3.3 Entwurf der Anbindung der Aktoren und Sensoren

3.3.1 Verbindungsstruktur

Der Porstbaustein stammt aus dem bisher bestehenden WASTE. Er kapselt die Schnittstelle zu dem Kontrollsystem. Der Port-Handler realisert die Umsetzung der Ports in Aktoren und Sensoren. Dazu benutzt er den Mapper. Dieser kapselt die Datenstruktur mit deren Hilfe diese Umsetzung realisert wird. Der Message-Handler dient zur Kommunikation mit den Testraum / Simulator und kapselt deren Schnittstelle.

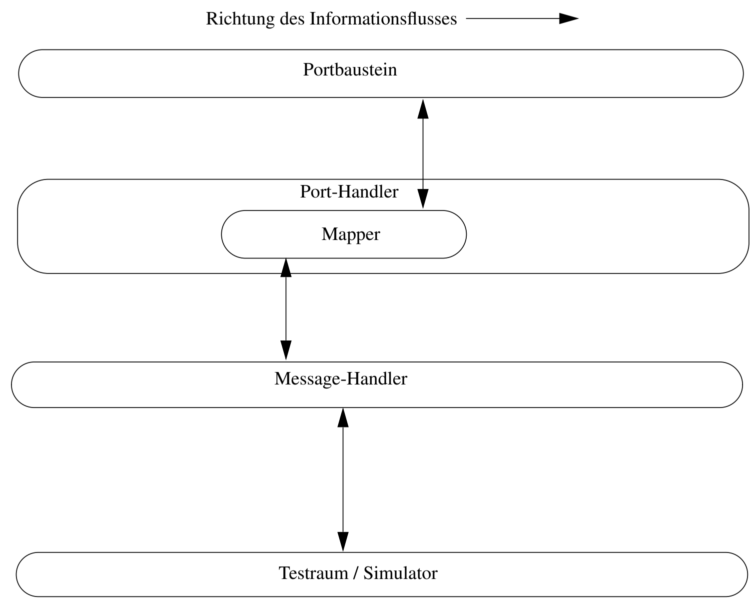


Abb. 3: Die Verbindungsstruktur

3.3.2 Portbaustein

Der Portbaustein stellt die von WASTE geforderten Methodenschnittstellen (d.h. die Methoden *CallBitInputPort*, *CallBitOutputPort*, *CallIntInputPort* und *CallIntOutputPort*) zum Kontrollsystem hin zur Verfügung. Dabei überprüft er, ob die aufgerufene Methode zum angegebenen Port paßt. Ist dies der Fall, so reicht er diesen Aufruf an den Port-Handler weiter. Dabei werden die Methoden *getBitValue*, *getIntvalue*, *setBitValue* und *setIntValue* des Port-Handlers benutzt. Die Zimmernummern werden verworfen, da der Testraum nur ein Zimmer repräsentiert.

3.3.3 Port-Handler

Der Port-Handler realisiert die Umsetzung der Ports in Aktoren und Sensoren. Er benutzt dabei den Mapper, der die Datenstruktur, mit der die Umsetzung realisiert wird, beinhaltet und kapselt. Der Port-Handler enthält nur Durchreichmethoden, die Methodenaufrufe des Portbausteins bzw. des Message-Handlers an den Mapper weiter reichen. Somit stellt er lediglich fest definierte Schnittstellen zum Mapper zur Verfügung.

3.3.4 Mapper

3.3.4.1 Grundsätzliche Entscheidungen

Da die Antwortzeiten des Testraums nicht näher spezifiziert sind (sowohl Befehl als auch Antwort werden über das Internet transportiert), der Scheduler einer CallBack-Methode jedoch nur eine ganz bestimmte Zeitspanne zuweißt, ist es gefährlich jeden Aufruf einer Methode des Portbausteins direkt an den Testraum weiterzuleiten. Sei beispielsweise angenommen diese Zeitspanne betrüge zehn Millisekunden und die Abarbeitung einer bestimmten CallBack-Methode dauere nun schon neun Millisekunden an. Wenn diese CallBack-Methode jetzt einen Aufruf *CallBitInputPort* an den Portbaustein richten würde und dieser Aufruf direkt über den Port-Handler, den Mapper, Message-Handler und das Internet an den Testraum weitergeleitet würde, könnte es geschehen, daß bevor die Antwort vom Testraum übers Internet, dem Message-Handler, den Mapper, den Port-Handler und den Portbaustein zurückgegeben werden kann, die letzte Millisekunde verstrichen ist, und der Simulator der CallBack-Methode die Kontrolle entzieht. Dadurch würde das gesamte System einen undefinierten Zustand annehmen. Gelöst wird dieses Problem dadurch, daß der Portbaustein (genauer der Mapper) die Werte aller Sensoren bzw. Aktoren zwischenspeichert und diese dann nach und nach in einer eigenen CallBack-Methode aktualisiert (d.h. die Sensoren des Testraumes abfragt und Befehle an die Aktoren sendet). Dadurch kommt es vor, daß das Kontollsystem Sensorwerte erhält, die nicht aktuell sind bzw. daß Befehle an Aktoren erst nach einer Verzögerung weitergeleitet werden. Diese zeitliche Verzögerung liegt im Sekundenbereich und ist gegenüber der Reaktionszeit des gesammten System (diese liegt im Minutenbereich) vernachlässigbar klein. Die Aktualisierung der Sensoren- und Aktorenwerte erfolgt in der CallBack-Methode des Mappers. Sie durchläuft zyklisch alle Aktoren und Sensoren und veranlaßt bei Bedarf den Message-Handler dazu Nachrichten an den Testraum zu senden. Empfängt der Message-Handler neue Sensorenwerte, meldet er diese an den Mapper, der sie in die Mappermatrix aufnimmt.

3.3.4.2 Umsetzung von Ports in Aktoren und Sensoren

Die Umsetzung (Mapping) der Ports in Aktoren und Sensoren wird mit Hilfe einer zweidimensionalen Matrix, der MapperMatix, realisiert. Ihre Einträge haben je zwei Felder. Das eine Feld (genannt Name-Feld) enthält einen String der Länge vier (das entspricht der Länge der Namen der Sensoren und Aktoren des Testraumes), das andere Feld (genannt Value-Feld) einen Integerwert. Die Dimensionen der Matrix ergeben sich wie folgt: Die Anzahl der Spalten ist gleich der Anzahl der zu behandelden Ports (hier elf). Die Anzahl der Zeilen ist gleich der maximalen Anzahl der Aktoren/Sensoren, die einem Port zugeordnet sind (hier drei). In den Name-Feldern werden die Namen der zu dem betreffenden Port

gehörenden Aktoren bzw. Sensoren gespeichert; in den Value-Feldern der dazu gehörige Wert. So lassen sich Ports und Aktoren bzw. Sensoren leicht aufeinander abbilden.

Beim Umgang mit der Matrix sind weitere Informationen von Nutzen. Zum einen ist es praktisch, die wirkliche Anzahl realer Aktoren bzw. Sensoren eines Ports schnell auslesen zu können, zum anderen verlangen Aktoren und Sensoren in der CallBack-Methode des Mappers verschiedene Behandlungen. Um diese Informationen lokal eng beieinander zu haben, werden sie zusätzlich in der Matrix gespeichert. Sie wird dazu um zwei Zeilen (Zeile 0 und Zeile 4) erweitert. Dies geschieht wie folgt: In den Elementen der ersten Zeile der Matrix wird im Value-Feld die Anzahl der realen Aktoren bzw. Sensoren gespeichert. Erst ab der zweiten Zeile stehen dort die Namen und Werte der Aktoren und Sensoren. In der letzten Zeile ihr wird eine Art Status gespeichert. Er ist als Integerwert kodiert und sagt z.B. aus, ob es sich bei diesem Port um einen Aktor oder einen Sensor handelt.

Da alle binäre Sensoren in den Event-Modus geschaltet werden können, müssen diese nicht extra „von Hand“ aktualisiert werden. Sie melden sich selbst, sobald sie ihren Zustand verändert haben. Um sie besonders zu kennzeichnen erhalten sie einen Status, der sie als aktive Sensoren kennzeichnet.

Der Regensensor liefert die Menge des seid dem letztem Reset gefallenen Regens im Millimeter. Die minimal mögliche Differenz zwischen zwei Werten beträgt zur Zeit 0.1 mm. WASTE dagegen erwartet einen Sensor, der auf die aktuelle Stärke des Niederschlags reagiert. Bei einem Niederschlag von mehr als 10 mm/h soll er eine 1 sonst eine 0 zurückgeben. Die Menge des gefallenen Regens muß also ständig überwacht und die beobachteten Veränderungen in die Einheit mm/h umgerechnet werden. Um dies zu vereinfachen, wird ein Intervall bestimmt, das zwischen zwei aufeinanderfolgenden Abfragen des Regensensors liegen muß. Es soll zum einen so lang sein, daß wenn eine Änderung in der Menge des Regens zwischen zwei aufeinanderfolgende Abfragen festgestellt wird auch der Niederschlag mindestens ein Stärke von 10mm/h hat. Zum anderen soll es so kurz wie möglich sein, um die Reaktionszeit nicht unnötig zu verlängern. Dieses Intervall läßt sich mit folgender Formel berechnen:

Intervall [s]= min. Auflösung[mm] * (3600[s/h] / Schwellwert[mm/h])

Sie ergibt sich wie folgt: der Quotient aus 3600[s/h] und dem Schwellwert in mm/h ergibt einen Wert, der in Sekunden angibt, wie lange es dauert, bis eine Niederschlagsmenge vom 1mm bei einer Stärke von 10mm/h gefallen ist. Dieser Wert wird mit der minimalen Auflösung in mm multipliziert und man erhält einen Wert, der in Sekunden angibt, wie lang es bei einer Stärke von 10mm/h dauert, bis der Regensensor mit eben dieser minimalen Auflösung ein Veränderung erkennt. Da der Regensensor eine ganz spezielle Behandlung benötigt, bekommt er einen eigenen Status, der dies kenntlich macht, zugeordnet.

Ob der in der Matrix gespeicherte Wert eines Aktors, an den Testraum übermittelt werden muß hängt davon ab, ob sich dieser Wert seit der letzten Überprüfung verändert hat (d.h. WASTE hat die Methode *CallBitOutput* bzw. *CallIntOutput* mit dem Port, der diesem Aktor entspricht aufgerufen). Diese beiden Zustände werden durch zwei weitere Stati repräsentiert.

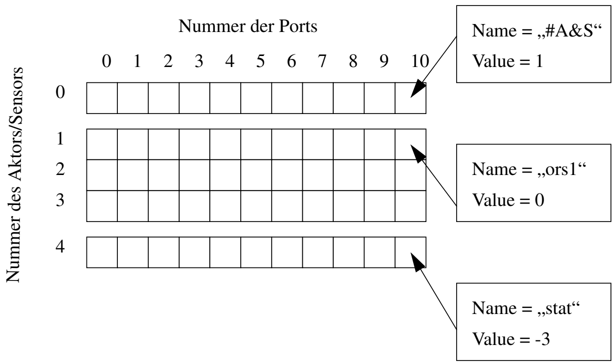
Die Steuerung der Heizkörperventile ist nicht nur von den Werten abhängig, die von WASTE über den Portbaustein und den Port-Handler in die Matrix geschrieben wurden, sondern auch von der aktuellen Raumtemperatur. So kann es vorkommen, daß allein durch Änderung der Raumtemperatur und obwohl der Port nicht angesprochen wurde, die Ventile geöffnet bzw. geschlossen werden müssen. Für diese Sensoren (die also immer betrachtet werden müssen) wird ein weiterer Status hinzugefügt.

Insgesamt ergeben sich die folgenden Stati:

Status	Bedeutung
-3	Dieser Port steht für einen Sensor, der in einer eigenen Routine behandelt wird (zur Zeit nur der *Regensensor*).
-2	Dieser Port steht für einen binären, aktiven Sensor, braucht also nie aktualisiert zu werden.
-1	Dieser Port steht für einen Sensor, muß also immer wieder aktualisiert werden.
0	Dieser Port steht für einen Aktor, der immer wieder aktualisiert werden muß (zur Zeit nur die *Heizkörperventile*).
1	Dieser Port steht für einen Aktor, der seit dem letzten Check nicht angesprochen wurde, muß also auch nicht aktualisiert werden.
>1	Dieser Port steht für einen Aktor, der seit dem letzten Check angesprochen wurde, er muß also aktualisiert werden. Außerdem gibt der Status an, wie viele der realen Aktoren noch zu aktualisieren sind. (Bei jeder Aktualisierung eines dieser Aktoren wird der Status um eins verringert; ist er gleich eins, so sind alle realen Aktoren dieses Ports aktualisiert.)

Tabelle 17: Stati der Aktoren und Sensoren

Die Matrix hat insgesamt folgenden Aufbau:



Der Mapper muß nun Methoden bereitstellen, mit denen der Port-Handler Sensorenwerte aus der Matrix auslesen und neue Werte für Aktoren eintragen kann. Außerdem muß eine Methode existieren, mit der der Port-Handler im Auftrag des Message-Handlers neu empfangene Sensorenwerte an den Mapper melden kann.

3.3.5 Message-Handler

Der Message-Handler wurde aus dem Message-Handler der Benutzungsschnittstelle entwickelt. Wie dieser arbeitet auch er als Client (das Verfahren zum Verbindungsaufbau wird ohne Änderung übernommen). Er stellt Methoden zum Senden (*sendRequest* und *sendOrder*) zur Verfügung und verarbeitet in seiner CallBack-Methode eingehende Nachrichten. Außerdem stellt er Methoden (*makeActive* und *makePassive*) zur Verfügung, mit deren Hilfe binäre Sensoren in den Event-Modus und wieder in den Normal-Modus geschaltet werden können. Der Nachrichtentransport zwischen WASTE und dem Testraum unterscheidet sich aber etwas von dem zwischen WASTE und der Benutzungsschnitstelle. Bei beiden Anbindungen kommen zwar Socketstreams aus [Cod97] zum Einsatz (diese werden noch um Operatoren zum Senden und Empfangen einzelnener Zeichen erweitert), jedoch werden zwischen WASTE und dem Testraum reine Zeichenfolgen ohne Header übertragen. Als Endmarkierung für jede Nachricht (sowohl zum als auch vom Testraum) dient ein ‚^A‘ (=‚\001‘). Daran kann jeder der Kommunikationspartner erkennen, wann er eine Nachricht vollständig empfangen hat.

Der Message-Handler kann folgende Aufträge vom Port-Handler (d.h. eigentlich vom Mapper) erhalten:

- Setze den Aktor *actor* auf den Wert *value* (Methode *sendOrder* (*actor*, *value*)).
- Frage den Sensor *sensor* ab (Methode *sendRequest* (*sensor*)).
- Schalte den Sensor *sensor* in den Event-Mode (Methode *makeActive* (*sensor*)).
- Schalte den Sensor *sensor* in den Normal-Mode (Methode *makePassive* (*sensor*)).

In jedem dieser Fälle baut er den entsprechenden Befehlsstring auf und verschickt ihn.

Da der Testraum auf jede von ihm empfangene Nachricht mit einer Antwort, Bestätigung oder Fehlermeldung reagiert, überwacht der Message-Handler seinen Verbindungsendpunkt auf eingehende Nachrichten. Anhand des ersten gelesenen Zeichens ermittelt er um welche Art von Nachricht es sich handelt. Bestätigungen beginnen mit ‚A‘, Fehlermeldungen mit ‚E‘ - alles weitere (in der zur Zeit gültigen Spezifikation des Schnittstelle des Testraumes) müssen Antworten sein, die Sensorenwerte enthalten. Nun liest er die Nachricht vollständig ein (d.h. bis zum ‚^A‘). Da die Spezifikation der Schnittstelle des Testraumes in Zukunft erweitert werden könnte (insb. um Nachrichtentypen die auch mit 'A' bzw. 'E' beginnen, aber keine Bestätigungen bzw. Fehlermeldungen sind, wird die komplett empfangene Nachricht zusätzlich überprüft, ob sie wirklich eine Bestätigung bzw. Fehlermeldung war. Bestätigungen werden ignoriert, Fehlermeldungen und unbekannte Nachrichtentypen führen dagegen zum Abbruch. War die eingegangene Nachricht die Meldung eines Sensorenwertes, extrahiert der Message-Handler aus ihr den Namen des Sensors und den Wert und meldet beides mit der Methode *updateValue* an den Port-Handler. Schließlich aktualisiert der Mapper den Wert dieses Sensors in seiner Matrix. Das Sende- und Empfangsprotokoll unterscheiden sich nur marginal (beim Sendeprotokoll werden keine Header, sondern Befehlsstrings aufgebaut und beim Empfangsprotokoll wird nicht anhand eines Nachrichtentyps, sondern anhand des erstem empfangenen Zeichens entschieden, um welchen Typ Nachricht es sich handelt) von denen des Message-Handlers des Zimmers. Deshalb werden sie hier nicht mehr aufgeführt.

3.3.6 Initialisierung

Der Verbindungsaufbau erfolgt genau wie der Verbindungsaufbau zwischen Benutzungsschnittstelle und Zimmer. Selbst der Kode, der den Verbindungsaufbau auf Seite des Clients (Benutzungsschnittstelle) realisiert wird vollständig übernommen. Allerdings hat dieser Message-Handler keinen Monitor. Sendeanforderungen stellt hier der Mapper, der Teil des Port-Handlers ist. Dieser nimmt auch unmittelbar nachdem das Kontrollsystem (das ursprüngliche WASTE) vollständig initialisiert und der Scheduler gestartet wurde seine Arbeit auf, denn WASTE arbeitet auch unabhängig von den Benutzereingaben mit den Defaultwerten. Aus diesem Grund wird hier nicht gewartet, bis der Benutzer den Start-Button gedrückt hat.

Die value-Felder der Mappermatrix im Bereich der Aktoren und Sensoren (d.h. der Zeilen eins bis drei) werden bei der Initialisierung mit Initial- bzw. mit Defaultwerten belegt. Bei Ports, die Sensoren repräsentieren, dies mangels realer Sensoren aber nicht realisieren können, wird der Default-Wert im entsprechenden value-Feld der ersten Zeile eingetragen.

Die gesamte Matrix wird wie folgt initialisiert (der besseren Lesbarkeit halber ist die Matrix um 90 Grad gedreht):

	0	**1**	**2**	**3**	**4**
Port 0	name = „#A&S“ value = 1	name = „rts2“ value = 0	name = „defa“ value = 0	name = „defa“ value = 0	name = „stat“ value = -2
Port 1	name = „#A&S“ value = 1	name = „imd1“ value = 1	name = „defa“ value = 0	name = „defa“ value = 0	name = „stat“ value = -1
Port 2	name = „#A&S“ value = 3	name = „wtm1“ value = 0	name = „wtm2“ value = 0	name = „wtm3“ value = 0	name = „stat“ value = -2
Port 3	name = „#A&S“ value = 3	name = „rva1“ value = 0	name = „rva2“ value = 0	name = „rva3“ value = 0	name = „stat“ value = 0
Port 4	name = „#A&S“ value = 3	name = „suc1“ value = 1	name = „suc2“ value = 1	name = „suc3“ value = 1	name = „stat“ value = -2
Port 5	name = „#A&S“ value = 0	name = „defa“ value = 1	name = „defa“ value = 0	name = „defa“ value = 0	name = „stat“ value = -2
Port 6	name = „#A&S“ value = 3	name = „wop1“ value = 0	name = „wop2“ value = 0	name = „wop3“ value = 0	name = „stat“ value = 1
Port 7	name = „#A&S“ value = 0	name = „defa“ value = 0	name = „defa“ value = 0	name = „defa“ value = 0	name = „stat“ value = 1
Port 8	name = „#A&S“ value = 0	name = „defa“ value = 0	name = „defa“ value = 0	name = „defa“ value = 0	name = „stat“ value = 1
Port 9	name = „#A&S“ value = 1	name = „ots1“ value = 0	name = „defa“ value = 0	name = „defa“ value = 0	name = „stat“ value = -1

	0	**1**	**2**	**3**	**4**
Port 10	name = „#A&S“ value = 1	name = „ors1“ value = 0	name = „defa“ value = 0	name = „defa“ value = 0	name = „stat“ value = -3

3.3.7 Terminierung

Der Verbindungsabbau erfolgt während der durch den Benutzer ausgelösten Terminierung des Gesamtsystems. Er wird in den Destruktor des Message-Handlers integriert. Da auf die Schnittstelle des Testraums bei dieser Anbindung kein Einfluß genommen werden kann, und diese Schnittstelle außer dem einfachen Schließen der Sockets keinen Verbindungsaufbau unterstützt, muß auf eine andere Art und Weise sicher gestellt werden, daß bei Abbau der Verbindung keine Nachrichten mehr unterwegs sind. Dazu müssen die Sensoren, die in den Event-Modus geschaltet wurden, wieder in den normalen Modus zurück geschaltet werden (dies geschieht im Destruktor des Mappers). Anschließend ließt der Message-Handler seinen Verbindungsendpunkt eine durch die maximale Antwortzeit bestimmte Zeit (hier werden drei Sekunden angenommen) aus. Alle empfangenen Nachrichten werden dabei ignoriert. Da sicher gestellt wurde, daß während dieser Zeit keine Nachricht an den Testraum gesendet wurde, sind mit sehr hoher Wahrscheinlichkeit sowohl der Nachrichtenkanal als auch alle Puffer leer. Nun kann der Message-Handler seinen Socket schließen. Der Testraum bemerkt dies automatisch und schließt daraufhin seinen Socket.

3.3.8 Simulator

Die AG Zimmermann stellt einen Simulator bereit, der die gesamte Funktionalität des Testraumes simuliert. Dieser Simulator ist mit Visual-Wave realisiert und auf den Rechnern 'sep1' und 'sep2' ablauffähig. Er stellt eine grafische Oberfläche zum Anzeigen und Manipulieren der Aktoren und Sensoren des Testraums und der daran angeschlossenen Wetterstation zur Verfügung. Außerdem besitzt er die gleiche Socketschnittstelle wie der reale Testraum.

Da ein Testen des Systems in dem realen Testraum sehr schwierig ist (man müßte eine Zugangsberechtigung zu den Aktoren erhalten) und eine falsche Ansteuerung der Aktoren zu Schäden an dem Raum und seinem Invertar führen könnte, wird sowohl aus praktischen Gründen (da die Tests z.T. auf unterschiedlichen Umweltbsdingungen basieren, diese man im Testraum allerdings nicht künstlich schaffen kann, wären sie nur dann durchführbar, wenn die Umweltbedingungen zufällig mit den geforderten des Tests übereinstimmten) und Sicherheitsgründen die Anbindung an den Simulator und nicht an den realen Testraum realisiert. Um das System später an den Testraum anzubinden, sind nur die Daten des Hosts und des Ports des Testraumes (hauspc1.informatik.uni-kl.de, 32418) statt denen des Simulators (im Regelfall sep1.informatik.uni-kl.de, 32418) einzugeben. Die Schnittstelle ist bei beiden die gleiche.

Bei Tests hat sich herausgestellt, daß die Behandlung vieler Nachrichten innerhalb kurzer Zeit auf Seite des Simulators so viel Rechenzeit verbraucht, daß für die Simulation nicht mehr genügend Ressourcen bereit stehen. Um dies zu umgehen, werden nicht in jedem Zyklus des Schedulers Nachrichten an den Simulator geschickt. Nach einigen Tests hat sich eine minimale Zeitspanne von zwei Sekunden zwischen der Sendung zweier aufeinanderfolgender Nachrichten als akzeptabel erwiesen. Die Reaktionszeit des Gesamtsystem wird dadurch zwar negativ beeinflußt, aber da es sich nur um Größen im Sekundenbereich handelt, ist diese Beeinflussung vertretbar. Außerdem sind hiervon in erster Linie die normalen Sensoren, wie zum Beispiel die Temperaturfühler, betroffen. Diese ändern ihre Werte nicht so schnell, als daß diese Verzögerung von Belang wäre. Weiterhin werden im Simulator die Inhalte der einzelnen Fenstern,in denen die Werte der Aktoren und Sensoren angezeigt werden, in einstellbaren Intervallen aktualisiert. Wählt man hier einen zu kleinen Wert, ist der Simulator so sehr mit dem Aktualisieren der Fenster beschäftigt, daß kaum Rechenzeit für die eigentliche Simulation verbleibt. Bei meinen Test habe ich mit einem Intervall von dreißig Sekunden ein noch akzeptables Verhalten beobachtet.

3.3.9 Objektmodell

Das folgende Objektmodell beschreibt nur den Teil des Systems, der für die Anbindung von Relevanz ist. Der Simulator wird durch ein Subsystem abstrahiert.