Übersicht

Motivation

Die Hauptschwierigkeit bei der Erstellung von Software ist deren enorme Komplexität. Bei der Softwareentwicklung werden Daten (Variablen) durch Operationen (Funktionen, Methoden) modifiziert bzw. verarbeitet. Auch bei kleinen Applikationen ist die Zahl der notwendigen Funktionen und Variablen sehr groß. Diesem Komplexitätsproblem zu begegnen, wird in allen Phasen der Softwareentwicklung (Analyse, Entwurf/Design, Umsetzung, Test) versucht, das Problem im handhabbare Module/Komponenten zu zerlegen und diese zu bearbeiten. Demzufolge haben sich unterschiedliche Vorgehensmodelle, Programmiersprachen, Architekturen und Testverfahren ergeben. Alle dienen einzig dazu, die Komplexität von Softwareentwicklungen in den Griff zu bekommen.

Steed.Net bildet eine Plattform und Programmierumgebung um mittels Zustandsmaschinen diese Komplexität in einfachen, ausführbaren Diagrammen zu modellieren.

steed.net und User Interfaces

Heutige User Interfaces sind sehr Komplex und bestehen aus einer Vielzahl von Fenstern, Masken, Dialogen. Alle diese Elemente ändern je nach Daten, Fortschritt der Bearbeitung und Userinteraktionen ihre Eigenschaften bzw. es werden ganz andere Masken oder Fenster angezeigt. Dies erfordert eine enormen Aufwand um in der Programmierung der Logik eines User Interfaces.
Zustandsmaschinen können uns auch bei der Implementierung von Masken und Dialogen unterstützen. Zum einen kann die Reihenfolge und Bedingungen zum anzeigen von bestimmten User Interface Elementen in form einer Zustandsmaschine realisiert werden. Zum anderen sind Änderungen von Eigenschaften von User Interface Elementen je nach Systemzustand wünschenswert. Eine Lösung ist die Erweiterung einer Form oder eines Controls um eine zugehörige Zustandsmaschine. Sowie eine Möglichkeit die UI Elemente je nach Zustand entsprechende Eigenschaften zuzuweisen.

steed.Net Plattform

Die Steed.Net Plattform besteht aus mehreren Modulen diese sind:

Zustandsmaschinen

- Die reine Zustandsmaschinen-Componente zur Modellierung von Abläufen ohne User-Interface z.B. für Server-Komponenten.
- Zustandsmaschinen-Control-Komponente – dies ist ein herkömmliches UserControl mit einer eingeschlossen Zustandsmaschine zur Modellierung von Abläufen im Userinterface und der Möglichkeit Control-Eigenschaften in Abhängigkeit von den Zustand der eigenen Zustandsmaschine sowohl auch der entfernten bzw. referenzierten Zustandsmaschine festzulegen. (Dependency View)
- Zustandsmaschinen Form Componenten dies ist eine herkömmliche System.Windows.Form mit eingeschlossener Zustandsmaschine -> siehe b.

Router
Der Router dient der Erstellung der Zustandsmaschinen und der Vermittlung von Ereignissen/Triggern zwischen Zustandsmaschinen.

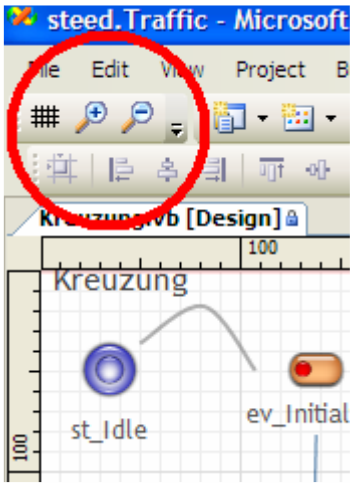
Kommunikationsprovider
Dient der Übertragung von Ereignissen zwischen Zustandsmaschinen über Prozess- und Rechnergrenzen. Zur Implementierung eigener Übertragungsmechanismen ist ein öffentliches Interface IComProvider zu implementieren.

Logprovider
Ermöglicht das schreiben, speichern und auswerten (bzw. Abfragen) von zur Laufzeit generierten Information über Fehler oder Debug-Information.
Eigene Logprovider können über das Interface ILogProvider implementiert werden.

steed.Net AddIn im VisualStudio

Das steed.Net AddIn ist im Studio mit folgenden Elementen Vertreten:

ToolBar/Commandbar:



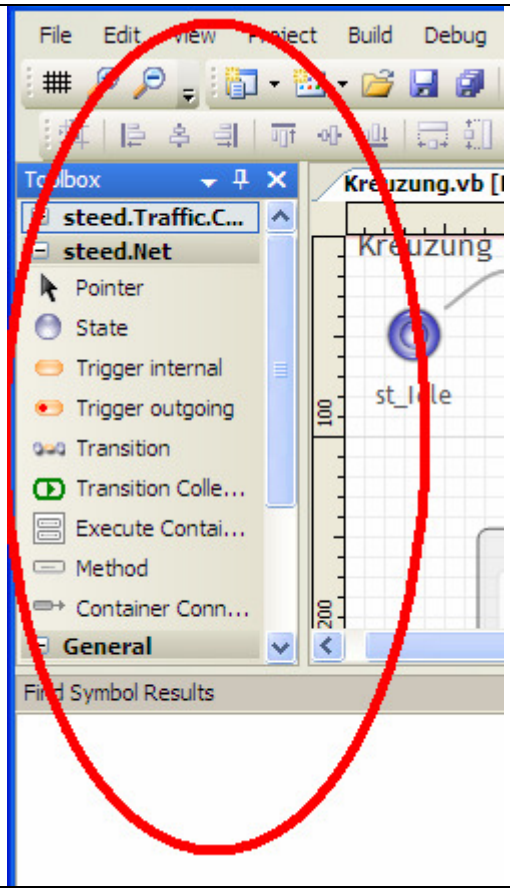
ist nur sichtbar wenn das [state] method AddIn geladen wurde.
Die Buttons auf der Commandbar sind nur enabled wenn Sie den Statemaschinen-Designer als aktives Fenster offnen haben.

 schaltet das Grid im Designer ab / an.

 vergrößert die gesamt Ansicht des Designers

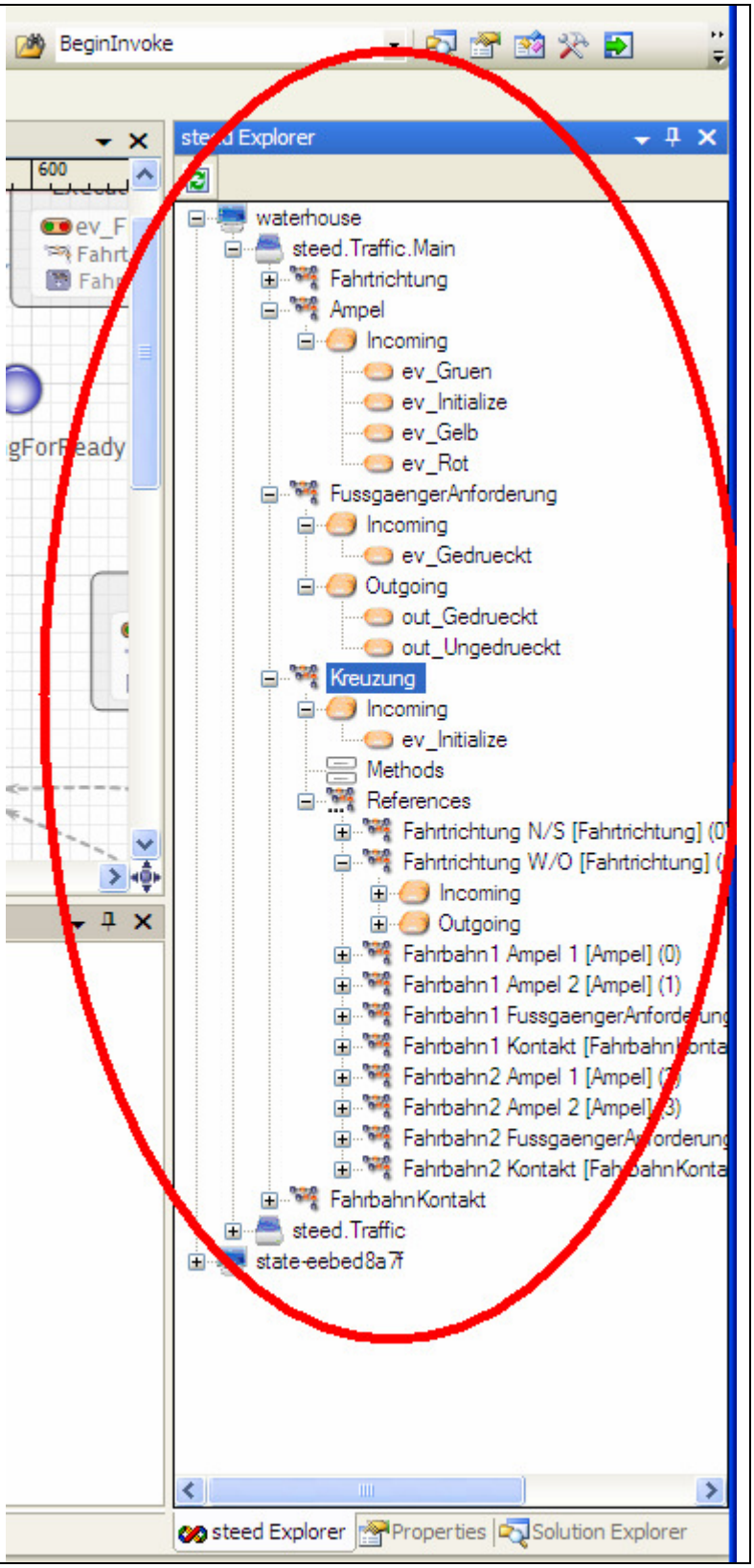
 verkleinert die Ansicht.

ToolBox

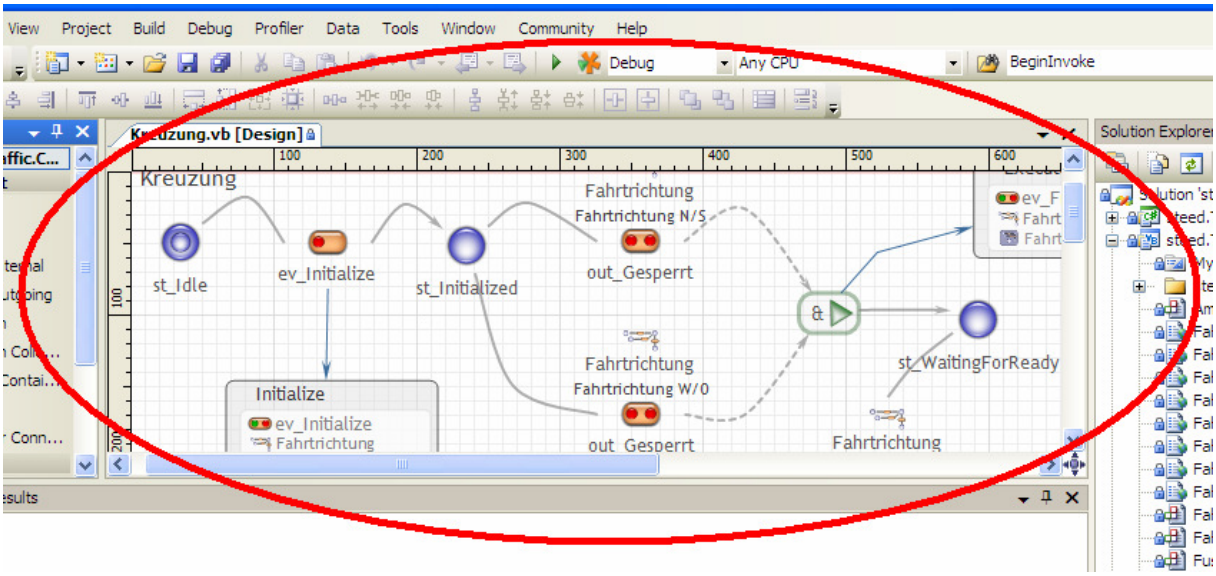
Zeigt Ihnen die Beschreibungs-Elemente von steed.Net Anhand dieser Elemente lässt sich nun eine gesamter Applikations-Prozess beschreiben bzw. designen.	

steed Explorer

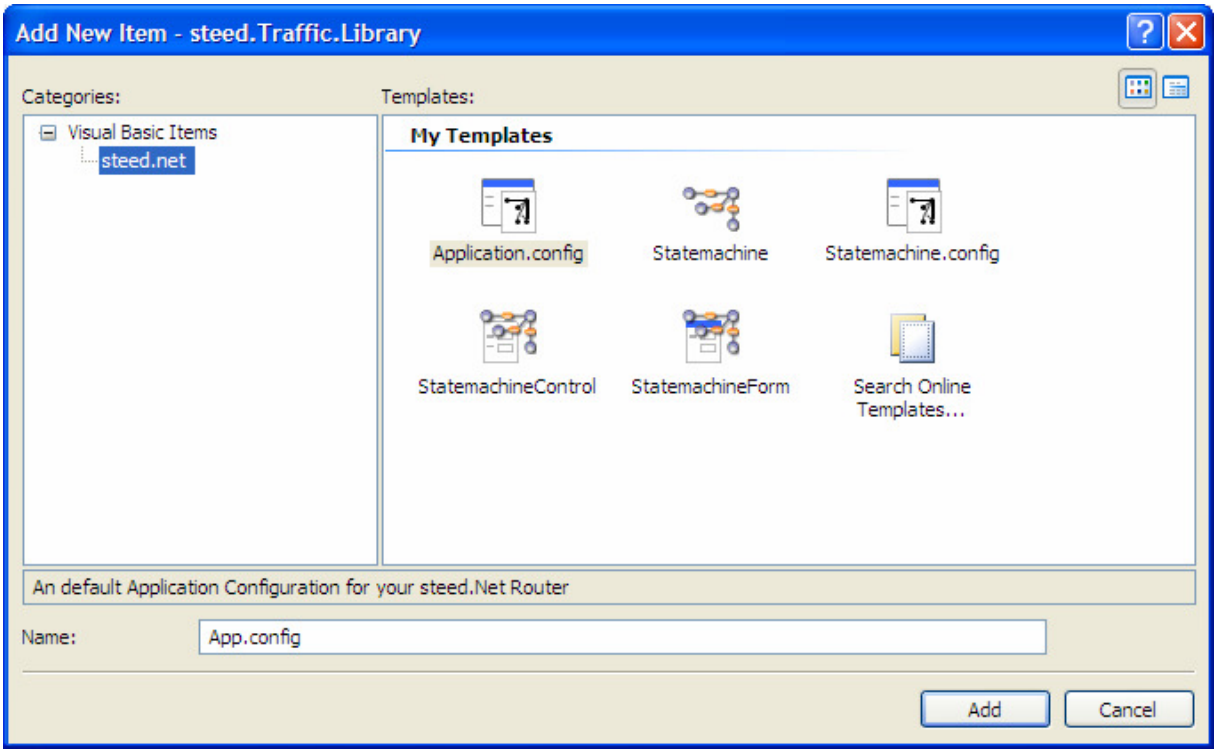
Dieses Fenster zeigt Ihnen, unter anderem alle in Ihrer Solution befindlichen Zustandsmaschinen an.
Die gerade aktuelle Zustandsmaschine, also die die Sie im Designer geöffnet haben wir in dem TreeView markiert.
Aus dem TreeView heraus sind sie in der Lage Referenzen auf andere Zustandsmaschinen auf Ihre gerade designte zu ziehen.
In der Enterprise-Edition von steed.Net werden Ihnen in diesem Fenster auch alle im Netzwerk und der lokalen Maschine gefundenen Instanzen von Router und deren Zustandsmaschinen angezeigt.



steed Designer



Projekt Items



Zustandsmaschine
Zustandsmaschinen / Statemachines Endlicher Automat (EA, engl. finite state machine - FSM)

Zustandsmaschinen sind endliche Automaten und bestehen aus 1-n Zuständen. Die Funktionalität wird durch die Definition von Ereignissen und deren Behandlung realisiert. Die Behandlung eines Ereignisses mit Daten wird als Transition bezeichnet und beinhaltet 0 bis n Funktionsmethoden. Bei der Entwicklung einer Zustandsmaschine werden im ersten Schritt die möglichen Zustände eines Systems oder Moduls identifiziert. Zu jedem Zustand werden alle regulären/ valide Ereignisse definiert, die in diesem Zustand eintreten können.

Alle Ereignisse (Trigger) die ausgelöst werden gelangen zum zentralen Router der in jeder Appdomain existiert. Dieser sammelt alle nachrichten und Verteilt diese zu dem jeweiligen Empfänger. Diese Verteilung erfolgt lokal sowie über Prozess und Rechner grenzen hinweg.
Aus diesem Grund kommt dem Router eine Zentrale Rolle in der steed.net Plattform zu.

steed Router

Der Router ist eine Komponente, die im Speziellen Zustandsmaschinen (IHostableObjects) lokal oder im Netz finden, hosten und Nachrichten an diese vermitteln kann.
Finden und Hosten… Ein Router sucht einen anderen Router im Netz und erhält Informationen über die vom Router gehosteten Objekte. Jedes IHostableObject (IHOB) hat einen eindeutigen Namen, der beim Host (Router) in einer Hosting-Liste eingetragen ist. Die Funktionen einer Zustandsmaschine sind bereits vordefiniert. Die Router können mit Hilfe von Communikationsprovider (IComProvider) miteinander kommunizieren. Wir stellen zwei Comunikationsprovider Microsoft Message Queue und .net Remoting zur Verfügung.
Deswegen sind die verschiedenen Zustandsmaschinen im Netz zu erreichen.
Der Client fragt den lokalen Router nach speziellen Hostable Objects, um eine Verbindung mit dem Router herzustellen, der die gewünschten Hostable Objects hat. Der Router sucht zunächst die Hostable Objects lokal. Falls sie nicht gefunden werden, wird die Anfrage aufs Netz ausgedehnt. Bei einem positiven Suchergebnis wird eine Verbindung hergestellt mit der „Ref. ID“ als Rückmeldung. Über die „Ref. ID“ bekommt der Client alle benötigten Informationen über die gewünschten IHostableObjects und kann jeder Zeit mit ihnen arbeiten.
Hosten… In diesem Fall hat der Router nicht nur die Suchfunktion, sondern kann auch selbst IHostableObjects hosten.
Da bedeutet dass der Router bei einer Anfrage nach einer bestimmten Zustandsmaschine (je nach Konfiguration der Zustandsmaschine) eine neue Instanz erstellt und eine „Ref ID“ liefert, oder eine vorhandene Instanz bzw. deren „Ref ID“ liefert.
Vermitteln… Es existieren zwei verschiedene Klassen von Zustandsmaschinen (HOB): Single- und Multi-Funktion. Eine Single-Zustandsmaschine (HOB) wird nur einmal instanziert und jegliche Anfragen auf neue Instanzen werden auf diese eine Umgeleitet. Andere Clients haben die Möglichkeit, auf sie zuzugreifen und damit auch zu arbeiten. Z.B. Mandant einer Buchhaltung

Eine Multi-Zustandsmaschine (HOB) ist für mehrere Clients da. Sie wird für jede Anfrage einer neuen Instanz neu erstellt und somit arbeitet jeder Client mit seiner eigenen Instanz.
In jeder steed.net Applikation ist also ein Router vorhanden, dies ist durch die statische Klasse „Statemachine.Router.Router“ gegeben.
Dieser muss in jeder Applikation gestartet und am Ende auch gestoppt werden.
Start: Statemachine.Router.Router.Start(bool Syncron);
Stop: Statemachine.Router.Router.Stop(bool Syncron);
Die Übergabe eines booleschen Wertes macht es Möglich den Start oder das Beenden des Router synchron (True) oder asynchron (False) auszuführen.
Applikationen mit einem UI sollten Router.Start mit dem Parameter „True“ aufrufen, da dadurch sichergestellt ist, das eventuell auf der MainForm befindliche Zustandsmaschinen einen „fertig“ gestarteten Router vorfinden. Allerdings kann sich das Startverhalten ändern indem Sie den Parameter auf „False“ setzten.
Applikationen die als Service gestaltet werden können Router.Start mit dem Parameter „False“ starten.

```
static class Program
  {
     /// <summary>
     /// Der Haupteinstiegspunkt für die Anwendung.
     /// </summary>
     [STAThread]
     static void Main()
     {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Statemethod.Router.Router.Start(true);
        Application.Run(new frmMain());
        Statemethod.Router.Router.Stop(true);
    }
  }
```

Des Weiteren ist eine App.Config Datei für eine detaillierte Konfiguration des Routers vorhanden.
Das Hinzufügen einer „eigenen“ Routerkonfiguration ist optional, da der Router mit einer default Konfiguration arbeitet.
→ siehe RouterConfig. Eine default Config Datei ist leicht durch Add->new Item-> Application.Confing in der steed.Net Section.