

Plus CD!

Stimmen zu EclipseCon, JAX und Eclipse Forum Europe >> 10

4.09

Deutschland € 9,80
Österreich € 10,80, Schweiz CHF 19,20



eclipse
MAGAZIN

eclipse

MAGAZIN

www.eclipse-magazin.de

CD-INHALT

Exklusiver Buchauszug:

„Eclipse Web Tools Plattform“
von Kai Brüssau, Kapitel 3:
Java Servlets **entwickler.press**

Chord Scale Generator

Eclipse-Tool für Saiteninstrumente

Eclipse-Tools

Java Workflow Tooling (JWT) 0.5.0,
Bonita 4.1, FraSCAti 0.5

eDine

Innovatives Restaurant-
Management

WTP

Dali JPA Tools, Ajax
Toolkit Framework

WEB

TOOLS

PLATTFORM

Modellierung zur Laufzeit >> 31

CDO Model Repository im Einsatz

Eclipse versus Maven >> 68

Kampf der Builder

Eclipse RCP für Musiker >> 54

Musiktool Chord Scale Generator

Java Workflow Tooling (JWT) >> 89

Vom abstrakten Geschäftsmodell zum lauffähigen Code

RCP-Tests mit SWTBot >> 75

Eclipse-Magazin-Tutorial



0 4

4 196603 209806

D 68864

Datenträger enthält
Info- und
Lehrprogramme
gemäß §14 JuSchG

Best of Eclipse DemoCamps: Grafische Editoren für große Metamodelle

GenGMF: Jetzt wird gezaubert ...



>> ENRICO SCHNEPEL

Bei der Erstellung von grafischen Editoren mit dem Graphical Modeling Framework (GMF) können die erforderlichen Modelle abhängig von der Struktur des Metamodells sehr groß werden. Dabei steigt die in den GMF-Modellen Graphical Def Model und Mapping Def Model enthaltene Komplexität im Vergleich zum Metamodell überproportional an und ist damit schwer zu handhaben. GenGMF ist eine Erweiterung für das GMF und erlaubt die Generierung dieser Modelle. Hierzu wird mit einer GMF-ähnlichen domänenspezifischen Sprache ein Modell erstellt, das im Vergleich zu den aus ihm generierten GMF-Modellen bis zu 80 Prozent weniger Modellelemente enthält. GenGMF war Gegenstand zweier Kurzpräsentationen bei den Eclipse DemoCamps in Berlin und Leipzig und soll nun auch den Lesern des Eclipse Magazins im Rahmen einer zweiteiligen Artikelserie vorgestellt werden.

Ausführung werden bei einigen Wizards weitere Informationen vom Entwickler abgefragt. Durch den iterativen-interaktiven Ansatz ist es jederzeit möglich, in den Prozess einzugreifen und das Modell direkt zu verändern, um anschließend weiter mit den Wizards zu arbeiten. Dies ist bei den Wizards aus dem GMF nur sehr bedingt möglich. Ein GenGMF-Modell enthält Templates, die Diagrammfiguren beschreiben, sowie Deskriptoren, mit denen die Templates, Metamodellelemente sowie Tooldefinitionen einander zugeordnet werden.

Templates

Mit den GenGMF Templates können die Figuren für den grafischen Editor beschrieben werden. Dies erfolgt analog zu einem GMFGraph-Modell, d. h. mit den gleichen Modellelementen. Bei GenGMF werden jedoch alle zu einer Figur gehörenden Modellelemente zu einem Template zusammengefasst und sind nicht wild über das gesamte Modell verstreut. Ein weiterer Vorteil bei der Verwendung von Templates ist die Vermeidung von Redundanz in den modellierten Informationen, denn bei GMF werden häufig ganze bereits modellierte Bäume dupliziert, nur um z. B. für einen weiteren Diagrammknoten eine andere Hintergrundfarbe zu definieren. Letztendlich kann mit einem Template eine Figur definiert werden, die für diverse Knoten oder Verbindungen das „gemeinsame Layout“ beschreibt. Später kann dieses Layout noch dynamisch für jeden einzelnen Knoten-/Verbindungstypen verändert werden. In GenGMF werden drei Arten von Templates verwendet: *Node*-, *Compartment*- sowie *EdgeTemplates*. Die *Node*- und *Compartment* Templates werden zum Beschreiben von Knotenlay-



Die Erstellung von grafischen Editoren mit dem Graphical Modeling Framework (GMF) ist oft eine sehr zeitintensive Aufgabe, da die den Editor beschreibenden GMF-Modelle bezüglich ihrer Größe ausufernd sein können. Nicht selten weisen sie die 20-fache Knotenanzahl des Metamodells auf. Da in den Modellen zusätzlich ungefähr ebenso viele Referenzen manuell gesetzt werden müssen, steigt neben der Komplexität auch die Fehlerwahrscheinlichkeit beim Editieren der Modelle überproportional an. Mit GenGMF wird die Komplexität auf eine andere Weise abgebildet und ermöglicht so das Erstellen eines kleineren Modells. Erreicht wird dies durch Anwendung des

„Don't repeat yourself“-Prinzips (DRY) und durch den Einsatz von Templates für sich wiederholende GMFGraph-Modellkonstrukte. Entwickelt wurde GenGMF im Rahmen einer Diplomarbeit, in der eine detaillierte Dokumentation enthalten ist.

Zauberer in Aktion

Zur Vorbereitung auf die Demonstrationen bei den DemoCamps (Kasten: „Eclipse DemoCamps“) wurden Wizards in den Editor integriert. Sie ermöglichen die Erstellung eines GenGMF-Modells zu einem vorgegebenen Metamodell innerhalb weniger Minuten. Dabei wird der Entwickler bei der iterativen Bearbeitung des Modells interaktiv unterstützt, indem zu einem bestimmten Modellelement verschiedene Wizards über das Kontextmenü aufgerufen werden können. Bei der

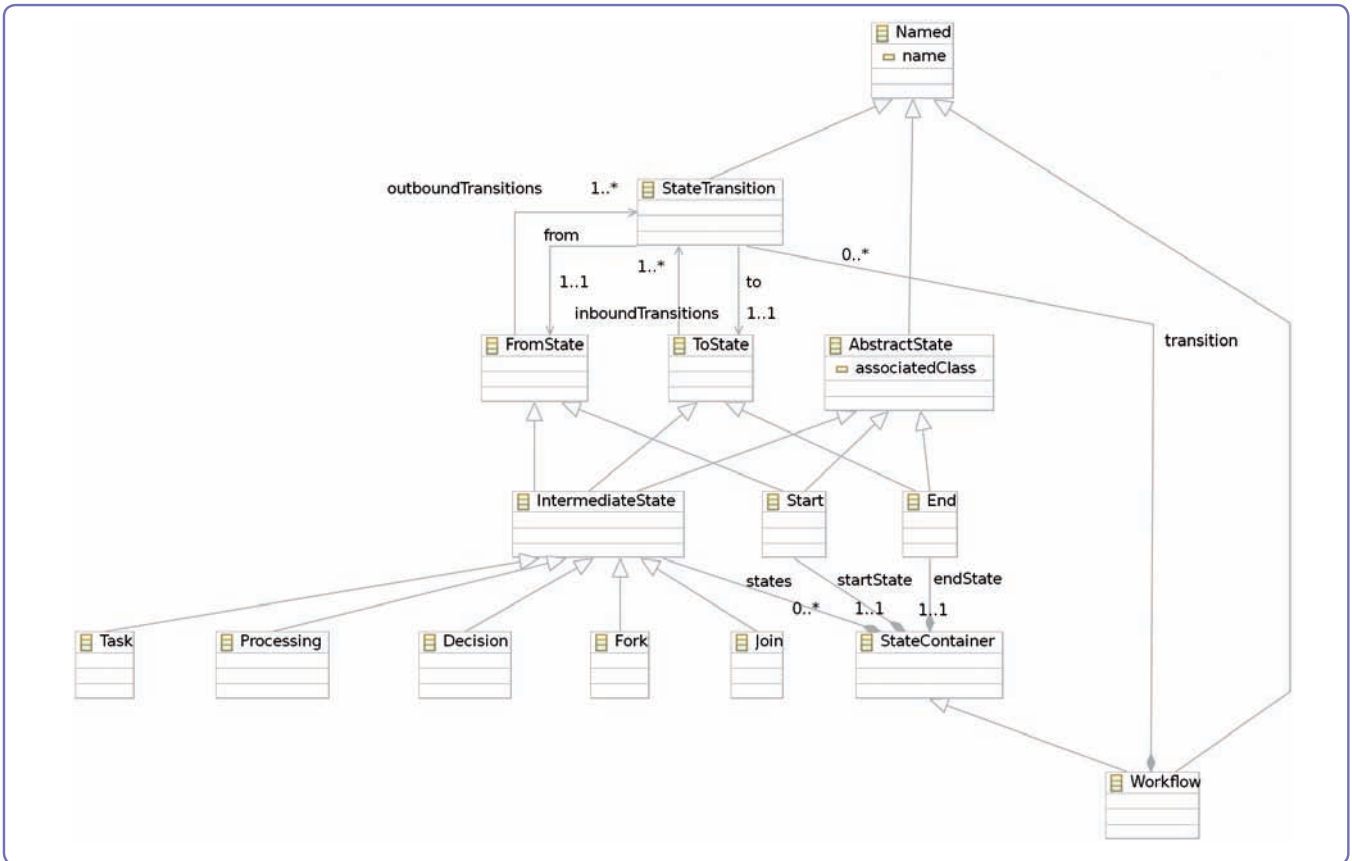


Abb. 1: Workflow-Metamodell

outs verwendet, wobei Compartments auch Kindelemente im Diagramm enthalten können. Das EdgeTemplate dient zum Modellieren von Verbindungen zwischen Knoten. Bei der Verwendung der Wizards werden die Templates iterativ aufgebaut, d. h. es wird zuerst ein rudimentäres Template zur Verfügung gestellt, das keine weiteren „Extras“ wie Labels usw. enthält. Dieses Template kann um Labels erweitert werden. Für diese werden dabei automatisch die *ChildAccess*- und *DiagramLabel*-Elemente initialisiert.

Deskriptoren

Deskriptoren sind mit den Reference/Mapping-Konstrukten aus dem GMF-Map-Modell vergleichbar und verknüpfen ein Template mit einem Metamodellelement und einem Tool aus einem GMFTool-Modell. Im Gegensatz zu den Templates bilden sie damit die Unterschiede zwischen den einzelnen Instanzen eines Templates, den Knoten- oder Verbindungsbeschreibungen, ab. Zu jedem Template-Typ gibt es auch bei den Deskriptoren korrespondierende Typen. Dies sind *Node*- und *CompartmentDesc* für Knoten sowie *Edge*- und *ReferenceEdgeDesc* für Verbindungen. Im Gegen-

satz zum GMFMap LinkMapping wird bei GenGMF jedoch explizit zwischen element- und den referenzbasierten Verbindungen unterschieden, was Fehler in der Modellierung des grafischen Editors vermeidet. Die elementbasierten Deskriptoren *Node*-, *Compartment*- und *EdgeDesc* können als Kinder *LabelDesc*-Elemente enthalten, um Attribute aus dem Metamodell in den Knoten oder neben einer Verbindung darzustellen. Diese Funktionalität ist vergleichbar mit den

GMFMap LabelMappings. Die oben angesprochenen Compartment-Kindknoten werden in *CompartmentChildDesc*-Elementen beschrieben. Hier werden die möglichen Kinder anhand der entsprechenden Deskriptoren referenziert, anstatt wie beim GMF das Mapping-nahezu identisch-noch einmal anzugeben.

Transformation

Aus einem GenGMF-Modell werden die beiden Modelle GMFGraph und GMF-

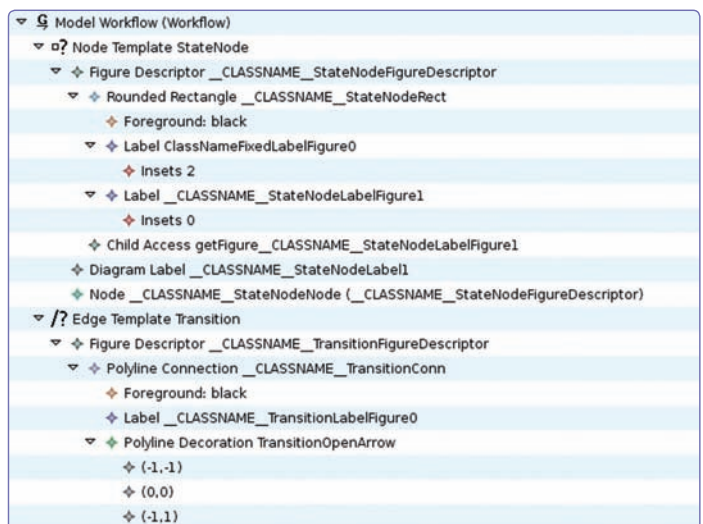


Abb. 2: GenGMF Templates



Best of Eclipse DemoCamps

Das zweite Berliner Eclipse DemoCamp

Bereits zum dritten Mal hat die Eclipse Foundation zum weltweiten Demo-Campen aufgerufen. In Berlin wurden die Zelte am 27. November 2008 zum zweiten Mal im newthinking store [1] in Berlin Mitte aufgeschlagen. Das Spektrum der vielen spannenden Vorträge reichte von kleinen Projekten und Diplomarbeiten bis hin zu ausgereiften Produkten.

Das zweite Berliner Eclipse DemoCamp [2] wurde von der b+m Informatik AG [3] (Enrico Schnepel) in Kooperation mit dem Fraunhofer Institut FOKUS [4] (Tom Ritter) organisiert. Das Camp startete um 18:00 Uhr, sodass die Teilnehmer genug Zeit hatten, nach der Arbeit ihre Zelte in Berlin-Mitte aufzubauen. Das Interesse am Berliner DemoCamp lag über unseren Erwartungen und so befürchteten wir, dass es für die Teilnehmer etwas eng werden könnte. Zum Glück fand sich schließlich doch für jeden ein Platz. Insgesamt gab es acht Vorträge, die in drei Blöcken präsentiert wurden. Zwischen den Blöcken waren 15-minütige Pausen eingeplant, die zum Diskutieren und Netzwerken genutzt wurden, und in denen das mit belegten Stullen gefüllte Buffet geplündert werden konnte. Die lockere Atmosphäre des DemoCamps machte es wirklich leicht, mit anderen ins Gespräch zu kommen. So ließen sich neue Kontakte mit Personen aus der Region knüpfen. Man traf aber auch alte

Bekannte, mit denen man über das Gehörte plaudern konnte.

Demonstrationen

Die drei Vorträge des ersten Blocks kreisten um das Thema „Austausch von Daten bzw. Modellen“. Den Anfang machte dabei Eike Stepper mit seinem Vortrag über das CDO-Model Repository [5] (siehe Artikel „Modeling goes Enterprise II“ von Eike Stepper in diesem Heft) und über die Frage, wie Objekte aus dem Eclipse Modeling Framework (EMF) über das Netzwerk synchronisiert werden können. Anschließend stellte Tom Ritter das Produkt ModelBus [6] des Fraunhofer Instituts FOKUS vor, das eine Infrastruktur zur Verarbeitung von Modellen über die Eclipse-Instanz hinaus zur Verfügung stellt. Im letzten Vortrag vor der Pause wurde uns von Volker Wegert ein SAP R/3 Connector für Eclipse-RCP-Anwendungen (RCER) [7] präsentiert.

Anschließend folgte ein Block zum Thema „grafische Editoren“. Den An-

fang machte Jens von Pilgrim mit dem Eclipse-Projekt GEF3D [8] zur Bearbeitung von Modellen im dreidimensionalen Raum. Der nächste Vortrag kam von Enrico Schnepel über das GenGMF-Projekt [9] (Artikel in diesem Heft: „Jetzt wird gezaubert“). Martin Flügge bot uns einen ungeplanten, jedoch an dieser Stelle thematisch passenden Vortrag über das Dawn-Projekt [10]. Mit Dawn synchronisiert Flügge auf dem GMF aufbauende Editoren verschiedener Eclipse-Instanzen über das Netzwerk und stellt diese zusätzlich auf einer Webseite dar.

Der letzte Block widmete sich ganzen Entwicklungsumgebungen. Mit der TTworkbench [11], vorgestellt von Dirk Borowski, können z. B. Protokolle systematisch getestet werden. Stephan Herrmann präsentierte abschließend ObjectTeams [12], eine Eclipse-Erweiterung, die Java um neue dynamische Sprachkonstrukte bereichert.

Zeitraffer

Das Ausrufen von Eclipse DemoCamps durch die Eclipse Foundation ist fast schon zur Tradition geworden. Berlin war bereits zweimal mit von der Partie. Beim ersten DemoCamp im November 2007, das von Ralph Bergmann von der Java User Group Berlin-Brandenburg organisiert wurde, gab es viele positive Rückmeldungen. Auch das zweite DemoCamp wurde durchweg positiv aufgenommen. Das dritte Berliner DemoCamp ist bereits in Planung und passend zum Galileo-Release für Juni 2009 angekündigt [13].

Enrico Schnepel und Tom Ritter



Abb.1: Zweites Eclipse DemoCamp in Berlin

Map automatisch generiert. Bei dieser Transformation werden die Templates für jeden referenzierenden Deskriptor instanziiert. Dabei wird jedes Vorkommen der Zeichenfolge `__CLASSNAME__` durch den Namen der im Deskriptor verknüpften Metamodellklasse ersetzt. So kann z.B. der Klassenname, ähnlich den UML-Stereotypen, in einem Diagramm angezeigt werden. Dies wird auch explizit durch entsprechende Wizards unterstützt.

Ein Beispiel

Als Beispiel soll an dieser Stelle – wie so oft – ein Workflow modelliert werden. Die dem Heft beiliegende CD enthält hierzu ein vorbereitetes Projekt *net.randomice.gengmf.demo.workflow*. Um die Stärken von GenGMF zu demonstrieren, werden verschiedene State-Typen definiert. Abbildung 1 zeigt das entsprechende Metamodell. Neben den klassischen Start- und Endknoten sollen diverse Knotentypen auf der Diagrammfläche platziert werden können. Dies sind *Task* (Benutzerinteraktion), *Process* (Datenverarbeitung), *Decision* (Entscheidung) sowie *Fork* (Beginn) und *Join* (Ende der Nebenläufigkeit). Verbunden werden die einzelnen Knoten mit einer Transition. Jedes Element hat einen Namen.

GenGMF-Modelle werden über den GenGMF Model Wizard unter `FILE | NEW | OTHER` mit dem Wurzelement *Model* erstellt. Nach dem Erstellen muss als Erstes das Metamodell über das Kontextmenü und dann *Load Resource* verknüpft werden. Damit die Editorunterstützung korrekt funktioniert, muss nur noch das Wurzelement des Metamodells für den Editor in der Eigenschaft *Root-Element* eingetragen werden. Des Weiteren sollte ein Name (z. B. *Workflow*) für das GenGMF-Modell vergeben werden. Diese Schritte sind im Demoprojekt bereits vollzogen worden.

Modellieren der Templates

Für den Editor werden zwei Templates benötigt – ein Node Template für die diversen Knotentypen und ein Edge Template für die Verbindung (Transition). Für alle benannten Elemente sollen die Namen angezeigt werden – entweder im Knoten oder neben der Verbindung. Die Knoten sollen zusätzlich einen textuellen Hinweis auf den Typ des Knotens, ähnlich wie die Stereotypen aus der UML, enthalten. Die Wizards zum Erstellen von Templates wer-

den direkt vom Modellelement aus über das Kontextmenü aufgerufen. Der abgefragte Name eines Templates sollte dabei die Gruppe von Elementen repräsentieren, für die das Template erstellt wird. Für den Workflow-Editor bieten sich die Namen *StateNod* und *Transition* an. Mithilfe der Wizards im Kontextmenü des jeweiligen Templates werden nun die Figuren noch weiter modifiziert. Bei den Knoten soll ein fester Typtext über einem variablen Namenstext stehen und muss daher zu Beginn der Figurbeschreibung vorkommen. Er lässt sich über den Wizard *Create new ClassName label* erzeugen. Anschließend kann das dynamische Label zum Anzeigen des Namens des Knotens bzw. der Verbindung mit dem Wizard *Create new feature label* erstellt werden. Die Verbindung soll mit einer die Richtung anzeigenden Pfeilspitze ausgestattet werden. Hierzu kann einer der drei Wizards *Add [open/closed/filled] target arrow* genutzt werden. Mithilfe der einzelnen Wizards wurde eine komplexe Baumstruktur in den Templates aufgebaut. Sie ist in Abbildung 2 dargestellt und enthält neben den jeweiligen Figurdefinitionen auch die Beschreibungen für die Notation des Modells für das Diagramm.

Modellieren der Deskriptoren

Nach dem Erstellen der Templates werden diese nun mit den einzelnen Elementen aus dem Metamodell verknüpft. Die passenden Wizards (*Create new [...] descriptor using template "[...]"*) sind vom entsprechenden Template aus aufzurufen

Anzeige

Eclipse DemoCamps

GenGMF wurde bei den Eclipse DemoCamps in Berlin und Leipzig vorgestellt. In der Vorbereitung hierzu wurde die Bearbeitung der Modelle mit der Einführung von Wizards erheblich vereinfacht und beschleunigt. Diese Entwicklung ist in die aktuelle Version GenGMF 2.0 eingeflossen.

GenGMF - Short Facts

Mit den GenGMF Wizards können grafische Editoren für ein bestehendes EMF-basiertes Metamodell innerhalb weniger Minuten erstellt werden. Das Layout der einzelnen Figuren wird durch Templates beschrieben, und Deskriptoren kombinieren diese mit den Metamodellelementen. Die GMF-Modelle können anschließend automatisch aus dem GenGMF-Modell generiert werden.

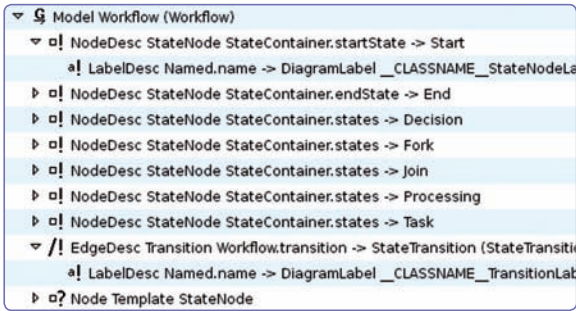


Abb. 3: GenGMF-Deskriptoren und das integrierte Toolmodell

und verlangen einige zusätzliche Angaben vom Entwickler. Dabei werden die Auswahlmöglichkeiten in den einzelnen Dialogen auf die zum bisher erstellten Modell passenden Möglichkeiten beschränkt. Die Definition der Knotentypen mit den Deskriptoren wird für den Workflow-Editor beispielhaft anhand des Startknotens beschrieben. Für alle elementbasierten Deskriptoren fragt der Wizard zuerst ab, welche Containment-Referenz des Modellwurzelements für den Deskriptor gelten soll. Für den Startknoten muss die Referenz *startState* ausgewählt und bestätigt werden. Da zu der gewählten Referenz nur ein Knotentyp passt, wird der Schritt zu dessen Auswahl übersprungen. Im letzten Schritt wird durch den Wizard abgefragt, ob ein entsprechendes Tool für die Palette erstellt werden soll, was ebenfalls bestätigt wird. Diese Schritte müssen nun für die anderen Knotentypen der Containment-Referenzen *endState* sowie *states* wiederholt werden. Da bei der *states*-Referenz mehrere Knotentypen (*extends IntermediateState*) passen, werden diese abgefragt und entsprechend für jedes Element ein Deskriptor sowie ein Toolelement angelegt.

Für die Verbindung wird ähnlich vorgefahren. Da im Workflow-Metamodell mit der Klasse *Transition* die Verbindungen elementbasiert abgebildet werden, muss auf dem *EdgeDesc*-Element auch der entsprechende Wizard aufgerufen werden. Nach der Auswahl der Referenz *Workflow.transition* können in den dar-

auffolgenden Dialogen die Standardeinstellungen bestätigt werden. Dabei wurden die Referenzen im Metamodell mit Namen wie *from* bzw. *to* automatisch als Quell- bzw. Zielreferenz erkannt und entsprechend im GenGMF-Modell eingesetzt. Des Weiteren haben die Wizards für jeden Deskriptor ein *LabelDesc*-Kindelement angelegt. Attribute wie *name* oder *title* aus der Metamodellklasse wurden dabei mit den in den Templates definierten Featurelabels verknüpft.

Für jeden Deskriptor wurde ein Toolelement angelegt. In GMF wird dies normalerweise in einer separaten *.gmftool*-Datei abgelegt. Für GenGMF wurde jedoch das ganze GMF Tooling DefModel als zweites Wurzelement in das GenGMF-Modell integriert. Ist dies nicht gewünscht, muss vor dem Erstellen der Deskriptoren ein separates Toolmodell angelegt und das Wurzelement im GenGMF-Modellelement unter *Tools* verknüpft werden. Das fertige GenGMF-Modell mit zwei exemplarisch aufgeklappten Deskriptoren und Toolmodell ist in Abbildung 3 zu sehen.

Generieren der GMF-Modelle

Jetzt können die GMF-Modelle generiert werden. Hierzu wird im Package Explorer im Menü der *.gengmf*-Datei der Punkt *Generate GMF-Graph and -Map Model* aufgerufen. Die GMF-Modelle wurden automatisch durch die GenGMF-Transformation erstellt und müssen nicht weiter bearbeitet

werden. Es wird mit dem normalen GMF-Entwicklungsprozess fortgefahren, d. h. man erstellt das GMF-Generatormodell aus dem GMFMap-Modell, um dann den grafischen Editor zu generieren. Diese Aktionen können auch über das GMF Dashboard ausgeführt werden, wenn die Modelle über *Select* entsprechend verknüpft werden. Dabei muss als Tooling Def Model das erstellte GenGMF-Modell angegeben werden, da die Wizards hier die Tooldefinitionen hinterlegt haben. Das generierte Diagramm-Plug-in benötigt zwei generierte Plug-ins aus dem EMF-Kontext. Diese werden über das Kontextmenü des Wurzelknotens der *.genmodel*-Datei erzeugt. Das Diagramm-Plug-in kann als Eclipse-Anwendung gestartet werden. In der Runtime-Instanz kann ein neues Workflow-Diagramm über FILE | NEW | OTHER angelegt werden. Im Editor können alle vorher modellierten Diagrammelemente erzeugt und mit einer Transition verbunden werden (Abb. 4). Voilà!

Fazit

GenGMF wurde benutzt, um einen grafischen Editor mit diversen Knotentypen zu erstellen. Dabei unterstützt GenGMF den Entwickler mit Wizards, die kontextsensitiv aufgerufen werden und durch die Generierung des Editormodells führen. Im nächsten Heft wird die Erstellung von Compartments zum Darstellen von inhaltlichen Beziehungen sowie das Scripting für GenGMF behandelt.



Enrico Schnepel arbeitet bei der b+m Informatik AG als Softwarearchitekt im Bereich der modellgetriebenen Softwareentwicklung und domänenspezifischen Sprachen.

GenGMF ist als Teil seiner Diplomarbeit entstanden und wird inzwischen in verschiedenen Projekten eingesetzt.

>> Links & Literatur

- [1] Schnepel, Enrico: GenGMF - a GMF model generator: <http://gengmf.randomice.net/>
- [2] Schnepel, Enrico: Entwicklung eines grafischen Editors zur Unterstützung eines modellgetriebenen Softwareentwicklungsprozesses, Diplomarbeit (2008), Fachhochschule für Technik und Wirtschaft, Berlin: http://www.randomice.net/files/Diplomarbeit_Schnepel.pdf
- [3] Stahl, Thomas, et. al.: Modellgetriebene Softwareentwicklung, 2. Edition (2007)
- [4] b+m Informatik AG: <http://engineering.bmiag.de/>

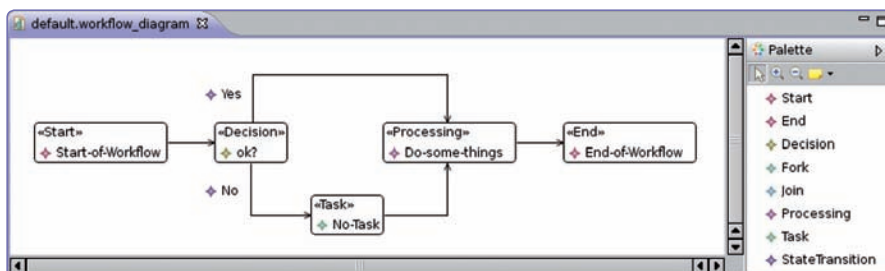


Abb. 4: Modell im fertigen Editor