

Graphische Dialogmodellierung in Eclipse mit Hilfe des Graphical Editor Framework (GEF)

Dr. Jürgen Lind

iteratec GmbH

1. Juli 2004

Definition: Web-basierte Benutzerschnittstelle

- Ein WebClient läuft vollständig innerhalb der Browser-Umgebung ab.
- Im Gegensatz zum Native Client keine eigenständige Client-Applikation, die lokal auf dem Rechner des Anwenders zu installieren ist.
- Die eigentliche Ablauflogik des Client wird in einem Web-Container – beispielsweise einer Servlet-Engine – ausgeführt.
- Die Kommunikation zwischen Browser und Web-Container findet auf Basis des http(s)-Protokolls statt.

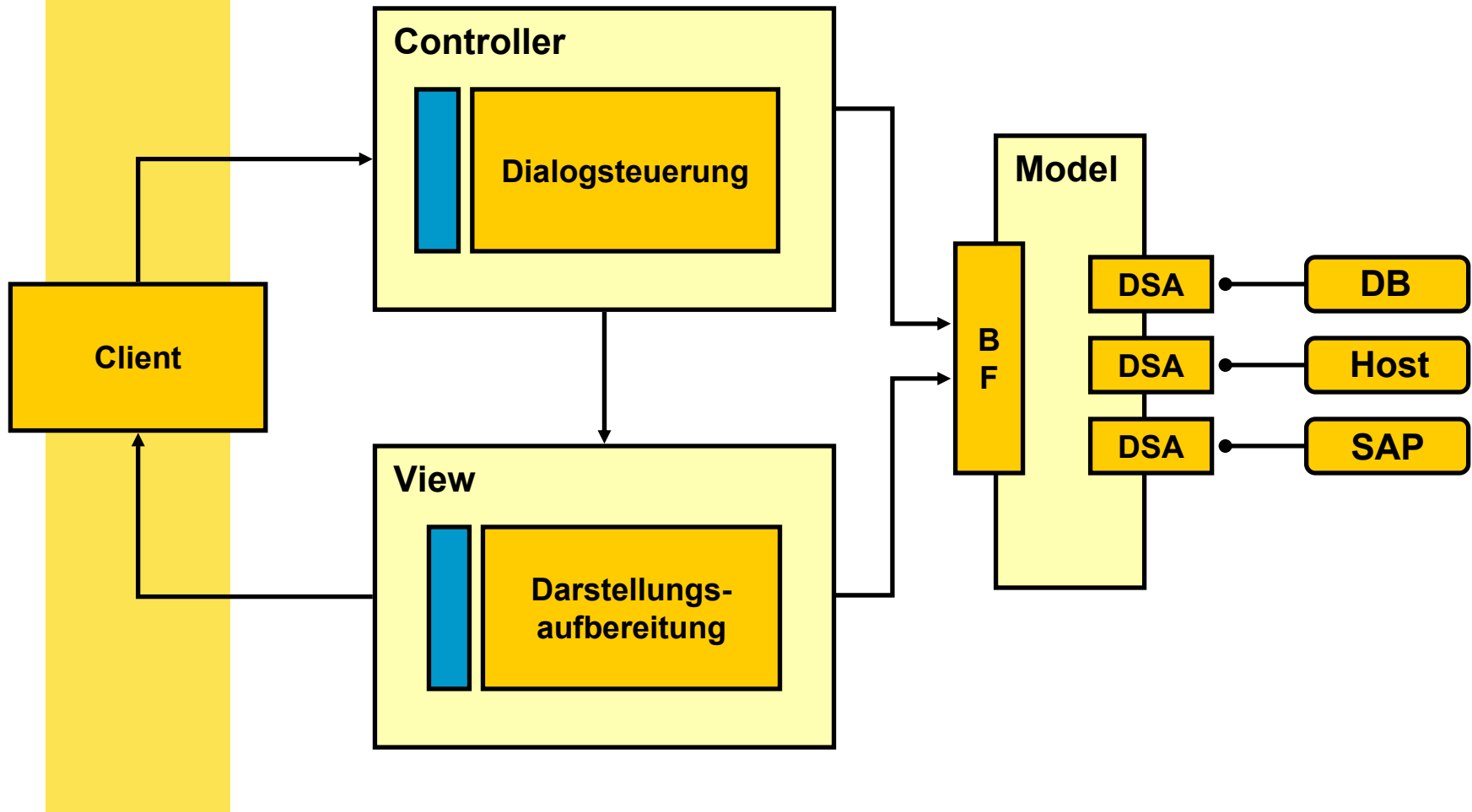
WebClient: Überblick

Iteratec WebClient:

Architektur und Framework für web-basierte Anwendungen



WebClient Architektur: Übersicht



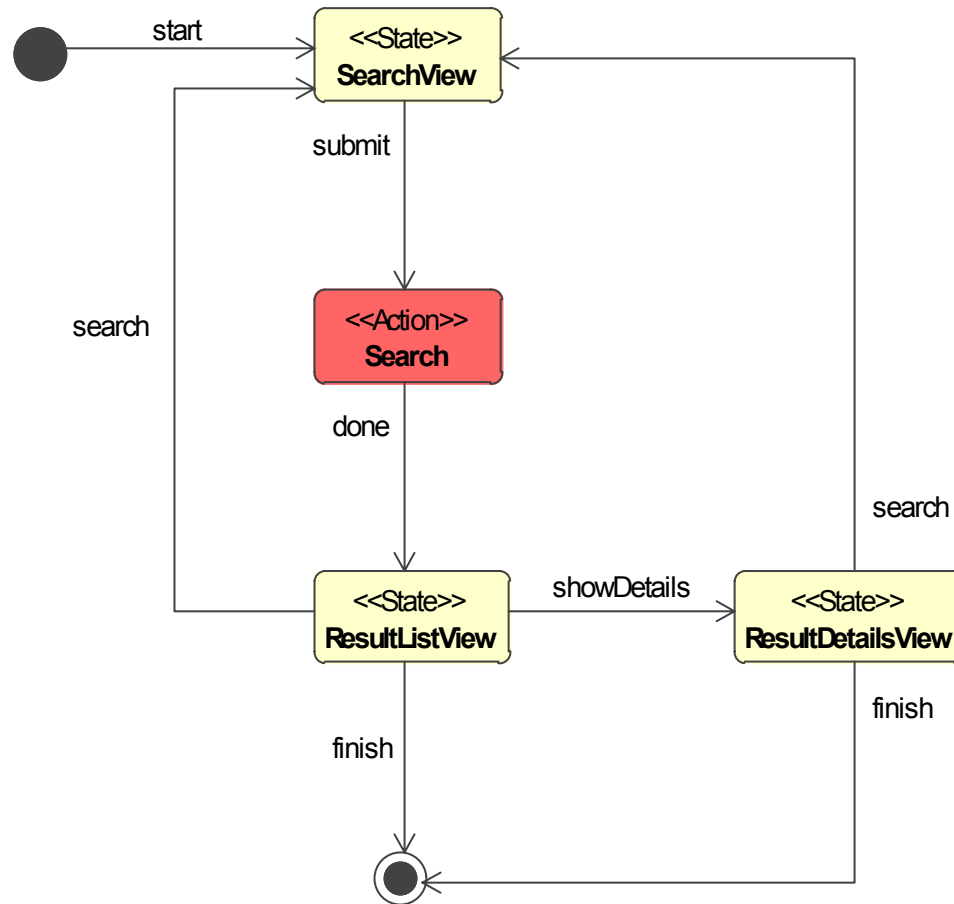
Dialogsteuerung

- Ein Dialog besteht aus einer oder mehreren Masken (screens) und repräsentiert einen fachlich abgeschlossenen Arbeitsvorgang, Beispiel: Suchen
- Ein Anwendung besteht in der Regel aus mehreren Dialogen, die über ein Navigationsfenster angesprochen werden
- Zu jeder Zeit ist genau einer dieser Dialog aktiv
- Jeder Dialog verfügt über ein Dialoggedächtnis, in dem die dialogrelevanten Daten zustandsübergreifend abgelegt werden können.
- Zusätzlich gibt es noch ein dialogübergreifendes Gedächtnis, über das verschiedene Dialog miteinander kommunizieren können.

Dialogmodellierung

- Basiert auf einem formalen Modell mit
 - Zuständen (States)
 - Aktionen (Actions)
 - Ereignissen (Events)
- Übergangsfunktion definiert die erlaubten Zustandsänderungen
- Aktionen werden durch Ereignisse ausgelöst und liefern die Resultate nach außen
- Strukturierung von größeren Maschinen mit Hilfe eines “include”-Mechanismus
- Dialogmodellierung unabhängig von der zugrunde liegende Technologie

Dialogmaschinen: Notation I



Dialogmaschinen: Notation II

```
<dialog-machine name="Beispiel">
  <state name="initial" isInitial="true">
    <transition event="start" target="SearchView"/>
  </state>
  <state name="SearchView">
    <transition event="submit" target="Search"/>
  </state>
  <state name="ResultDetailsView">
    <transition event="search" target="SearchView"/>
    <transition event="finish" target="final"/>
  </state>
  <state name="ResultListView">
    <transition event="finish" target="final"/>
    <transition event="showDetails" target="ResultDetailsView"/>
    <transition event="search" target="SearchView"/>
  </state>
  <state name="final" isTerminal="true">
  </state>
  <action name="Search">
    <transition event="done" target="ResultListView"/>
  </action>
</dialog-machine>
```

Anforderungen

- Grafische Darstellung der WebClient Dialogmaschinen
- Erstellung/Veränderung des Layouts
- Persistenz des Layouts
- Anbindung der Sourcen wie Java Code oder JSP Quelltexte

Ansatz I: XMI

- Erstellung und Layout der Dialogmaschinen als UML Finite-State-Machine Modelle
- Export der Modelle über XMI
- Konvertierung zu Dialogspezifikationen über
 - XSLT
 - Plain Java

Ansatz I: Vorteile

- Es existiert eine Vielzahl von UML Werkzeugen
- FSMs haben eine definierte Semantik
- Anpassbarkeit/Erweiterbarkeit durch Stereotypes und Tagged-Values

Ansatz I: Nachteile

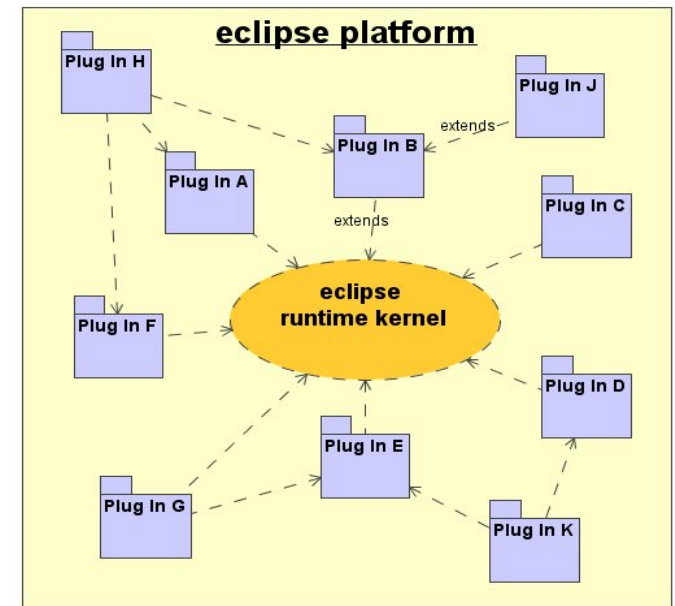
- Jedes Werkzeug hat einen eigenen XMI Dialekt, dies erfordert u.U. die Festlegung auf ein bestimmtes Werkzeug oder eine kleine Auswahl
- Kommerzielle Werkzeuge haben einen zu großen Leistungsumfang → zu hohe Lizenzkosten für wenig benötigte Funktionalität
- Ausprogrammierte Konvertierung erfordert umfangreiche Anpassungen bei Versionsänderungen des Werkzeugs
- Fehlende Integration in die Entwicklungsumgebung (Source-Code Navigation etc.)

Ansatz II: Eclipse und GEF

- Modellierung erfolgt in der Entwicklungsumgebung
- Datenstrukturen werden von Framework zur Verfügung gestellt

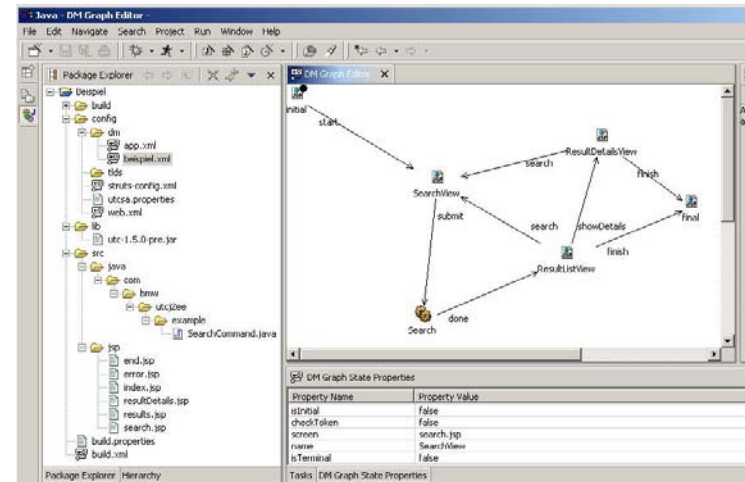
Technische Rahmenbedingung: eclipse

- Eclipse-Plattform =
Kleiner Laufzeitkern + Plug-ins
- Unterstützt die Entwicklung verschiedenster Werkzeuge je nach Anwendungsfall
- Offene Plattform
- Geeignet für beliebige Inhalte (Java, HTML, JSP, XML, etc.)
- Betriebssystem-übergreifend verfügbar (Windows, Linux, Mac OS)



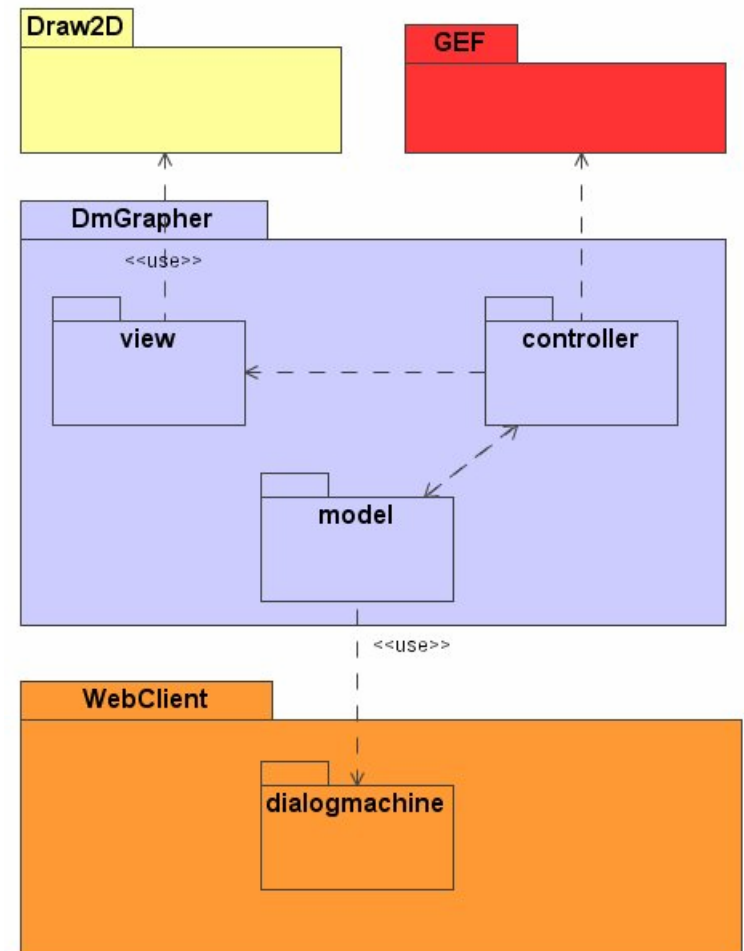
Technische Rahmenbedingungen: GEF

- Unterstützung zur Visualisierung und Bearbeitung von beliebigen Modellen innerhalb der eclipse
- Unabhängig vom Anwendungsfall
- Bereitstellen von Standard-Funktionen wie
 - Bewegen von Objekten und Objektgruppen
 - Zooming
 - Undo/Redo Basisfunktionen

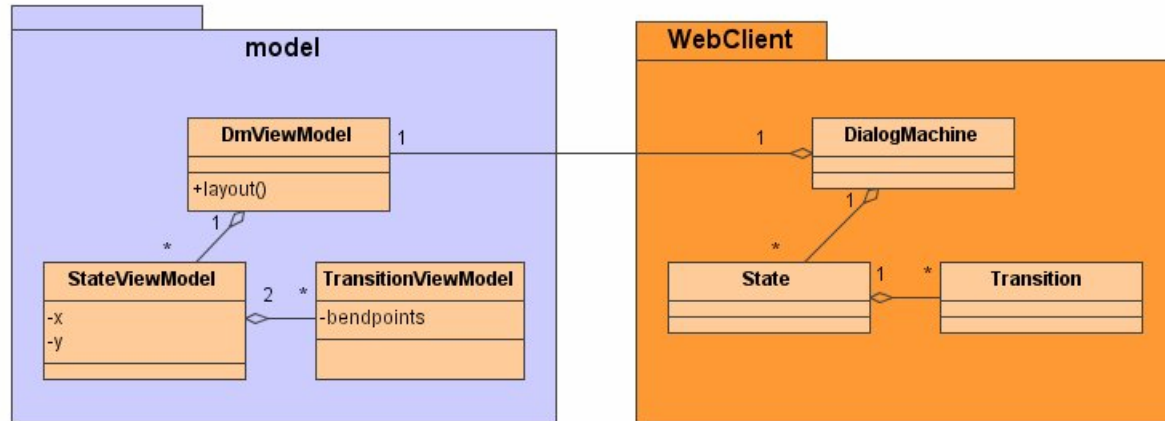


Grobarchitektur

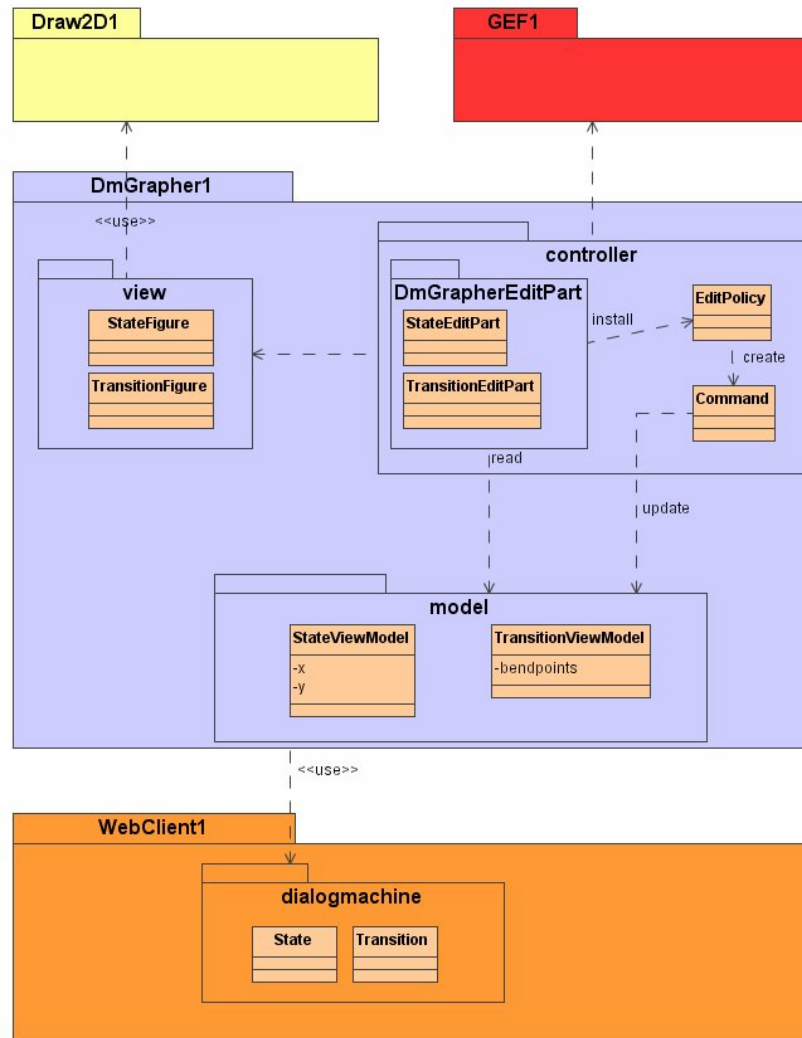
- Draw2D und GEF zur Visualisierung
- GEF zum Editieren
- Modell wird vom WebClient bereitgestellt



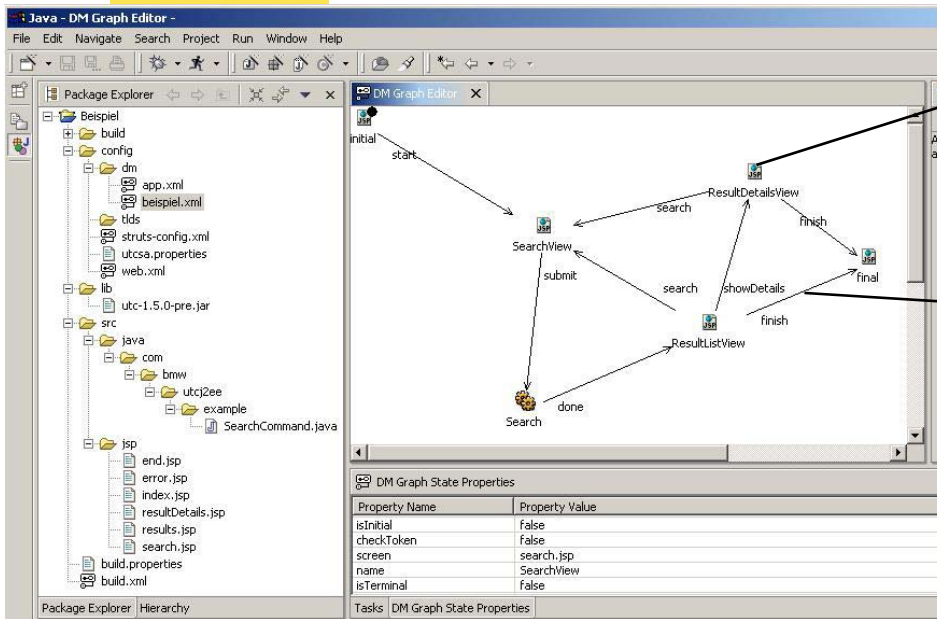
Model



Details



Persistenz von Layoutinformation



```
<dialog-machine-graph name="Beispiel">
  <node name="Search" x="-287" y="171" >
  </node>
  <node name="initial" x="-383" y="81" >
  </node>
  <node name="SearchView" x="-298" y="84" >
  </node>
  <node name="final" x="-278" y="342" >
  </node>
  <node name="ResultDetailsView" x="-125" y="200" >
  <connection name="finish" >
    <bendpoint x="-149" y="350" />
  </connection>
  </node>
  <node name="ResultListView" x="-304" y="252" >
    <connection name="search" >
      <bendpoint x="-348" y="236" />
      <bendpoint x="-345" y="125" />
    </connection>
  </node>
</dialog-machine-graph>
```

Ansatz II: Vorteile

- Kein eigenes CASE Tool erforderlich
- Round-Trip
- Nahtlose Integration in die Entwicklungsumgebung

Ansatz II: Nachteile

- Proprietäres Format
- Wenig Dokumentation zu GEF

Zusammenfassung und Ausblick

- DMGrapher Plug-In unterstützt die Entwicklung einer UTC/SA Anwendung im Eclipse
 - Grafische Darstellung des Dialogautomaten
 - Anbindung an die Sourcen
- Realisierung erfolgte mit GEF
 - Bereitstellung von wichtiger Basisfunktionalität
 - WhiteBox Framework → Verfügbarkeit des Quellcodes
- Weiterentwicklung des Plug-Ins bei iteratec GmbH

Weiterführende Verweise

- Eclipse
 - <http://www.eclipse.org>
- GEF
 - <http://www.eclipse.org/gef>
- UTC/SA Framework
 - J.Lind, M.Luber, *BMW-Standardarchitektur für web-basierte Anwendungen*, Javasppektrum 11/2002

Vielen Dank für Ihre Aufmerksamkeit
Fragen?