dienst-Anfrage an die Komponente übergeben werden. (Genau genommen sind die Initialisierungs-Parameter Dienst-Parameter des init-Dienstes.)

In der tabellarischen Darstellung werden Typen und Parameter mit einem * gekennzeichnet sein, wenn sie noch nicht exakt festgelegt sind. Eine weitere Präzisierung erfolgt dann in den entsprechenden daraus abgeleiteten Beschreibungen.

3 Allgemeine Muster von Eigenschaften

Viele der später definierten Dienste, die eine Komponente einer anderen zur Verfügung stellt, werden ähnliche Muster in ihrem Verhalten aufweisen. Deswegen sollen bereits hier folgende Typen und Eigenschaften allgemein formuliert werden, so daß sie später wiederverwendet werden können.

Die folgenden Typen werden dabei als gegeben vorausgesetzt:

Typ	Typdefinition	Beschreibung
Address		Adresse einer Komponente
Nat		Natürliche Zahlen
RequestId		Referenz-ID der Anfrage aus Sicht des Servers
Identifier		Eindeutiger Bezeichner der Instanz einer Komponente
Interaction (T)		Interaktion mit einem Parameter vom Typ T
Location		Räumliche Lage der Instanz
Real		Reelle Zahlen
Time		Die Zeit
TimeInterval	Time x Time	Zeitintervall: untere Grenze: .min, obere: .max

3.1 Single-access

Ein Interaktionspunkt IP erfüllt die *single-access (delayCS, delaySC)* Eigenschaft, falls er folgendem Schema entspricht:

3.1.1 Typen

Typ	Typdefinition	Beschreibung
TimeInterval		Zeitinterval: untere Grenze: .min, obere: .max

3.1.2 Parameter

Parameter	Typ	Beschreibung
delayCS	TimeInterval	Client to Server Laufzeitschranken
delaySC	TimeInterval	Server to Client Laufzeitschranken

3.1.3 Eigenschaften

- IP ist mit max. zwei Komponenten verbunden. Dienste eines Servers S an IP können also nur von *genau einem* Client C angesprochen werden.
- IP propagiert jede Interaktion von C zu S. Dies erfolgt im Zeitintervall [delayCS.min, delayCS.max].
- IP propagiert jede Interaktion von S zu C. Dies erfolgt im Zeitintervall [delaySC.min, delaySC.max].
- IP erfindet keine weiteren Interaktionen.

3.2 Multi-access

Ein Interaktionspunkt IP erfüllt die *multi-access (delayCS, delaySC)* Eigenschaft, falls er folgendem Schema entspricht:

3.2.1 Typen

Typ	Typdefinition	Beschreibung
Address		Adresse einer Komponente
TimeInterval		Zeitintervall: untere Grenze: .min, obere: .max

3.2.2 Parameter

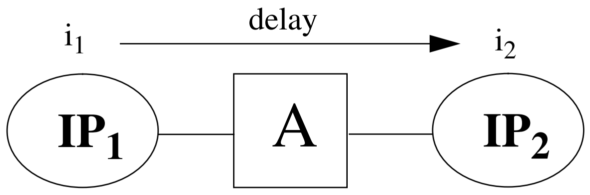
Parameter	Typ	Beschreibung
delayCS	TimeInterval	Client to Server Laufzeitschranken
delaySC	TimeInterval	Server to Client Laufzeitschranken

3.2.3 Eigenschaften

- *** p erlaeutern
- IP darf mit mehr als zwei Komponenten verbunden sein. Dienste von S an IP können also von mehreren Komponenten gleichzeitig verwendet werden.
- IP propagiert jede Interaktion i (p) von C als i (Address client = C, p) zu S. Dies erfolgt im Zeitintervall [delayCS.min, delayCS.max].
- IP propagiert jede Interaktion i (Address client = C, p) von S als i (p) zu C. Dies erfolgt im Zeitintervall [delaySC.min, delaySC.max].
- IP erfindet keine weiteren Interaktionen.

3.3 Bounded Age

A erfüllt die Eigenschaft *boundedAge* (IP_1, i_1, IP_2, i_2, *delay*), falls folgendes Schema zutrifft:



3.3.1 Typen

Typ	Typdefinition	Beschreibung
TimeInterval		Zeitintervall: untere Grenze: .min, obere: .max

3.3.2 Parameter

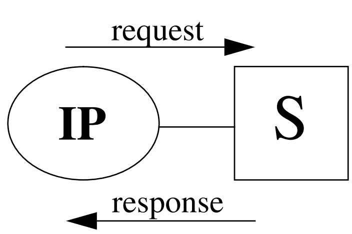
Parameter	Typ	Beschreibung
delay	TimeInterval	IP_1 to IP_2 interaction delay

Eigenschaften:

- Ist A bereit, eine Interaktion vom Typ i_2 an IP_2 durchzuführen, dann gab es zuvor eine Interaktion vom Typ i_1 an IP_1.
- Ist A bereit, eine Interaktion vom Typ i_2 an IP_2 durchzuführen, dann wurde während delay.min Zeiteinheiten zuvor keine Interaktion vom Typ i_1 an IP_1 akzeptiert.
- Hat A eine Interaktion vom Typ i_1 an IP_1 akzeptiert, dann wird höchstens delay.max Zeiteinheiten später eine Interaktion vom Typ i_2 an IP_2 geben.
- In dem Intervall zwischen dem Akzeptieren von i_1 an IP_1 und dem Akzeptieren von i_2 an IP_2 ist A niemals bereit, ein weiteres i_1 an IP_1 entgegenzunehmen.

3.4 Single-reply

Ein Dienst X wird als *addressing single-reply (delayReqRec)* Dienst bezeichnet, falls er folgendem Schema entspricht:



3.4.1 Typen

Typ	Typdefinition	Beschreibung
Address		Adresse einer Komponente
ErrorDesc		Fehlerbeschreibung
TimeInterval		Zeitintervall: untere Grenze: .min, obere: .max

3.4.2 Parameter

Parameter	Typ	Beschreibung
delayReqResp	TimeInterval	request to receipt delay

Interaktionstypen:

Name	Parameter	Beschreibung
• X.request	(Address adr)	Auftrag
• X.response	(Address adr)	Antwort
• X.error	(Address adr, ErrorDesc)	Fehlermeldung

Eigenschaften:

- Entweder es folgt auf eine Anfrage X.request (Address adr = a) genau eine Antwort X.response (Address adr = a) innerhalb des Zeitintervalls delayReqResp oder

  es wird X.error (Address adr = a) anstelle des X.response bei Mißerfolg in derselben Zeit zurückgegeben.
- Zwischen einem X.request() und dem nachfolgenden X.response() bzw. X.error werden keine weiteren X.request() angenommen.

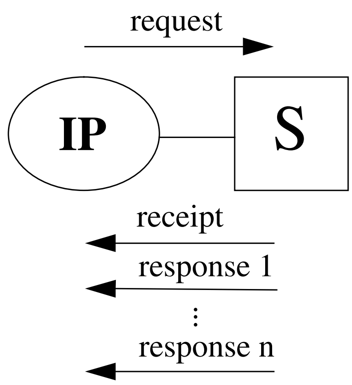
Der "cancel"-Interaktionstyp macht bei single-reply Diensten keinen Sinn, da die Anfrage mit der Antwort terminiert. Es werden weder Quittung noch RequestID zurückgegeben.

Ohne Berücksichtigung der Zeit sei *single-reply* $=_{\mathrm{Def}}$ *single-reply* $([0, \infty])$.

Wird in obigem Schema der *Address*-Parameter weggelassen, dann heißt die Eigenschaft auch einfach *single-reply.*

3.5 Multi-reply

Ein Dienst X erfüllt die *addressing multi-reply (n, delayReqRec, delayRespResp)* Eigenschaft, falls er folgendem Schema entspricht:.



3.5.1 Typen

Typ	Typdefinition	Beschreibung
Address		Adresse einer Komponente
ErrorDesc		Fehlerbeschreibung
RequestId		Auftrags-Identifikator
TimeInterval		Zeitintervall: untere Grenze: .min, obere: .max

3.5.2 Parameter

Parameter	Typ	Beschreibung
delayReqRec	TimeInterval	request to receipt delay
delayRespResp	TimeInterval	response to response delay
n	Nat	Zahl der Antworten

Interaktionstypen::

Name	**Parameter**	**Beschreibung**
• X.request	(Address adr)	Auftrag
• X.receipt	(Address adr, RequestId rid)	Quittung
• X.response	(Addresss adr, RequestId rid)	Antwort
• X.cancel	(Address adr, RequestId rid)	Abbruch
• X.error	(Address adr, RequestId rid, ErrorDesc)	Fehler

Eigenschaften:

- Entweder folgt auf eine Anfrage X.request (Address adr = a) genau ein X.receipt (Address adr = a, RequestId rid = r) innerhalb des Zeitintervalls delayReqRec oder

  es wird X.error (Address adr = a, RequestId rid = r) anstelle des X.receipt bei Mißerfolg in derselben Zeit zurückgegeben.
- In dem Intervall zwischen einem X.receipt (Address adr = a, RequestId rid = r) und einem X.cancel (Address adr = a, RequestId rid = r) wird niemals ein X.receipt (Address adr = a, RequestId rid = r) zurückgegeben, d.h. die Kombination aus Adresse und RequestId ist eindeutig.
- Auf ein X.receipt (Address adr = a, RequestId rid = r) folgt ein X.response (Address adr = a, RequestId rid = r) innerhalb des Zeitintervalls delayRespResp.
- Auf ein X.response (Address adr = a, RequestId rid = r) folgt ein weiteres X.response (Address adr = a, RequestId rid = r) innerhalb des Zeitintervalls delayRespResp.
- Es werden höchstens n Antworten X.response (Address adr = a, RequestId rid = r) verschickt.

  In dem Intervall zwischen einem X.cancel (Address adr = a, RequestId rid = r) und einem darauffolgendem X.receipt (Address adr = a, RequestId rid = r) wird niemals ein X.response (Address adr = a, RequestId rid = r) oder ein X.error (Address adr = a, RequestId rid = r) zurückgegeben.

Wird in obigem Schema der *Address*-Parameter weggelassen, dann heißt die Eigenschaft auch einfach *multi-reply.*

3.5.3 Acyclic-multi-reply

Im Normalfall können die Antworten des o.g. multi-reply Dienstes in variierenden Zeitabständen erfolgen. Der Dienst wird daher auch als *azyklisch* bezeichnet.

- *acyclic-multi-reply(n, delayReqRec) multi-reply(n, delayRec, [0. ∞])*

3.5.4 Cyclic-multi-reply

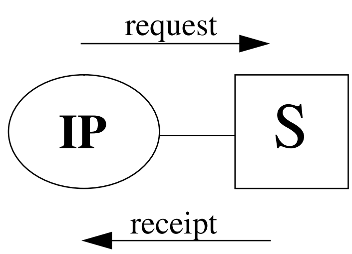
Sind dagegen die zeitlichen Abstände zwischen den Antworten äquidistant (mit einer max. zulässigen Schwankungsbreite), dann wird der Multi-reply Dienst als *zyklisch* bezeichnet.

- *cyclic-multi-reply* $=_{\mathrm{Def}}$ *multi-reply*

3.6 Receipt-reply

Bei einigen Diensten (Bsp.: Sensormanagement) müssen keine Daten rückübermittelt werden. Dann können die response-Dienstelemente entfallen.

Ein Dienst X erfüllt die *addressing receipt-reply (delayReqRec)* Eigenschaft, falls er folgendem Schema entspricht:



3.6.1 Typen

Typ	Typdefinition	Beschreibung
Address		Adresse einer Komponente
ErrorDesc		Fehlerbeschreibung
TimeInterval		Zeitintervall: untere Grenze: .min, obere: .max

3.6.2 Parameter

Parameter	Typ	Beschreibung
delayReqRec	TimeInterval	request to receipt delay

Interaktionstypen::

Name	Parameter	Beschreibung
• X.request	(Address adr)	Auftrag
• X.receipt	(Address adr)	Quittung
• X.error	(Address adr, ErrorDesc)	Fehler

Eigenschaften:

- Eine Anfrage X.request (Address adr = a) wird mit genau einem X.receipt (Address adr =a) quittiert. Dies erfolgt im Zeitintervall delayReqRec.

Ohne Berücksichtigung der Zeit sei *receipt-reply* $=_{\mathrm{Def}}$ *receipt-reply ([0,* ∞ *]).*

Wird in obigem Schema der *Address*-Parameter weggelassen, dann heißt die Eigenschaft auch einfach *multi-reply.*

4 Ableitbare Eigenschaften

Das komplette Teilsystem bestehend aus IP und S (s.o.) bietet ebenfalls einen single-reply (t) - Dienst X, falls X single-reply (t_S) Dienst von S ist, IP ein single-access ($t_{IP/S}$, $t_{S/IP}$)- Interaktionspunkt ist und t.max >= t_S.max + t_{IP}.max und t.min <= t_S.min + t_{IP}.min.

Entsprechendes gilt für die folgenden Eigenschaften und für Multi-access Interaktionspunkte.

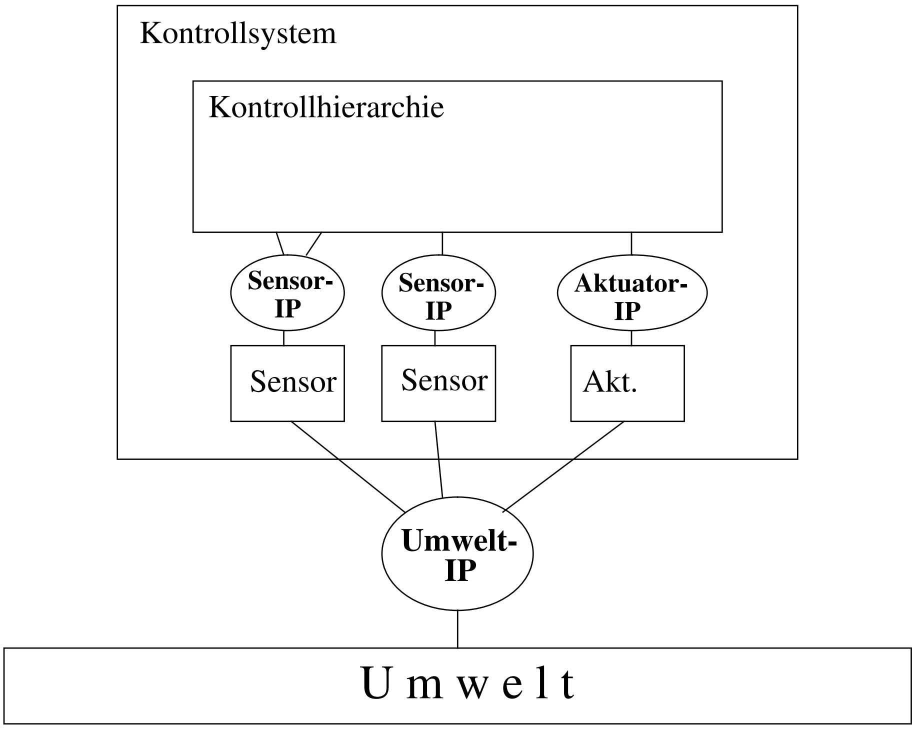
II Modellierung der Umwelt

1 Einleitung

Eine vollständige Modellierung der Physik der Umwelt des Kontrollsystems wird hier natürlich nicht angeboten. Wichtig ist es aber, präzisere Rahmenbedingungen für die Beschreibung des *Informationsflusses* zwischen Umwelt und Kontrollsystem zu schaffen. Eine weitere Motivation lag in der Forderung, die Umwelt möglichst leicht durch einen Simulator ersetzen zu können.

Zur Vereinfachung der Darstellung wird dazu dieselbe Methode wie zur Beschreibung innerhalb des Kontrollsystems verwendet. Wir erwarten also, daß die "Komponente" Umwelt am Umwelt-IP ununterbrochen aktuelle Informationen über ihren Zustand anbietet, etwa in Form eines Dienstes *transfer* mit Interaktionstyp *transfer.temperature (T)* mit $T \in \mathcal{R}eal$. Welche konkreten Informationen bzw. Dienste ein Sensor am Umwelt-IP erwartet, wird jeweils bei der Beschreibung des Sensors angegeben. Für Aktuatoren wird analog verfahren.

Unter Verwendung der Architekturbeschreibung kann die Abbildung vom Anfang etwas abstrakter so zusammengefaßt werden:



2 Typen

Typ	Typdefinition	Beschreibung
Interaction (T)		Interaktion mit einem Parameter vom Typ T
PhysValue*		Typ der physikalischen Größe

3 Parameter

Parameter	Typ	Beschreibung
transfer.quantity*	Interaction (PhysValue)	Platzhalter für die Interaktion der Umwelt am Umwelt-IP, die die phys. Größe anbietet

4 Dienste

Eigenschaften:

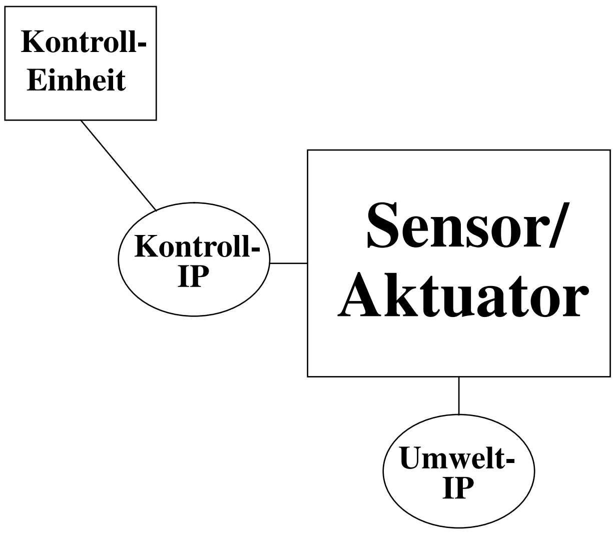
- Jede Interaktion transfer.quantity(x) der Umwelt am Umwelt-IP wird ständig erfolgreich durchgeführt.

III Gemeinsamkeiten von Sensoren & Aktuatoren

1 Architektur

Zur Kontrolle ihres Betriebszustands sind sowohl Sensoren als auch Aktuatoren über einen speziellen Interaktionspunkt mit jeweils genau einer Kontrolleinheit aus ihrer Umgebung verbunden. Eine *Kontrolleinheit* ist genau eine der folgenden Komponenten des Kontrollsystems: *Kontrollzelle*, *Kontrollgruppe* oder *Kontrollregion*.

- Der *Kontroll-Interaktionspunkt (Kontroll-IP)* dient der Übergabe von Befehlen zur Sensor-/Aktuator-Steuerung. Der Kontroll-IP ist ein *singel-access*-IP, d.h. es gibt für den Sensor bzw. Aktuator genau eine verantwortliche Kontrolleinheit. Dadurch kann es bei den am Kontroll-IP angebotenen Management-Diensten nicht zu Konflikten kommen. Die Identität der Kontrolleinheit kümmert den Sensor/Aktuator überhaupt nicht.



2 Typen

Typ	Typdefinition	Beschreibung
ErrorDesc *	Aufzählung	Art des Fehlers bei Fehlverhalten
TableSize	positive Integerzahl	Größe der Tabellen für die Aufträge der Kontrolleinheiten. Entspricht der max. Anzahl der bearbeitbaren zyklischen Aufträge oder Event-Aufträge.

3 Instanziierungs-Parameter

Parameter	Typ	Beschreibung
identifier*	Identifier	Eindeutiger Bezeichner der Instanz der Komponente
location*	Location	Räumliche Lage der Instanz der Komponente

4 Dienste

Hier werden die allgemeinen Dienste beschrieben, die jeder Sensor/Aktuator den Benutzern (Kontrolleinheiten) mindestens zur Verfügung stellt. Es wurde dabei keine Festlegung bzgl. der Implementierung von Interaktionen vorgenommen.

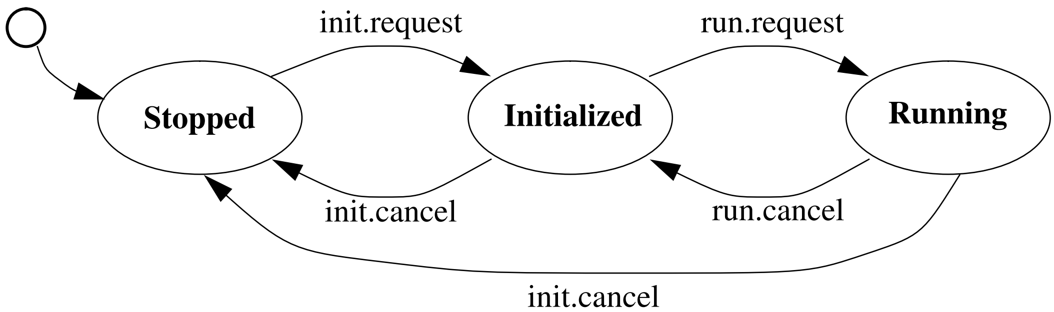
4.1 Verfügbarkeit von Diensten

Sowohl Sensor als auch Aktuator werden an weiteren Interaktionspunkten noch weitere Dienste anbieten. Diese Dienste sind allerdings nicht immer verfügbar.

Zur Beschreibung der Verfügbarkeit dieser Dienste lassen sich folgende Prädikate definieren und als Zustände interpretieren:

- **Stopped**: Der Sensor/Aktuator akzeptiert am Kontroll-IP nur den init.request, an anderen Interaktionspunkten reagiert er gar nicht. Dieser Zustand ist der Startzustand bei Inbetriebnahme.
- **Initialized**: Der Sensor/Aktuator ist neu initialisiert, er akzeptiert ausschließlich init.cancel bzw. run.request am Kontroll-IP. Alle anderen Anfragen beantwortet er mit einem .error ("Not Running").
- **Running**: Dieser Zustand ist der Arbeitsmodus des Sensors. In diesem Zustand werden alle später beschriebenen Dienst-Aufträge außer init.request und run.request entgegengenommen.

Daraus ergibt sich folgendes Zustandsübergangsdiagramm:



4.2 Dienste am Kontroll-Interaktionspunkt (Management)

Die Management-Dienste können nicht von allen Kontrolleinheiten verwendet werden. Die Steuerung bleibt der Kontrolleinheit am *single-access* Kontroll-IP vorbehalten.

4.2.1 Initialisierung und Rücksetzen

Dienstname: **init**

Interaktionstypen:

Name	Parameter	Beschreibung
• init.request	(TableSize)	Die TableSize bestimmt die max. Anzahl Aufträge, die der Sensor/Aktuator gleichzeitig bearbeiten kann. Die Parametereinstellung wirkt sensor-weit.

• init.receipt	()	Die Initialisierung wurde erfolgreich durchgeführt.
• init.error	(ErrorDesc)	Fehler
• init.cancel	()	Die momentanen Einstellungen werden gelöscht. Alle Kontrolleinheiten, die gerade einen Dienst D verwenden werden von dem Dienstausfall mittels D.error("Init.cancel") informiert.

Eigenschaften:

- receipt-reply

4.2.2 Starten und Stoppen

Dienstname: **run**

Interaktionstypen:

Name	**Parameter**	**Beschreibung**
• run.request	()	Aktivieren des Sensors/Aktuators. Zulassen von Anfragen der Kontrolleinheiten.
• run.receipt	()	Die Komponente wurde erfolgreich gestartet.
• un.cancel	()	Deaktivieren. Alle Kontrolleinheiten, die gerade einen Dienst D verwenden, werden von dem Dienstausfall mittels D.error("Run.cancel") informiert.
• run.error	(ErrorDesc)	Fehler

Eigenschaften:

- receipt-reply

IV Sensoren

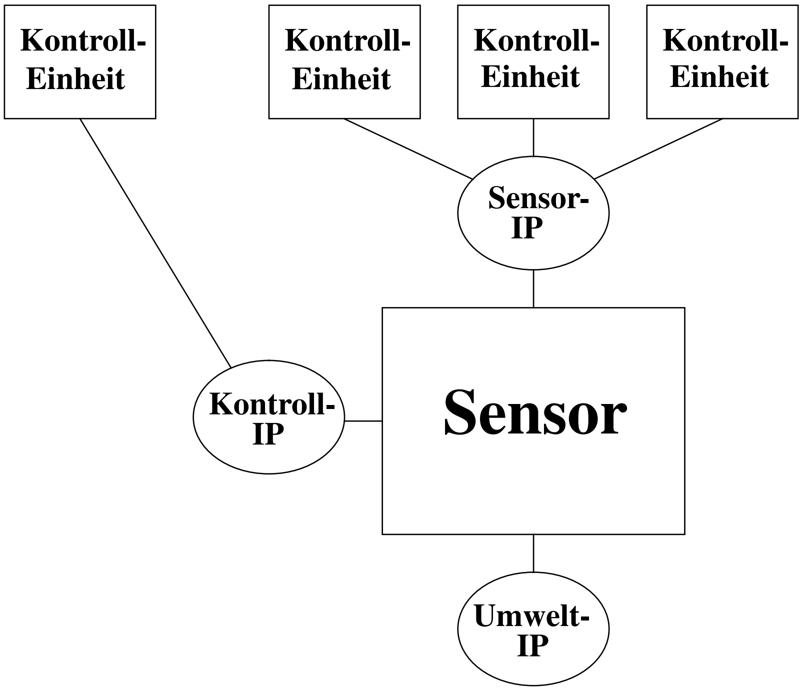
1 Einleitung

Die *Sensoren* sind aufgeteilt in zwei Unterklassen *Bereichssensor* und *Binärer Sensor*, die sich im Typ ihrer Rückgabewerte unterscheiden. Näheres siehe in den zugehörigen Dokument-Teilen rangeSensor.doc, bzw. binarySensor.doc.
Hier erfolgt eine Beschreibung derjenigen Eigenschaften, die allen Sensoren gemeinsam sind.

2 Architektur

Ein Sensor kommuniziert über zwei Interaktionspunkte mit Kontrolleinheiten aus seiner Umgebung.

- Über den Sensor-*Interaktionspunkt (Sensor-IP)* kann auf die Meßwerte des Sensors zugegriffen werden,
- der *Kontroll-Interaktionspunkt (Kontroll-IP)* dient der Übergabe von Befehlen zur Sensor-Steuerung (Beschreibung s.v.) .



3 Typen

Typ	Typdefinition	Beschreibung
AnswerModus	{immer nach Eintreten des Events, n-mal}	Angabe zur Antwortgenerierung bei Eintreten des Events
ErrorDesc*	Aufzählung	Art des Fehlers bei Fehlverhalten
EventModus*		Ursache für das Auslösen eines Events

Typ	Typdefinition	Beschreibung
Address	Adressen aller Kontrollein-heiten	Adresse der Absender-Kontroll-einheit
EventParameter*		Parameter zur Eventgenerierung, in Abhängigkeit vom EventModus
Number	[-1, 0, MaxInteger] -1 bedeutet ∞	Anzahl der beantragten Anworten (Vektoren oder Meßwerte) für zyklische Anfragen, Eventgenerierung und Statistiken.
Umwelt:: PhysValue*		Typ der physikalischen Größe
RequestID	Integer	ID der Anfrage; vom Sensor vergeben, um multi-reply Anfragen (s.h.) zu gruppieren
TableSize	positive Integerzahl	Größe der Tabellen für die Aufträge der Kontrolleinheiten. Entspricht der max. Anzahl der bearbeitbaren zyklischen Aufträge oder Event-Aufträge
Unit*		Einheit des Meßwertes
Value*	[valueMin, valueMin + valueStep, ... , valueMax]	Typ des Meßwerts, der vom Sensor ermittelt wird

4 Parameter

Parameter	Typ	Beschreibung
eventReqRecDelay*	Time	Verzögerung zwischen Anfrage und Quittung
Umwelt:: transfer.quantity*	Interaction (PhysValue)	Platzhalter für die Interaktion der Umwelt am Umwelt-IP, die die phys. Größe anbietet
valueAge*	Time	Altersintervall eines Sensorwerts
valueMin*	Real	Kleinstmöglicher Rückgabewert
valueMax*	Real	Größtmöglicher Rückgabewert
valueReqRespDelay*	Time	Verzögerung zwischen Anfrage und Antwort
valueConv*	PhysValue -> Value	Konvertierungsfunktion (Meßgenauigkeit)
valueStep*	Real	Auflösung

5 Dienste

5.1 Dienste am Sensor-Interaktionspunkt (Meßwert-Erfassung)

Es ist zu beachten, daß sämtliche Dienste am Sensor-IP erst nach erfolgreicher Initialisierung und Starten des Sensors am Kontroll-IP zur Verfügung stehen (s. Dienste *init* und *run* im Abschnitt *Sensor-Management* und Abschnitt *Verfügbarkeit*).

5.1.1 Polling

Dienstname: **valueReq**

*Interaktionstypen:*I

Name	**Parameter**	**Beschreibung**
• valueReq.request	(Address)	Auftrag
• valueReq.response	(Address, Value v)	Es wird der momentane Meßwert (Value) an die anfragende Kontrolleinheit Address) zurückgegeben.
• valueReq.error	(Address, ErrorDesc)	Fehler

Eigenschaften:

- addressing single-reply (valueReqRespDelay)
- boundedAge (Umwelt-IP, transfer.quantity(x), Sensor-IP, valueReq.response(Value v= x'), valueAge) mit x' = valueConv (x)

5.1.2 Eventgenerierung

Dienstname: **eventReq**

Beschreibung: An die anfragende Kontrolleinheit wird eine Antwort geschickt, falls der im EventModus eingestellte Fall eintritt. Der Rückgabewert ist der erste Wert unmittelbar nach Eventauslösung.

Interaktionstypen:

Name	**Parameter**
• eventReq.request	(Address, Number n, EventModus, EventPar, AnswerModus)
• eventReq.receipt	(Address, RequestId)
• eventReq.response	(Address, RequestId, Value)
• eventReq.cancel	(Address, RequestId)
• eventReq.error	(Address, RequestId, ErrorDesc)

Eigenschaften:

- addressing acyclic-multi-reply (n, eventReqRecDelay)
- boundedAge (Umwelt-IP, transfer.quantity(x), Sensor-IP, eventReq.response (Value v=x'), valueAge) mit x' = valueConv (x)

5.1.3 Abfrage von Parametern

Dienstname: **getParameterValues**

Beschreibung: Sensor-weit eindeutige Parametereinstellungen können abgefragt werden. Bestimmte Eigenschaften des Sensors (hier: ResponsibleUnit) sind nicht fest vorgegeben, sondern können verändert werden. Die Werte der Parameter sollen während einer Initialisierung eingestellt werden, und für den nachfolgenden normalen Betrieb des Sensors konstant bleiben. Die Funktion *getParameterValues* gibt die Werte der Parameter aus. Die Funktion wird überflüssig, falls die Werte der Parameter bereits von einer übergeordneten Instanz den Kontrolleinheiten bekanntgegeben werden

Interaktionstypen:

Name	**Parameter**	**Beschreibung**
• getParameterValues.request	(Address)	Auftrag
• getParameterValues.response	(Address, TableSize)	Antwort

Eigenschaften:

- addressing single-reply

V Bereichssensoren (range sensors)

1 Einleitung

Eine Reihe von Sensoren erfaßt Meßwerte innerhalb eines gegebenen Werte*bereichs*. Wir werden in den folgenden Abschnitten die gemeinsamen Dienste dieser Sensorklasse beschreiben. Information zu einzelnen Sensoren kann den Datenbättern entnommen werden, die für jeden Sensortyp existieren.

2 Architektur

Siehe *Sensor.*

3 Typen

Typ	Typdefinition	Beschreibung
EventModus	- bei Änderung des Values - bei Verlassen eines Intervalls (relativ) - bei Verlassen des Intervalls (absolut)	Ursache für das Auslösen eines Events
EventParameter *		Parameter zur Eventgenerierung, in Abhängigkeit vom EventModus
	% des Wertebereiches des Values	- bei Änderung des PhysicalValue: Angabe der Granularität
	Menge von Tupeln der Form: (lowerLimit, upperLimit)	- bei Verlassen eines Intervalls (relativ/absolut) *
Unit*		Einheit des Meßwerts
Value*	[valueMin, valueMin + valueStep, ... , valueMax]	Typ des Meßwerts, der vom Sensor ermittelt wird

4 Parameter

Parameter	Typ	Beschreibung
cyclicReqLower*	Time	max. Abweichung vom Sendezyklus nach unten
cyclicReqRecDelay	Time	Verzögerung zwischen Anfrage und Quittung
cyclicReqUpper*	Time	max. Abweichung vom Sendezyklus nach oben

5 Dienste

5.1 Dienste am Sensor-Interaktionspunkt (Meßwert-Erfassung)

Neben den Standard-Diensten für Polling und Events ist bei Bereichssensoren das zyklische Senden von Meßdaten sinnvoll:

5.1.1 Zyklisches Senden

Dienstname: **cyclicReq**

Interaktionstypen:

Name	**Parameter**	**Beschreibung**
• cyclicReq.request	(Address, Number n, Time interval)	Der interval-Parameter gibt an, in welchem Zeitabstand die angeforderten Werte versendet werden sollen. Die Quittung und die erste Anwort werden innerhalb einer IntervalTime nach Erhalt des Auftrages zurückgeliefert. Die Number gibt die Zahl der zu versendenden Meßwerte an.
• cyclicReq.receipt	(Address, RequestId)	Die Quittung teilt der Kontrollzelle die RequestId mit, die die folgenden Nachrichten ebenfalls kennzeichnet (für den Fall, daß von einer Kontrollzelle mehrere Anfragen aktiv sind)
• cyclicReq.response	(Address, RequestId, Value v)	An die anfragende Kontrollzelle wird zyklisch der jeweils aktuellen Meßwert (Value) geschickt.
• cyclicReq.cancel	(Address, RequestId)	Die RequestId kann für spätere Stornierungen (Funktion cancelReq) benutzt werden.
• cyclicReq.error	(Address, RequestId, ErrorDesc)	Fehler

Eigenschaften:

- addressing cyclic-multi-reply (n, cyclicReqRecDelay, [interval-cyclicReqLower, interval+cyclicRequestUpper])
- boundedAge (Umwelt-IP, transfer.quantity(x), Sensor-IP, cyclicReq.response(Value v=x’), valueAge) mit x’ = valueConv (x)

VI Binäre Sensoren (binary sensors)

1 Einleitung

Die Sensor-Unterklasse der binären Sensoren liefert nur binäre Meßwerte zurück. Aus diesem Grund ist die Funktionalität gegenüber den Bereichs-Sensoren etwas eingeschränkt.
Jeder binäre Sensor ist insbesondere auch Sensor, so daß er alle allgemeinen Sensor-Eigenschaften erbt.

2 Architektur

Siehe *Sensor/*

3 Typen

Typen	Typdefinition	Beschreibung
EventModus	{"0->1","1->0", "anyChange"}	Ursache für das Auslösen eines Events
EventParameter	∅	Parameter zur Eventgenerierung, in Abhängigkeit vom EventModus
Value	{0,1}	Typ des Meßwerts, der vom Sensor ermittelt wird
ErrorDesc*	Aufzählung	Art des Fehlers bei Fehlverhalten
Unit	Bool	Einheit des Meßwerts

4 Parameter

5 Dienste

Es kommen keine neuen Dienste hinzu.

VII Aktuatoren

1 Einleitung

Aktuatoren sind diejenigen Objekte, mit denen das Kontrollsystem Einfluß auf seine Umgebung nimmt. Das Kontrollsystem möchte durch entsprechende Befehle bzw. Einstellungen bestimmte Zustände der realen Welt erzielen. Soll-Werte werden vorgegeben und die Aufgabe des Aktuators ist, die kontrollierte (beeinflußte) Größe dementsprechend zu stellen (Ist-Wert). Dies kann wiederum indirekte anderer physikalische Größen beeinflussen.

Es werden nur Aktuatoren angeboten, die eine Größe direkt kontrollieren (stellen). Regelung wird nicht vorgenommen. Es existiert kein Aktuator, der bestimmte Regelverfahren einschließt (z.B. Aktuator *regleZimmerTemperatur*). Jeder Aktuator ist gleichzeitig ein Sensor für die entsprechende gestellte Größe. Schon daraus folgt die Aufteilung der Aktuatoren in *Binäre Aktuatoren* und *Bereichsaktuatoren*.

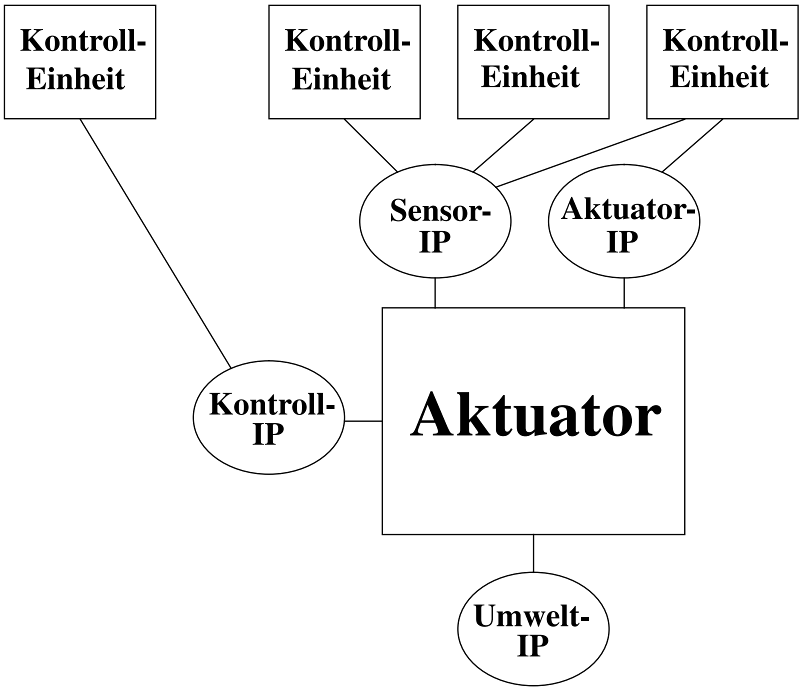
Es ist zu beachten, daß der Ist-Wert meist nicht direkt vom Aktuator gemessen wird. Da der Aktuator aber auch als Sensor funktionieren soll, muß er eine Repräsentation des Ist-Werts speichern. Wegen der möglichen Ungenauigkeit, wird dieser Wert *angenommener Ist-Wert (AssumedValue)* genannt. Zum Beispiel mißt der blin-rotator nicht den Öffnungswinkel der Lamellen, sondern wird diesen Wert über die Zeitspannen der Aktivierung der 3-Punkt-Steuerung ermitteln. Eine mögliche Abweichung zwischen angenommenem Ist-Wert und tatsächlichem Ist-Wert sind die Folge. Dieser Toleranzbereich sollte für jeden einzelnen Aktuatortyp vorgegeben werden können.

2 Architektur

Jeder Aktuator verhält sich grundsätzlich wie ein Sensor für die ... Größe für Im Normalbetrieb ist ein Aktuator über drei Interaktionspunkte mit Kontrolleinheiten aus seiner Umgebung verbunden. Eine *Kontrolleinheit* ist genau eine der folgenden Komponenten des Kontrollsystems: *Kontrollzelle*, *Kontrollgruppe* oder *Kontrollregion*.

- Über den Sensor-*Interaktionspunkt (Sensor-IP)* kann auf die Meßwerte des Sensorteils zugegriffen werden,
- der *Kontroll-Interaktionspunkt (Kontroll-IP)* dient der Übergabe von Befehlen zur Sensor-Steuerung..
- Über den zusätzlichen *Aktuator-Interaktionspunkt (Aktuator-IP)* bietet er Dienste zum Einstellen der Größe für genau eine Kontrolleinheit an

.



3 Typen

Parameter	Typdefinition	Beschreibung
ErrorDesc*	Aufzählung	Art des Fehlers bei Fehlverhalten
Identifier*		Eindeutiger Bezeichner der Instanz des abstrakten Aktuators
Location*		räumliche Lage des physikalischen Aktuators
TargetValue*	[valueMin, valueMin + targetValueStep, ... , valueMax]	einzustellender Soll-Wert
Unit*		Einheit der kontrollierten Größe
Value*	[valueMin, valueMin + valueStep, ... , valueMax]	Durch den Sensor-Teil ermittelter Ist-Wert

4 Parameter

Parameter	Typ	Beschreibung
setReqRecDelay*	TimeInterval	Verzögerung zwischen Anfrage und Quittung

Parameter	Typ	Beschreibung
setReqRecRespDelay*	TimeInterval,	Zeitintervall zwischen Quittung bis zur Rückmeldung einer erfolgreichen Stell-Aktion oder neuem Auftrag
valueMin*	Real	Kleinstmöglicher Rückgabewert
valueMax*	Real	Größtmöglicher Rückgabewert
targetValueAge	TimeInterval	Zeitintervall zwischen einem request und einer erfolgreichen Stell-Aktion
targetValueConv	TargetValue -> PhysValue	Konvertierungsfunktion (Stellgenauigkeit)
targetValueStep*	Real, > valueStep	Auflösung bei der Einstellung

5 Dienste

5.1 Dienste am Aktuator-Interaktionspunkt (Stellwert-Vorgabe)

Es ist zu beachten, daß sämtliche Dienste am Aktuator-IP erst nach erfolgreicher Initialisierung und Starten des Sensors am Kontroll-IP zur Verfügung stehen (s. Dienste *init* und *run* im Abschnitt *Sensor-Management* und Abschnitt *Verfügbarkeit*).

5.1.1 Einstellen eines Wertes

Es wird davon ausgegangen, daß der Aktuator Informationen über sein Stell-Zustand mit der selben Genauigkeit und demselben Wertebereich wie beim Setzen der Stellgroesse zur Verfuegung stellt

Dienstname: **setReq**

Interaktionstypen:

Name	Parameter	Beschreibung
• setReq.request	(TargetValue v)	Stellt den Soll-Wert ein. Ein Auftrag wird als *laufend* bezeichnet, falls ein setReq.request() eingegangen ist, aber weder ein ..response() oder .error() zurückgeschickt noch ein .cancel() eingetroffen sind. Geht während der Bearbeitung (d.h. nach Erhalt eines setReq.request() und vor der Rückgabe eines setReq.response()) ein weiterer setReq.request() ein, so hat dieser Vorrang und der alte wird verworfen.
• setReq.receipt	()	Bestätigt den Auftrags-Eingang.
• setReq.response	(TragetValue v)	Bestätigt die erfolgreiche Durchführung des letzten laufenden Auftrags setReq.request (v), d.h. unmittelbar danach gilt: Sollwert = Istwert (bis auf Rundung) Es ist zu beachten, daß die Einstellung des Wertes einige Zeit, je nach Aktuatortyp sogar bis zu Minuten, in Anspruch nehmen kann. !!! Sollte eine Maximalzeit bzw. die Anzahl der Versuche vorgegeben bzw. bei der Instanziierung eingestellt werden können ??

Name	**Parameter**	**Beschreibung**
• setReq.error	(ErrorDesc)	Fehler

Eigenschaften:

- receipt-reply (setReqRecDelay)
- boundedAge (Aktuator-IP, setReq.receipt (TargetValue v = x),
  Aktuator-IP, setReq.response (TargetValue v = x) <u>exor</u> setReq.request (TargetValue v = x'), setReqRecRespDelay)
- boundedAge (Aktuator-IP, setReq.response (TargetValue v = x'), Umwelt-IP, transfer.quantity (x), targetValueAge) mit x' = targetValueConv (x)

Change Log

Bei der Überarbeitung des alten Dokumenten-Satzes wurden einige Dokumente gestrichen, da sie nur noch wenige Seiten enthielten, und in sensuators.doc zusammengefaßt. Die in diesen Dokumenten vorher durchgeführten Änderungen sind hier aufgezählt. Die Änderungen an sensuators.doc sind ganz am Ende zu finden.

1 interface.doc

Change 1: 06.03.96

- Inhalte in sensuators.doc übernommen und dann Dokument gekillt.

2 sensor.doc

Change 2: 22.-25.01.96 gegenüber alter Version:

- Beschreibung der architekturellen Umgebung hinzugenommen
- Umbenennung einiger Dienste zur Konfliktvermeidung mit Begriffen aus der Kommunikationswelt
- Umwandlung des "Cancel"-*Dienstes* in einen *Interaktionstyp* ".cancel" für die entsprechenden Funktionalitäten
- Umwandlung des "releaseErrorEvent"-Dienstes in einen Interaktionstyp für die entsprechenden Funktionalitäten zur Anpassung an das allg. Fehlerkonzept (s. Vervollständigungs-Doku, "Dienstausfall-Fehler"). Insbesondere werden Fehlermeldungen jetzt auch an Kontrolleinheiten verschickt, die den Sensor benutzen.
- Statistik-Dienste sind - aus Gründen der Einschränkung - (vorläufig) gestrichen.
- Zustände "Waiting" und "DataError" zwecks Einschränkung gestrichen.
- Die Error-Zustände gestrichen, da sie nicht mit der Vervollständigungs-Doku konform sind. Globale Fehler-Zustände können einen teilweisen Dienstausfall nicht modellieren.
- Die Dienste "initParameters" und "delete" in "init.request" und "init.cancel" verwandelt.
- Die Dienste "run" und "stop" in "run.request" und "run.cancel" verwandelt, wobei run.cancel jetzt wieder zurück in den "Initialized"-Zustand geht. Irgendwie sollte das eigentlich möglich sein. Komplett stoppen kann man dann durch run.cancel mit nachfolgendem init.cancel.
- Die "Clear-Box-Beschreibung" erstmal gestrichen, da erstmal eine Schnittstellen-Beschreibung wichtiger ist.
- Die Parameter des init-Dienstes Vectorlength, TracingRate und SamplingRate gestrichen, da Vectorlength und TracingRate nur für Statistiken (ebenfalls gestrichen) wurden und die SamplingRate erst bei den Bereichs-Sensoren sinnvoll wird.
- Beim "Polling" das Versenden der Quittung gestrichen, da es ja wohl ausreicht, wenn die Antwort ankommt.
- Die ganzen internen (und damit für den Benutzer unsichtbare) Ereignisse gestrichen. Diese sind durch die Streichung des Waiting- und der gesamten Error-Zustände überflüssig geworden.

- Das Zustandsübergangsdiagramm ziemlich gekürzt. Insbesondere enthält es jetzt keine Kanten für interne Ereignisse mehr.
- Die Tabelle zur Zusammenfassung erstmal gestrichen, weil die derzeitige gekürzte Version eigentlich übersichtlich genug ist.
- Den Begriff *response* in single-response, multi-response, receipt-response durch *reply* ersetzt, um Irritationen mit dem Interaktionstyp *.response* zu vermeiden.
- Da *abstrakter Sensor* und *physikalischer Sensor* zusammen den eigentlichen Sensor ergeben, den das Kontrollsystem sieht, macht es keinen Sinn, bei der Schnittstellenbeschreibung von irgendetwas anderem als *dem Sensor* zu reden. Also: 'abstrakt' überall gestrichen.
- Zur Vereinfachung den Begriff *Kontrollinstanz* aufgegeben. Die damit bezeichnete Kontrolleinheit ist eindeutig durch ihre Anbindung an den Kontroll-IP festgelegt. Das spart einen weiteren verwirrenden Begriff. (Kann man noch diskutieren ...).
- Der Begriff *Kontrollzelle* war im Text fast überall als Synonym für *Kontrolleinheit* gebraucht worden. Alle Vorkommen entsprechend ersetzt.

Change 3: 09.02.96

- Status-Parameter bei eventReq.response genauso überflüssig wie bei value.Req.response. Also gestrichen.
- Tabelle unsortiert. Also sortiert.

Change 4: 21.02.96, 29.02.96

- Architektur-Bild etwas umgestellt (entsprechend der vorherrschenden Vorstellung, daß die Sensoren "unten" liegen und um einen Interaktionspunkt zur Umwelt erweitert (Hüstel).
- Die Sensor-Innereien gelöscht, da diese jetzt - wie alle anderen Dinge, die von einem konkreten physikalischen Sensor-Produkt abhängig sind - in products.doc erläutert werden.
- Die allgemeine Beschreibung der architekturellen und Dienst-Eigenschaften in das Ober-Dokument instances.doc befördert.
- Den *Daten-IP* in *Sensor-IP* umgetauft, da er dann vernünftig von dem inzwischen entstandenen *Aktor-IP* unterschieden werden kann.

Change 5: 04.03.96

- Die Dienstbeschreibungen etwas systematischer gestaltet.
- Die Managment-Dienste in den gemeinsamen Teil für Aktuatoren und Sensoren überführt.
- Das Sensor-Dokument beendet und als Kapitel in sensuators.doc aufgenommen.

3 rangeSensor.doc

Change 6: 09.02.96, gegenüber alter Version V1.9, 23.11.95

- Statistik-Dienste gestrichen.
- Anpassung des cyclicReq-Dienstes an sensor.doc-Darstellung
  Streichung des Status-Parameters in cyclicReq.response (s. sensor.doc)

- Tabelle unsortiert. Also sortiert

Change 7: 06.03.96

- Dokument eufgelöst und Inhalt nach sensuators,doc überführt.

4 binarySensor.doc

Change 8: 09.02.96 gegenüber alter VersionV1.4, 23.11.95:

- Statistik-Dienste gestrichen.
- Anpassung an sensor.doc-Darstellung
- Tabelle unsortiert. Also sortiert

Change 9: 06.03.96

- Dokument beendet. Inhalte nach sensuators.doc

5 actuator.doc

Change 10: 29.02.96 ff. gegenüber alter Version V1.1, 31.10.95:

- Alle Anteile, die der Beschreibung des Sensor-Anteils eines Aktors dienten, entfernt, da diese in der Sensor-Beschreibung enthalten sind.
- Zusammenfassung aller darüberhinausgehenden Eigenscahften in einen Dienst setReq.
- Die vielen Value-Typen etwas gekürzt. Zunänchst reichen zwei: Value, wie bei den Sensoren und TargetValue für den Soll-Wert.
- Das Architektur-Bild angepaßt an die Sensor-Version.
- Tabelle entsprechend gekürzt

Change 11: 06.03.96

- Dokument beendet. Inhalte nach sensuators.doc

6 sensuators.doc

Change 12: 06.03.96

- Die o.g. Dokumente zusammenkopiert.
- Die *single-access/multi-access* Eigenschaften bezogen sich bisher auf Dienste. Sinnvoller ist es aber sie als architekturelle Eigenschaften von Interaktionspunkten zu formulieren.
- Ein Problem bei der Initialisierung: Nach der Initialisierung soll es genau eine für das Management verantwortliche Kontrolleinheit geben. Aber wer darf den Init-Dienst benutzen ?? Alle? Aus Sicht des Sensuatorsr ist das ziemlich egal, es dürfte sich die Kontrolleinheit am Kontroll-IP sogar dynamisch verändern. Dann ist aber die Übergabe der *ResponsibleUnit* bei der Initialisierung nicht notwendig. Also: gestrichen !
- Den init.respond (Status) - Interaktionstyp gelöscht. Entweder es klappt, dann gibt es ein init.receipt() oder nicht, dann gibt es einen init.error (...). Falls irgendwann mal der Status

gebraucht wird, dann sollte ein expliziter status-Dienst eingerichtet werden.

- Der *Acknowledge*-Dienstparameter ist aus demselben Grund überflüssig, da entweder ein .recipt zurückkommt oder aber ein .error(), der dem alten .receipt(NAK) entspricht. Also alle Vorkommen beseitigt. Effizienzunterschiede gibt es auch keine.

Change 13: 11.03.96

- Jede Input/Output/Parameter-Tabelle aufgeteilt in je eine Tabelle zur Typdefinition (Großschreibung) und eine zur Auflistung von Instanziierungs-Parametern (Kleinschreibung).

Change 14: 13.-19.03.96

- Bei der Beschreibung der allgemienen Dienstmuster dasselbe Schema eingesetzt wie zur Beschreibung speziellerer Dienste.
- Die *SenderAddress* muß bei jeder Interaktion des Sensuators als Parameter enthalten sein, damit die Zuordnung der Interaktionen zum Benutzer klar ist. Entsprechende Stellen erweitert. Zur Verallgemeinerung umgetauft in *Address*. Den multi-access-IP um die Fähigkeit erweitert, mit solchen Adressen zurecht kommen zu können.
- Zur Angleichuung der Begriffe mit anderen Dokumenten den Begriff *Kontrollogik* durch *Kontrollhierarchie* ersetzt.
- Einige kleine Korrekturen nach Vorschlägen von Martin Kronenburg.
- SetReq.cancel-Interaktion bei Aktuatoren gestrichen, da es keinen Vorteil bringt, die laufende Einstellung irgendwo zu stoppen und nicht zu wissen wo. Daf%ur ist aber ein Überschreiben des Stellwertes (fast) jederzeit möglich.