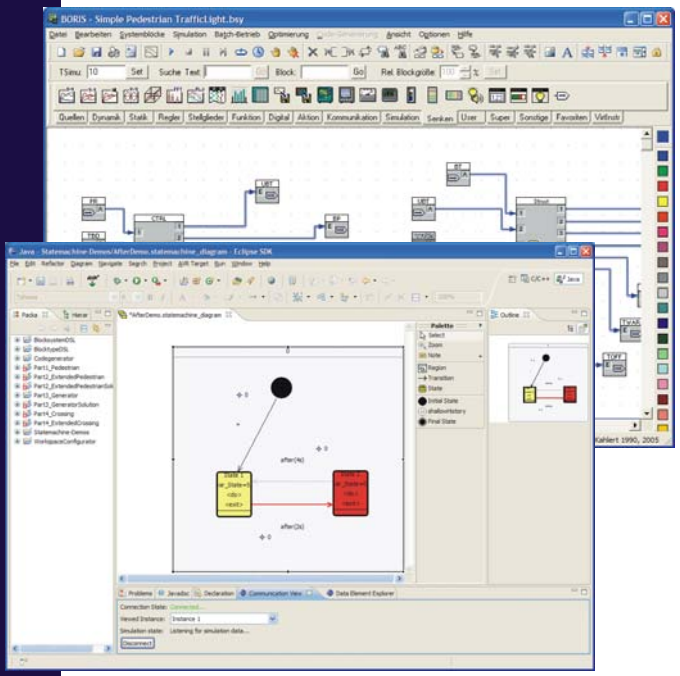
WinFACT



WinFACT

State Machine Workbench

Vorläufiges Benutzerhandbuch

Ingenieurbüro Dr. Kahlert

Ludwig-Erhard-Str. 45
D-59065 Hamm

WinFACT

WINDOWS FUZZY AND CONTROL TOOLS

VORLÄUFIGES
BENUTZERHANDBUCH

STATE MACHINE WORKBENCH

Release 0.7

Ingenieurbüro Dr. Kahlert
Ludwig-Erhard-Str. 45 D-59065 Hamm
Tel. 0 23 81/926 996 Fax 0 23 81/926 997

Inhalt

1 Einführung

1.1 Leistungsumfang

Die *State Machine Workbench* von WinFACT erlaubt den Entwurf, die Simulation sowie die Codegenerierung (optional) von Zustandsautomaten. Diese können innerhalb der blockorientierten Simulationsumgebung BORIS in beliebige Systemstrukturen eingebunden werden, sodass hybride Systeme aus kontinuierlichen und ereignisdiskreten Systemblöcken verarbeitet werden können. Ein Zustandsautomat kann aus folgenden Elementen bestehen:

- Initial State (Anfangszustand)
- Final State (Endzustand)
- State (Zustand)
- Region
- Transition (Zustandsübergang)
- Shallow history
- Deep history
- Choice (Verzweigung)
- Event (Ereignis)
- Variable

Zum Entwurf des Zustandsautomaten steht ein komfortabler grafischer Editor zur Verfügung, der in die freie Entwicklungsumgebung ECLIPSE [ECL] integriert ist. Innerhalb von BORIS wird der Zustandsautomat dann durch einen entsprechenden Systemblock abgebildet, der wie ein „gewöhnlicher“ Systemblock in beliebige Systemstrukturen eingebunden werden kann. Während der Simulation kann auf Wunsch eine Kommunikation zwischen BORIS und dem

grafischen Editor aktiviert werden, die durch Highlighting aktiver Zustände und Transitionen einen Einblick in das „Innenleben“ des Zustandsautomaten ermöglicht und damit beispielsweise ein komfortables Debugging ermöglicht. Selbstverständlich kann optional auch eine Generierung von C-Code für die Gesamtstruktur oder auch nur den Zustandsautomaten erfolgen.

> In der **Demo-Version** der *State Machine Workbench* ist die Laufzeit des Zustandsautomaten-Blocks unter BORIS auf 30 Sekunden begrenzt; beim Starten einer Simulation erscheint ein entsprechendes Hinweisfenster. Enthält eine Simulationsstruktur mehrere Zustandsautomaten, so erscheint auch dieses Hinweisfenster entsprechend mehrfach.

Die *State Machine Workbench* ist für Lehre, Forschung und Industrie in verschiedenen Versionen und Lizenztypen verfügbar. Einzelheiten dazu finden Sie im Internet unter *www.winfact.de*.

Die Entwicklung der *State Machine Workbench* wurde im Rahmen des *Zukunftswettbewerbs Ruhrgebiet* vom Land NRW finanziell unterstützt[*].

1.2 Installation

Die *State Machine Workbench* setzt sich aus zwei Paketen zusammen, die nacheinander installiert werden müssen:

- dem Statechart-Editor
- dem *Statemachine-Add-On* für das blockorientierte Simulationssystem BORIS

Die Installation beider Pakete setzt eine bestehende Installation des Programmpakets WinFACT 7 voraus; sollten Sie dieses noch nicht installiert haben, so holen Sie dies zunächst nach, bevor Sie fortfahren. Da der Statechart-Editor mindestens eine JAVA-Umgebung der Version 6 benötigt, muss gegebenenfalls auch eine aktualisierte JAVA-Installation erfolgen.

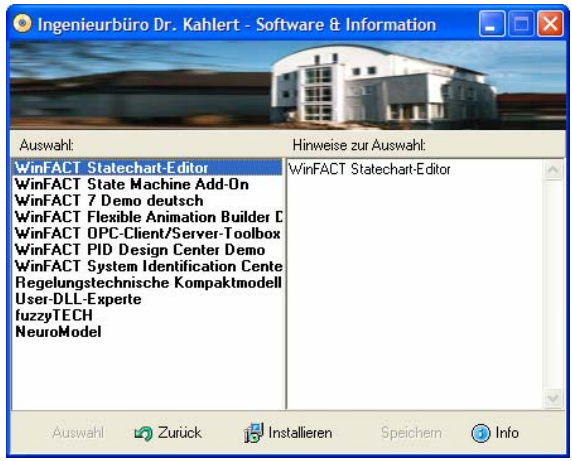
1.2.1 Installation und Konfigurieren des Statechart-Editors

Zur Installation des Statechart-Editors gehen Sie wie folgt vor:

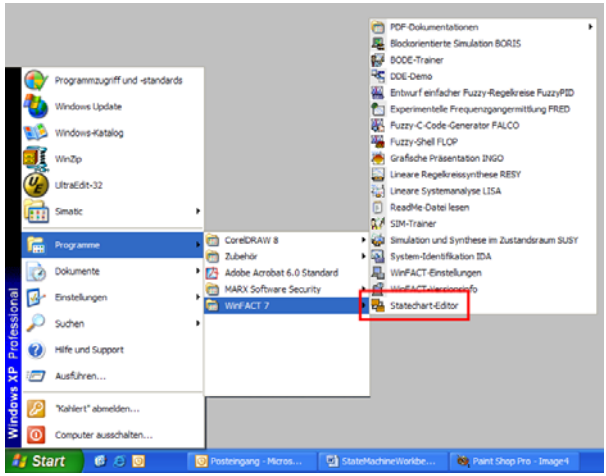
1. Legen Sie die Installations-CD in Ihr CD-Laufwerk ein. Das Installationsprogramm startet nach kurzer Zeit automatisch.

[*] Forschungsprojekt *Modellbasierte generative Softwareentwicklung für Eingebettete Systeme*, kurz *MDA4E*, 2006-2008 [MDA]

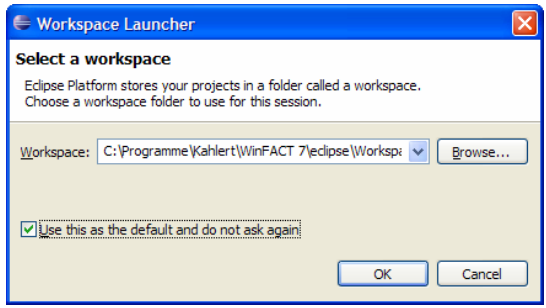
2. Wählen Sie aus der zu installierenden Software das Paket *WinFACT Statechart-Editor* und betätigen Sie die Schaltfläche *Installieren*:



3. Folgen Sie den Anweisungen des Installationsprogramms. Es ist dabei dringend zu empfehlen, den Statechart-Editor in dasselbe Verzeichnis zu installieren, in dem sich auch Ihre WinFACT 7-Installation befindet (standardmäßig *c:\programme\kahlert\winfact 7*). Nach Beendigung der Installation finden Sie den Statechart-Editor in Ihrer WinFACT 7-Programmgruppe:

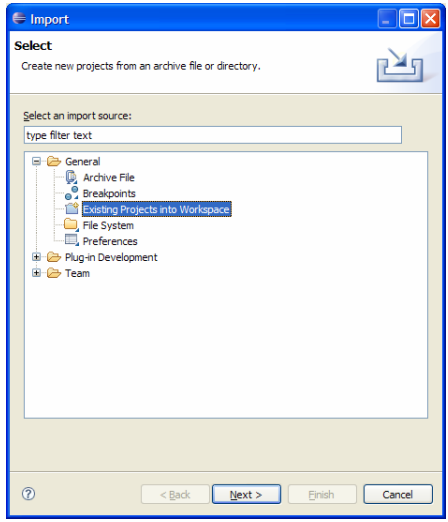


4. Starten Sie den Statechart-Editor. Nach Verschwinden des ECLIPSE-Logos werden Sie nun zunächst aufgefordert, einen sogenannten *Workspace* (Arbeitsbereich) für Ihre Projekte anzugeben:

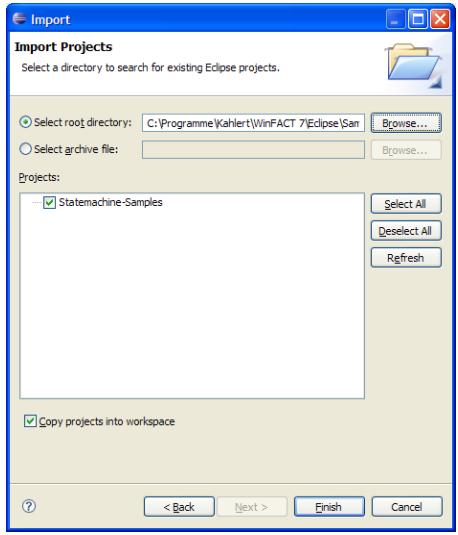


   Über die *Browse*...-Schaltfläche können Sie hier prinzipiell ein beliebiges Verzeichnis auswählen; empfehlenswert ist jedoch wie in obiger Bildschirmgrafik gezeigt das Unterverzeichnis *\ECLIPSE\Workspace* Ihrer WinFACT 7-Installation, da in diesem Fall das Erscheinungsbild des Editors automatisch vorkonfiguriert wird. Durch Aktivierung des Optionsfeldes *Use this as the default and do not ask again* sorgen Sie dafür, dass das ausgewählte Verzeichnis bei allen nachfolgenden Aufrufen des Editors als voreingestellter Workspace benutzt wird. Wollen Sie später gegebenenfalls einmal den Workspace wechseln (was in der Regel aber nicht sinnvoll ist), so können Sie dies über die Menüoption FILE ▸ SWITCH WORKSPACE erreichen.

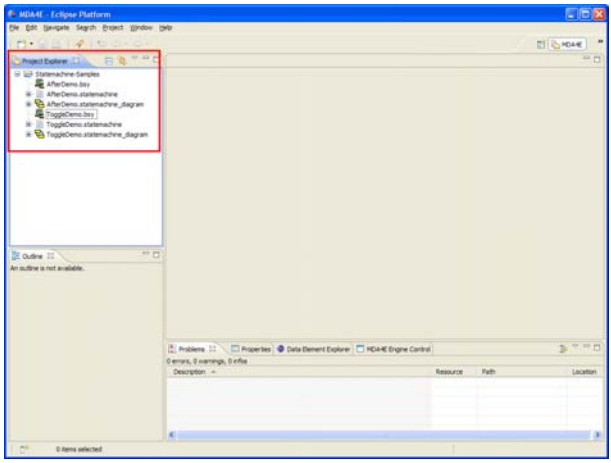
5. Abschließend müssen nun noch die mitgelieferten Beispiele in den aktiven Workspace kopiert werden. Wählen Sie dazu im Editor die Menüfolge FILE ▸ IMPORT... und im daraufhin erscheinenden Dialog die Option *Existing Projects into Workspace*:



6. Fahren Sie mit der Schaltfläche *Next* > fort und wählen Sie im nachfolgenden Dialog das Unterverzeichnis *\ECLIPSE\Samples* Ihrer WinFACT 7-Installation aus; dieses enthält alle Beispiele in einem Projekt namens *Statemachine-Samples*:



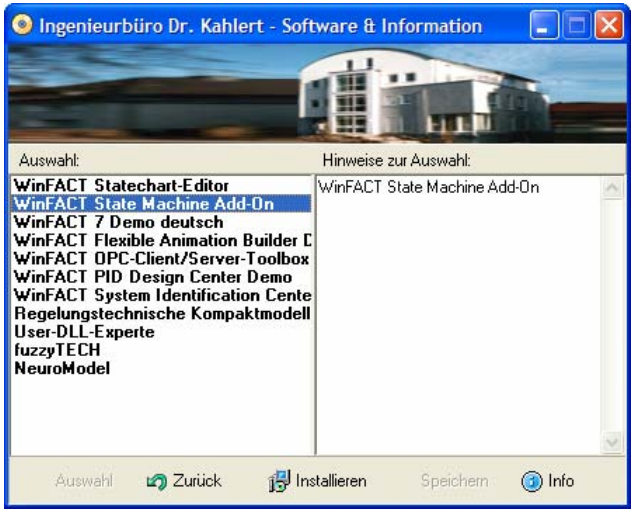
7. Kehren Sie über die Schaltfläche *Finish* wieder ins Programm-Hauptfenster zurück. Im *Project Explorer* sollten Sie nun das importierte Projekt wiederfinden. Über das Plus-Symbol können Sie sich alle zum Projekt gehörenden Dateien anzeigen lassen:



1.2.2 Installation des Statemachine-Add-Ons für BORIS

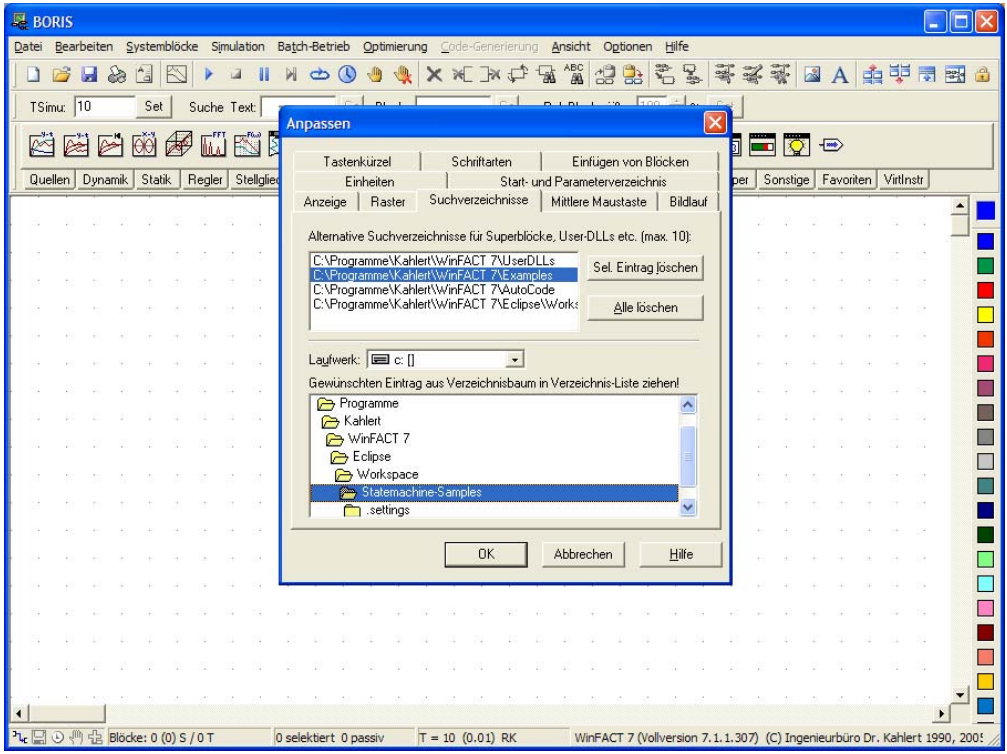
Zur Installation des Statemachine-Add-Ons gehen Sie wie folgt vor:

1. Legen Sie die Installations-CD in Ihr CD-Laufwerk ein. Das Installationsprogramm startet nach kurzer Zeit automatisch.
2. Wählen Sie aus der zu installierenden Software das Paket *WinFACT State Machine Add-On* und betätigen Sie die Schaltfläche *Installieren*:



3. Folgen Sie den Anweisungen des Installationsprogramms. Achten Sie bei der Installation insbesondere darauf, als Installationsverzeichnis das Verzeichnis Ihrer WinFACT 7-Installation anzugeben, da nur dann ein einwandfreies Zusammenspiel aller Komponenten sichergestellt ist.

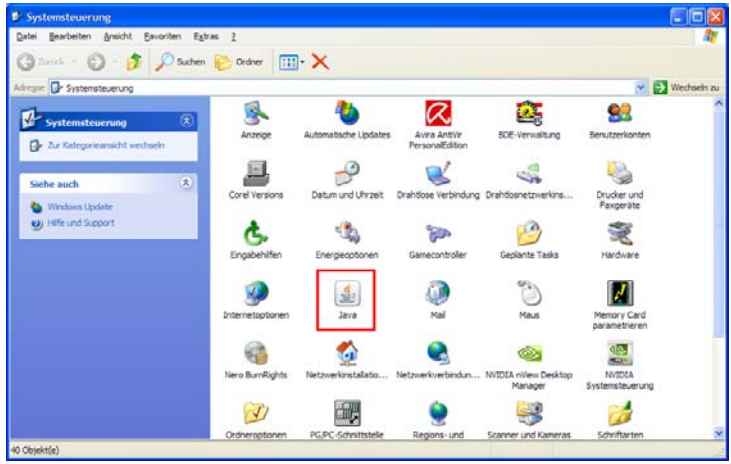
4. Damit alle mitgelieferten Beispiele problemlos laufen, muss abschließend noch das Unterverzeichnis *\Statemachine-Samples* (Beispielverzeichnis) des bei der Installation des Statechart-Editors (siehe Abschnitt 1.2.1) angegebenen Workspaces in die Liste der BORIS-Suchverzeichnisse aufgenommen werden. Starten Sie dazu BORIS aus Ihrer WinFACT 7-Programmgruppe und rufen Sie über OPTIONEN ▸ ANPASSEN... den Dialog zur Verwaltung der Suchverzeichnisse auf. Fügen Sie dann wie in nachfolgender Bildschirmgrafik gezeigt das Beispielverzeichnis (standardmäßig *c:\programme\kahlert\winfact 7\eclipse\workspace\statemachine-samples*) per Drag&Drop in die Liste der Suchverzeichnisse ein. Beenden Sie BORIS danach, bevor Sie das erste Mal mit den Beispieldateien arbeiten!



1.2.3 Installation von JAVA

Der Statechart-Editor benötigt JAVA Version 6 oder höher. Um festzustellen, ob dieses bereits auf Ihrem Rechner installiert ist, gibt es zwei Möglichkeiten:

1. Wechseln Sie in die Windows-Systemsteuerung (START ▸ EINSTELLUNGEN ▸ SYSTEMSTEUERUNG). Sofern JAVA auf Ihrem Rechner bereits installiert ist, finden Sie wie in nachfolgender Bildschirmgrafik gezeigt ein entsprechendes Icon. Über einen Doppelklick auf dieses Icon gelangen Sie in einen Dialog, der Ihnen die benötigten Versionsinformationen anzeigt.



Java-Icon in der Windows-Systemsteuerung

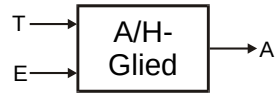
2. Folgen Sie dem Internet-Link www.java.com/de. Dort finden Sie einen Link, über den die auf Ihrem Rechner installierte JAVA-Version automatisch ermittelt wird (siehe nachfolgende Bildschirmgrafik).

Java im Internet

Sollte auf Ihrem Rechner noch kein JAVA installiert sein oder dieses nicht mindestens die Versionsnummer 6 besitzen, so finden Sie unter dem unter 2. angegebenen Link auch die Möglichkeit zum kostenlosen Download der aktuellen Version.

1.3 Einführendes Beispiel

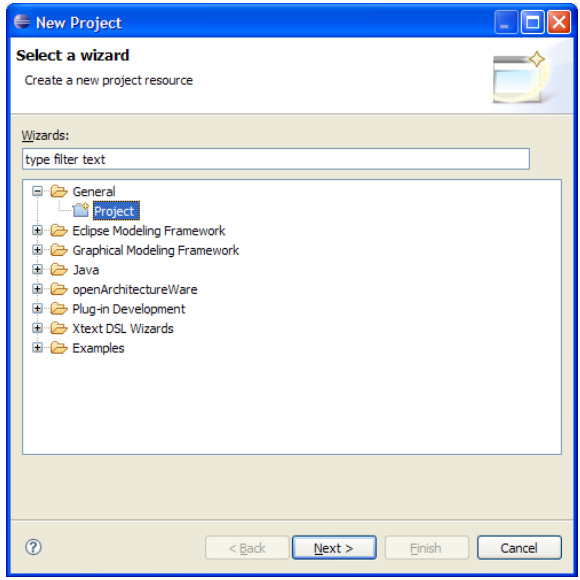
Bevor eine detaillierte Beschreibung des Leistungsumfangs der *State Machine Workbench* erfolgt, sollen die grundlegenden Funktionen anhand eines einfachen Einführungsbeispiels aufgezeigt werden. Dazu soll ein sogenanntes *Abtast-Halteglied* (Sample/Hold-Element) mit Hilfe eines Zustandsautomaten nachgebildet werden, wie es nachfolgende Grafik zeigt. Das am Eingang E anliegende Signal wird bei Auftreten einer positiven Flanke am Triggereingang T auf den Ausgang A geschaltet und dort gehalten, bis der nächste Triggerimpuls auftritt.



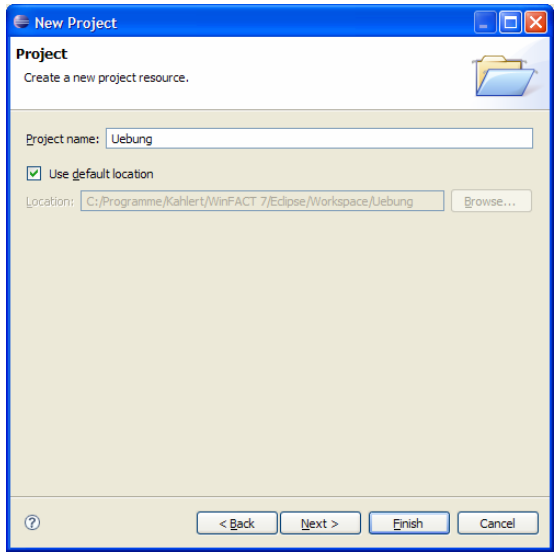
Abtast-Halteglied

Gehen Sie zum Entwurf des Zustandsautomaten wie folgt vor:

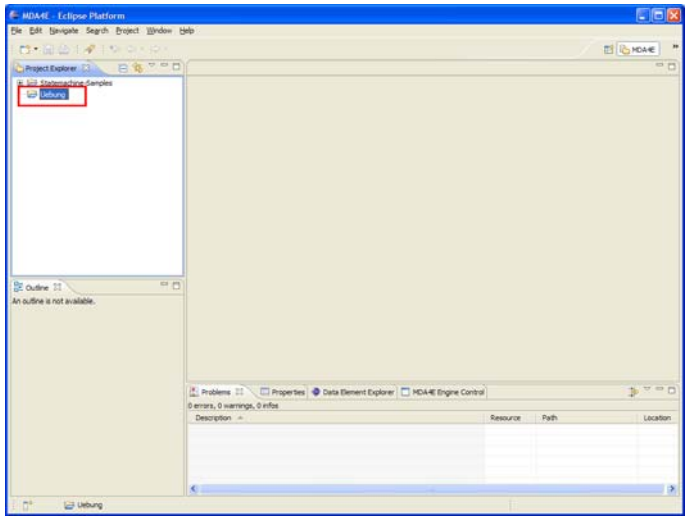
1. Starten Sie – sofern nicht bereits geschehen - den Statechart-Editor.
2. Legen Sie ein neues Projekt mit dem Namen *Uebung* an. Wählen Sie dazu zunächst die Menüfolge FILE ▸ NEW ▸ PROJECT... und geben Sie als Projekttyp *General* an:



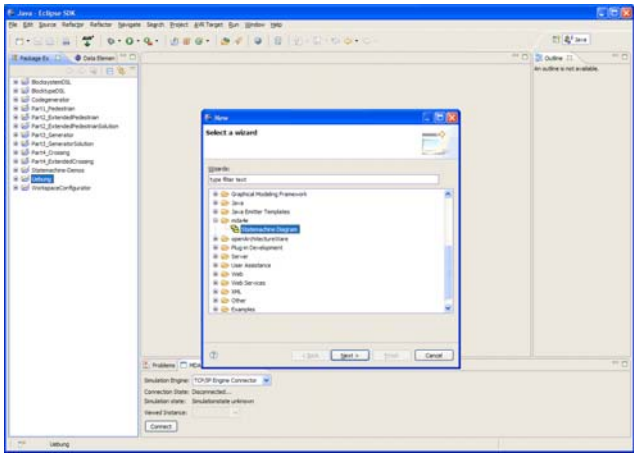
3. Fahren Sie mit der Schaltfläche *Next* > fort und geben Sie als Projektnamen *Uebung* an:



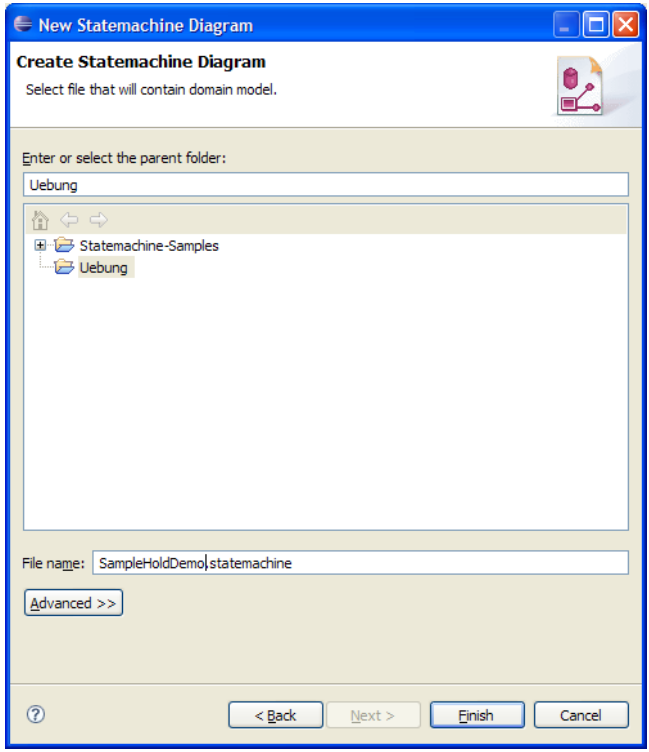
4. Bestätigen Sie über die Schaltfläche *Finish*. Im *Project Explorer* wird nun ein neuer Projektordner mit dem Namen *Uebung* angezeigt:



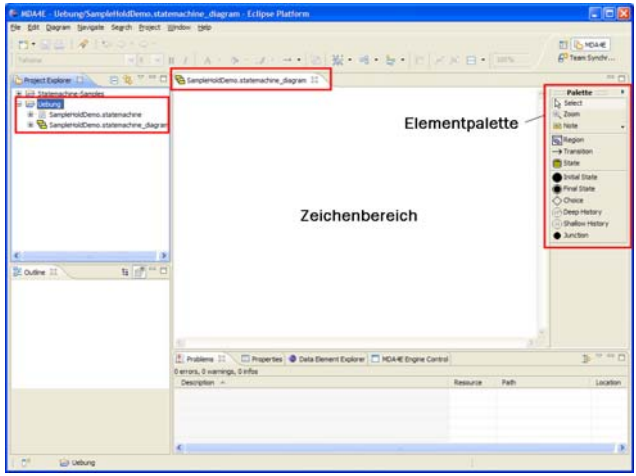
5. Klicken Sie mit der rechten Maustaste auf den neuen Projektordner und wählen Sie im daraufhin erscheinenden Kontextmenü die Menüfolge NEW ▸ OTHER… Es erscheint nunmehr ein Dialog zur Auswahl des Dateityps. Hier wählen Sie im Ordner *mda4e* den Typ *Statemachine Diagram* und fahren mit der Betätigung der Schaltfläche *Next* > fort:



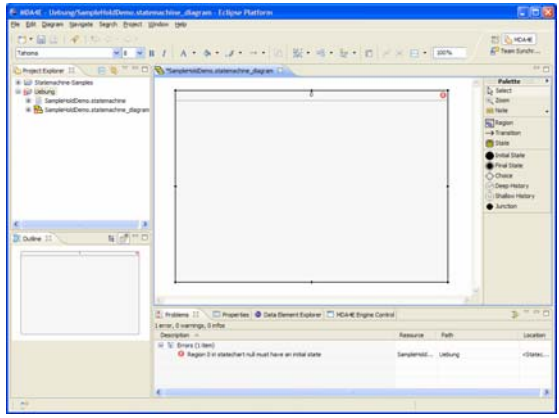
6. Nunmehr kann der Dateiname für den Zustandsautomaten gewählt werden. Jeder Zustandsautomat besteht aus zwei Dateien, wobei die erste das Zustandsdiagramm (d. h. die grafische Repräsentation des Automaten) darstellt und die Dateiendung *statemachine_diagram* besitzt, während die zweite (und für die spätere Einbindung in BORIS relevante) Datei mit der Dateiendung *statemachine* den logischen Aufbau des Zustandsautomaten enthält. Ändern Sie den voreingestellten Dateinamen *default.statemachine* auf *SampleHoldDemo.statemachine*:



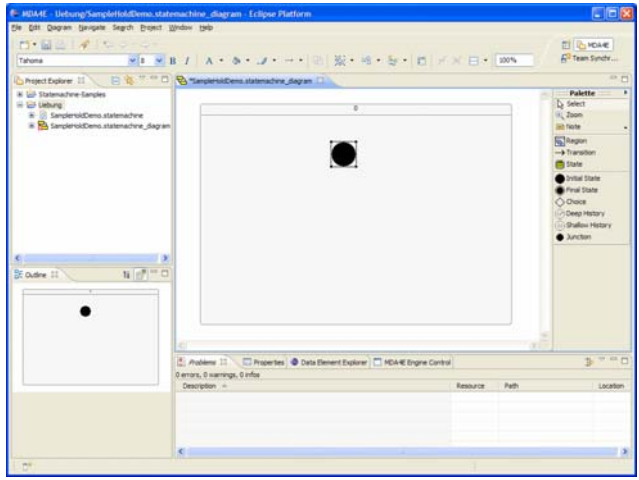
Schließen Sie den Dialog mit *Finish*. Im Projektordner finden Sie nun die beiden neuen Projektdateien; außerdem wurde der eigentliche Grafikeditor für das zu erstellende Zustandsdiagramm geöffnet. Der Zeichenbereich ist naturgemäß zunächst noch jungfräulich leer. Rechts vom Zeichenbereich befindet sich die Elementpalette mit den Editorwerkzeugen bzw. Komponenten des Zustandsautomaten:



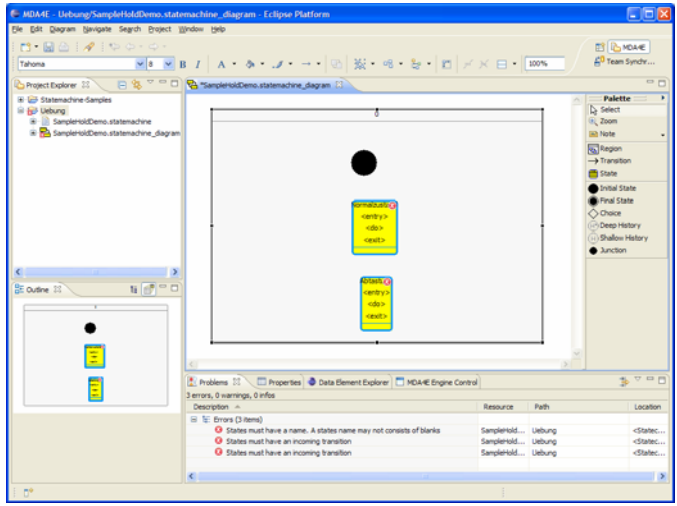
7. Als Erstes muss zunächst eine Region eingefügt werden. Klicken Sie dazu in der Elementpalette auf die Option *Region* und ziehen Sie anschließend mit der Maus bei festgehaltener Maustaste eine Region auf, die in etwa der Größe des Zeichenbereichs entspricht. Nach dem Loslassen der Maustaste wird am oberen Rand der erstellten Region die Eingabe einer eindeutigen ID verlangt; übernehmen Sie den voreingestellten Wert von 0:



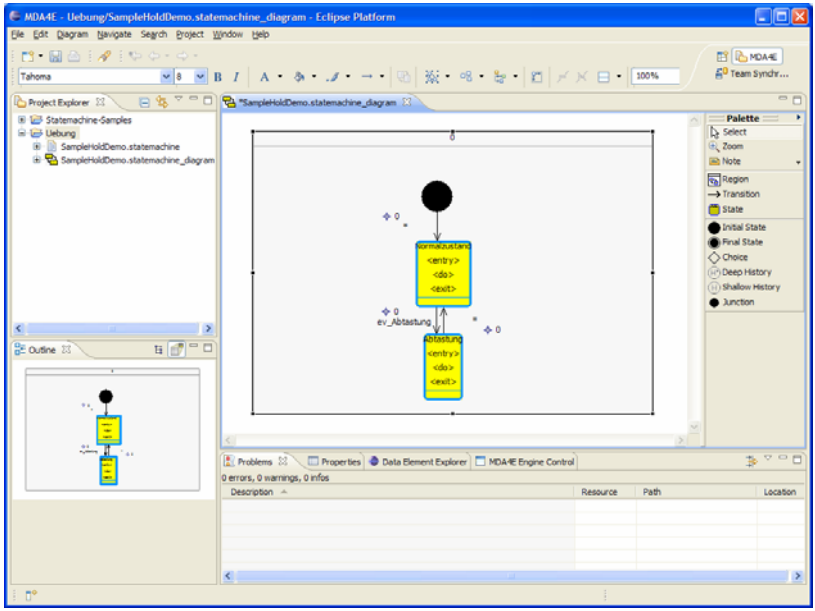
8. Nun muss der Anfangszustand definiert werden. Klicken Sie dazu in der Elementpalette auf *Initial State* und anschließend im oberen Bereich der zuvor aufgezogenen Region etwa in die Mitte; der Anfangszustand wird automatisch in seiner Standardgröße eingefügt:



9. Als nächstes wollen wir zwei Zustände einfügen: einen „Normalzustand“ und einen Zustand, der im Abtastvorgang kurzzeitig angenommen wird. Klicken Sie dazu in der Elementpalette zunächst auf *State* und anschließend etwa in die Mitte des Zeichenbereichs. Es erscheint wiederum ein Editierfeld zur Benennung des eingefügten Zustands; nennen Sie ihn *Normalzustand*. Fügen Sie auf dieselbe Weise anschließend unterhalb dieses Zustands einen zweiten Zustand ein, den Sie mit *Abtastung* bezeichnen. Ihr Editor sollte nun etwa wie folgt aussehen:



10. Nun müssen drei Transitionen definiert werden: vom Anfangszustand in den Normalzustand, vom Normalzustand in den Abtastzustand und von dort zurück in den Normalzustand. Klicken Sie dazu in der Elementpalette zunächst auf *Transition*. Klicken Sie anschließend in das Innere des Anfangszustands und ziehen Sie die Transition bei festgehaltener Maustaste in das Innere des Normalzustands. Nach Loslassen der Maustaste kann die Transition benannt werden; da dies für diese Transition nicht erforderlich ist, können Sie das entsprechende Editierfeld direkt verlassen. Anschließend fügen Sie auf dieselbe Weise eine Transition vom Normalzustand in den Abtastzustand ein; diese muss jedoch benannt werden. Geben Sie ihr die Bezeichnung *ev_Abtastung*. Abschließend ziehen Sie eine Transition vom Abtastzustand zurück in den Normalzustand; diese kann wieder unbezeichnet bleiben. Das Zustandsdiagramm sollte nun etwa wie folgt aussehen:



11. Für das Ein- und Ausgangssignal des späteren BORIS-Blocks müssen wir nun zwei Variablen definieren. Zunächst erstellen wir die Variable, die das Eingangssignal des BORIS-Blocks verarbeitet. Dazu wechseln Sie im unteren Fensterbereich auf die *Data Element Explorer*-Ansicht. Klicken Sie nun mit der rechten Maustaste in dieser Ansicht auf den Eintrag *Variables* und wählen Sie die Menüoption *Create Variable*. Es erscheint nunmehr der Dialog zur Definition von Variablen. Geben Sie folgende Werte an:

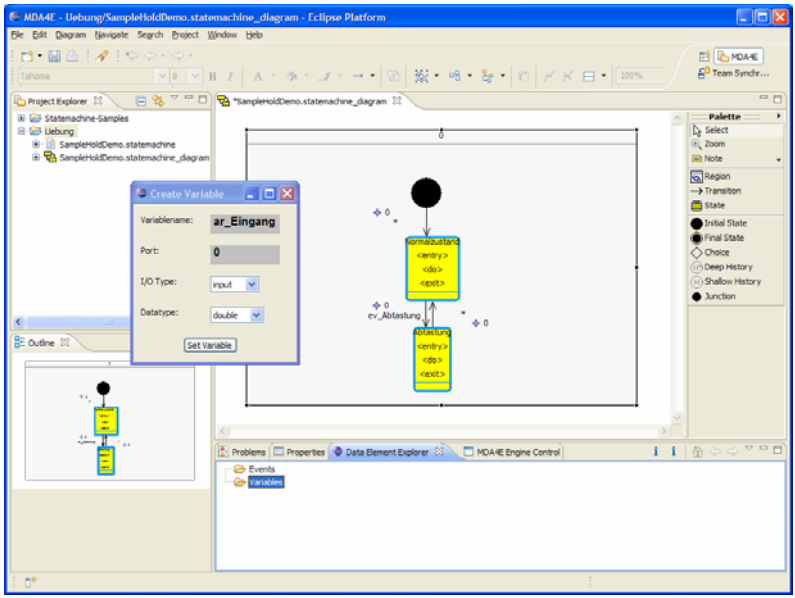
Variable name: *var_Eingang*

Port: *0*

I/O Type: *Input*

Data type: *Double*

Bestätigen Sie Ihre Eingaben über die Schaltfläche *Set Variable*. Wir haben dadurch eine Fließkomma-Eingangsvariable definiert, die den Namen *var_Eingang* trägt:



Auf dieselbe Weise definieren wir nun eine zweite Variable, die den Ausgang des BORIS-Blocks repräsentiert, wie folgt:

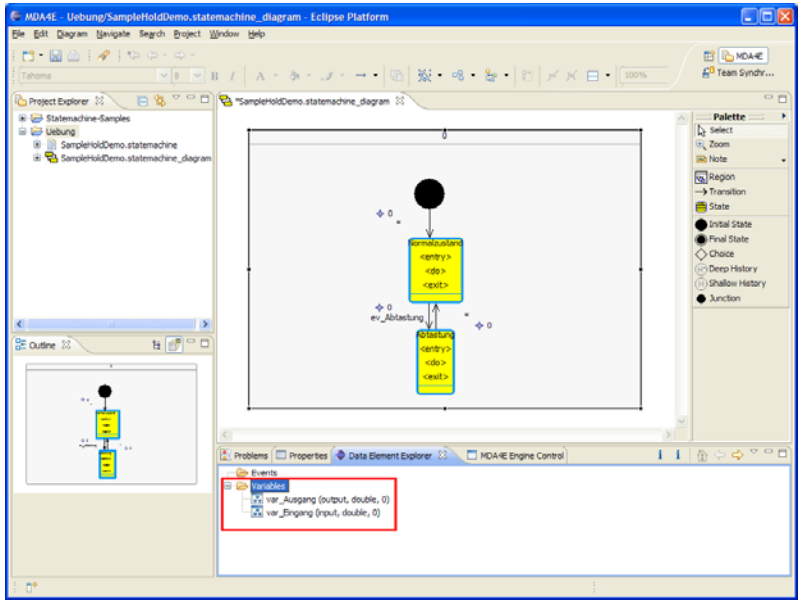
Variable name: *var_Ausgang*

Port: *0*

I/O Type: *Output*

Data type: *Double*

Beide Variablen werden nach ihrer Definition im *Data Element Explorer* angezeigt:



12. Nun müssen wir noch den Event spezifizieren, der die Abtastung auslöst. Dieser muss denselben Namen tragen wie die entsprechende Transition, hier also *ev_Abtastung*. Dazu klicken Sie im *Data Element Explorer* mit der rechten Maustaste auf den Eintrag *Events* und wählen die Option *Create Event*. Spezifizieren Sie den Event wie folgt:

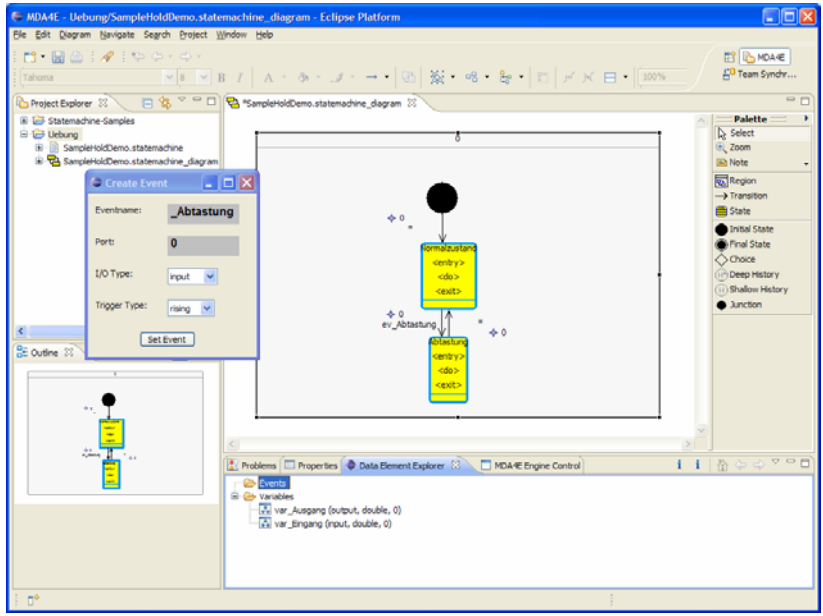
    Event name: *ev_Abtastung*

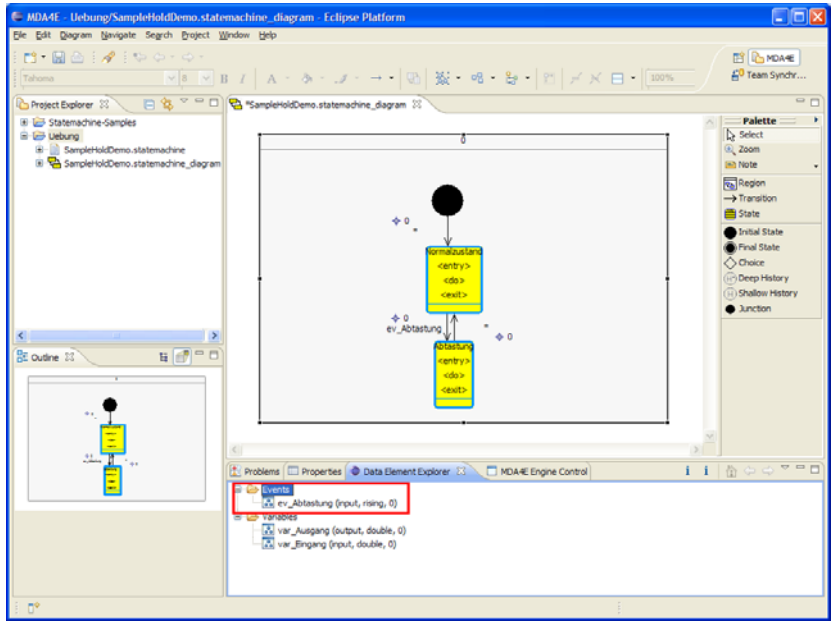
    Port: *0*

    I/O Type: *Input*

    Trigger type: *Rising*

    Bestätigen Sie Ihre Eingaben über die Schaltfläche *Set Event*. Der spezifizierte Event wird ebenfalls im *Data Element Explorer* angezeigt:

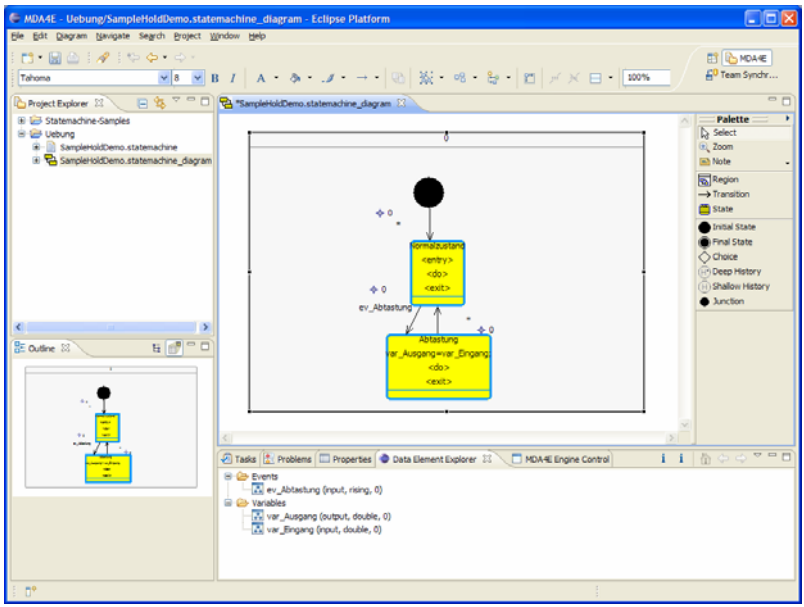

MDA4E - Uebung/SampleHoldDemo.statemachine_diagram - Eclipse Platform
Create Event
Eventname:
_Abtastung
Port:
0
I/O Type:
input
Trigger Type:
rising
Set Event
Normalzustand
<entry>
<do>
<exit>
ev_Abtastung
Abtastung
Palette
Select
Zoom
Note
Region
Transition
State
Initial State
Final State
Choice
Deep History
Shallow History
Junction
Problems
Properties
Data Element Explorer
MDA4E Engine Control
Events
Variables
var_Ausgang (output, double, 0)
var_Eingang (input, double, 0)


MDA4E - Uebung/SampleHoldDemo.statemachine_diagram - Eclipse Platform
Normalzustand
<entry>
<do>
<exit>
ev_Abtastung
Abtastung
Palette
Select
Zoom
Note
Region
Transition
State
Initial State
Final State
Choice
Deep History
Shallow History
Junction
Problems
Properties
Data Element Explorer
MDA4E Engine Control
Events
ev_Abtastung (input, rising, 0)
Variables
var_Ausgang (output, double, 0)
var_Eingang (input, double, 0)

13. Nunmehr müssen wir noch den eigentlichen Abtastvorgang „programmieren“. Klicken Sie dazu zwei Mal mit der Maus in den <*entry*>-Eintrag der unteren Transition und geben Sie in das Editierfeld dann den Programmcode

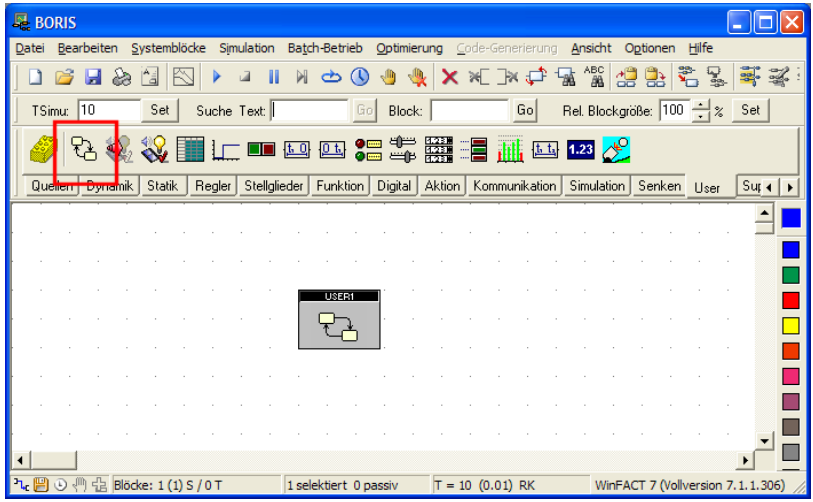
    ```
    var_Ausgang = var_Eingang;
    ```

    ein (Semikolon nicht vergessen!):

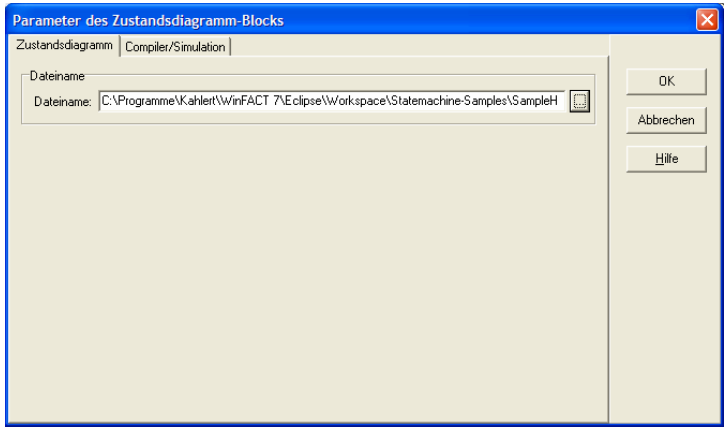


14. Der Zustandsautomat ist nunmehr fertig gestellt; speichern Sie das Beispiel über die Menüfolge FILE ▸ SAVE oder die Toolbar-Schaltfläche mit dem Diskettensymbol ab. Schließen Sie den Editor aber noch *nicht*!

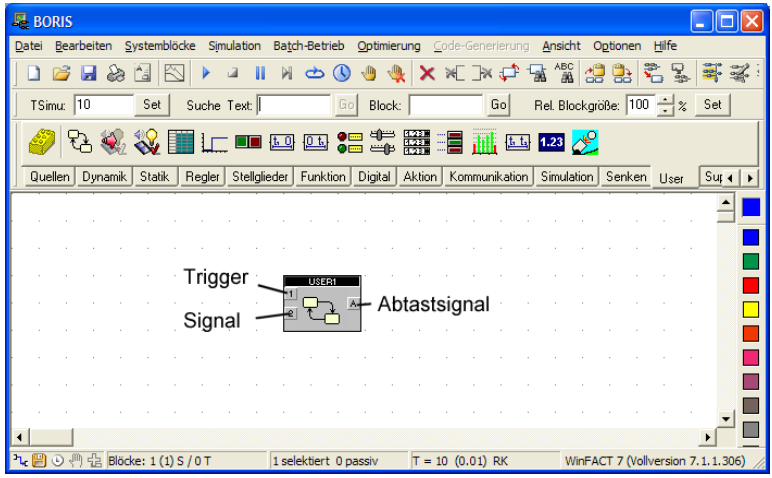
15. Starten Sie nun BORIS und fügen Sie über den Reiter *User* der Systemblock-Toolbar einen leeren Zustandsautomaten-Block ein:



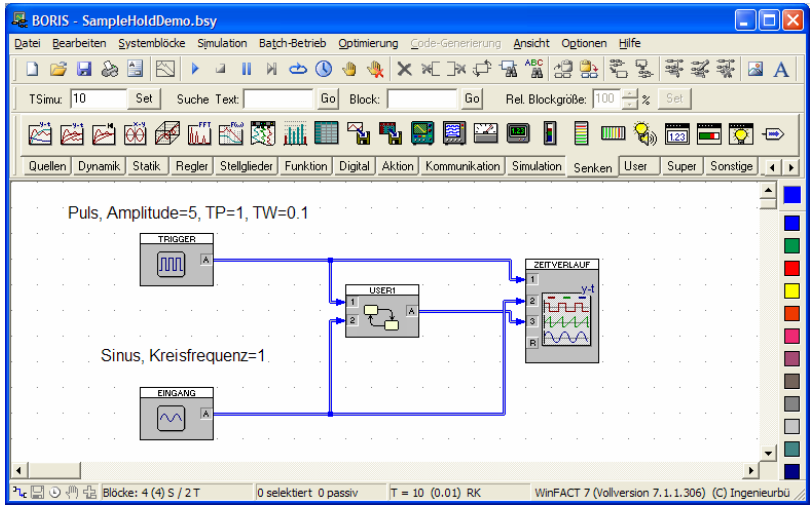
Öffnen Sie durch einen Doppelklick auf den Block und anschließende Betätigung der Schaltfläche *Dialog…* den Parameterdialog des Zustandsautomaten-Blocks und geben Sie die zuvor erstellte Datei *SampleHoldDemo.statemachine* an; diese befindet sich im Unterverzeichnis *\ECLIPSE\workspace* Ihrer WinFACT 7-Installation:



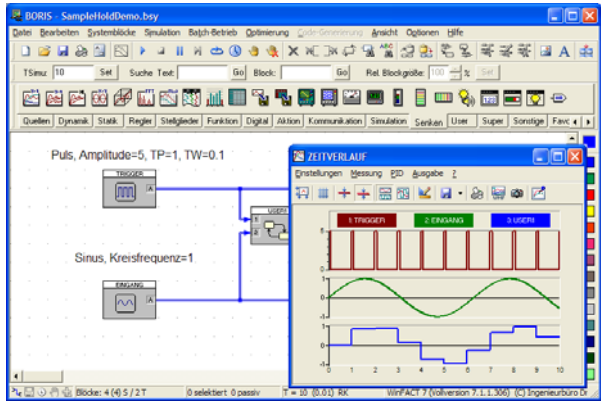
Die übrigen Einstellungen im Parameterdialog sind zunächst ohne Belang; schließen Sie ihn daher über die Schaltfläche *OK*. Der Zustandsautomaten-Block hat nunmehr zwei Eingänge (Triggersignal oben und Eingangssignal unten) sowie einen Ausgang (abgetastetes Signal):



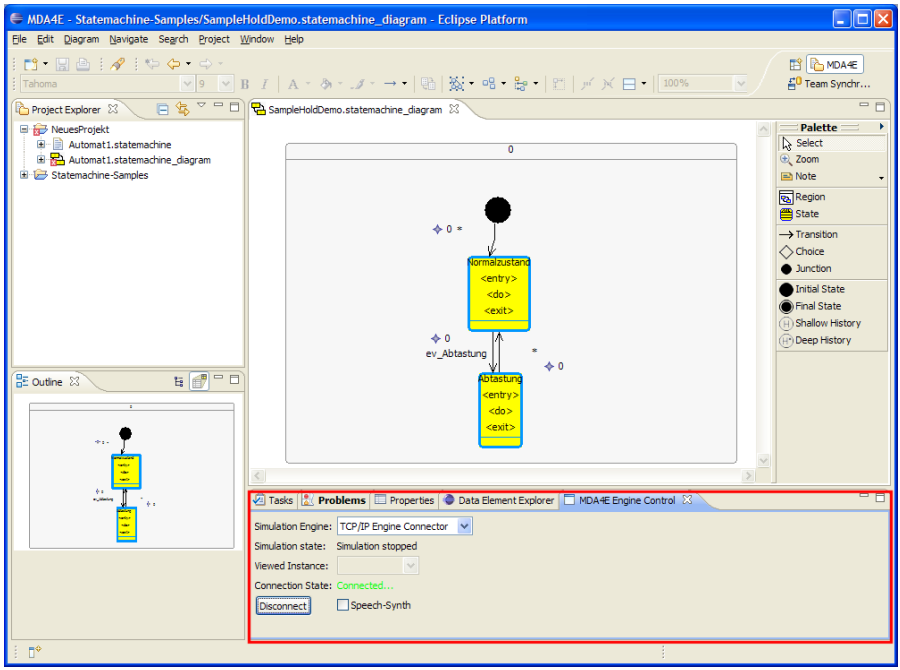
16. Als Eingangssignal wählen wir ein Sinussignal mit der Kreisfrequenz 1, als Triggersignal ein Rechtecksignal mit der Amplitude 5 und einer Periodendauer von 1 s. Alle Signale führen wir auf einen Zeitverlauf-Block, sodass unsere Gesamtstruktur wie folgt aussieht:



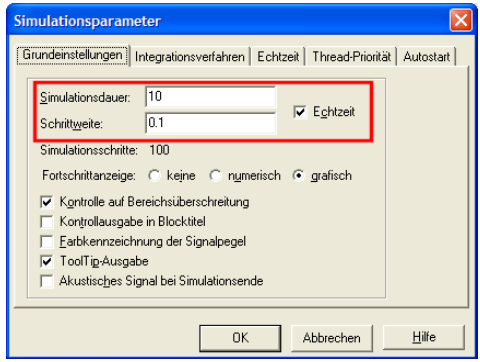
17. Starten Sie die Simulation. Sie sollten folgende Ergebnisse erhalten:



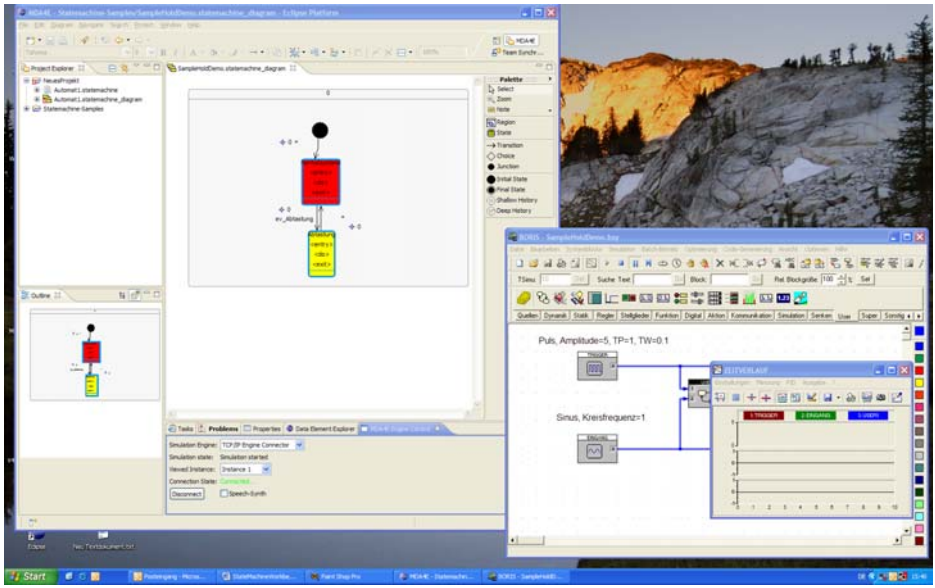
18. Abschließend soll nun die Simulation mit paralleler Kommunikation mit dem Statechart-Editor erfolgen. Wechseln Sie dazu im Editor im unteren Fensterbereich auf die Ansicht *MDA4E Engine Control* und verbinden Sie den Editor mit BORIS über die Schaltfläche *Connect*:



19. Ordnen Sie das Editor-Fenster sowie das BORIS-Fenster so an, dass beide Fenster vollständig sichtbar sind. Ändern Sie die Simulationsschrittweite in BORIS auf 0.1 s und aktivieren Sie die Echtzeit-Simulation über die Menüoption SIMULATION ▸ PARAMETER…:



Starten Sie nun erneut die Simulation unter BORIS. Die von BORIS ermittelten Ergebnisse werden nunmehr während der Simulation an den Statechart-Editor geschickt, der sie durch entsprechendes Highlighting aktiver Zustände und Transitionen visualisiert:



20. Herzlichen Glückwunsch – Sie haben Ihren ersten Zustandsautomaten erfolgreich in Betrieb genommen! In den nachfolgenden Kapiteln erfahren Sie nun Genaueres zu den vielfältigen Möglichkeiten der *State Machine Workbench*.

2 Elemente von Zustandsautomaten

Auf der Elementpalette des Statechart-Editors sind sämtliche Elementtypen hinterlegt, aus denen sich Zustandsautomaten aufbauen lassen. Hier sollen nun die einzelnen Typen und deren Bedeutung innerhalb von Zustandsautomaten erläutert werden. Die Elementtypen sind in zwei Kategorien unterteilt. Zum einen gibt es die grundlegenden Elemente *Region, Transition* und *State*, zum anderen gibt es eine Reihe von sogenannten Pseudozuständen, *Initial State, Final State, Choice, Deep History, Shallow History* und *Junction*, von denen Sie den erstgenannten bereits verwendet haben.

2.1 Regionen (Regions)

Ein Zustandsautomat wird immer in grafisch abgegrenzten Gebieten, den *Regions*, aufgebaut. Diese Schachtelung ermöglicht es, mehrere Zustandsautomaten auf der obersten Ebene anzulegen, die dann auch parallel abgearbeitet werden. Da eine parallele Abarbeitung mehrerer Aufgaben (hier Zustandsautomaten) auf einem einzelnen Prozessor nicht möglich ist, wurden Prioritäten eingeführt, die eine Abarbeitungsreihenfolge fest vorgeben. Diese Prioritäten werden als Ganzzahlen in der Titelzeile einer Region aufgeführt. Eine größere Zahl bedeutet hierbei eine höhere Priorität.

Eine Region kann aber nicht nur direkt auf der Zeichenfläche des Editors verwendet werden, sondern auch innerhalb eines Zustands und dieses ebenfalls mehrfach. Mit anderen Worten: Ein Zustand kann wieder einen oder mehrere Zustandsautomaten (*Regionen*) beinhalten, die ebenfalls wieder in einer durch Prioritäten vorgegebenen Abarbeitungsreihenfolge ausgeführt werden.

2.2 Zustände (States)

Über den Elementtyp *State* fügen Sie Zustände zu einem Automaten hinzu. Ein Zustand wird wie oben beschrieben immer in einer *Region* platziert und muss innerhalb „seiner“ Region einen eindeutigen Namen haben. Während der Simulation gibt es in jeder Region genau einen aktiven Zustand. Jeder Zustand kann also seinen Simulationszustand von passiv zu aktiv und umgekehrt wechseln. Man sagt auch, ein Zustand wird betreten bzw. verlassen.

Jeder Zustand enthält nach dem Einfügen zunächst die drei Felder <*entry*>, <*do*> und <*exit*>. Diese stehen für jeweils eine frei definierbare Aktion bzw. Aktivität. Nur bei <*do*> spricht man von einer Aktivität, da diese solange resp. immer wieder ausgeführt wird, wie der Zustand aktiv ist (s. u.). Wie die Namen der Felder vermuten lassen, werden diese bei bestimmten Ereignissen ausgeführt.

So wird die

<*entry*>- Aktion bei dem Betreten des Zustands, die

<*do*>- Aktion bei jedem Simulationsschritt und die

<*exit*>-Aktion beim Verlassen des Zustands ausgeführt.

Es kann vorkommen, dass ein Zustand betreten und direkt wieder verlassen wird. In diesem Fall bleibt die <*do*>-Aktivität unberücksichtigt!

Generell müssen die drei Felder in syntaktisch korrektem ANSI-C verfasst werden und sollten möglichst nur kurze Anweisungen enthalten. Es sollen hier kurz ein paar Beispiele für übliche Anweisungen vorgestellt werden:

```
1.  a--;
2.  a++;
3.  b += a * 2;
4.  b -= abs(a);
5.  c = (b != 0 && a > 0) ? a / (double) b : 1;
```

Damit diese Anweisungen syntaktisch gültig sind, müssen die verwendeten Bezeichner (hier `a`, `b` und `c`) im *Data Element Explorer* des Editors als Variablen definiert worden sein (s. u.). Die Gültigkeit aller Anweisungen wird erst durch den BORIS-Block festgestellt, der Ihnen ggf. die entsprechenden Fehlermeldungen eines im Hintergrund arbeitenden C-Compilers in einem Meldungsfenster anzeigt.

Das Auffinden eines Fehlers innerhalb des Diagramms ist unter Umständen keine einfache Sache (der Compiler kennt dieses nicht und bezieht seine Ausgaben immer auf einen durch den BORIS-Block erzeugten C-Code).

Wir empfehlen Ihnen, die folgenden Tipps zu beherzigen:

- Jede Anweisung muss mit einem Semikolon abgeschlossen werden.
  Beispiel für eine Entry-Funktion:

```
ia=0; ib=20; ic=3.3;
```

- Verwenden Sie keine Schleifen (`while()... , for()... , do... while()`).
  Die *do*-Funktionalität wird in jedem Simulationsschritt ausgeführt, solange der Zustand aktiv ist. Eine Abbruchbedingung der Schleife kann als Bedingung an einer abgehenden Transition formuliert werden.
- Vermeiden Sie Verzweigungen (z. B. if-/switch-case – Anweisungen).
  Diese sollten besser durch die Pseudozustände Entscheidungsknoten (*Choice*) oder Verbindungsstelle (*Junction*) ausgedrückt werden.
- Vermeiden Sie komplexe, nicht auf Anhieb überschaubare Ausdrücke.
  Der in den oben stehenden Beispielen letzte Ausdruck ist so gerade noch überschaubar.
- Falls Sie eine komplexere Funktion benötigen, schreiben Sie diese Funktion besser in einem separaten C-Modul (C-Datei), das Sie durch eine Headerdatei einbinden und ggf. vorab testen. Derartige Funktionen sollten seiteneffektfrei sein, d. h. gleiche Funktionsargumente führen immer zum gleichen Ergebnis; es gibt keine weiteren Abhängigkeiten. Wäre dem nicht so, hätten Sie eine zustandsbehaftete Funktion, die besser durch ein Zustandsdiagramm beschrieben wird.

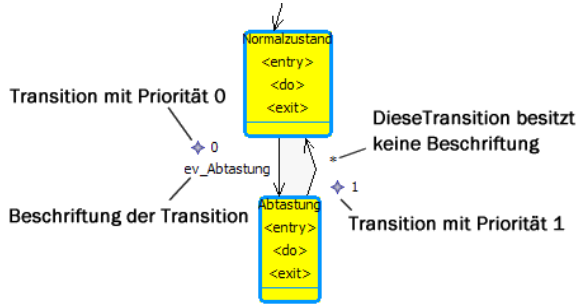
2.3 Transitionen (Transitions)

Damit ein Wechsel von einem Zustand in einen anderen stattfinden kann, ist eine Übergangsfunktion zu definieren. Diese wird im Diagramm durch einen Pfeil dargestellt. An der Kante des Pfeils sind zwei Angaben erforderlich:

- Eine Beschriftung, die Angaben macht, wann bzw. unter welchen Bedingungen der Übergang durchgeführt werden soll und mit welcher Aktion dieser verbunden sein soll. Wird keine Bedingung festgelegt, wird die Transition lediglich mit einem Sternchen (*) beschriftet.

- Eine Priorität. Diese wird an der Transition mit einer vorangestellten Raute gekennzeichnet.

Nachfolgender Ausschnitt aus einem Zustandsdiagramm verdeutlicht dies am Beispiel zweier Transitionen.



Beschriftung und Priorität von Transitionen

Die Beschriftung ist wie folgt anzugeben:

```
e1, e2, ... [Bedingungsausdruck] / Aktion
```

Hierbei sind `e1, e2,` usw. die Namen der Ereignisse, bei denen der Übergang vom Ausgangszustand zum Zielzustand ausgelöst wird (in obigem Ausschnitt beispielsweise *ev_Abtastung*). Dabei reicht das Eintreten eines der gelisteten Ereignisse aus. Die Namen der Ereignisse müssen im *Data Element Explorer* definiert worden sein. Der Bedingungsausdruck erfordert, dass das durch diesen Ausdruck spezifizierte Resultat ungleich Null (nach ANSI-C Konvention *wahr*) sein muss, damit der Zustandswechsel erfolgt. Enthält eine Transition keine Bedingung, so verhält sie sich so, als wäre eine Bedingung angegeben, die immer *wahr* ist. Enthält eine Transition keine Ereignisse, so entspricht dies der Angabe eines Ereignisses das immer präsent ist.

Wird die Transition ausgeführt, dann werden zuvor (also bevor der Zielzustand betreten wird) die Anweisungen der Aktion ausgeführt. Die Anweisungen selbst sind wie bei den Aktionen eines Zustands in ANSI-C anzugeben und jeweils durch ein Semikolon abzuschließen. Als Aktionen können beispielsweise Ereignisse (Events) vom Typ *Output* dienen, die über den *Send*-Befehl ausgelöst werden. Soll etwa das Event *ev_Out1* ausgelöst werden, so gelingt dies über das Kommando

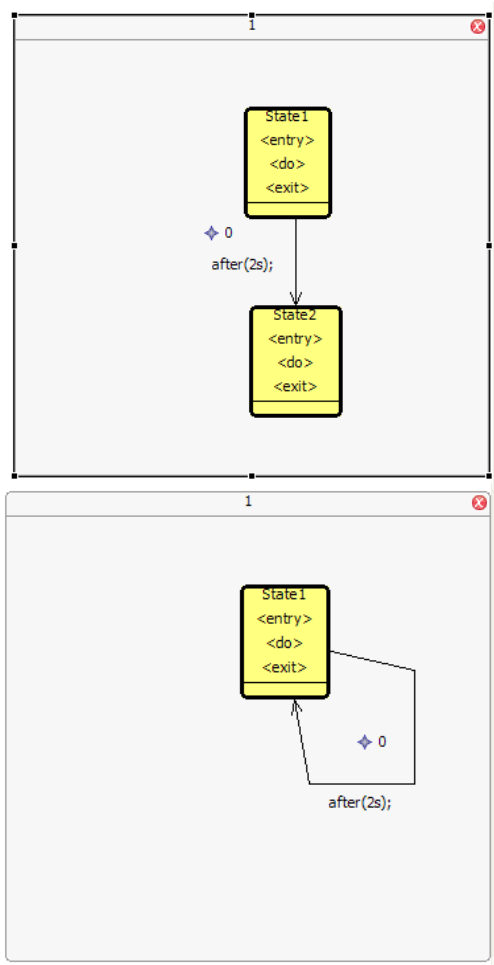
```
send(ev_Out1);
```

Eine besondere Bedeutung im Zusammenhang mit Transitionen hat die *after()*-Klausel, die die Realisierung zeitgesteuerter Übergänge erlaubt. Der Übergang zwischen den beteiligten

Zuständen erfolgt dabei automatisch nach der im *after()*-Befehl spezifizierten Zeitspanne. Soll also beispielsweise ein Übergang von Zustand *State1* in Zustand *State2* nach 2 s erfolgen, so kann dies wie in nachfolgender Grafik (oberes Teilbild) gezeigt durch die Transitionsbedingung

```
after(2s);
```

realisiert werden. Auch zeitgesteuerte Übergänge von einem Zustand auf sich selbst sind auf diese Weise möglich (unteres Teilbild).



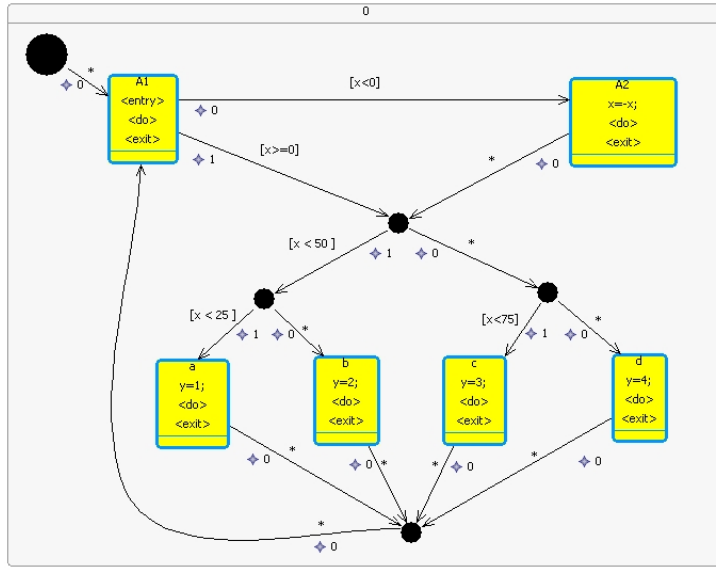
Zeitgesteuerte Übergänge mit Hilfe der after()*-Klausel*

Transitionen, die denselben Ausgangszustand haben, müssen unterschiedliche Prioritäten besitzen. Dies gilt auch dann, wenn die angegebenen Bedingungen sich gegenseitig ausschließen. Durch die Vergabe von Prioritäten erübrigt sich oft die Angabe von Bedingungen: Dadurch, dass die Transition mit der geringsten Priorität ohne Bedingung formuliert werden kann, dient diese sozusagen für alle nicht näher spezifizierten Fälle. Ein Beispiel hierzu finden Sie im Abschnitt *2.3.1 Verbindungsstelle (Junction)*.

Mit Hilfe der Pseudozustände *Verbindungsstelle* (Junction) und *Entscheidungsknoten* (Choice) können segmentierte Transitionen aufgebaut werden. Hierbei werden die oben angegebenen Beschreibungsteile auf den Transitionspfad verteilt, was in der Regel zu mehr Übersichtlichkeit und leichter verständlichen Automaten führt.

2.3.1 Verbindungsstelle (Junction)

Verbindungsstellen dienen eher der Strukturierung eines Automaten als seinem Verhalten. Setzen Sie Verbindungsstellen dann ein, wenn Übergänge von einem oder mehreren Ausgangszuständen zu einer Menge von Zielzuständen unter gleichen Bedingungen existieren. Das nachfolgende Zustandsdiagramm soll dieses näher veranschaulichen:



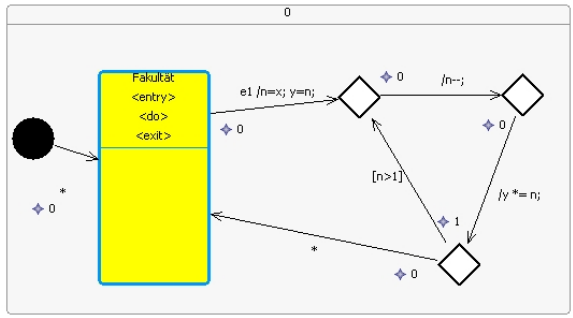
Dieses Zustandsdiagramm beschreibt eine Funktion, die in Abhängigkeit vom Wert der Variablen x die Variable y auf einen Wert der Menge {1, 2, 3, 4} setzt. Wesentlich ist hier aber nicht die Funktion, sondern die Einfachheit mit der, durch Hilfe von Verbindungsstellen,

komplexere Bedingungen auf die Segmente der Transition verteilt werden können. Ebenfalls können aufgrund der Priorität, die pro Segment zu definieren ist, einige Bedingungen entfallen. So wird y = 1, wenn |x| < 25, y = 2 wenn 25 <= |x| < 50, y = 3 wenn 50 <= |x| < 75 und y = 4 sonst.

Segmentierte Transitionen dürfen, da sie als eine Transition gelten, nur im ersten Segment eine Ereignisliste besitzen. Alle folgenden Segmente dürfen nur Bedingungen und Aktionen enthalten.

2.3.2 Entscheidungsstelle (Choice)

Die Entscheidungsstelle dient genau wie die Verbindungsstelle dem Aufbau segmentierter Transitionen. Die Entscheidung erfolgt hierbei allerdings nach Abarbeitung der Aktion an der Kante des ersten Segments, sofern eines der notierten Ereignisse eingetroffen ist und die ggf. angegebene Bedingung zu *wahr* ausgewertet wird. Hierdurch ist man in der Lage, Berechnungen durchzuführen. Als Beispiel die Bestimmung der Fakultät einer Ganzzahl:



Sobald das Ereignis `e1` eingetroffen ist, wird, ausgehend vom alleinigen Zustand *Fakultät*, die Aktion: `n=x; y=n;` ausgeführt. Aufgrund der folgenden zwei Segmente schließen sich die Anweisungen `n--;` und `y*=n;` an. In der letzten Entscheidungsstelle wird geprüft, ob `n>1` ist, und entsprechend verzweigt. Die Transition ist vollständig durchlaufen, wenn der Zustand *Fakultät* wieder erreicht ist.

Eine Ereignisliste ist auch bei dieser Art der Transition nur im ersten Segment (hier: `e1 / n=x; y=n;`) erlaubt.

Der Unterschied zur Verbindungsstelle ist, dass Aktionen direkt ausgeführt werden. Dies bedeutet, dass falls ein Segment aufgrund einer Bedingung nicht durchschritten wird, die Transition also „mitten drin“ abbricht, trotzdem alle bis zu diesem Punkt notierten Aktionen durchgeführt werden. Der Ausgangszustand bleibt in diesem Fall jedoch aktiv, d. h. es findet kein Zustandswechsel statt.

Der Zustandsautomaten-Block von BORIS erzwingt Entscheidungsstellen, die immer vollständig durchlaufen werden, indem er prüft, ob ein von einer Entscheidungsstelle ausgehendes Segment ohne Bedingung mit geringster Priorität existiert.

2.4 Pseudozustände

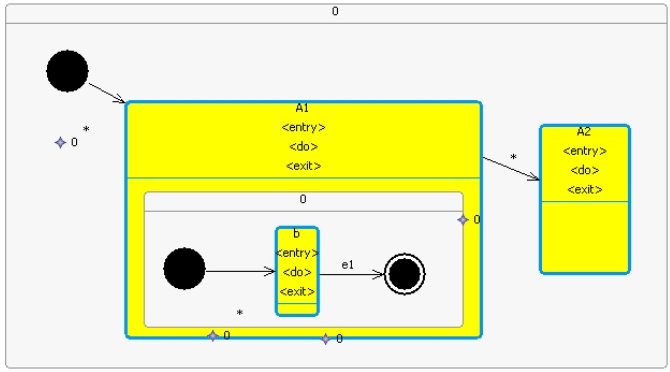
Pseudozustände sind Zustände, die keinen zeitlichen Aufenthalt bedeuten. Aus diesem Grund können derartige Zustände nicht betreten werden. Man benötigt diese Zustände vielmehr um Dinge auszudrücken, die auf andere Weise schlecht oder gar nicht ausgedrückt werden können.

2.4.1 Initialzustand (Initial State)

Der Initialzustand kennzeichnet lediglich den Zustand, der als erstes vom Automaten eingenommen werden soll. Er ist Startpunkt bei der Interpretation des Zustandsdiagramms, weshalb von ihm immer nur genau eine Transition bzw. ein Transitionssegment abgehen darf. Dieses darf weder eine Ereignisliste noch eine Bedingung enthalten.

2.4.2 Endzustand (Final State)

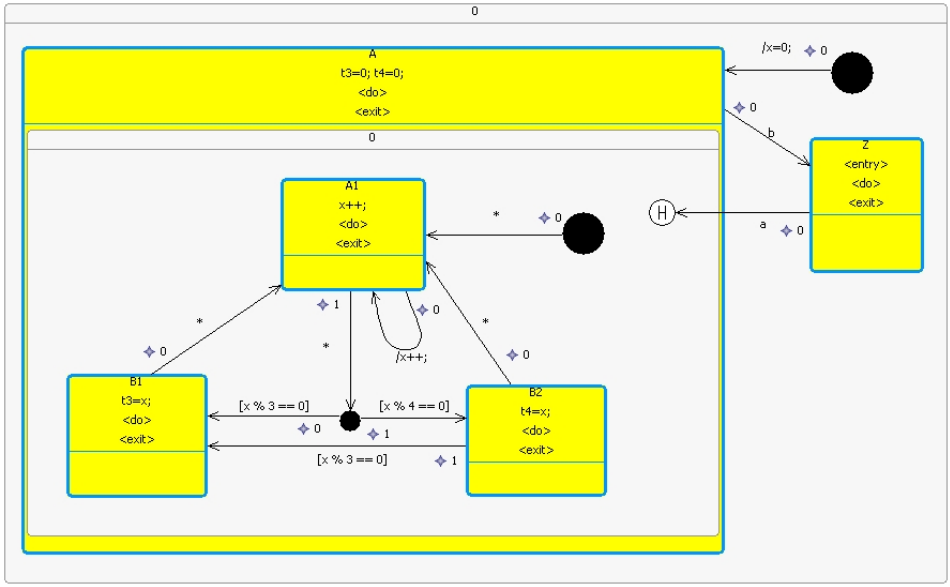
Der Endzustand markiert das Ende eines Zustandsautomaten. Befindet sich der Automat innerhalb eines anderen Zustandes und hat der übergeordnete Zustand eine abgehende Transition, die kein Ereignis hat, so schaltet diese Transition und der übergeordnete Zustand wird verlassen. Hierzu muss der Endzustand erreicht worden sein, was bedeutet, dass dies erst bei der nächsten Ausführung des Zustandsautomaten stattfinden kann.



Im obigen Beispiel werden im ersten Schritt die Zustände A1 und b betreten und bleiben in den nachfolgenden Schritten aktiv, bis das Ereignis e1 eintrifft. Jetzt wird b verlassen und der Endzustand erreicht. Bei der nächsten Ausführung des Zustandsautomaten wird erkannt, dass sich der in A1 enthaltene Zustandsautomat im Endzustand befindet und die Transition von A1 zu A2 schaltet.

2.4.3 Flacher/normaler History-Zustand (Shallow History State)

Manchmal möchte man einen Zustand, der durch ein weiteres untergeordnetes Zustandsdiagram beschrieben wird, in gleicher Weise aktivieren wie er verlassen wurde.

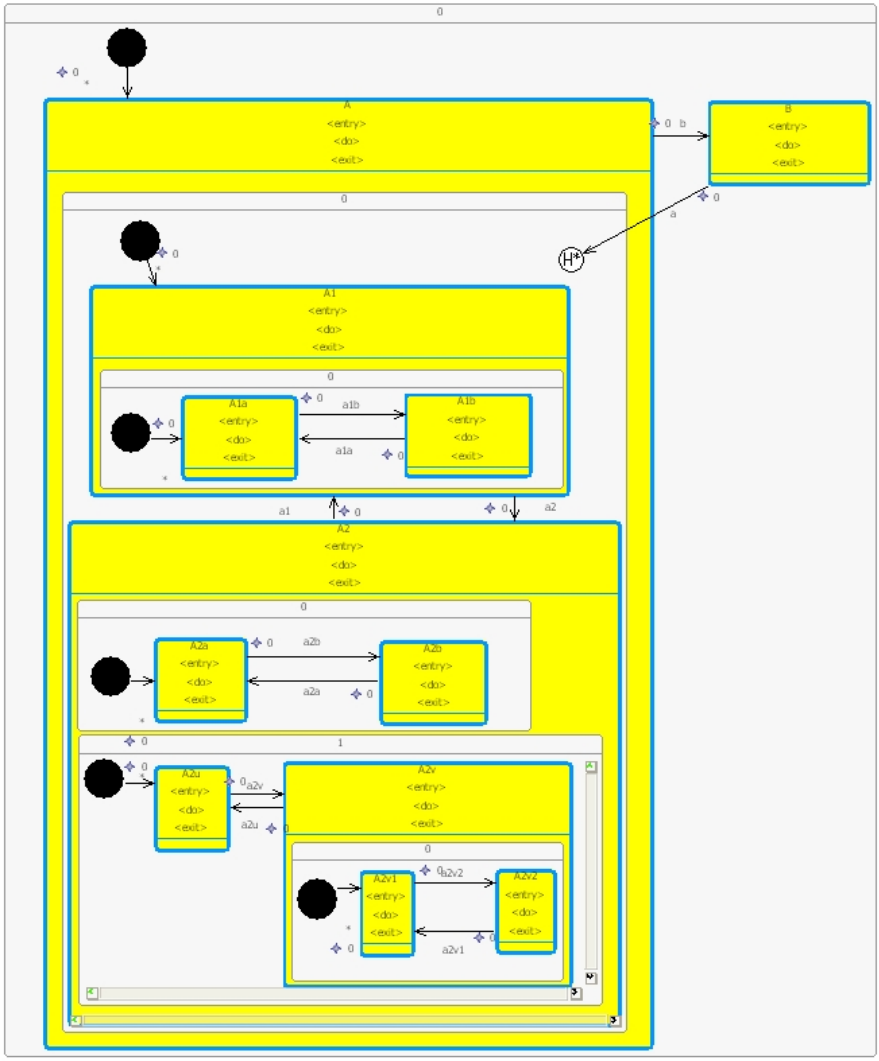


Im obigen Diagramm kann der Zustand A durch die Transition b (von A zu Z) verlassen werden. Dadurch wird auch der in A befindliche Zustandsautomat verlassen. Der aktive Unterzustand wird jedoch im durch H bezeichneten History-Zustand abgelegt. Diese Referenz auf den zuletzt aktiven Zustand innerhalb der Region, in der auch H ist, wird durch eine Transition zu H wieder aktiviert.

Der obige Zustandsautomat dient übrigens der Ausgabe aller Ganzzahlen, die durch drei und/oder durch vier teilbar sind (t3 und t4 sind Ausgangsvariablen).

2.4.4 Tiefer History-Zustand (Deep History State)

Der tiefe History Zustand speichert die komplette Hierarchie aller aktiven Zustände und ermöglicht es, diese wieder zu aktivieren. Die nachfolgende Grafik veranschaulicht den Zusammenhang.

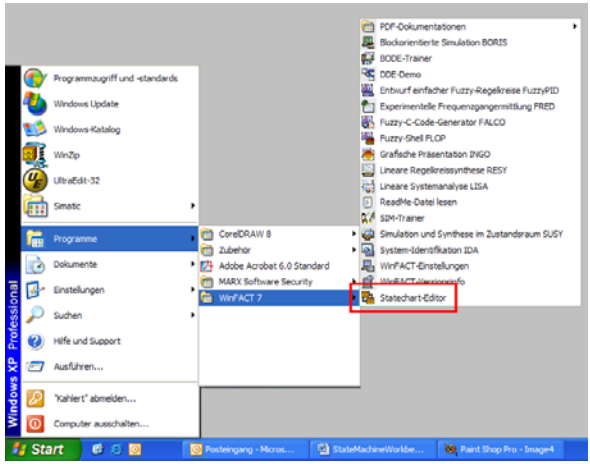


Im tiefen History-Zustand werden alle aktiven Zustände, die sich in der Region des linken Zustands (namentlich mit A gekennzeichnet) und in den darin befindlichen Regionen existieren, beim Verlassen von A zwischengespeichert, damit diese beim Schalten der Transition von B zu A wieder aktiviert werden können.

3 Der Statechart-Editor

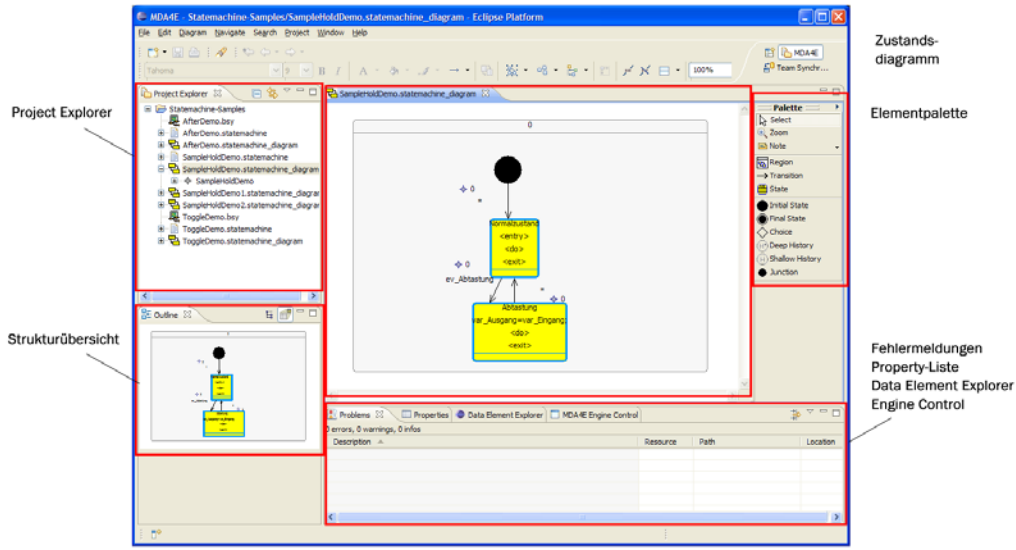
3.1 Aufruf und Konfigurierung des Statechart-Editors

Der Statechart-Editor ist Bestandteil (Plug-In) der freien Entwicklungsumgebung ECLIPSE und wird aus Ihrer WinFACT 7-Programmgruppe gestartet:



Der Statechart-Editor in der WinFACT 7-Programmgruppe

Nachfolgende Bildschirmgrafik zeigt den Aufbau des Editor-Hauptfensters:



Komponenten des Editor-Hauptfensters

Das Hauptfenster des Editors weist im Wesentlichen die folgenden Komponenten auf:

- Die *Arbeitsfläche* (Zeichenfläche) zum Aufbau des Zustandsdiagramms im Innenbereich des Editors. Diese Arbeitsfläche ist scrollfähig; außerdem kann die Ansicht über die rechts oberhalb der Zeichenfläche befindliche Listbox gezoomt werden.
- Die *Elementpalette* rechts neben der Arbeitsfläche mit den unterschiedlichen Komponenten von Zustandsautomaten.
- Eine *Strukturübersicht* (*Outline*); diese bietet unabhängig vom Ausschnitt der Arbeitsfläche einen Überblick über die Gesamtstruktur.
- Eine *Projektübersicht* (*Project Explorer*) am linken Fensterrand. Diese bietet eine Übersicht über alle im aktuellen ECLIPSE-Workspace befindlichen Projekte.

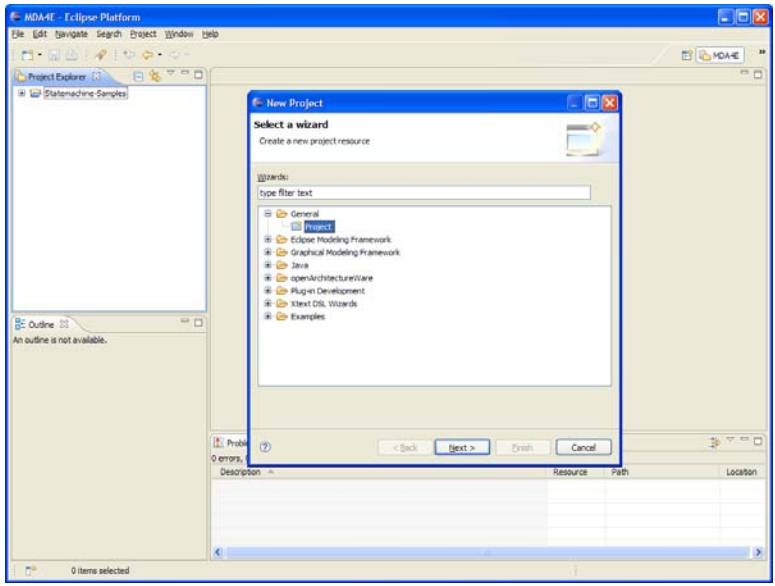
Am unteren Fensterrand befinden sich zudem folgende Komponenten, zwischen denen über entsprechende Registerkarten umgeschaltet werden kann:

- Eine Liste mit *Fehlermeldungen* (*Problems*), die auf Fehler im aktuellen Zustandsdiagramm (z. B. fehlerhafte Transitionen) hinweist.
- Ein *Eigenschaftseditor* (*Properties*), der die Anzeige und Bearbeitung der Eigenschaften des aktuell selektierten Zustandsdiagramm-Elementes ermöglicht.

- Eine *Variablen- und Eventübersicht* (*Data Element Explorer*). Hier finden Sie eine Liste aller im aktuellen Zustandsdiagramm deklarierten Variablen und Ereignisse (Events).
- Ein *Kommunikationsfenster* (*MDA4E Engine Control*); dieses liefert Ihnen die erforderlichen Einstellungen und Informationen für die Online-Kommunikation zwischen BORIS und dem Statechart-Editor.

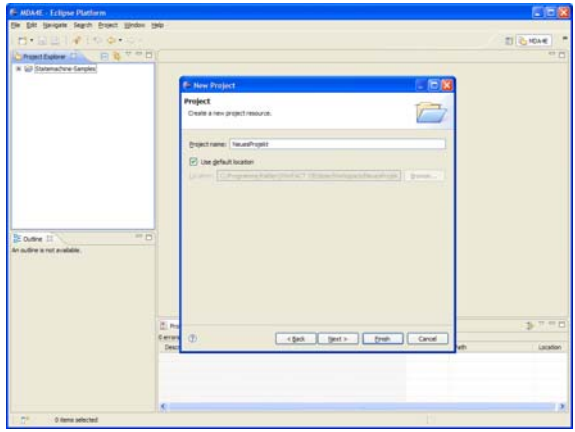
Zum Anlegen eines neuen Projektes gehen Sie wie folgt vor:

1. Klicken Sie mit der rechten Maustaste innerhalb des Project Explorers und wählen Sie im daraufhin erscheinenden Kontextmenü die Option NEW ▸ PROJECT... und wählen Sie im nachfolgend erscheinenden Dialog den Projekttyp *General*. Bestätigen Sie Ihre Wahl über die Schaltfläche *Next >*.



Anlegen eines neuen Projektes

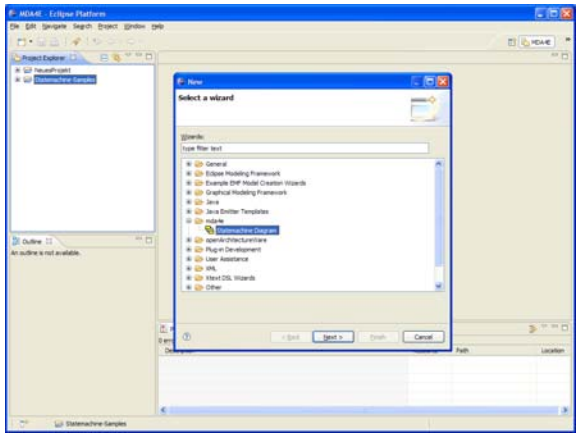
2. Tragen Sie im nun erscheinenden Dialog den Namen des neu anzulegenden Projektes (z. B. *NeuesProjekt*) ein und verlassen Sie den Dialog über die *Finish*-Schaltfläche. Das neu angelegte Projekt erscheint anschließend im Verzeichnisbaum des *Project Explorers* am linken Rand des Editors.



Angabe des Projektnamens (hier NeuesProjekt*)*

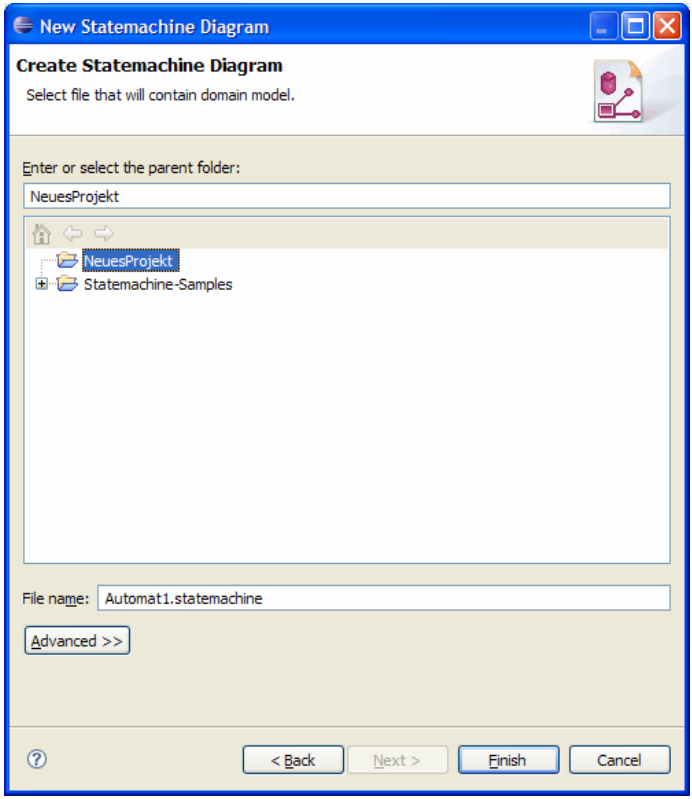
Möchten Sie einem bereits existierenden Projekt – z. B. dem gerade angelegten – einen Zustandsautomaten hinzufügen, so gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf den zugehörigen Projektordner und wählen Sie im daraufhin erscheinenden Kontextmenü die Menüfolge NEW ▸ OTHER… Es erscheint nunmehr ein Dialog zur Auswahl des Dateityps. Hier wählen Sie im Ordner *mda4e* den Typ *Statemachine Diagram* und fahren mit der Betätigung der Schaltfläche *Next >* fort:



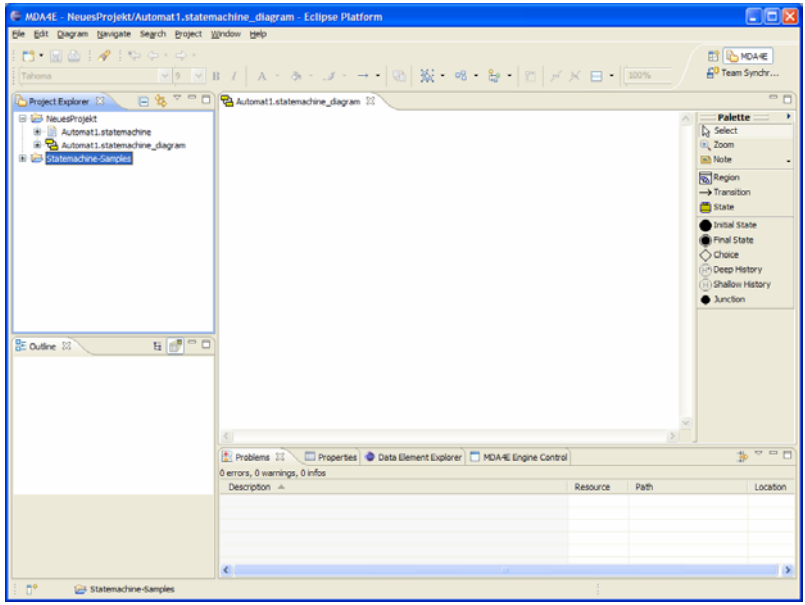
Anlegen eines neuen Zustandsautomaten

2. Nunmehr kann der Dateiname für den Zustandsautomaten gewählt werden. Jeder Zustandsautomat besteht aus zwei Dateien, wobei die erste das Zustandsdiagramm (d. h. die grafische Repräsentation des Automaten) darstellt und die Dateiendung *statemachine_diagram* besitzt, während die zweite (und für die spätere Einbindung in BORIS relevante) Datei mit der Dateiendung *statemachine* den logischen Aufbau des Zustandsautomaten enthält. Ändern Sie den voreingestellten Dateinamen *default.statemachine* auf den gewünschten Namen, z. B. auf *Automat1.statemachine*:



Eingabe des Dateinamens für den Zustandsautomaten

3. Bestätigen Sie den Dialog mit *Finish*. Im Projektordner finden Sie nun die beiden neuen Projektdateien; außerdem wurde der eigentliche Grafikeditor für das zu erstellende Zustandsdiagramm geöffnet. Der Zeichenbereich ist naturgemäß zunächst noch jungfräulich leer.

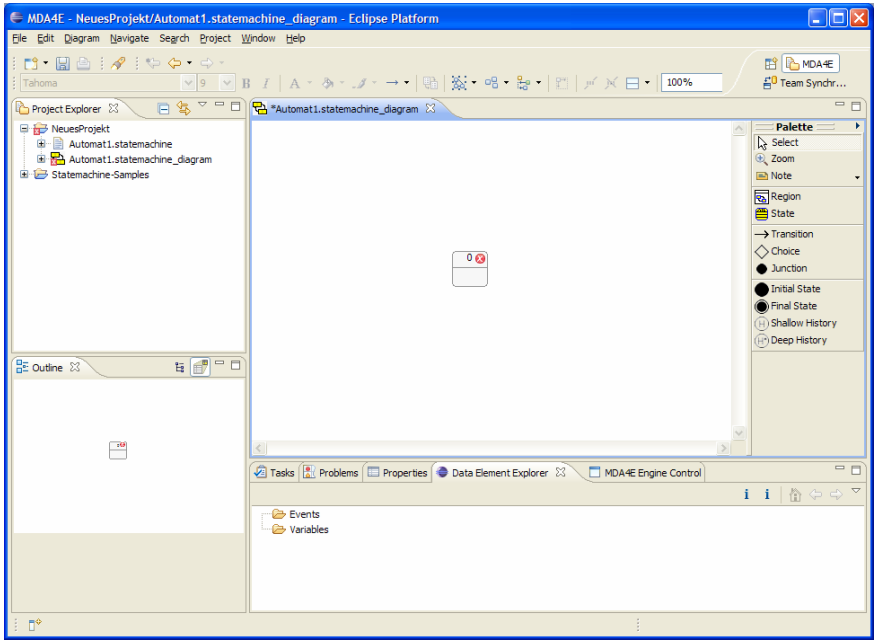


Editor nach Anlegen eines neuen Zustandsautomaten

3.2 Aufbau des Statecharts

Um ein neues Element in ein Zustandsdiagramm einzufügen, gehen Sie wie folgt vor:

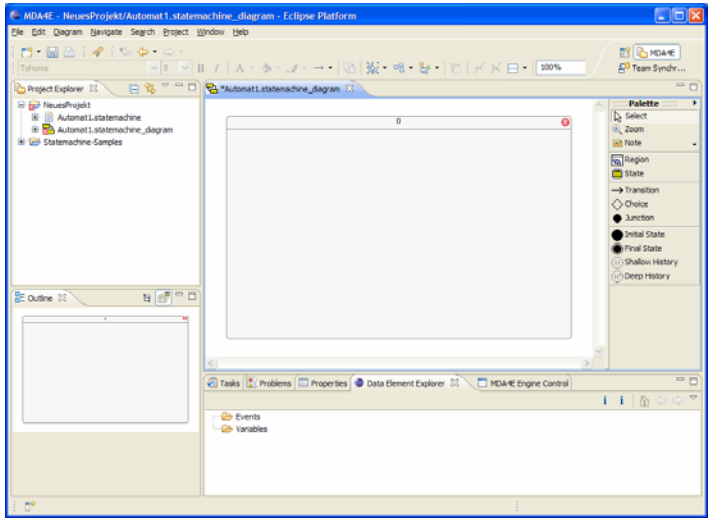
1. Klicken Sie mit der linken Maustaste in der Elementpalette auf den einzufügenden Elementtyp
2. Fahren Sie mit der Maus in den Zeichenbereich und klicken Sie mit der linken Maustaste an die Stelle, an der das Element positioniert werden soll. Das Element erscheint nun in seiner voreingestellten Größe um Zeichenbereich. Nachfolgende Grafik zeigt den Editor nach Einfügen einer neuen *Region*.



Editor nach Einfügen einer Region

3. Soll das Element verschoben werden, selektieren Sie es zunächst durch Anklicken und verschieben es anschließend mit festgehaltener Maustaste.

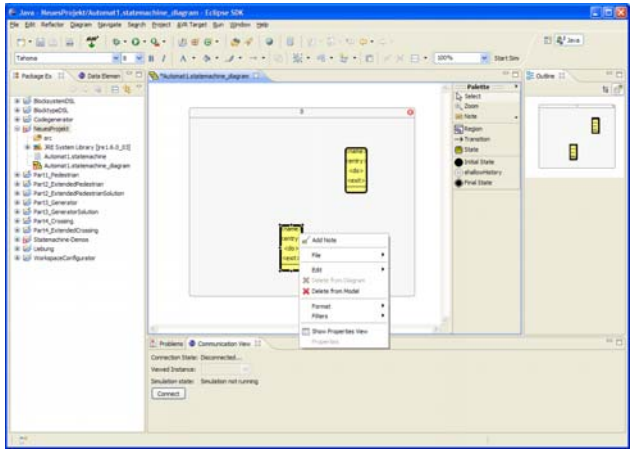
4. Soll das Element vergrößert werden, selektieren Sie es zunächst. Das Element erhält nun an seinen Rändern quadratische „Anfasser“, über die Sie seine Größe in der gewünschten Art und Weise anpassen können. Nachfolgende Bildschirmgrafik zeigt den Editor, nachdem die Region in dieser Weise aufgezogen wurde.



Editor nach Vergrößern der Region

Soll ein Element *gelöscht* werden, so erreichen Sie dies wie folgt:

1. Selektieren Sie das Element zunächst durch Anklicken mit der rechten Maustaste.
2. Wählen Sie im daraufhin erscheinenden Kontextmenü die Option *Delete from model*.

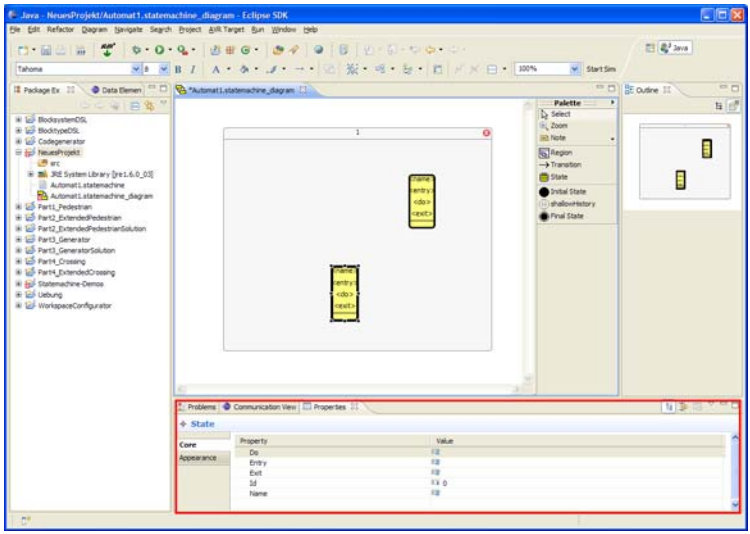


Löschen von Elementen

Um Parameter von Elementen zu ändern, gibt es verschiedene Möglichkeiten. Einige Parameter (z. B. die Priorität einer Region) lassen sich direkt innerhalb des Zeichenfensters durch Anklicken mit der Maus und Eingabe des neuen Werts ändern. Alternativ kann jedoch auch der *Eigenschaftseditor* (*Property view*) benutzt werden. Dazu gehen Sie wie folgt vor:

1. Selektieren Sie das Element zunächst durch Anklicken mit der rechten Maustaste.
2. Wählen Sie im daraufhin erscheinenden Kontextmenü die Option *Show properties view*.

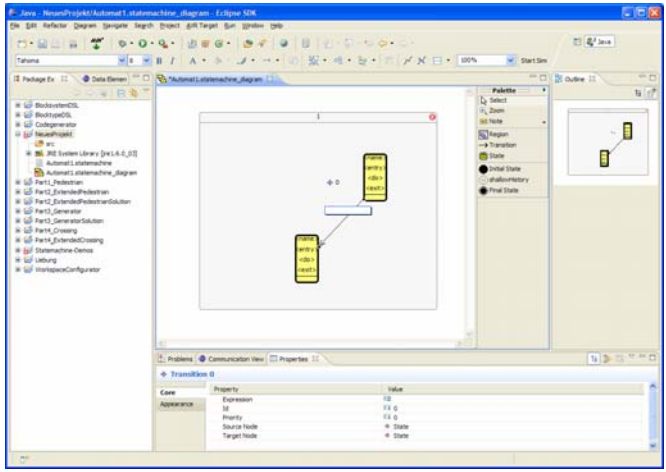
Im unteren Fensterbereich erscheint nun der Eigenschaftseditor, über den sich sämtliche Elementeigenschaften bearbeiten lassen.



Eigenschaftseditor

Eine *Transition* wird wie folgt erstellt:

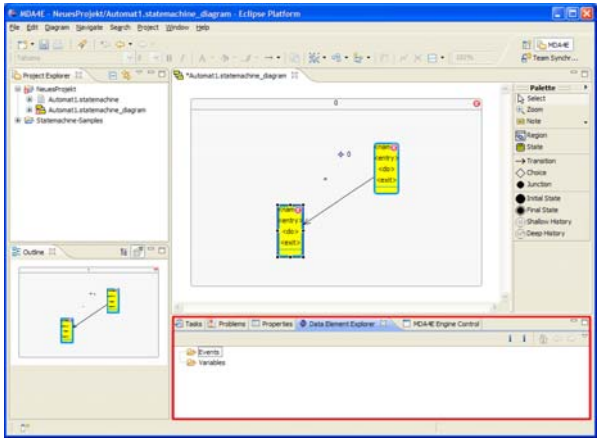
1. Klicken Sie in der Elementpalette zunächst auf den Eintrag *Transition*.
2. Klicken Sie mit der Maus nun zunächst in denjenigen State, von dem die Transition ausgehen soll.
3. Bewegen Sie die Maus nun *bei festgehaltener* Maustaste in den State, in den sie führen soll, und lassen Sie die Maustaste dort los.
4. Geben Sie im nun erscheinenden Editierfeld die Transitionsbedingung(en) an.



Einfügen einer Transition

3.3 Ereignisse (Events)

Ereignisse (Events) werden über den *Data Element Explorer* definiert und angezeigt.

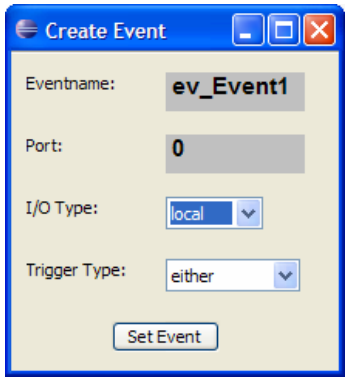


Data Element Explorer (hier noch leer)

Um ein neues Ereignis zu definieren, gehen Sie wie folgt vor:

1. Klicken Sie im *Data Element Explorer* mit der rechten Maustaste auf den Eintrag *Events* und wählen Sie im daraufhin erscheinenden Kontextmenü den Eintrag *Create Event*.

   Es erscheint der nachfolgende Dialog. Geben Sie die gewünschten Parameter ein und verlassen Sie den Dialog über die Schaltfläche *Set Event*.

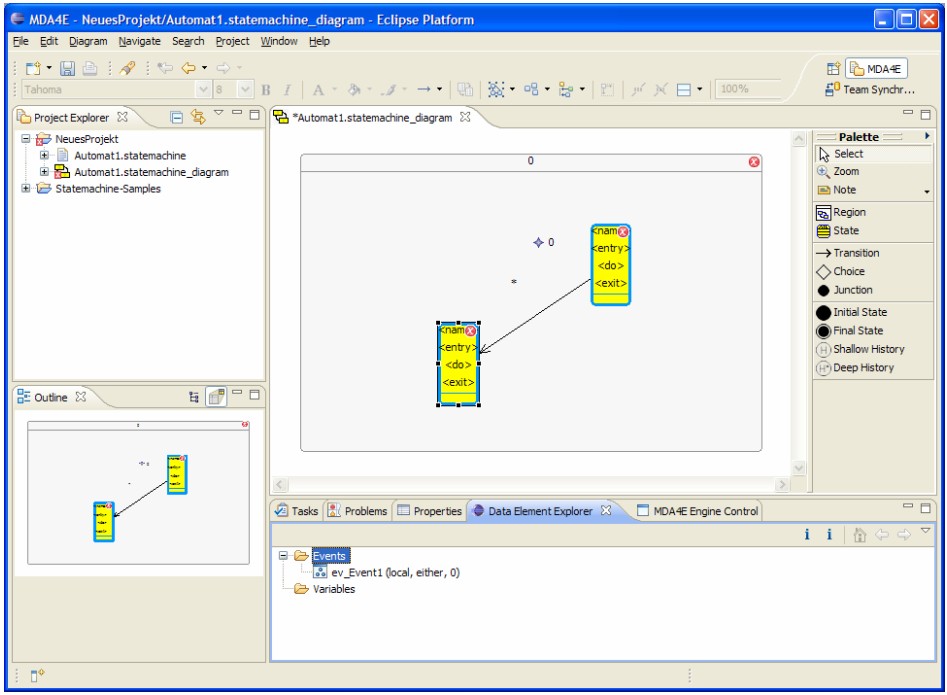


Anlegen eines neuen Events

Die einzelnen Parameter haben folgende Bedeutung:

Parameter	**Bedeutung**
Eventname	Legt den Namen des Events fest (muss eindeutig sein!)
Port	Bestimmt die Reihenfolge der Events in Bezug auf die Zuordnung zu den Ein- bzw. Ausgängen von BORIS-Zustandsautomat-Blöcken
I/O Type	Legt fest, ob der Event nur lokal oder als Ein- bzw. Ausgang eines BORIS-Zustandsautomaten benutzt werden soll
Trigger Type	Legt fest, ob das Event bei steigender und/oder fallender Flanke ausgelöst werden soll oder aber über einen Funktionsaufruf (zurzeit noch nicht implementiert)

Nach dem Anlegen des Events erscheint dieser im *Data Element Explorer* (siehe nachfolgende Bildschirmgrafik) und kann von dort gegebenenfalls durch einen Doppelklick mit der Maus bearbeitet werden.



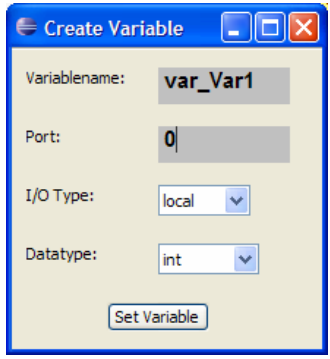
Editor nach Anlegen des Events

3.4 Variablen

Variablen werden wie Ereignisse über den *Data Element Explorer* definiert und angezeigt. Soll eine neue Variable angelegt werden, so gehen Sie wie folgt vor:

1. Klicken Sie im *Data Element Explorer* mit der rechten Maustaste auf den Eintrag Events und wählen Sie im daraufhin erscheinenden Kontextmenü den Eintrag *Create Variable*.

   Es erscheint der nachfolgende Dialog. Geben Sie die gewünschten Parameter ein und verlassen Sie den Dialog über die Schaltfläche *Set Variable*.



Anlegen einer neuen Variablen

Die einzelnen Parameter haben folgende Bedeutung:

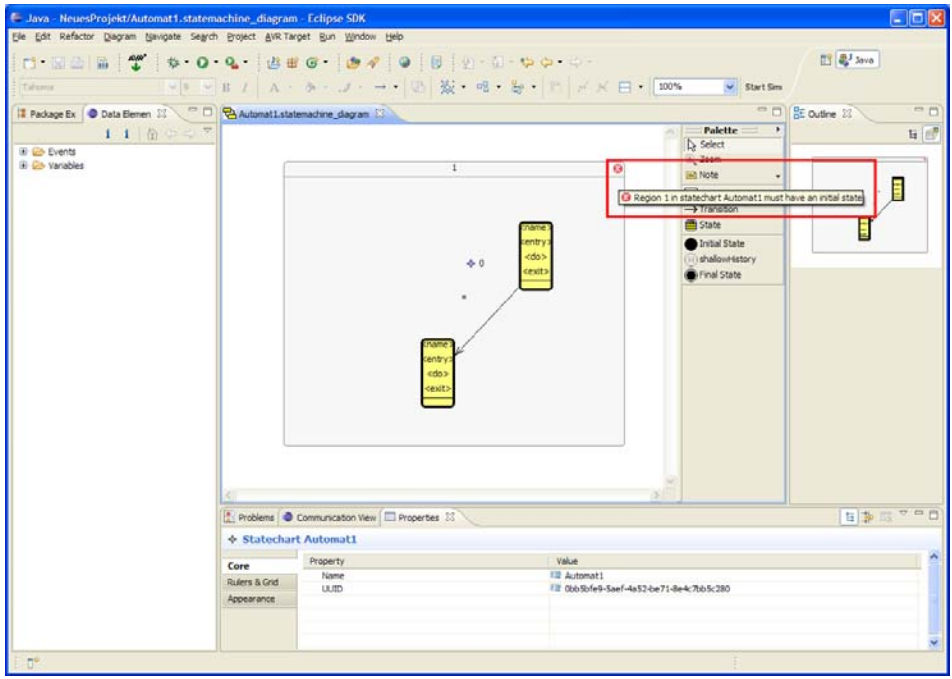
Parameter	Bedeutung
Variablename	Legt den Namen der Variablen fest (muss eindeutig sein!)
Port	Bestimmt die Reihenfolge der Variablen in Bezug auf die Zuordnung zu den Ein- bzw. Ausgängen von BORIS-Zustandsautomat-Blöcken
I/O Type	Legt fest, ob die Variable nur lokal oder als Ein- bzw. Ausgang eines BORIS-Zustandsautomaten benutzt werden soll
Datatype	Bestimmt den Datentypen der Variablen (Ganzzahl, Fließkomma oder Boolean)

Nach dem Anlegen der Variablen erscheint diese wie die Ereignisse im *Data Element Explorer* und kann von dort gegebenenfalls durch einen Doppelklick mit der Maus bearbeitet werden.

3.5 Überprüfungen zur Entwurfszeit

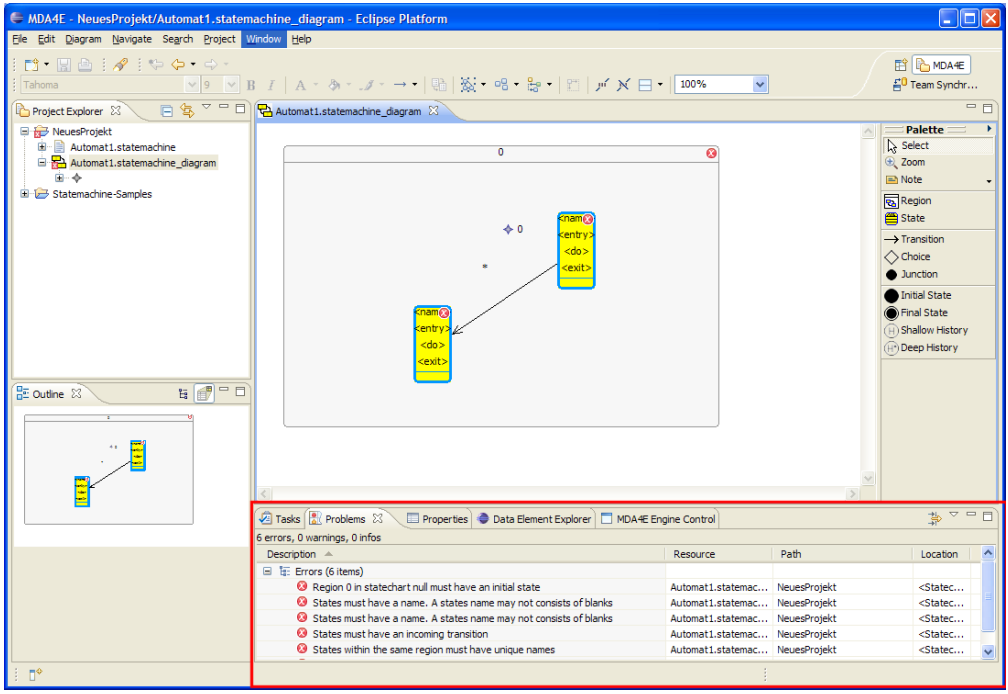
Beim Entwurf von Zustandsautomaten ist eine Reihe von Entwurfsrichtlinien zu beachten, von denen ein Großteil bereits zur Entwurfszeit quasi „im Hintergrund“ überprüft wird; werden Richtlinien verletzt, so wird dies unmittelbar angezeigt, sodass das Zustandsdiagramm direkt korrigiert werden kann. Dazu erscheint innerhalb des Zeichenfensters ein Warnsignal in Form eines weißen Kreuzes auf einem roten Grund, bei dessen Überschreiten mit der Maus ein entsprechender Warnhinweis erscheint. Nachfolgende Bildschirmgrafik zeigt dazu

ein Beispiel: Hier besitzt die angelegte Region (noch) keinen Anfangszustand, sodass der entsprechende Zustandsautomat noch nicht vollständig ist. Wird nunmehr ein *Initial State* eingefügt, verschwindet das Warnsymbol nach einigen Sekunden automatisch.



Überprüfung der Entwurfsrichtlinien

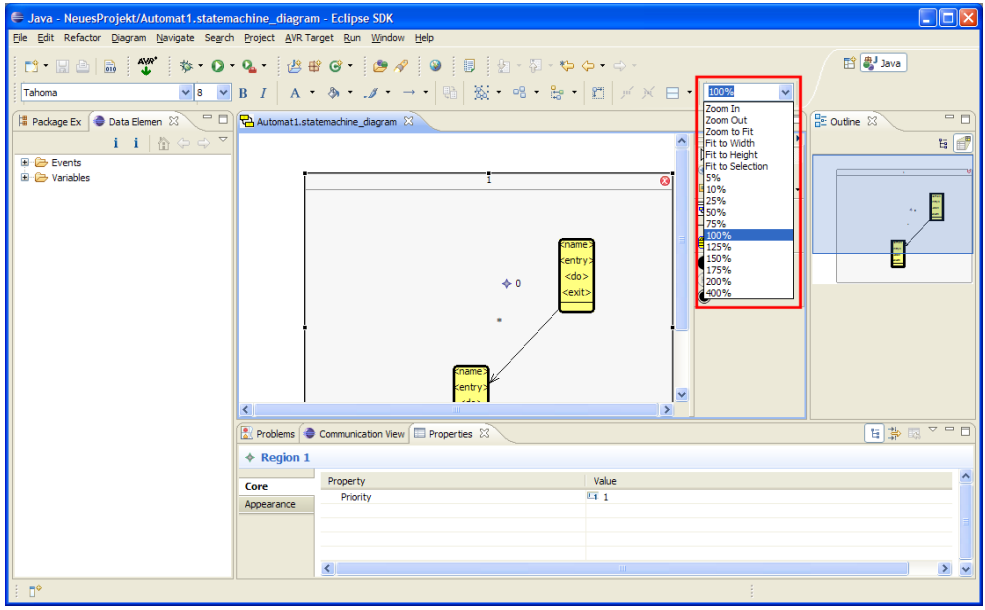
Außerdem werden alle aktuellen Fehler- bzw. Warnmeldungen in einer Liste in der *Problems View* unterhalb des Zeichenbereichs angezeigt (siehe nachfolgende Bildschirmgrafik).



Anzeige von Fehlern und Warnungen im Problems*-Fenster*

3.6 Ändern des Bildschirmausschnitts

Bei komplexeren Zustandsdiagrammen reicht der Zeichenbereich häufig nicht aus, um das komplette Diagramm darzustellen bzw. Details zu erkennen. Der Editor bietet daher die Möglichkeit, den sichtbaren Ausschnitt nach Belieben anzupassen. Die entsprechenden Optionen finden Sie im Listenfenster in der oberen rechten Ecke des Editors (siehe nachfolgende Bildschirmgrafik).



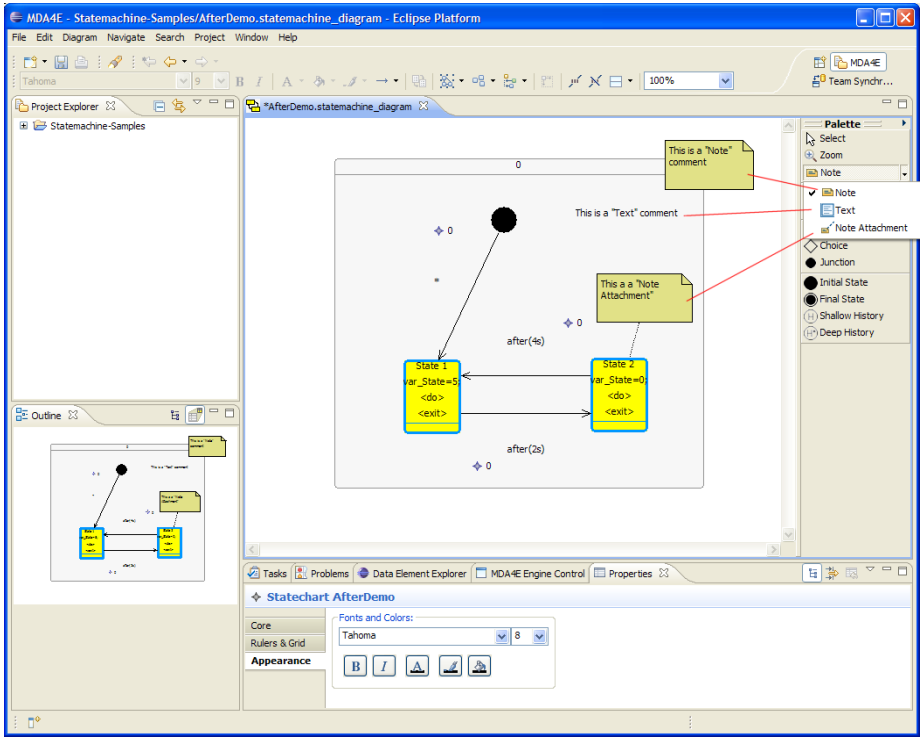
Wahl des Bildschirmausschnitts

3.7 Einfügen von Kommentaren

Zur besseren Dokumentation von Zustandsdiagrammen bietet der Statechart-Editor drei unterschiedliche Arten von Kommentartexten, die ebenfalls über die Elementpalette verfügbar sind:

Note	Notizzettel („Post it“)
Text	Einfacher Kommentartext
Note Attachment	Notizzettel mit Referenzlinie

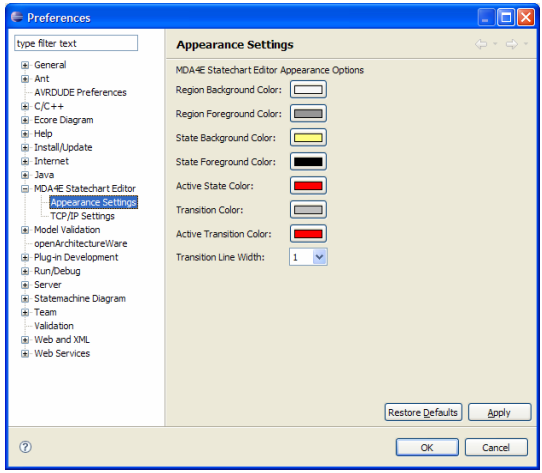
Nachfolgende Bildschirmgrafik zeigt jeweils ein Beispiel für die unterschiedlichen Arten von Kommentartexten. Die Eigenschaften (Textfarbe, Font, Hintergrundfarbe etc.) des Kommentartextes kann jeweils über die *Property View* eingestellt werden.



Arten von Kommentartexten

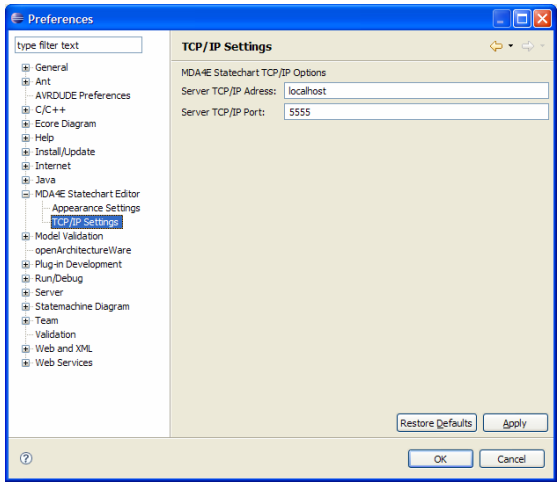
3.8 Weitere Optionen

Eine Reihe von Editor-Optionen lassen sich über den Menüpunkt WINDOW ▸ PREFERENCES... erreichen. Unter dem Menüpunkt *Appearence Settings* befinden sich zunächst diverse Farbeinstellungen:



Farbeinstellungen

Unter dem Menüpunkt *TCP/IP Settings* finden Sie die Einstellungen zur TCP/IP-Kommunikation zwischen dem Statechart-Editor und BORIS; hier ist die IP-Adresse des BORIS-Rechners sowie die Nummer des Ports anzugeben, über den die Kommunikation erfolgen soll:

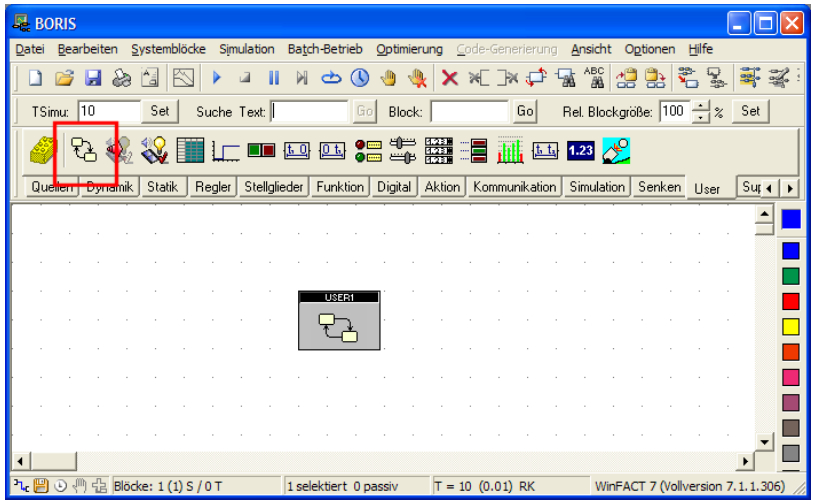


TCP/IP-Einstellungen

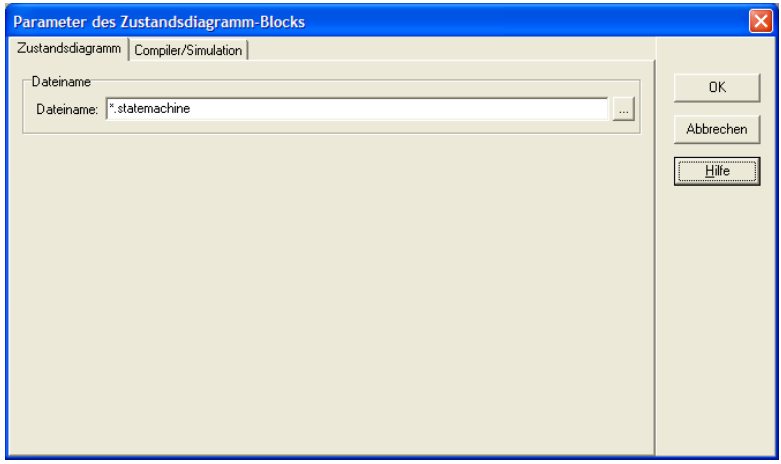
4 Statemachine-Simulation unter BORIS

4.1 Einfügen und Parametrieren eines Statemachine-Systemblocks

Ein (zunächst leerer) Zustandsautomaten-Block wird unter BORIS über das zugehörige Icon auf dem Reiter *User* der Systemblock-Toolbar eingefügt. Der eingefügte Block besitzt zunächst weder Ein- noch Ausgänge:

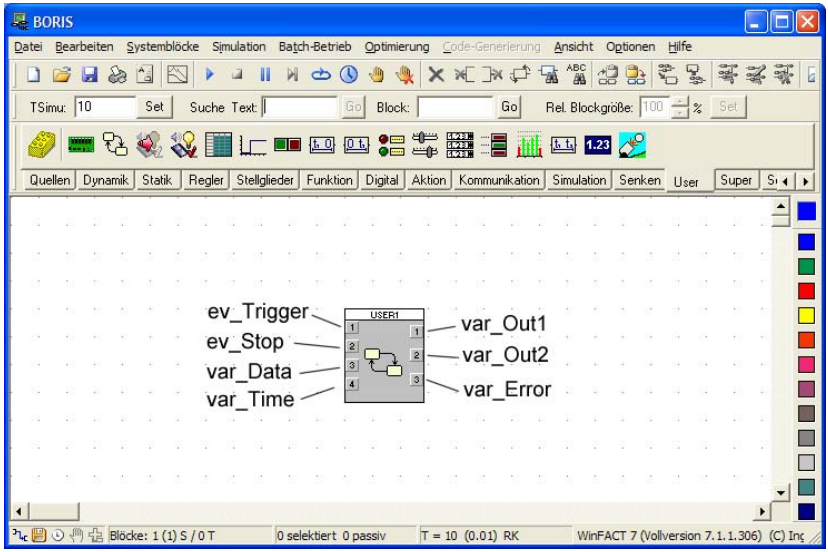


Durch einen Doppelklick auf den Systemblock und Betätigung der Schaltfläche *Dialog…* im daraufhin erscheinenden Dialog gelangen Sie in den eigentlichen Parameterdialog des Zustandsautomatenblocks. Dieser ermöglicht über den mit *Zustandsdiagramm* beschrifteten Reiter zunächst die Auswahl des Zustandsautomaten:



Nach der Auswahl des Zustandsautomaten und Rückkehr ins BORIS-Hauptfenster erhält der Systemblock nunmehr die innerhalb des Zustandsautomaten definierten Ein- und Ausgänge, d. h. Events und Variablen. Dabei spielen die im Zustandsautomaten-Editor vergebenen Portnummern eine entscheidende Rolle, da sie die Reihenfolge festlegen, in der die Events und Variablen angeordnet werden. Zunächst werden alle Events in der Reihenfolge steigender Portnummer zugeordnet, anschließend die Variablen. Nachfolgende Bildschirmgrafik zeigt beispielhaft die Ein-/Ausgangsbelegung eines Zustandsautomaten-Blocks, für den folgende Events und Variablen definiert wurden:

Name	**Typ**	**Portnummer**
ev_Trigger	Event-Eingang	0
ev_Stop	Event-Eingang	1
var_Data	Eingangsvariable	0
var_Time	Eingangsvariable	1
var_Out1	Ausgangsvariable	0
var_Out2	Ausgangsvariable	1
var_Error	Ausgangsvariable	2

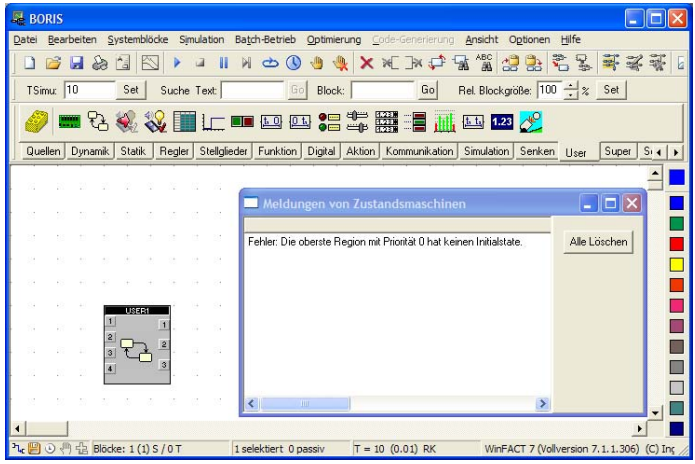


Der Block kann nun wie gewohnt mit anderen Systemblöcken verbunden werden.

Wird das einem bereits in der Systemstruktur befindlichen Zustandsautomaten-Block zugrunde liegende Zustandsdiagramm im Statechart-Editor in der Weise geändert, dass sich die Anzahl der Ein- und/oder Ausgänge ändert, so ist zu beachten, dass BORIS diese Tatsache *nicht* automatisch erkennt! Um den BORIS-Systemblock anzupassen, muss in diesem Fall daher durch Doppelklick auf den Block die zugehörige Zustandsautomatendatei neu eingelesen werden.

4.2 Fehler- und Kontrollmeldungen des Zustandsautomaten-Blocks

BORIS überprüft in regelmäßigen Abständen, ob die einem Zustandsautomaten-Block zugrunde liegende Datei im Editor modifiziert wurde und führt gegebenenfalls automatisch eine Neucompilierung durch. Treten dabei Probleme irgendwelcher Art auf, erscheint ein Kontrollfenster, in dem eine entsprechende Meldung auf dieses Problem hinweist. Nachfolgende Bildschirmgrafik zeigt dieses Kontrollfenster für den Fall, dass ein Zustandsautomat keinen Anfangszustand (Initial State) aufweist. Das Kontrollfenster kann jederzeit geschlossen werden; es erscheint bei Bedarf dann wieder automatisch.

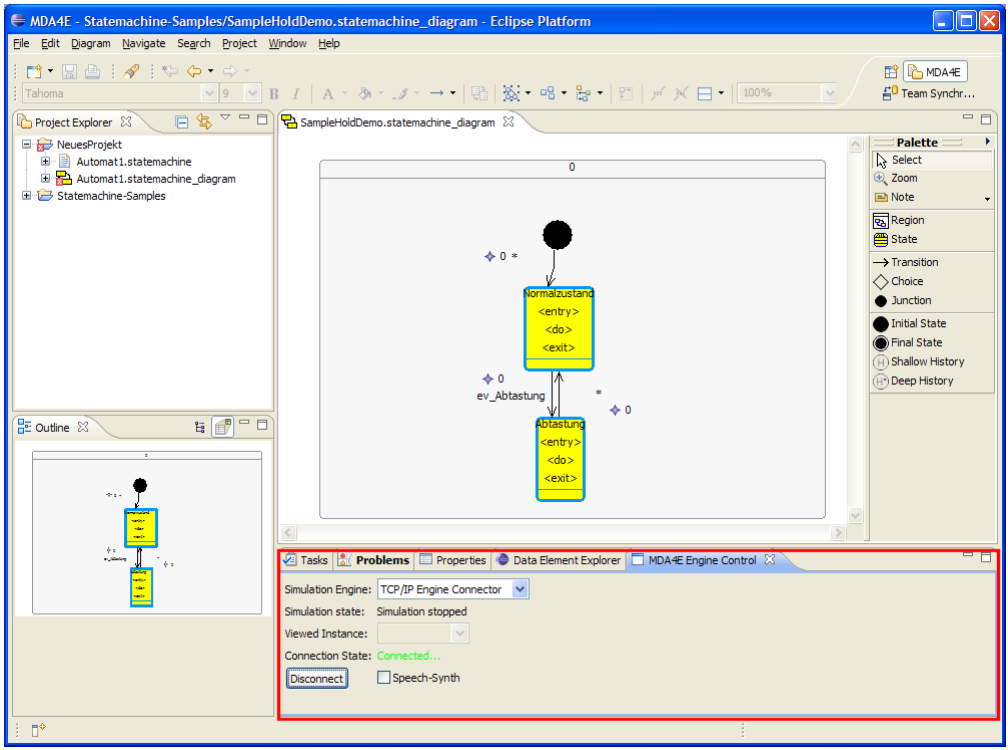


4.3 Kommunikation mit dem Statechart-Editor

Während der Simulation können die Abläufe im „Inneren“ des Zustandsautomaten auf Wunsch im Statechart-Editor visualisiert werden; dieses Leistungsmerkmal ist insbesondere für Demonstrationszwecke oder zur Fehlersuche von Interesse. Die Kommunikation zwischen BORIS und dem Statechart-Editor findet dabei über das TCP/IP-Protokoll statt, sodass beide Anwendungen gegebenenfalls auch auf unterschiedlichen Rechnern laufen können. Die entsprechenden BORIS-seitigen Einstellungen finden Sie auf dem Reiter *Compiler/Simulation* im Parameterdialog des Zustandsautomaten-Blocks:

Innerhalb von BORIS muss lediglich der TCP-Port (standardmäßig 5555) angegeben werden, über den die Kommunikation stattfinden soll. Die Angaben zu Rechnername und IP-Adresse dienen lediglich der Information.

Der Verbindungsaufbau selbst erfolgt aus dem Statechart-Editor über die *MDA4E Engine Control*. Durch Betätigung der Schaltfläche *Connect* wird eine Verbindung zu BORIS aufgebaut; die Felder *Connection State* und *Simulation State* zeigen den aktuellen Zustand von BORIS an. Befindet sich derselbe Zustandsautomat mehrfach innerhalb einer BORIS-Systemstruktur, so kann über die Listbox *Viewed Instance* der darzustellende Block ausgewählt werden. Wird die Option *Speech-Synth* aktiviert, erfolgt zusätzlich zum Highlighting aktiver States eine Sprachausgabe:



Da während der Kommunikation zwischen BORIS und dem Statechart-Editor keinerlei Zeitsynchronisation stattfindet, sollte BORIS ein wenig „gebremst“ werden, da die Visualisierung innerhalb des Statechart-Editors ansonsten nach einer gewissen Zeit „hinterherhinkt“. Es empfiehlt sich daher, BORIS in solchen Fällen im Echtzeitmodus (beispielsweise mit ei-

ner Schrittweite von 100 ms oder mehr) zu betreiben oder aber den Einzelschrittmodus von BORIS zu benutzen.

4.4 Compiler-Einstellungen

Für die Compilierung der Zustandsautomaten zu Beginn einer BORIS-Simulation ist ein C-Compiler erforderlich, der automatisch mit der *State Machine Workbench* installiert wird. Soll anstelle dieses Compilers ein anderer Compiler zum Einsatz kommen, so finden Sie die entsprechenden Einstellungen auf dem Reiter *Compiler/Simulation* im Parameterdialog des Zustandsautomaten-Blocks:

Änderungen an diesen Einstellungen sollten nur von erfahrenen Anwendern vorgenommen werden, da eine fehlerfreie Abarbeitung der Zustandsautomaten in BORIS ansonsten nicht gewährleistet werden kann!

5 C-Code-Generierung

Sofern der *AutoCode-Generator* für das blockorientierte Simulationssystem BORIS erworben wurde, können Zustandsautomaten wie „normale“ BORIS-Systemblöcke zu ANSI-C-Code generiert werden. Dies gilt selbstverständlich nicht nur für den Zustandsautomaten selbst, sondern auch für Strukturen aus mehreren Zustandsautomaten und/oder anderen Systemblöcken. Hinweise dazu finden Sie im Benutzer-Handbuch des AutoCode-Generators.

6 Mitgelieferte Beispiele

Für den einfachen Einstieg sowie als Anregung für eigene Entwicklungen werden mit der *State Machine Workbench* einige Beispiele geliefert, die Sie nach der Installation im Projektordner (*Project Explorer*) des Statechart-Editors finden. Jedes Beispiel besteht aus drei Dateien, die sich lediglich in der Dateierweiterung unterscheiden. Dateien mit der Endung *statemachine_diagram* enthalten jeweils das zum Beispiel gehörende Zustandsdiagramm (grafische Repräsentation des Zustandsautomaten), Dateien mit der Endung *statemachine* den eigentlichen Zustandsautomaten (logischer Aufbau), Dateien mit der Endung *bsy* die zum Beispiel gehörige BORIS-Simulationsstruktur (sofern vorhanden).

Dateiname	**Beispiel**
AfterDemo.xyz	Demo der *after()*-Klausel
Ampelsteuerung.xyz	Ampelsteuerung
FunctionByChoiceTest.xyz	Demo des *Choice*-Elements
Heatcontrol.xyz	Heizungsregelung
JunctionTest.xyz	Demo des *Junction*-Elements
LevelControl.xyz	Füllstandsregelung

PedestrianTrafficLightGER.xyz	Fußgängerampel
Schiebedach.xyz	Schiebedach
SingleDirection.xyz	Einfache Verkehrsampel
ToggleDemo.xyz	Zustandsautomat als Flip-Flop
TrafficLightGER.xyz	Komplexe Verkehrsampel
SampleHoldDemo.xyz	Zustandsautomat als Abtast-Halteglied

7 Literatur

[ECL] www.ECLIPSE.org

[MDA] www.mda4e.org Forschungsprojekt: Modellgetriebene Softwareentwicklung für Embedded Systems

[XML] Hauser, T.: XML Standards, Entwickler.press 2006

[AUT] Hopcroft, J. E., et al: Einführung in die Automatentheorie, Addison-Wesley 2003

[WF7] Benutzerhandbuch WinFACT 7, Ingenieurbüro Dr. Kahlert, Hamm