Echtzeit

Nr. 1, Juli 2014

Mitteilungen
des GI/GMA/ITG-Fachausschusses
Echtzeitsysteme

GESELLSCHAFT FÜR INFORMATIK E.V.

VDI VDE VDI/VDE-Gesellschaft
Mess- und Automatisierungstechnik

ITG INFORMATIONSTECHNISCHE
GESELLSCHAFT IM VDE

Impressum

Herausgeber	GI/GMA/ITG-Fachausschuss Echtzeitsysteme http://www.real-time.de
Sprecher	Prof. Dr. Dr. Wolfgang A. Halang FernUniversität in Hagen Lehrstuhl für Informationstechnik 58084 Hagen wolfgang.halang@fernuni-hagen.de
Stellvertreter	Prof. Dr. Dieter Zöbel Universität Koblenz-Landau Institut für Softwaretechnik 56016 Koblenz zoebel@uni-koblenz.de
Chefredaktion	Prof. Dr.-Ing. habil. Herwig Unger FernUniversität in Hagen Lehrstuhl für Kommunikationsnetze 58084 Hagen herwig.unger@fernuni-hagen.de
Redaktion, Layout	Dipl.-Ing. Jutta Düring FernUniversität in Hagen Lehrstuhl für Informationstechnik 58084 Hagen jutta.duering@fernuni-hagen.de
ISSN	2199-9244

Redaktionell abgeschlossen am 07. Juli 2014

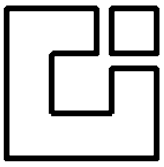
Einreichung von Beiträgen:
Alle Leserinnen und Leser sind aufgerufen, das Mitteilungsblatt auch zukünftig durch Beiträge mit zu gestalten, um den Informations- und Meinungsaustausch zwischen allen an den Fragen der Echtzeitprogrammierung Interessierten zu fördern.

In dieser Ausgabe:

1 Eine kurze Geschichte der Zeit

Jutta Düring, Fachausschuss Echtzeitsysteme

Der Fachausschuss wurde 2003 von „Echtzeitprogrammierung – PEARL“ in „Echtzeitsysteme“ und die Tagung 2008 von „PEARL“ in „Echtzeit“ umbenannt. Konsequenterweise hat die Fachausschussleitung nun auch dem Rundbrief einen neuen Namen gegeben. Wir sind stolz, Ihnen die Erstausgabe unseres neuen Rundbriefs Echtzeit vorstellen zu dürfen. Wir hoffen, es gefällt.



Die erste Ausgabe der PEARL-News erschien im Jahre 1992; ausgestattet mit dem alten GI-Logo (rechts) und dem alten Schriftzug in der heute nicht mehr gebräuchlichen Schriftart OCR. Bis 1997 gaben die Herren Hans Windauer (Sprecher), Peter Holleczek (Stellvertreter) und Leberecht Frevert (Redaktion) die Zeitschrift heraus. Als Redakteure folgten mit verschieden langen Amtszeiten die Herren Wolfgang A. Halang, Rainer Müller, Rolf Müller, Reinhard Baran und Herwig Unger.

Für ganze 15 Jahre, von 1999 an bis zur ersten Ausgabe im Januar diesen Jahres begleitete uns das bunte Pearl-Quadrat. Einige werden es zukünftig vielleicht vermissen. Auch wenn der Begriff PEARL nun ganz aus der Namensgebung verschwunden ist, hat sich der Fachausschuss die Pflege dieser bewährten höheren Programmiersprache weiterhin auf seine Fahnen geschrieben. Hierzu gehören sowohl die Entwicklung des Linux-Compilers OpenPearl als auch die Neufassung der DIN 66253-2 und DIN 66253 Teil 3 insbesondere bezüglich funktionaler Sicherheit.

2 Ein Benchmark in PEARL für umwelttechnische Simulationen

Christina K. Houben, Wolfgang Gödde
FernUniversität in Hagen

Die Programmiersprache PEARL unterstützt den Echtzeitbetrieb sowie Nebenläufigkeit, Portabilität und Sicherheit in ihr erstellter Software, wodurch sie besonders für die Automatisierungstechnik geeignet ist. Auch in der Umwelttechnik wird automatisiert, zum Beispiel bei der Gewinnung erneuerbarer Energie, der Mülltrennung in der Abfallverwertung oder der Autoabgasfilterung im Verkehr.

In der Umwelttechnik wird angestrebt, Anlagen so auszulegen, dass das Klima möglichst wenig beeinflusst und die biologische Diversität nicht gefährdet wird. Um die Effizienz der Automatisierungssysteme in Bezug auf den Umweltschutz zu ermitteln, werden akkumulative Simulationen durchgeführt. Aus diesem Grund stellen wir im Folgenden die Architektur eines Benchmarks in PEARL vor, das die zeitlichen Veränderungen der Umweltbedingungen unter Einfluss eines technischen Systems simuliert.

Wir verwenden PEARL, weil auf diese Art der Benchmark leicht mit einem in PEARL geschriebenen Automatisierungsprogramm verbunden werden kann. Darüber hinaus bietet sich so die

Gelegenheit, Schwächen von PEARL90 anhand unseres Beispiels aufzuzeigen, um diese Erkenntnisse für die Weiterentwicklung von PEARL90 zu PEARL2020 zu nutzen und die Schwächen zu eliminieren. Deshalb geben wir nicht nur Ausschnitte unseres PEARL90-Codes an, sondern auch, wie dieser Code in PEARL2020 aussehen würde.

Bei Kleinprojekten wie einer Photovoltaikanlage für private Haushalte, einer Meerwasserentsalzungsanlage für eine Gruppe von Nomaden oder Wasser-Recycling eines Hallenbads ermöglicht unser Benchmark, eingebettete Systeme zu testen und zu validieren, *bevor* hohe Investitionskosten für einen Prototypen anfallen. Bei Großprojekten, die ohnehin ohne Prototypen auskommen müssen, können mit dem Umwelttechnik-Benchmark Unit-Tests für Teilsysteme durchgeführt werden, ohne von der Fertigstellung des Gesamtsystems abhängig zu sein. Der Benchmark ist nicht dafür vorgesehen, die ökonomische Nachhaltigkeit einer Anlage zu ermitteln, weil die Beantwortung der Fragen zur Finanzierung und Rendite mit wenigen Eingaben auskommen, die einfach verrechnet werden können, nämlich hauptsächlich die jährlich erwartete Produktion, Betriebs- und Investitionskosten.

2.1 Grundlagen der Umwelttechnik

Die Teilbereiche der Umwelttechnik befassen sich mit Luft, Wasser, Boden, Energie, Abfall und Verkehr, siehe Abb. 1. Daher ist es sinnvoll, ein PEARL-Modul für jeden Teilbereich zu schreiben. Zusätzlich muss es ein Modul mit einem globalen Strukturtypen `UmweltZustand` und der global zugänglichen Instanz `AktuellerZustand` geben, weil die einzelnen Teilbereiche sich gegenseitig beeinflussen. Beispielsweise bewirkt ein Sturzregen eine Veränderung der Konzentration von Giftstoffen im Abwasser, so dass der Wirkungsgrad von Klärwerken verringert wird. Um ein realitätsgetreues Abbild der Natur in unserem Benchmark zu erreichen, geben wir im Folgenden eine Zusammenfassung der Einführung in die Umwelttechnik nach [1] und [2].

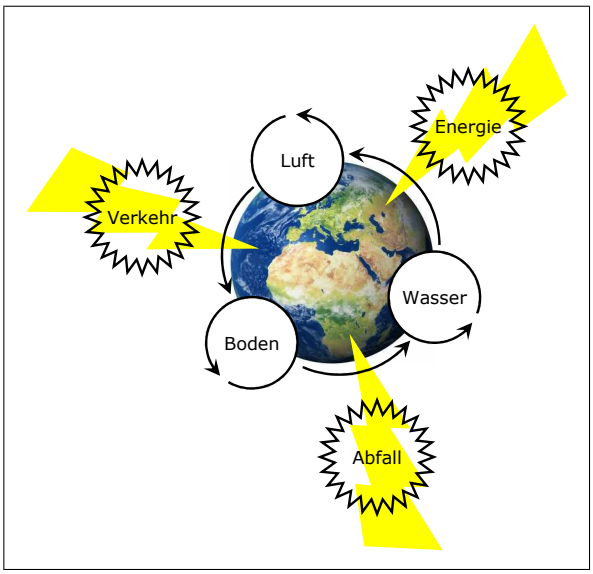


Abbildung 1: Teilprobleme

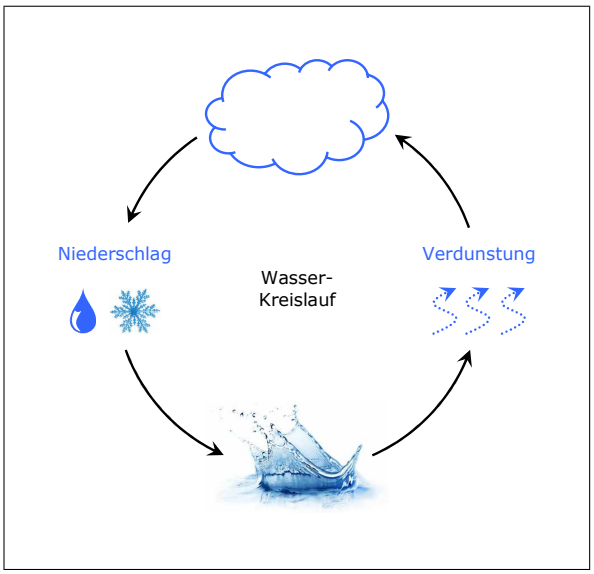


Abbildung 2: Wasserkreislauf

Unsere geophysikalische Umwelt besteht aus den drei Bereichen Atmosphäre bzw. Luft, Lithosphäre bzw. Boden und Hydrosphäre bzw. Wasser, siehe Abb. 1. Bevor die Industrialisierung ihren Anfang nahm, standen alle drei Bereiche in einem Gleichgewicht zueinander: Eine Pflanze nimmt *Wasser* und Nährstoffe aus dem *Boden* auf, wodurch sie mittels Photosynthese das

Kohlendioxid aus der *Luft* in Sauerstoff zum Atmen für Lebewesen umwandelt. Auch innerhalb der einzelnen Teilbereiche gibt es Kreisläufe, z.B. den Wasserkreislauf in der Hydrosphäre, siehe Abb. 2. Es gibt jedoch auch komplizierte Kreisläufe, die man eigentlich nicht mehr als *Kreis*läufe bezeichnen kann, sondern eher als verwobene gerichtete Graphen. Hierzu zählen die Kreisläufe von Treibhausgasen wie Kohlendioxid.

Dieses Gleichgewicht ist seit Beginn der Industrialisierung gestört, wodurch neue Teilbereiche bzw. Teil*probleme* der Umwelttechnik erwachsen, nämlich Energie, Abfall und Verkehr. Es sind dies die Bereiche, in denen automatisiert wird. Unser Umwelttechnik-Benchmark stellt dafür eine Simulationsumgebung bereit, um das Erreichen folgender Ziele zu prüfen: Umweltschäden vermeiden bzw. weitgehend vermindern, Rohstoffe und Restenergie so weit wie möglich wiederverwerten und noch übrige Produkte am Ende des technischen Weges gefahrlos beseitigen.

2.2 Zusammenspiel der Module

In Abb. 3 zeigen wir den Aufbau unseres Umwelttechnik-Benchmarks in Form eines Paket- und Klassendiagramms. Darin haben wir durch Pfeile die Schnittstellen gekennzeichnet, über die Automatisierungs-Software und Simulations-Benchmark kommunizieren: Die Sensoren erhalten Eingabewerte wie Niederschlagsmenge oder Temperatur, die durch die Steuerung verarbeitet und als Regelvariablen an die Aktoren gesendet werden, diese wiederum liefern Eingabewerte für das Simulationspaket.

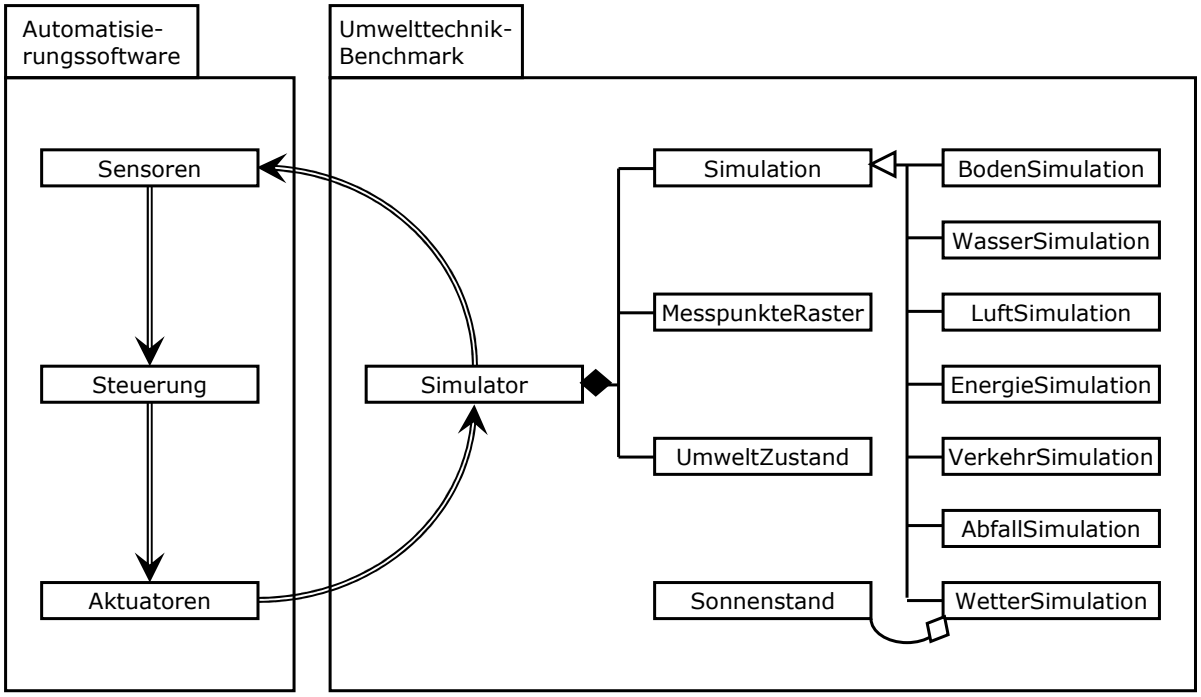


Abbildung 3: Übersicht über die Module des Benchmarks

Die einzige Schnittstelle zum Automatisierungssystem stellt also das Modul `Simulator` dar. `Simulator` besitzt ein Array der sieben verschiedenen Simulationsteilbereiche, Daten über das `MesspunkteRaster` und den aktuellen `UmweltZustand`. Der aktuelle `UmweltZustand` muss `GLOBAL` vereinbart werden, weil die verschiedenen Simulationen darauf zugreifen. Aber auch einige Prozeduren des `Simulators` müssen `GLOBAL` sein, weil sie die Schnittstelle zum Automatisierungssystem bilden. Dies führt dazu, dass nicht nur die eigentliche `Simulator`-Schnittstelle für das

Automatisierungsprogramm sichtbar ist, sondern auch der aktuelle `UmweltZustand`. Es wäre aber besser, wenn letzteres nur innerhalb des Benchmark-Pakets sichtbar wäre. Dies führt uns zu unserer ersten Neuerung für PEARL2020:

Verschieden weitreichende Sichtbarkeitsbeschränkungen:
PEARL2020 definiert statt des `GLOBAL`-Attributs einen `INTERFACE`-Teil für jedes Modul mit den Zugriffsbeschränkungen `PUBLIC` als Ersatz für `GLOBAL` und `PROTECTED` für den paketweiten Gebrauch. Alle Prozeduren und Variablen bleiben privat, die nicht im `INTERFACE`-Teil erscheinen.

Hieraus folgen sofort zwei weitere Neuerungen: Zum Einen muss es die Möglichkeit geben, Module zu Paketen zusammenzufassen, zum Anderen müssen die Zugriffsrechte auf Variablen in Schreib- und Leserechte unterteilt werden. Ersteres lösen wir wie folgt:

Zusammenfassen von Modulen zu Paketen:
Um Pakete zu definieren, ist in PEARL2020 keine Syntax notwendig. Ein Paket wird durch ein Verzeichnis im Dateisystem dargestellt. Dabei gelten Unterverzeichnisse als eigenständige Pakete, sowohl vom Ober- zum Unterverzeichnis als auch umgekehrt ist kein `PROTECTED`-Zugriff möglich.

Dies entspricht der Handhabung in Java. Für das Problem der Lese-/Schreibrechte gibt es zwei Lösungen: Zum Einen könnte man Variablen, auf die nur lesend zugegriffen werden soll, weiterhin geheimhalten, aber eine Prozedur zum Lesen veröffentlichen. Zum Anderen könnte man aber auch folgende elegantere Lösung wählen:

Variablen, die von außen nur gelesen werden dürfen:
Von einem externen Modul nur les-, aber nicht veränderbare `PUBLIC`- oder `PROTECTED`-Variablen erhalten im `INTERFACE` zusätzlich das Attribut `READ`.

Weiter geht es mit den einzelnen Simulationen von `BodenSimulation` bis `WasserSimulation`, die in Abb. 3 in einer Vererbungsbeziehung zum Modul `Simulation` stehen. In PEARL90 ist dies nicht möglich, hätte aber den Vorteil, dass in der Oberklasse `Simulation` Prozedursignaturen vorgegeben werden könnten, die genau so von allen erbenden Klassen implementiert werden müssten, was eine einheitliche Art des Zugriffs böte. Als ersten Schritt in diese Richtung führen wir neben dem `STRUCT`-Schlüsselwort auch `CLASS`en ein, um nicht nur Variablen und Zeiger auf Prozeduren zusammenfassen zu können, sondern auch Variablen und Prozeduren ohne Zeiger:

Klassen zur Kapselung von Variablen und Prozeduren:
PEARL2020 stellt das `CLASS`-Schlüsselwort bereit, welches die gleiche Syntax wie `STRUCT` besitzt, aber auch Prozeduren als Komponenten erlaubt.

Nun kommen wir zur Vererbung, die auch ihre Tücken hat, und zwar Vererbungsanomalie, Namenskonflikte, das Fragile-Base-Class- und das Diamant-Problem sowie CAT-Calls, siehe [3] für Einzelheiten. Vererbung hat aber auch Vorteile, nämlich Wiederverwendbarkeit, Vermeidung von Fehlern beim Code-Kopieren und Anpassung von Implementationen hinter der Schnittstelle, wodurch benutzende Module nicht beeinträchtigt werden, siehe [3]. Daher wählen wir in PEARL2020 einen Mittelweg und erlauben die Vererbung von Schnittstellen [4], nicht aber von Implementationen:

Vererbung von Schnittstellen:
PEARL2020 erlaubt die Vererbung von Schnittstellen mit `INTERFACE ... EXTENDS`.

Als letztes verbleibt in unserer Übersicht das Modul `Sonnenstand`, das von `WetterSimulation` benutzt wird, weil z.B. Temperatur und Einstrahlungsstärke von der Höhe der Sonne über dem Horizont abhängen und die Höhe ihrerseits von Datum und Uhrzeit. Einen frei verfügbaren Algorithmus hierfür findet man in [5]. In diesem Modul brauchen wir also einen Typen `DATE`, um nicht eine umständliche Extraktion des Datums aus einer Zeichenkette wie in Abb. 4 programmieren zu müssen:

Datentyp für Datumsangaben:
PEARL2020 hat neben den Echtzeitdatentypen `CLOCK` und `DURATION` auch einen Typen `DATE`, der im Format `TT.MM.JJJJ` angegeben wird. Zusätzlich stellt der Übersetzer die Funktionsprozedur `TODAY: PROC RETURNS(DATE)` für das aktuelle Datum zur Verfügung.

2.3 Das Schnittstellen-Modul SimulatorMod

Nun können wir einen genaueren Blick auf das Schnittstellen-Modul `SimulatorMod` werfen, siehe Quelltext in Abb. 5. Auf der linken Seite wird eine Struktur `UmweltParams` definiert, welche alle möglichen und messbaren Eigenschaften für sämtliche Simulationsbereiche enthält. In Kommentaren ist hinter der physikalischen Größe jeweils vermerkt, in welcher Einheit sie angegeben werden muss. Zahlenwerte ohne physikalische Einheiten bergen eine große Fehlerquelle. So scheiterte eine Mars-Mission, weil die Einheiten Fuß und Meter verwechselt wurden [6]. Auf der rechten Seite der Abbildung ist zu sehen, dass für jedes Attribut aus `UmweltParams` ein eigener Getter und Setter in `Simulator` programmiert werden muss. Hier ist zu überlegen, ob Aufrufe ähnlich Call-By-Name aus funktionalen Programmierparadigmen übernommen werden sollten, damit nur ein Setter und ein Getter implementiert werden muss, der als Parameter nicht nur die Position innerhalb des Rasters erhält, sondern auch den Namen der `UmweltParams`-Komponente, um darauf zuzugreifen und die gewünschte Prozessvariable zu interpolieren. Außerdem sehen wir im Quelltext der Abb. 5, dass für ein und dieselbe Klasse verschiedene `INTERFACE`s implementiert werden dürfen [4], weil somit eine elegante und übersichtliche Implementierung des Model-View-Controller-Prinzips möglich ist, siehe [3]. Dies führt uns zu drei weiteren Verbesserungsvorschlägen für PEARL:

Verifikation physikalischer Einheiten:
PEARL2020 sollte ein Konzept vorsehen, mit dem die korrekte Verwendung physikalischer Einheiten überprüft werden kann. Als Vorbilder können das Zusatzpaket aus Ada und das Typsystem von NewSpeak dienen.

Call-By-Name-Parameterübergaben:
Es ist zu überprüfen, inwieweit das Call-By-Name-Prinzip funktionaler Programmierparadigmen die Ausdrucksfähigkeit der Programmiersprache erhöht, ohne die funktionale Sicherheit negativ zu beeinflussen.

```
SPC DATE ENTRY RETURNS(CHAR(10)) GLOBAL;

! Eingabewerte:
!   String - Zeichenkette mit Asciizeichen von '0' bis '9'
!   Index  - Position des gewuenschten Zeichens von 1 bis 10
! Ausgabewerte:
!   die zu '0' bis '9' gehoerige Zahl 0 bis 9
! Berechnung:
!   Wandelt ein Zeichen einer Zeichenkette in eine Zahl um.
Ziffer2Fixed: PROCEDURE(String CHAR(10), Index FIXED)
  RETURNS(FIXED);
  DCL AsciiBias INV FIXED INIT(48);
  RETURN(TOFIXED(String.CHAR(Index)) - AsciiBias);
END;

! Eingabewerte:
!   String - Datum als Zeichenkette TT-MM-JJJJ formatiert
! Ausgabewerte:
!   Tag    - Tageszahl von 1 bis 31
!   Monat  - Monatszahl von 1 bis 12
!   Jahr   - Jahreszahl
! Berechnung:
!   Wandelt eine Zeichenkette im Format TT-MM-JJJJ so um,
!   dass Tag, Monat und Jahr als FIXED dargestellt werden.
String2Datum: PROCEDURE(String CHAR(10),
  Tag FIXED IDENT, Monat FIXED IDENT, Jahr FIXED IDENT);
  Tag   :=   10 * Ziffer2Fixed(String,  1) +
              1 * Ziffer2Fixed(String,  2);
  Monat :=   10 * Ziffer2Fixed(String,  4) +
              1 * Ziffer2Fixed(String,  5);
  Jahr  := 1000 * Ziffer2Fixed(String,  7) +
            100 * Ziffer2Fixed(String,  8) +
             10 * Ziffer2Fixed(String,  9) +
              1 * Ziffer2Fixed(String, 10);
END;
```

Abbildung 4: Extraktion des Datums aus einer Zeichenkette in PEARL90

Mehrere Schnittstellen innerhalb einer Klasse:
Ähnlich Modula-3 erlaubt PEARL2020 mehrere Schnittstellen innerhalb derselben Klasse.

In Tabelle 1 ist für fünf physikalische Größen aus `UmweltParams` beispielhaft angegeben, mit welchen Messgeräten sie erfasst werden, in welchen Wertebereichen sie liegen und welche Durchschnittswerte in Deutschland in den letzten Jahren für sie gemessen wurden. Außerdem sind einige Einzelereignisse wie eine Bö aufgezählt, welche in der Simulation mit einer bestimmten Wahrscheinlichkeit auftreten sollten.

Die physikalischen Größen lassen sich weiter unterteilen, so kann man z.B. bei der Einstrahlung zwischen globaler, diffuser und direkter Strahlung unterschieden, wobei ungefähr `GlobaleEinstr := DiffuseEinstr + DirekteEinstr` gilt. Nicht fokussierende Systeme (z.B. Photovoltaik) können die globale Strahlung nutzen, fokussierende Systeme (z.B. Concentrating Solar Power – CSP) nutzen hingegen nur die direkte. Als Messgeräte für die Einstrahlungsarten dienen ein Pyranometer für die Globalstrahlung und ein Pyrheliometer für die Direktstrahlung.

```
TYPE UmweltParams STRUCT [
  ! Luft-Bereich
  CO2Masse FLOAT,      ! kg
  SO2Masse FLOAT,      ! kg
  OzonMasse FLOAT,     ! kg
  ! Wasser-Bereich
  SalzGehalt FLOAT,    ! %
  ! Boden-Bereich
  Regenwuermer FLOAT,  ! 1
  ! Energie-Bereich
  EVerbrauch FLOAT,    ! J
  EEinspeisung FLOAT,  ! J
  ! Weitere Bereiche
  ...
  ! Wetter-Bereich
  Windstaerke FLOAT,   ! km/h
  Temperatur FLOAT,    ! C
  Niederschlag FLOAT   ! mm
] PROTECTED;

TYPE Raster STRUCT [
  (Hoehe, Breite) INV FLOAT
    INIT(100, 200),    ! m
  (Zeilen, Spalten) INV
    FIXED INIT(2, 2), ! 1
  Messpunkte(Zeilen,
    Spalten) UmweltParams
] PROTECTED;
```

```
TYPE Simulator CLASS [
  INTERFACE(SensorInterface) PUBLIC;
    HoleWindstaerke: PROC(Pos Punkt)
                     RETURNS(FLOAT);
    HoleTemperatur:  PROC(Pos Punkt)
                     RETURNS(FLOAT);
    ...
  INTERFACE(AktorInterfac) PUBLIC;
    SetzeCO2Masse: PROC(Wert FLOAT,
                        Pos Punkt);
    SetzeSO2Masse: PROC(Wert FLOAT,
                        Pos Punkt);
    ...
  INTERFACE(AutoSysInterface) PUBLIC;
    NaechsterZeitschritt:
      PROC(Zeitschritt DUR);
  INTERFACE(StandardInterfc) PRIVATE;
    AktuelleZeit CLOCK,
    Raster MesspunkteRaster,
    AktuellerZustand(Raster.Zeilen,
                     Raster.Spalten)
                     UmweltParameter;
    InterpolationsParameter:
      PROC(Pos Punkt,
      Zeilen(4) FIXED IDENT,
      Spalten(4) FIXED IDENT,
      Gewicht(4) FLOAT IDENT);
] PUBLIC;
```

Abbildung 5: PEARL2020-Spezifikationen im Simulator-Modul

Als letztes ist in Abb. 5 die Struktur `Raster` gelistet, welche ein Gitter aus Messpunkten über eine rechteckige Fläche legt. Beispiele für solche Raster sind in Abb. 6 zu sehen. Das 1×1-Raster dient zur Simulation der korrekten Funktionsweise eines einzelnen Photovoltaik-Paneels auf dem Dach eines Hauses. Die Anlage soll nur bei hoher Einstrahlung Strom erzeugen. Bei Sturm muss das Paneel automatisch in eine Schutzstellung gebracht werden. Das 2×2-Raster hilft, eine Klimaanlage für eine Sporthalle zu testen. Nur wenn die Außentemperatur an einer Hallenwand geeignet ist, die Halle auf eine Innentemperatur im Bereich $[19, 22]°C$ herab- oder herauf zu temperieren, soll die Fensterreihe an der Hallenseite automatisch geöffnet werden [7]. Das 3×3-Raster dient zum Testen einer Kläranlage, deren Wirkungsgrad gesenkt wird, wenn sich zu viel Regen- mit dem Schmutzwasser innerhalb eines bestimmten Stadtteils vermischt [1].

Tabelle 1: Physikalische Größen und Einzelereignisse des Wetters

Größe	Einheit	Messgerät	Durchschnitt	Einzelereignisse
Windstärke	km/h	Anemometer	$14\,km/h$	Bö ($[18, 400]\,km/h$)
Einstrahlung	W/m^2	Photometer	$165\,W/m^2$	
Lichtstärke	cd	Luxmeter		Sonneneruption (10 MeV)
Temperatur	$°C$	Thermometer	$9°C$	
Niederschlag	mm/h	Pyranometer	$700\,mm/h$	Sturzregen

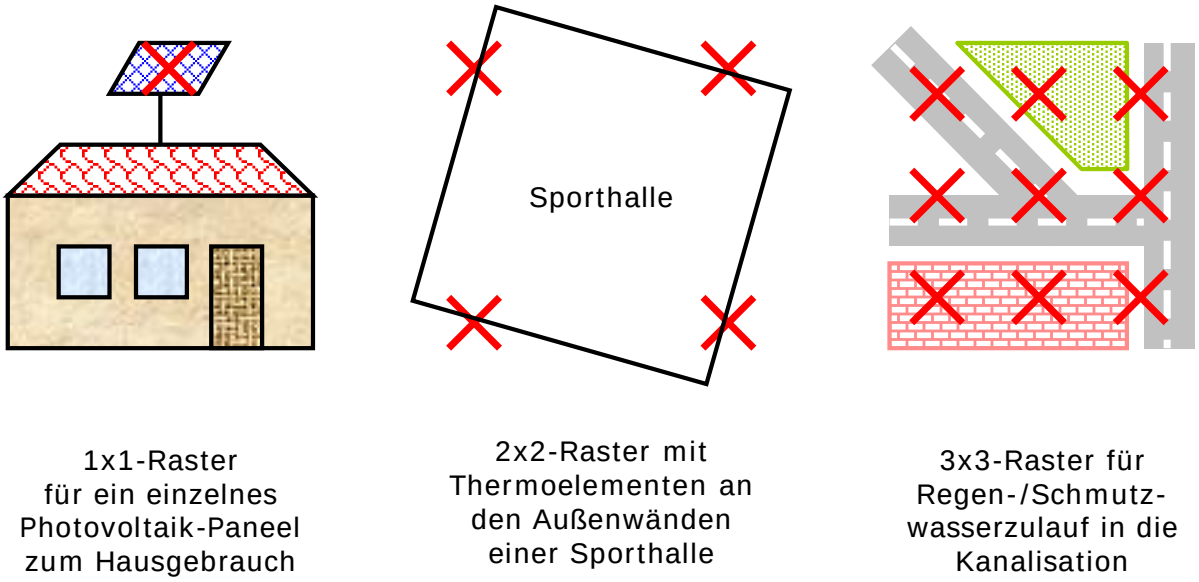


Abbildung 6: Verschiedene Anwendungsbeispiele mit Rastergrößen

2.4 Simulation der Umweltbedingungen

Innerhalb der Methode `Simulation.NaechsterZeitschritt(Zeitschritt DUR)` findet die eigentliche Simulation statt. Diese läuft nach dem Prinzip der finiten Differenzen ab, was für das Beispiel einer Wolke innerhalb von `WetterSimulation` in Abb. 7 angegeben ist. Weitere Beispiele für finite Differenzen und einen Simulationsbaukasten finden sich in [8]. Für einen zu simulierenden Zeitraum von einigen Jahren, wie dies bei Ertragsrechnungen der Fall ist, werden als Zeitschritte volle Stunden benutzt. Für PEARL2020 ist daher zu überlegen, eine Möglichkeit anzubieten, Standardwerte (hier: `1 HRS`) für Übergabevariablen festzulegen.

Angabe von Standardwerten für Eingabeparameter:
Es ist zu untersuchen, ob die optionale Angabe von Standardwerten für Ãœbergabeparameter von Prozeduren die funktionale Sicherheit fördert. Mit dem Standardwert `0` bzw. `NIL` bei der indirekten Initialisierung modulglobaler Variablen wird bspw. Korrektheit beim lesenden Zugriff ohne vorherige explizite Initialisierung suggeriert, was zu Fehlern führen kann, wenn ein Wert von `0` bzw. `NIL` unangemessen ist.

Innerhalb von `EnergieSimulation` wird für die zu ermittelnde Netzeinspeisung einer Photovoltaik-Anlage eine akkumulative Strategie eingesetzt, siehe Abb. 8.

```
DCL Wirkungsgrad INV FLOAT INIT(0.12);
MomentaneLeistung := Einstrahlung * Wirkungsgrad * Flaeche;
DeltaEnergie := MomentaneLeistung * Zeitschritt;
EnergieEinspeisung := EnergieEinspeisung + DeltaEnergie;
```

Abbildung 8: Akkumulative Berechnung der Energieeinspeisung

```
! Aktuelle Umwelt-Parameter
DCL Bewoelkung(Zeilen, Spalten) FLOAT;
DCL Windstaerke(Zeilen, Spalten) Punkt;
DCL Position Punkt;

! UmweltParameter fuer den naechsten Zeitschritt
DCL BewoelkungNeu(Zeilen, Spalten) FLOAT;
DCL PositionNeu Punkt;

! BewoelkungNeu ist ein Akkumulator-Array, welches am
! Anfang jedes neuen Schrittes auf Null gesetzt wird
FOR i FROM 1 TO Zeilen REPEAT
  FOR j FROM 1 TO Spalten REPEAT
    BewoelkungNeu(i, j) := 0;
  END;
END;

! Bewegung der Bewoelkung fuer einen Zeitschritt
FOR i FROM 1 TO Zeilen REPEAT
  FOR j FROM 1 TO Spalten REPEAT
    ! Neue Position der Teilwolke von Rasterpunkt i, j
    AktuellePos.X := i * Hoehe / Zeilen;
    AktuellePos.Y := j * Breite / Spalten;
    PositionNeu := Zeitschritt * Position * Windstaerke;
    ! Berechne die vier Nachbarn N1, N2, N3, N4
    ! von PositionNeu im Raster
    ...
    ! Berechne die Interpolationsgewichte G1, G2, G3, G4
    ! fuer die 4 Nachbarn
    ...
    ! Teile die Teilwolke auf ihre neue Position im Raster auf
    BewoelkungNeu(N1.X, N1.Y) :=
      BewoelkungNeu(N1.X, N1.Y) + G1 * Bewoelkung(i, j);
    ...
  END;
END;

! Transferiere BewoelkungNeu nach Bewoelkung ...
```

Abbildung 7: Berechnung der Bewölkung nach dem Prinzip der finiten Differenzen

2.5 Zusammenfassung und Ausblick

Wir haben die Architektur eines Benchmarks vorgestellt, der Umgebungsbedingungen für umwelttechnische Automatisierungssysteme simuliert, um diese vor dem Bau von Prototypen zu testen und vor der Durchführung teurer Feldversuche zu validieren. Dabei haben wir das Anwendungsbeispiel auch dazu genutzt, um verbesserungsbedürftige Eigenschaften von PEARL90 aufzudecken und Abhilfen in PEARL2020 vorzuschlagen. Kritik und weitere Vorschläge dazu nehmen wir unter christina.houben@fernuni-hagen.de gerne entgegen.

Momentan ist der Benchmark über eine große Anzahl an Gettern und Settern der Schnittstelle `Simulator` zugänglich. Eine Verbesserung wäre, statt der Getter und Setter einen Treiber einzubinden, der `BASIC`-Datenstationen zur Ein- und Ausgabe der Umweltparameter bereitstellt.

Literatur

[1] B. Philipp: *Einführung in die Umwelttechnik*, Aufl. 2, Vieweg, 1994

[2] G. Häberle et al.: *Fachwissen Umwelttechnik*, Aufl. 5, Europa-Lehrmittel, 2011

[3] C.K. Houben: Fostering Functional Safety of Real-time Systems with Concepts of Object Orientation, Proc. *19th Int. Conf. on Methods and Models in Automation and Robotics* (MMAR), 2014

[4] A.H. Frigeri, C.E. Pereira und W.A. Halang: An Object-Oriented Extension to PEARL90, Proc. *Int. Symp. on Object-Oriented Real-time Distributed Comp.* (ISORC), S. 265–274, 1998

[5] I. Reda und A. Andreas: Solar Position Algorithm for Solar Radiation Applications, NREL Report TP-560-34302, `http://rredc.nrel.gov/solar/codesandalgorithms/spa/`, 2008

[6] A.G. Stephenson et al.: Mars Climate Orbiter Mishap Investigation Board Phase I Report, NASA, 1999

[7] G. Thiele: *Software-Entwurf in PEARL-orientierter Form*, Teubner, 1993

[8] C.K. Houben: APSIS – Application for Simulation-Studies, Handbuch, Jülich Supercomputing Center, 2007

Bildnachweis

Erde in Abb. 1: iStockphoto aus dem PM-Artikel *Die Erde nimmt an Umfang zu*
Wasser in Abb. 2: Bild von `paradigma.de`
Tropfen in Abb. 2: Bild von `ds-dan.de` mit Nachbearbeitung der Farbe
Flocke in Abb. 2: Bild von `helpster.de` aus dem Artikel *Schneeflocken-Kostüm*

3 Echtzeitinformation für den öffentlichen Nahverkehr

K. Barghorn, J. Benra, H.-H. Blikslager, O. Fischer, L. Oelschläger, E. Schmoll und U. Willers
Jadehochschule Wilhelmshaven

3.1 Motivation

Die verstärkte Nutzung des öffentlichen Nahverkehrs wird allgemein als ein wesentlicher Beitrag zu Klimaschutz und sozialer Teilhabe angesehen. Letzteres betrifft meistens sehr junge (insbesondere Schüler) und ältere Verkehrsteilnehmer (die nicht oder nicht mehr selbst fahren können).

In urbanen Regionen ist der öffentliche Nahverkehr häufig sehr gut ausgebaut, anders sieht es dagegen in weniger dicht besiedelten Gebieten aus. Hier müssen oft Kosten und Nutzen gegeneinander abgewogen werden, was häufig zu einem minder attraktiven Angebot führt, obwohl allgemein bekannt ist, dass gerade eine gute Infrastruktur, zu der auch der öffentliche Nahverkehr gehört, die Attraktivität einer Region auch für die Industrie steigert.

Ein von der europäischen Union gefördertes Interreg IVb Projekt widmet sich dieser Problemstellung. Das Projekt ITRACT „Improving Transport and Accessibility through new Communication Technologies“ beschäftigt sich in 5 europäischen Ländern der Problematik unter verschiedenen Gesichtspunkten. Deutscher Partner in dem Projekt ist die Jadehochschule (Studienort Wilhelmshaven) mit ihrem Subpartner VEJ (Verkehrsverbund Ems-Jade).

In diesem Artikel wird ein Teilprojekt behandelt, das sich der Aufgabe stellt, die Echtzeitinformationen über den aktuellen Standort von Bussen auf kostengünstige Art und Weise der regionalen Wirtschaft zur Verfügung zu stellen.

3.2 Ausgangssituation

Momentan ist es für die Unternehmen, die in der Region Ems-Jade den Busverkehr im Wesentlichen zur Verfügung stellen, in vielen Fällen nicht möglich, teure Standardlösungen zum Mitkoppeln von Echtzeitinformationen zu bezahlen. Ebenso ist es von der Kostenseite häufig nicht möglich, die Echtzeitinformationen – sollten sie zur Verfügung stehen – ebenfalls kostengünstig zu verteilen, so sind z.B. kommerzielle Anzeigesysteme, wie sie in größeren Städten standardmäßig eingesetzt werden, häufig zu teuer.

3.3 Methodik des Projekts

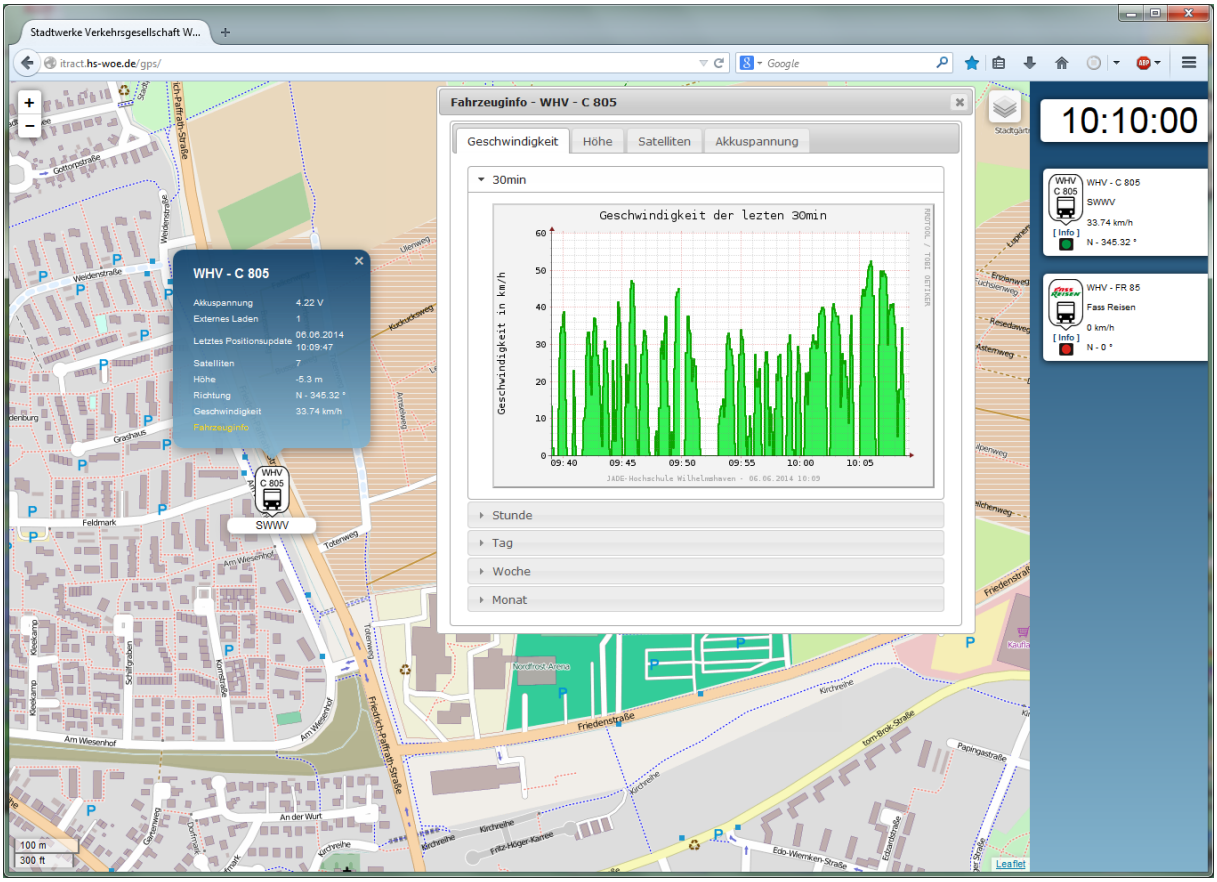
Zu Beginn des Projektes wurden sogenannte Innovation Workshops in allen Regionen des Projektes zusammen mit den Verkehrsbetrieben, Fahrgastvereinigungen etc. durchgeführt. Daraus ist eine Sammlung an möglichen Ideen entstanden, wie der öffentliche Nahverkehr verbessert und gestärkt werden könnte. Diese Möglichkeiten wurden auf Machbarkeit hin untersucht und es erfolgte eine Selektion realisierbarer Ideen, die dann durch Studierende im Rahmen von Projekten, Praxissemestern und Bachelorarbeiten bearbeitet wurden. Außerdem erfolgte ein Austausch zwischen den verschiedenen Regionen hinsichtlich einer eventuellen Anwendbarkeit in anderen Regionen. Dazu wurden verschiedene Grundbausteine, s.g. Building Blocks vereinbart, die jeweils Kernlösungen enthalten und sich leicht an andere Regionen anpassen lassen. So ist z.B. eine Applikation aus Wilhelmshaven mittlerweile auch im norwegischen Stavanger im Einsatz.

Nach einer Inbetriebnahme in den einzelnen Verkehrsunternehmen wird der Testbetrieb wissenschaftlich begleitet. Ebenso ist es ein Teil des Projektes, die verschiedenen Benutzergruppen bei der Nutzung der neuen Möglichkeiten zu unterstützen. Unter dem Stichwort „User Empowerment“ kann dies z.B. die Heranführung älterer Personen an die Vorteile der Nutzung von Smartphones, Tablets etc. sein. Nach Auswertung der Testphase ist geplant, auch die Möglichkeiten einer Verstetigung der gefundenen Lösungsansätze zu prüfen, um dauerhaft von den Ergebnissen des Projektes zu profitieren. Dies könnte beispielsweise durch die Bereitstellung von Softwarekomponenten durch überregionale Verkehrsträger geschehen.

3.4 Lösungsansätze

Ein Teil des Projektes widmet sich der Bereitstellung von Echtzeitinformationen der im Einsatz befindlichen Busse. Hierbei wird mit Hilfe des GPS (Global Positioning System) die Position des Busses erfasst und über eine Mobilfunkverbindung (GSM) an die Einsatzzentrale übermittelt. Dazu ist neben dem Einsatz kostengünstiger, kommerzieller GPS-Tracker auch die Nutzung von

Fahrgastinformationssystemen im Bus vorgesehen. Diese besitzen bereits ein GSM-Modul und stellen im Wechsel die jeweils nächsten Haltestellen als Perlschnuranzeige sowie positionsbezogene Werbung dar. Dadurch entfallen Kosten für die Datenverbindung in Höhe von circa 10 € pro Bus und Monat. Ein weiterer Vorteil ist, dass die gerade gefahrene Liniennummer vom Kassenrechner abgefragt werden kann und somit ein aufwändiges Verfahren zur Ermittlung der richtigen Buslinie aus den Positionsdaten entfällt. Die Linieninformation wird zwischen den einzelnen Komponenten (in diesen Fall Kassenrechner und Perlschnuranzeige) im standardisierten VDV 423 Protokoll übertragen und lässt sich somit herstellerunabhängig auslesen.



Aus den Positionsdaten kann dann eine dynamische Fahrplanauskunft erstellt werden, welche plattformunabhängig als HTML5 Seite auf Smartphones und Tablets angezeigt werden kann. Auch lassen sich dort kurzfristige Informationen wie beispielsweise Ausfälle aufgrund von Bauarbeiten oder Umleitungen anzeigen. All diese Informationen lassen sich browserbasiert an beliebigen Internetzugängen eingeben und verwalten. Sie werden in einer zentralen Datenbank gespeichert und der Öffentlichkeit auf einer OpenStreetMap basierten Karte übersichtlich präsentiert. Als Datenformat wurde der weltweite de-facto Standard für den Austausch von Fahrplaninformationen GTFS (General Transit Feed Specification) gewählt. Dieser verfügt in seiner Erweiterung GTFS-RT zudem über die Möglichkeit, Fahrplandaten in Echtzeit zu aktualisieren. Das GTFS-Format erlaubt es außerdem, auf eine große Zahl innovativer OpenSource Softwarekomponenten und Apps aus aller Welt zuzugreifen. Nachdem die Konvertierung der in einem proprietären Format vorliegenden Fahrplandaten zunächst im Projekt durchgeführt wurde, entwickeln sich mittlerweile Tendenzen, dieses Format auch von Seiten der überregionalen Verkehrsträger direkt bereitzustellen. Im Rahmen eines weiteren ITRACT-Teilprojekts wird derzeit ein Openstack Cloud-System entwickelt, welches auf der Basis von GTFS-Daten Software Services bereitstellt.

Diese Informationen lassen sich ebenfalls mit kostengünstigen HDMI-Monitoren und Kleinstrechnern (z.B. Raspberry Pi) stationär an Umsteigehaltestellen anzeigen.

Für das Realtime-Tracking wurden zunächst sehr preiswerte, handelsübliche GPS-Module verwendet, die jedoch ein geschlossenes System darstellen. Die Systemsoftware lässt sich nicht verändern. Aus diesem Grund wurde eine eigene, offene GPS-Tracking-Plattform entwickelt. Die erste Version dieses Gerätes besteht aus einem GPS-Receiver, einem Mikrocontroller und einem 3G-Funkmodul. Durch das Funkmodul werden die gesammelten Daten z.B. via UMTS an den Server gesendet. Für die Programmierung wurde die populäre „Arduino“ Open-Source-Plattform benutzt. Hierfür sind unterschiedliche Hardwaremodule preiswert verfügbar. Diese Baugruppen ließen sich für die Tracker Anwendung entsprechend kombinieren und durch eigene Hardware ergänzen. Bei dem so entwickelten Gerät sind nun alle Schnittstellen zugänglich. Auf der Basis dieser Hardware-Plattform konnte eine Software entwickelt werden, die den gestellten Anforderungen gerecht wird und bei neuen Gesichtspunkten entsprechend angepasst werden kann. Das Gerät wird zwar an die existierenden Ticket-Systeme im Bus angeschlossen, arbeitet jedoch unabhängig davon. Eine weitere Evolutionsstufe nutzt den vorhandenen Datenverkehr im Fahrzeug als passiver „Lauscher“ auf dem Datenbus.

Testbetrieb: Zurzeit läuft ein Testbetrieb mit Bussen, die mit dem Realzeit-Tracking System bestückt sind sowie ein Test mit der Monitoranzeige (auf dem Hochschulgelände der Jadehochschule). Bisher sehen die Ergebnisse ermutigend aus, so dass wahrscheinlich mittelfristig eine Bestückung der Busse der Region mit der neuen Technik erfolgen wird. Der Test wird ebenso die Benutzerfreundlichkeit der entstandenen Prototypen untersuchen. Für die Bewertung der Benutzerfreundlichkeit wurde eine Checkliste erarbeitet, die den Prototypen hinsichtlich der Effizienz, Effektivität, Aufgabenangemessenheit, Kontrollierbarkeit, Konsistenz, Fehlertoleranz, Individualisierbarkeit, Erlernbarkeit und Ästhetik betrachtet. Dabei kommen neben der Auswertung von Fragebögen auch der Einsatz weiterer professioneller wissenschaftlicher Methoden wie beispielsweise eine Eye-Tracking Studie zum Einsatz.

3.5 Ausblick

Nach Durchführung des Testbetriebs werden Workshops durchgeführt, die die Potenziale der Testprotoypen auf eine echte wirtschaftliche Nutzung abschätzen sollen. Danach werden sich die Verkehrsunternehmen der Region mit den Erkenntnissen auseinandersetzen.

weitere Informationen:
homepage http://www.jade-hs.de/ITRACT
tracking http://itract.hs-woe.de/gps/
facebook http://www.facebook.com/ItractDeutschland

Literatur:
VDV Schrift 423 Digitaler Standard für Telekommunikation im ÖPNV,
GTFS Spezifikation http://developers.google.com/transit/gtfs

4 Programm der Fachtagung Echtzeit 2014

Jutta Düring, Fachausschuss Echtzeitsysteme

Die Fachtagung „Echtzeit“ findet am 20./21. November 2014 wie gewohnt in Boppard am Rhein statt. Die Beiträge zum Leitthema „**Industrie 4.0 und Echtzeit**“ befassen sich mit Zeitsynchronisation, Weiterentwicklung von PEARL, Simulation, Plattformen und aktuellen Anwendungen. Die studentischen Beiträge sind dieses Jahr thematisch in das Programm eingebunden.

Die Tagung findet im üblichen Rahmen im Hotel Ebertor statt. Der Tagungsband erscheint ebenfalls wieder in der Reihe „Informatik aktuell“ des renommierten Springer-Verlages. In Anbetracht des interessanten Tagungsprogrammes und der immer anregend-freundlichen Atmosphäre in Boppard sollten Sie sich ab September unter `http://www.real-time.de/workshop.html` zur Teilnahme anmelden.

Erster Workshop-Tag: Donnerstag, der 20. November 2014

11:00 Uhr Treffen der Arbeitskreise

11:30 Uhr Mittagsimbiss (optional)

13:00 Uhr Begrüßung

13:15 Uhr Sitzung 1: *Zeitsynchronisation*

Zeitsynchronisation von Echtzeitmessungen verschiedener Signalquellen für HiL-Testverfahren
Spiteller, Trenkel (iSyst Intelligente Systeme GmbH)

Plug-and-Work für verteilte Echtzeitsysteme mit Zeitsynchronisation
Schriegel et al. (Fraunhofer IOSB-INA)

14:15 Uhr Pause

15:00 Uhr Sitzung 2: *Weiterentwicklung von PEARL*

Eine sicherheitsgerichtete Echtzeitprogrammiersprache für die Sicherheitsstufen SIL 3 und SIL 4 gemäß DIN EN 61508
Hillebrand (Universität Stuttgart/FernUniversität in Hagen)

Die Programmierumgebung OpenPEARL90
Müller, Schaible (Hochschule Furtwangen, FernUniversität in Hagen)

Konzeption und prototypische Umsetzung des E/A-Systems für einen PEARL-Compiler
Kölle (Hochschule Furtwangen)

16:30 Uhr Pause

17:00 Uhr Sitzung 3: *Simulation*

Sensorsimulation in HiL-Anwendungen
Trenkel, Spiteller (iSyst Intelligente Systeme GmbH)

Übersetzung von UML-Software-Spezifikationen in Simulationsmodelle
Walter (FernUniversität in Hagen)

18:00 Uhr Preisverleihung

18:15 Uhr Abendessen

20:00 Uhr Mitgliederversammlung des Fachausschusses

Zweiter Workshop-Tag: Freitag, der 21. November 2014

9:00 Uhr Sitzung 4: *Plattformen*

Der Raspberry Pi als Plattform für Fluoreszenzmessungen unter Echtzeitbedingungen
Lorenz et al. (HTW Dresden, Fraunhofer IWS, SBU Schirmer + Dr. Berthold Umwelttechnik GmbH)

Laufzeitvalidierung einer Plattform zur semantischen Integration von Feldgerätedaten
Reboredo (Bosch Rexroth AG)

Koprozessorgestützte Sicherheitsbedrohungen in Multi-OS-Systemen
Pöselt (Hochschule Darmstadt)

10:30 Uhr Pause

11:00 Uhr Sitzung 5: *Aktuelle Anwendungen*

Verwendungsfähigkeit von Android-CE-Geräten für CAR2X-Anwendungen am Beispiel einer Geschwindigkeitsregelung
Hotter (Hochschule Landshut)

Mobile Echtzeitkontrolle von Kommunikationskanälen
Kubek, Unger (FernUniversität in Hagen)

Kollaborative Fertigung mittels eines Multiagentensystems zur Vernetzung anlagenspezifischer Echtzeitsysteme
Regulin et al. (TU München)

12:30 Uhr Verabschiedung

12:45 Uhr Mittagsimbiss (optional)

5 Preisträger des Graduiertenwettbewerbs 2014

Jutta Düring, Fachausschuss Echtzeitsysteme

Die Gewinner des diesjährigen Graduiertenwettbewerbs sind:

Jürgen Hillebrand Definition einer sicherheitsgerichteten Programmiersprache (Masterarbeit FernUniversität in Hagen)

Holger Kölle Konzeption und prototypische Umsetzung eines I/O-Systems für einen PEARL-Compiler (Masterarbeit Hochschule Furtwangen)

Hermann Ludwig Lorenz Evaluationsplattform für Fluoreszenzmessungen (Masterarbeit Hochschule für Technik und Wirtschaft Dresden)

Der Fachausschuss gratuliert den Preisträgern und freut sich auf die Vorstellung der Arbeiten auf der Tagung.

6 Berichte aus den Arbeitskreisen

Jutta Düring, Fachausschuss Echtzeitsysteme

Der Fachausschuss hat zur Zeit drei aktive Arbeitskreise, welche sich über Mithilfe oder einen regen Gedankenaustausch freuen würden. Bitte wenden Sie sich bei Interesse an die jeweilige Kontaktperson.

6.1 Arbeitskreis Sprachpflege/Normung

Ziel des Arbeitskreises ist die Neufassung der DIN 66253-2 und DIN 66253-3 insbesondere bezüglich funktionaler Sicherheit. Es wurde eine Förderung durch das BMWi im Programm „Transfer von Forschungs- und Entwicklungsergebnissen durch Normung und Standardisierung“ beantragt. Projektleiterin ist Frau Christina K. Houben, wissenschaftliche Mitarbeiterin der FernUniversität in Hagen. Prof. Dr. Dr. Wolfgang A. Halang wurde zum Obmann des zuständigen DIN-Ausschusses NA043-01-22 AA gewählt. Finanziell wird der Ausschuss unterstützt von der Firma Think-Cell.

Kontakt: christina.houben@fernuni-hagen.de

6.2 Arbeitskreis PEARL-Compiler

Ziel des Arbeitskreises ist die Entwicklung eines offenen PEARL-Compilers unter Linux. Für den Übersetzer zeigt sich Marcel Schaible verantwortlich; für das Laufzeitsystem Prof. Rainer Müller. Es wird dringend Mitarbeit erwünscht bei der Erstellung der Projektdokumentation als Wiki sowie beim Compiler (Umsetzung weiterer PEARL-Sprachkonstrukte, Fehlerbehandlung).

Quelle: `http://sourceforge.net/projects/smallpearl/`
Der Compiler smallpearl wird zukünftig OpenPEARL heißen.

Kontakt: marcel.schaible@fernuni-hagen.de, rainer.mueller@hs-furtwangen.de

6.3 Arbeitskreis Echtzeitkommunikation

Die Interessen des Arbeitskreises liegen bei: Kommunikation in autonomen Systemen, Selbstorganisation, Echtzeitoptimierung von Netzen, Echtzeitnutzerbewertung, Informationsausbreitung und Nutzerbewertung in Sozialen Netzen. Einmal jährlich im Oktober findet für einen eingeladenen Kreis der Workshop „Autonome Systeme“ auf Mallorca statt. Das Doktorandenseminar wird jährlich in Dagstuhl angeboten. Ein BMBF-Antrag „Ambient Room“ ist in Vorbereitung.

Kontakt: herwig.unger@fernuni-hagen.de

6.4 Arbeitskreis Embedded Systems/RTOS-UH-PEARL

Der Arbeitskreis Embedded Systems/RTOS-UH-PEARL ist zur Zeit ruhend gesetzt, da der Nachfolger von Prof. Gerth den Schwerpunkt des Lehrstuhls auf Robotik ausgerichtet hat.