



Modellierung und Simulation mit Zustandsautomaten - Schwerpunkt BetterState

Dipl.-Inform. Thomas Stingl, Dipl.-Ing. Rico Dreier

Werkzeugbereitstellung: H. Witter, H. Hötterges, ISI

ASIM Meeting Aachen, 99/03/01

FZI Karlsruhe

Elektronische Systeme und Mikrosysteme

Prof. Dr.-Ing. K.-D. Müller-Glaser

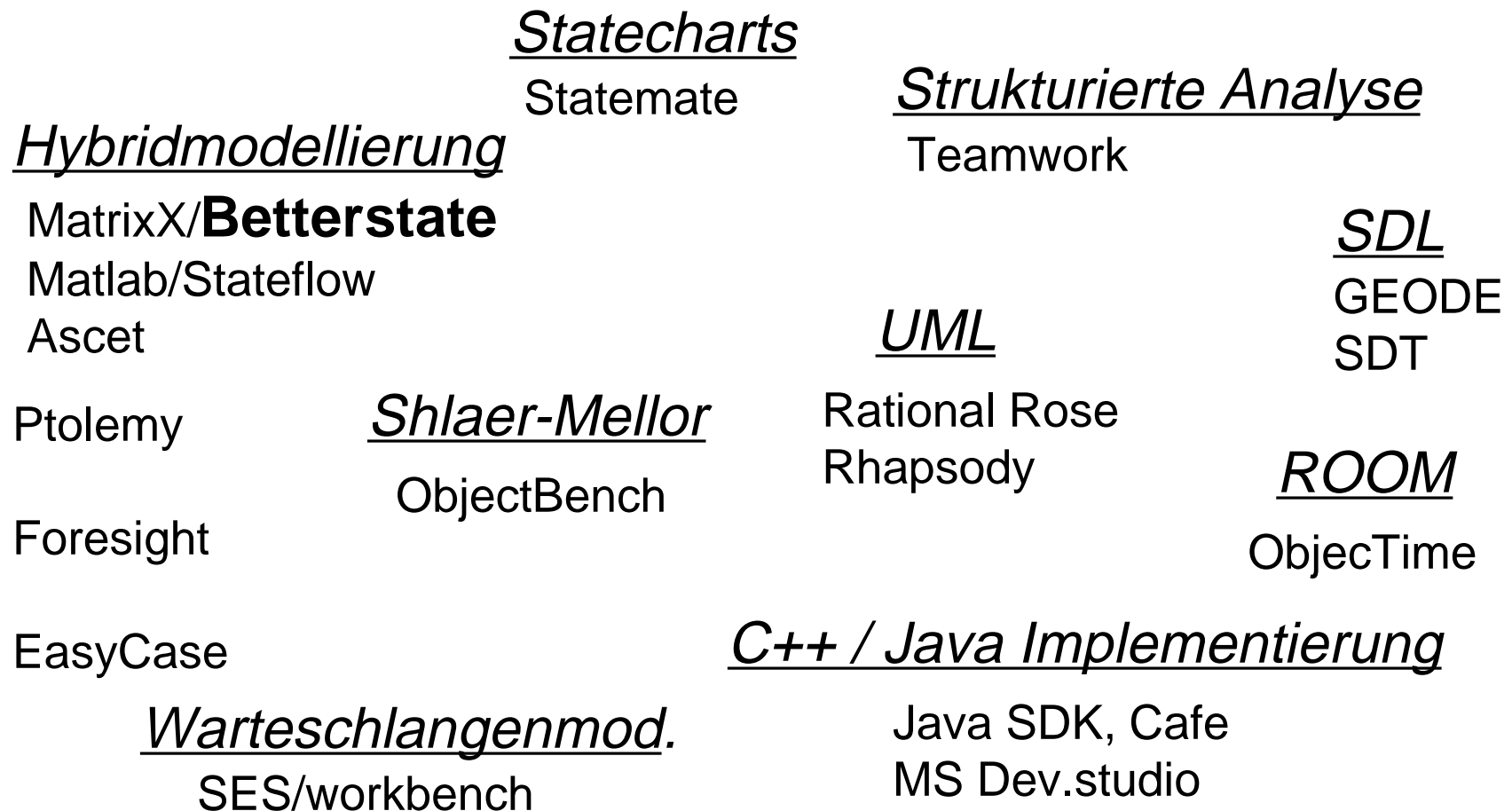
Dr. S. Schmerler

esm@fzi.de



- Grundlagen**
- Umfeld**
- Arbeitserfahrungen mit BetterState**

Einordnung - Entwicklungswerkzeuge im Umfeld

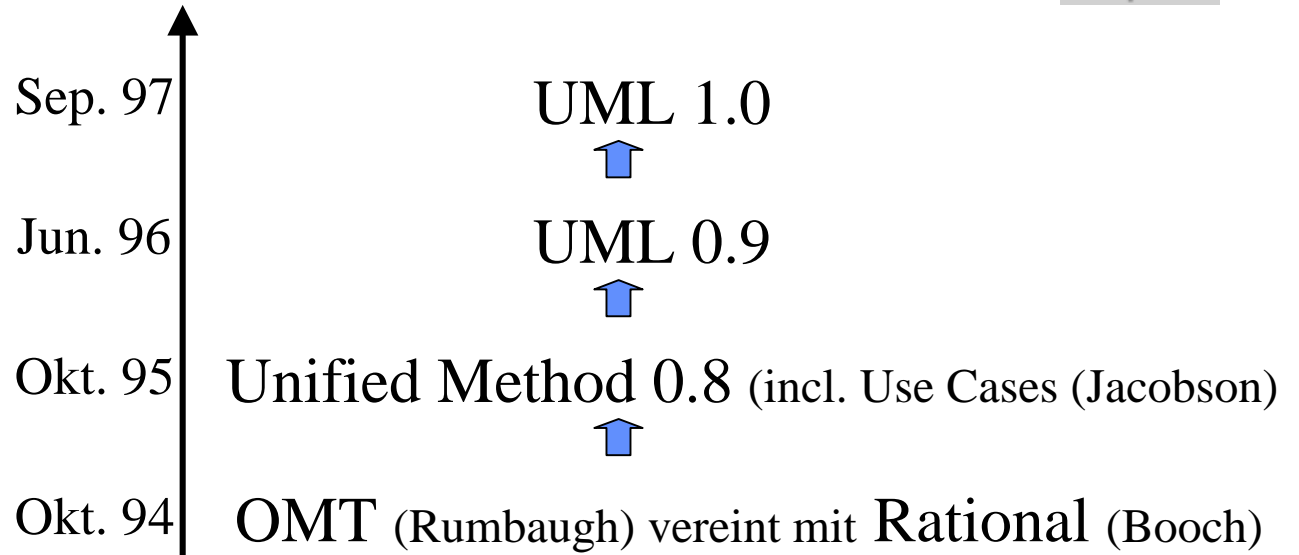


UML - Unified Modelling Language



□ Vorteile

- Sammelbecken der früheren OO-Ansätze
- Etabliert sich als Standard, immer mehr Werkzeuge
- Sehr übersichtliche Modellierung

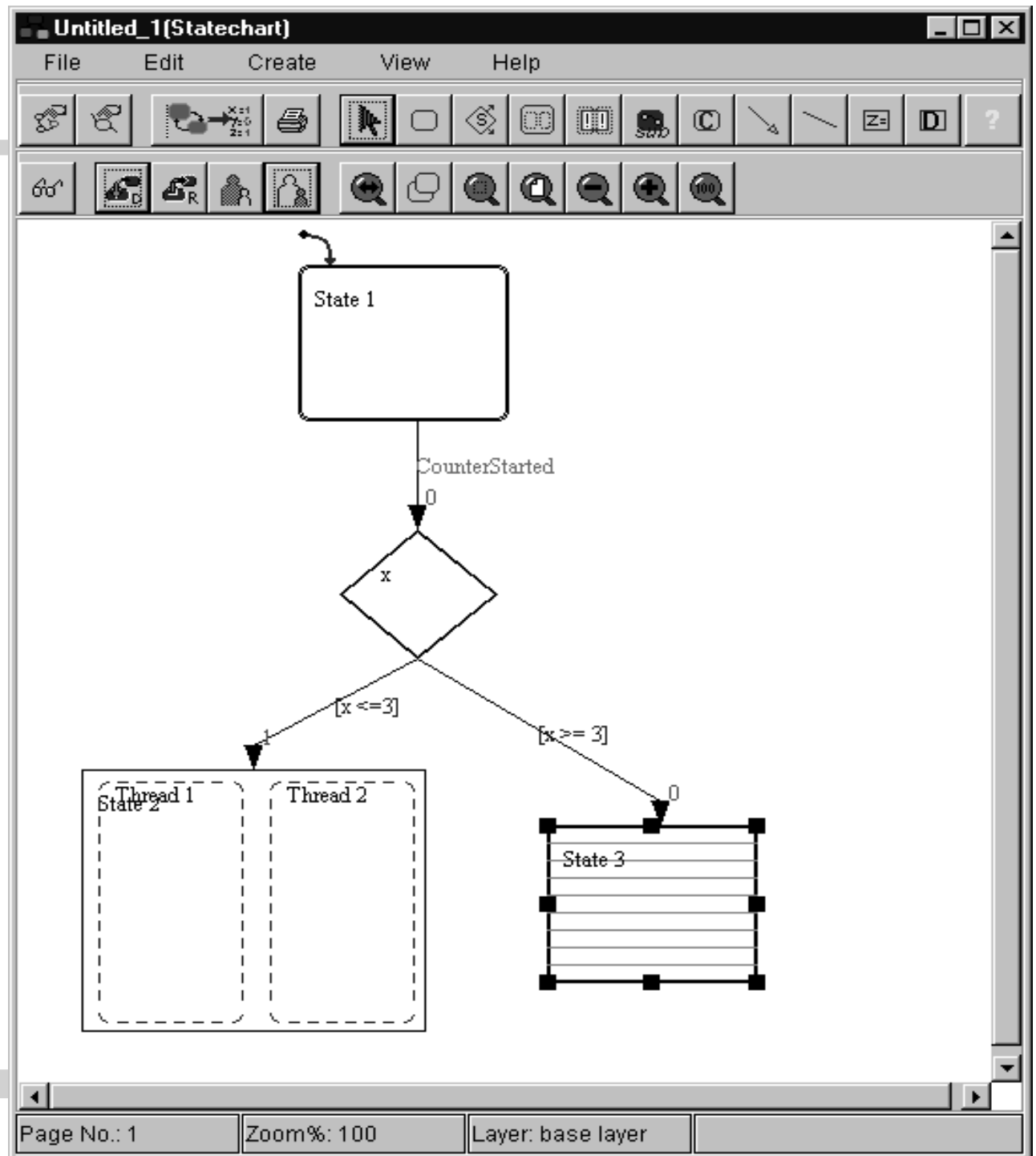


□ Probleme

- Fehlende Automatisierung zwischen den UML Komponenten
 - Use Cases, Klassendiagramme, Interaktionsdiagramme
- Codegenerierung problematisch
- Wichtiger Teil von Systemen hat kontinuierlichen Anteil, kann nicht direkt in UML modelliert werden
 - Verknüpfung von Werkzeugen für beide Welten ...

Beispielmodell

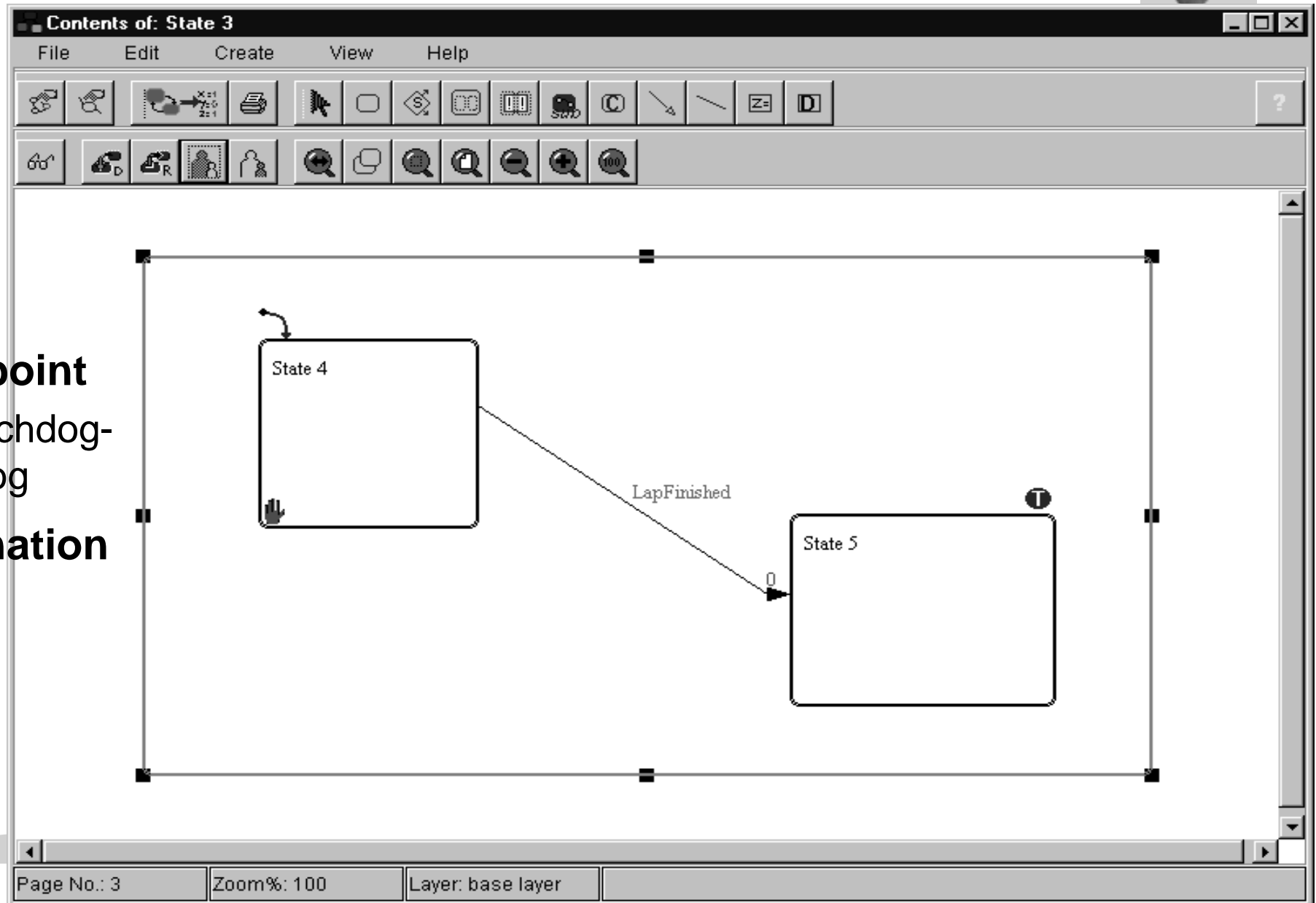
- ❑ BetterState 5.0 beta
- ❑ UML angepaßt
- ❑ Default entry
- ❑ Events, Conditions
- ❑ Visual Switch
- ❑ Parallelismus
- ❑ Prioritäten bei Transaktionen
- ❑ Hierarchie



Beispiel - Subchart



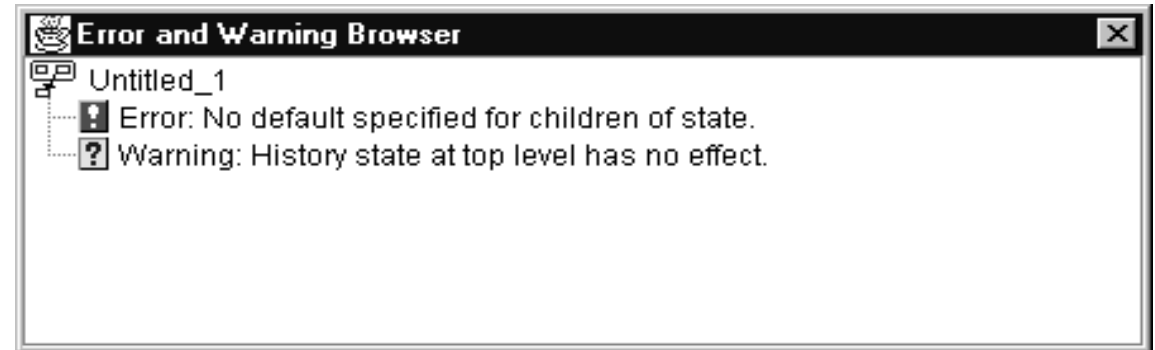
- **Breakpoint**
 - Watchdog-dialog
- **Termination state**





□ Timer realisierbar durch

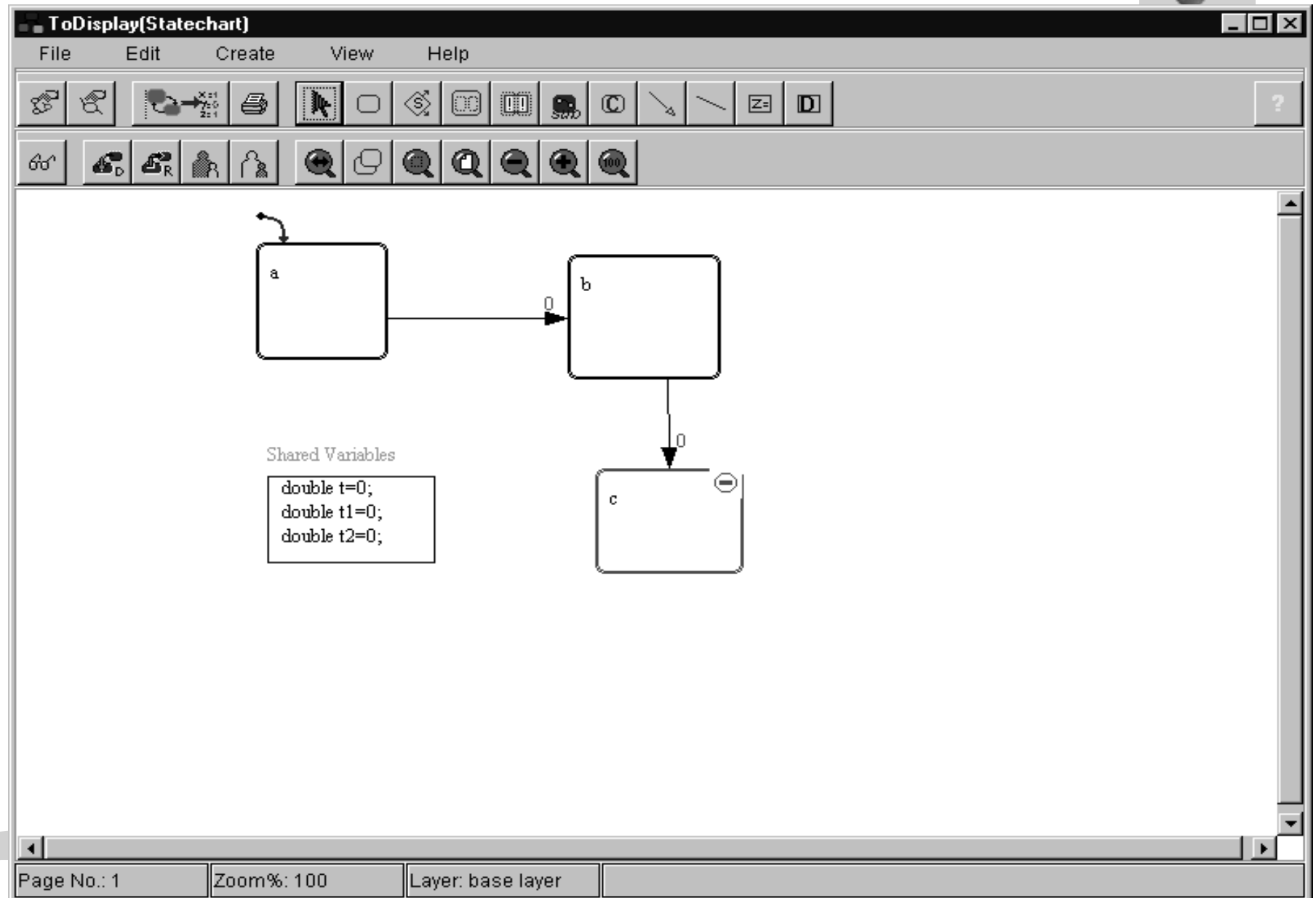
- Generator aus MatrixX
- C-Funktion



□ Simulation - animiertes Playback

- 1. Kopplung mit MatrixX
 - Verwendung von Taktgeneratoren
- 2. Stimuligenerierung mit C-Funktionen
 - z.B. bei Ein-/Austritt eines Zustands
- 3. Erzeugung der Stimuli interaktiv mittels VC++ Debugger
 - main() wird vom Benutzer erstellt, zur Kommunikation mit dem „Controller“

Einfaches Modell



Einfaches Modell, C-Code



State b (on-entry action)

```
t=timeGetTime();  
printf("Zeit %10.8e",t);
```

State c (on-entry action)

```
t1=timeGetTime();
```

State c (during action)

```
if ((t2>=t1+3000)&&(t2<t1+3001))  
    printf("Zeit %10.8e",t2);
```

```
// main.c  
#include "setevent.h"  
  
void main() {  
    BSReset();  
    while (1) {  
        BSFire();  
    }  
}  
  
// setevent.c  
int BSFire( )  
{  
    int nRetVal = 1;  
    nRetVal &= CHRT_ToDisplay();  
    return(nRetVal);  
}  
  
int BSReset( )  
{  
    int nRetVal = 1;  
    nRetVal &=  
CHRT_ToDisplay_BSReset();  
    return(nRetVal);  
}
```

Modellierung

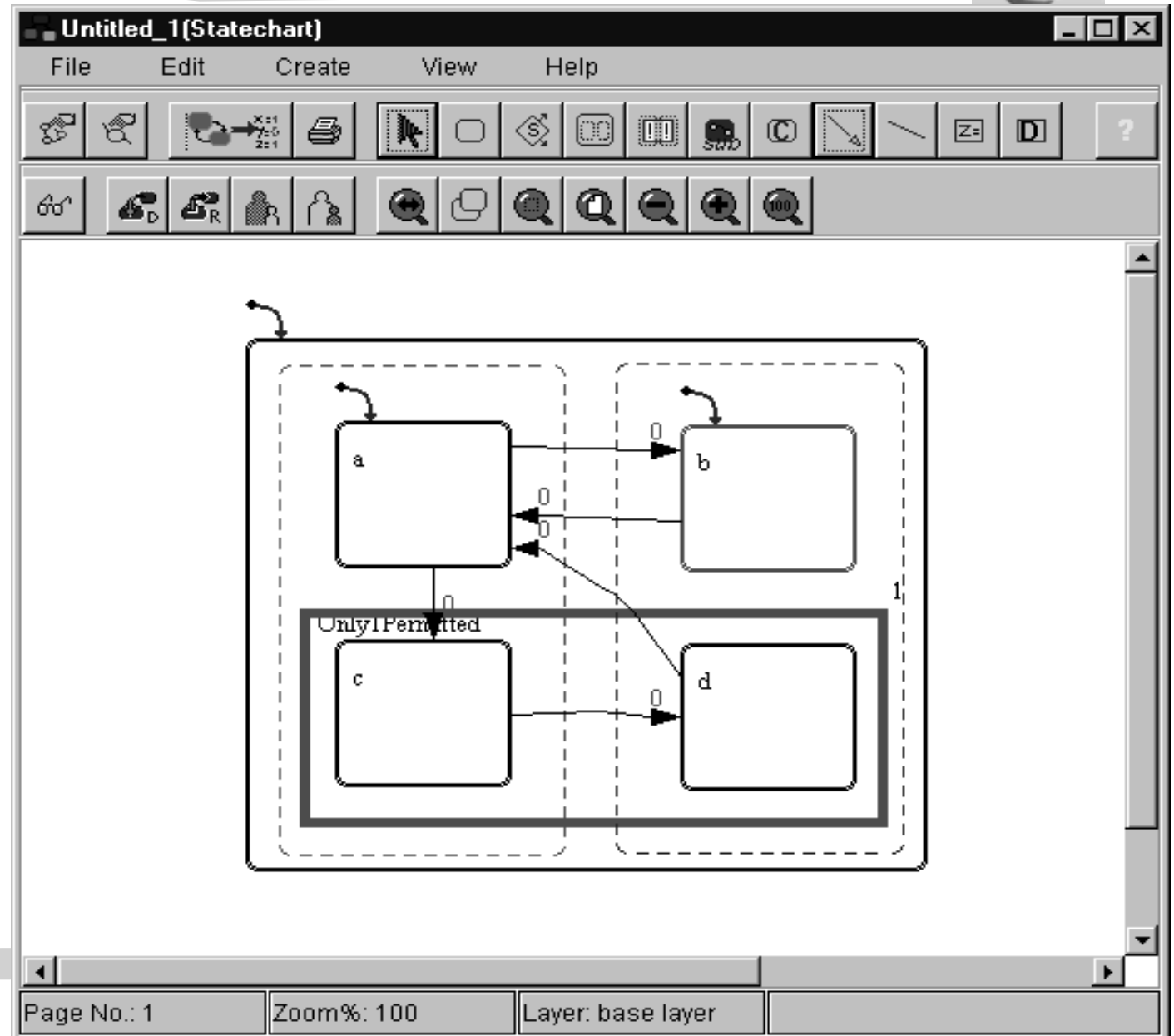


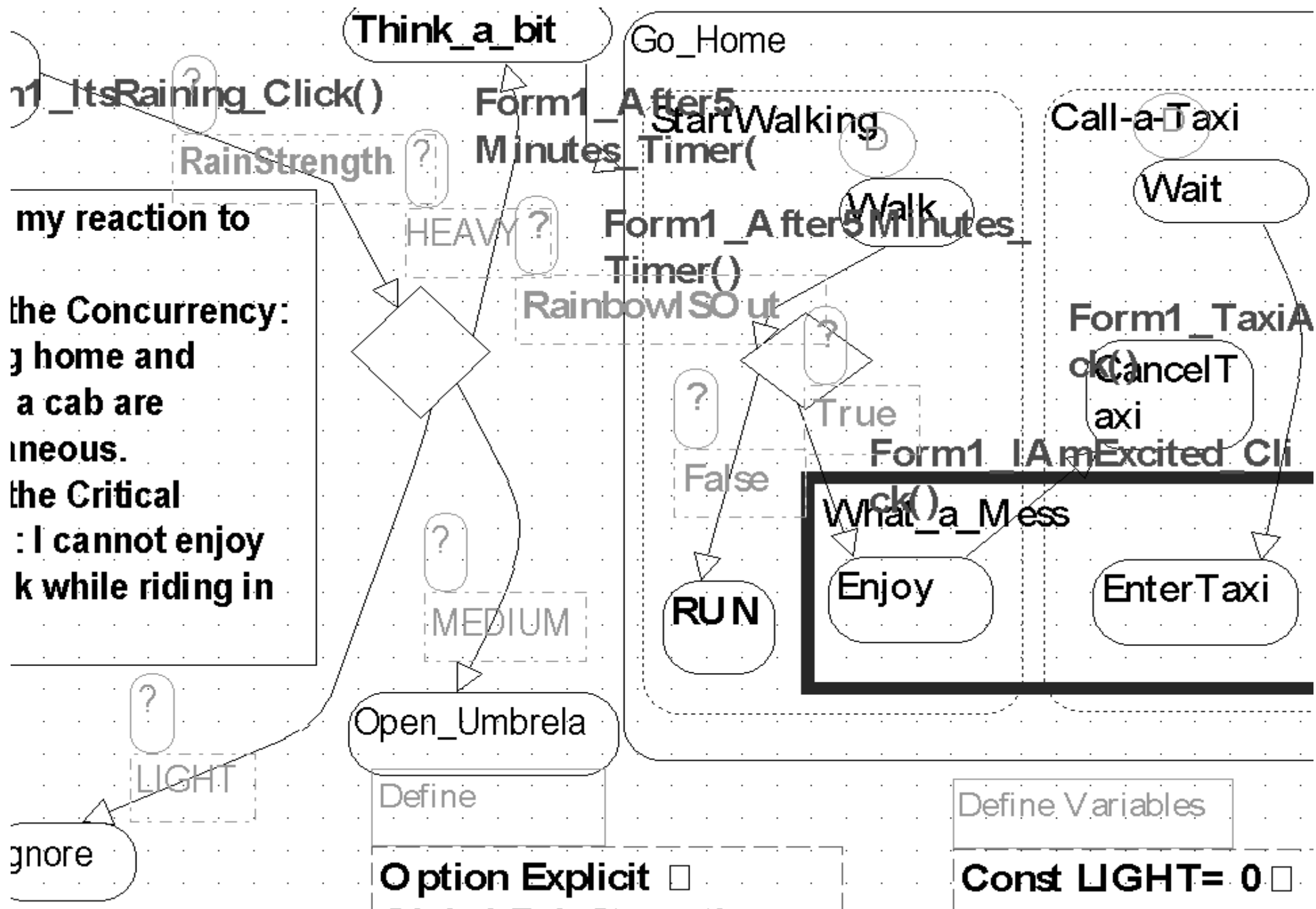
□ Zustandsmodellierung nach Harel

- auch parametrisierbare Charts (generische Statecharts)

□ Critical regions

- Anzahl gleichzeitig besuchbarer Zustände beschränkt
- Wird bei Simulation überwacht





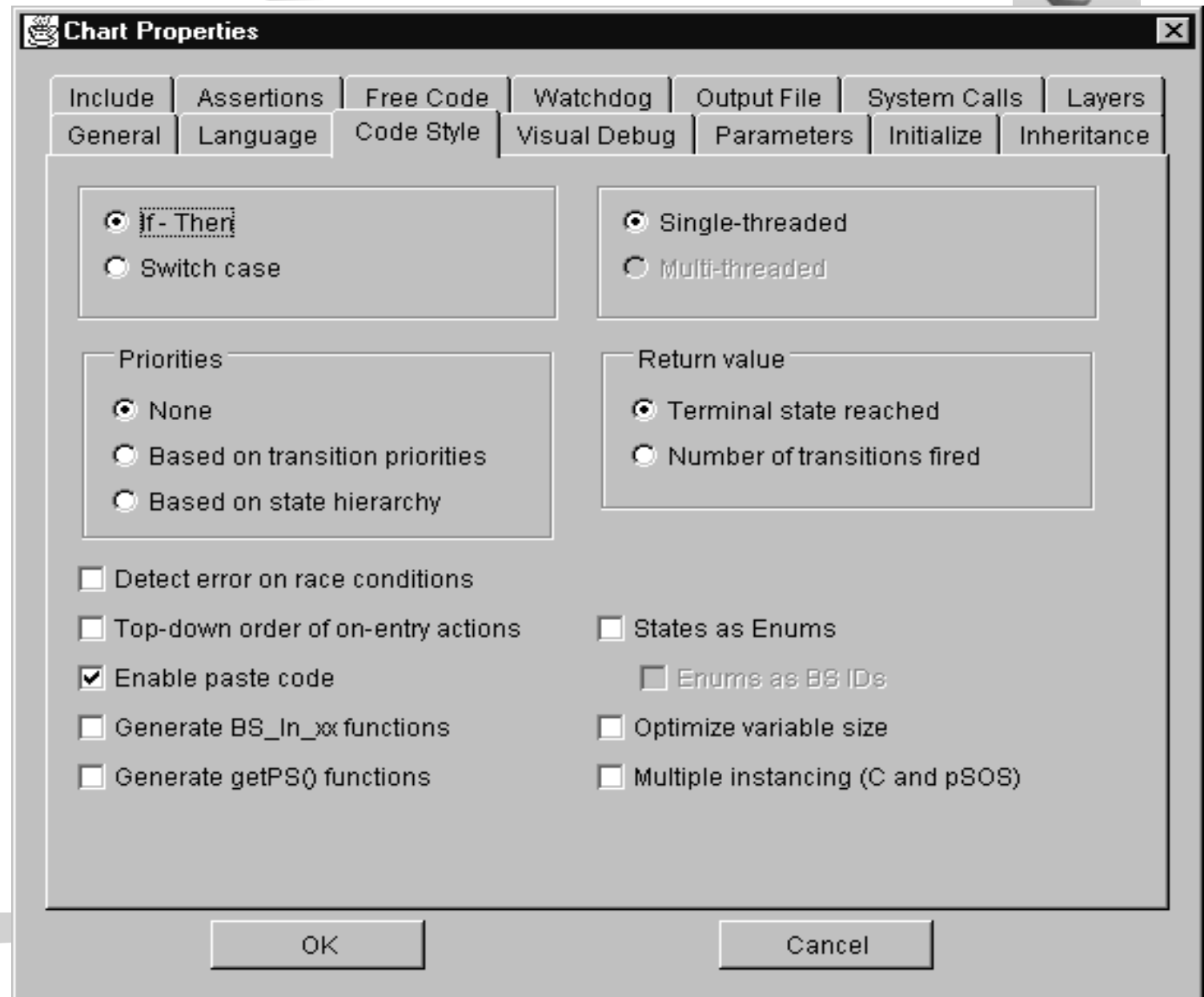
BetterState - Codeaspekte



Codegenerierung

- C
- C++
- pSOS

Konfigurierbar





- ❑ **Werkzeug zur Zustandsmodellierung, mit enger Schnittstelle zu MatrixX**
 - Intuitive Benutzerschnittstelle
- ❑ **Beschreibungsmittel**
 - Erw. Zustandsautomaten (Harel), Zustandsmodellierung entspricht UML
- ❑ **Weitere Schnittstellen**
 - Input Filter für Rational Rose Modelle
 - Simulationskopplung auch möglich z.B. an Visual C++
- ❑ **Benutzbarkeit**
 - Plattform
 - Windows; Solaris angekündigt
 - Dokumentationsgenerator
 - gut konfigurierbar