Gruppe: HK-07-2
Projektleiter: Moritz

Autor: Moritz
29.6.2007

Entwurfsbeschreibung

(Final)

Inhaltsverzeichnis

1 Allgemeines....2
1.1 Zielbeschreibung....2
1.2 GMF als Entwicklungsgrundlage....2
1.3 Eclipse und GMF als Produktgrundlage....2
2 Produktübersicht....3
2.1 Produktfunktionen....3
2.2 Grafische Oberfläche....3
3 Statisches Modell – Aufbau....4
3.1 Modell des Editors auf Basis von GMF....4
3.2 Struktur und Funktionen der Editor Packages....6
4 Dynamisches Modell – Funktionalität....7
4.1 Übersicht....7
4.2 Aktivitätsdiagram....8
4.3 Benutzung des Editors....9
4.4 Funktionsweise der EditPolicies....10
4.5 Anpassung der EditPolicies....11

1 Allgemeines

1.1 Zielbeschreibung

Das Produkt ist ein Editor in Form eines Eclipse Plugins. Dieser bietet, zu einem gegebenem Domain Modell (*.ecore), eine intuitivere Erstellung/Editierung von GMF *.graph, *.tool und *.map an. Aus diesen Dateien erzeugt GMF einen grafischen Editor für das gegebene Domain Modell.

1.2 GMF als Entwicklungsgrundlage

Das Plugin selber wurde mit GMF erstellt und funktioniert als ein GMF Editor Plugin. Dafür sind also ein *.ecore Modell und die dazugehörigen *.graph, *.tool und *.map Modelle entwickelt worden.

1.3 Eclipse und GMF als Produktgrundlage

Das GMF Framework setzt auf Eclipse auf. Weiterhin stellt GMF die grundlegende Funktionalität des Editors (Produkt) bereit. Zusätzlich wurde der Editor so modifiziert das er z.T die Funktionen aus der Zielbeschreibung bereitstellt.

2 Produktübersicht

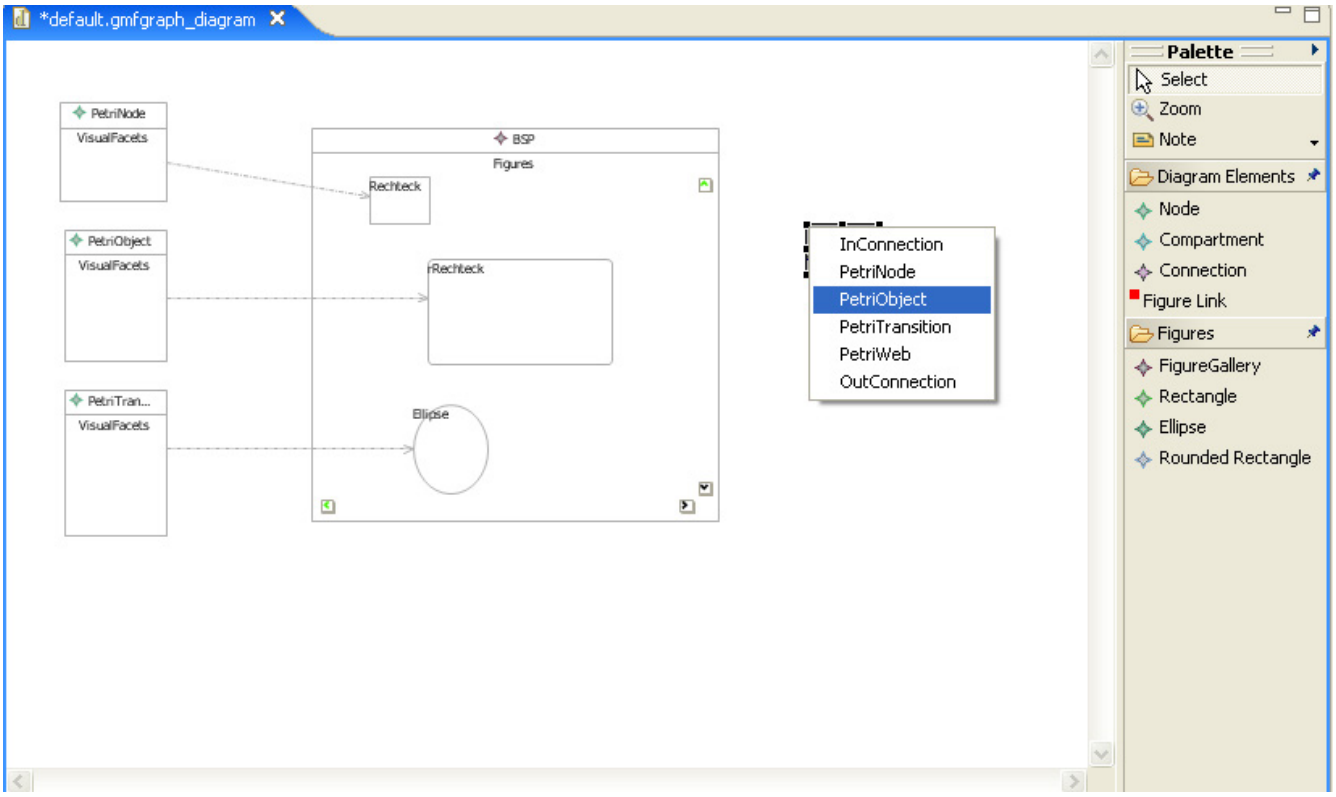
2.1 Produktfunktionen

Da der Editor ein mittels GMF erzeugtes Plugin ist, besteht er wie gewohnt, aus eine Zeichenfläche und der Toolbar aus der einfache Grafiken (Rechteck, Ellipse) ausgewählt werden können. Diese Grafiken können auf der Zeichenoberfläche gezeichnet werden und mit je einem Objekt aus dem vom Benutzer gegebenen *.ecore Modell assoziiert werden. Der Editor instanziiert, in der *.graph, die gezeichnete Grafik als Repräsentation für das gewünschte Objekt in der FigureGallery. Weiterhin wird für das Objekt auch ein CreationTool in der *.tool eingefügt, sodass man das Objekt im später erzeugte Editor, verwenden kann. Zusätzlich wird in der *.map das Mapping gesetzt. Der Name des assoziierten *.ecore Elements wird mittels eines Label auf der entsprechende Grafik kenntlich gemacht.

Desweiteren kann der Editor ein schon bearbeitetes Projekt weiter bearbeiten. Zum Schluß ist es möglich den Editor für das bearbeitete Projekt zu erstellen, was allerdings eine Funktionen ist, die GMF selber anbietet.

Weiterhin ist es möglich mehrere (unterschiedlich benannte) Modelle in einem Projekt zu bearbeiten.

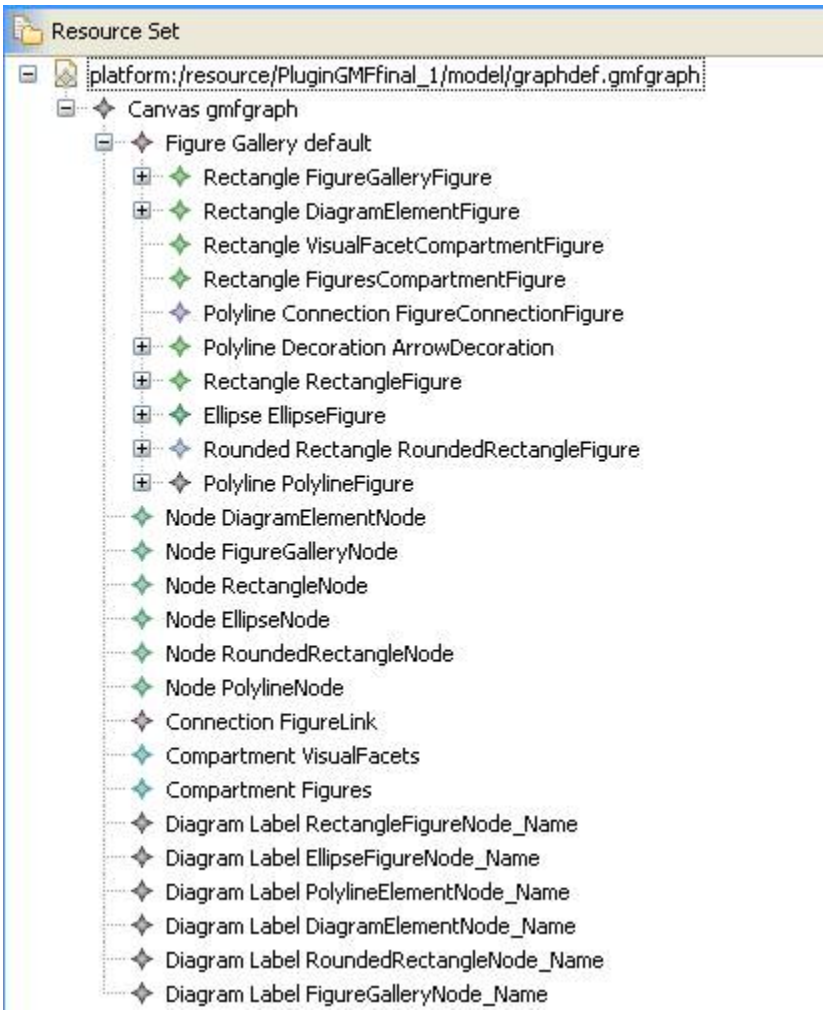
2.2 Grafische Oberfläche



3 Statisches Modell – Aufbau

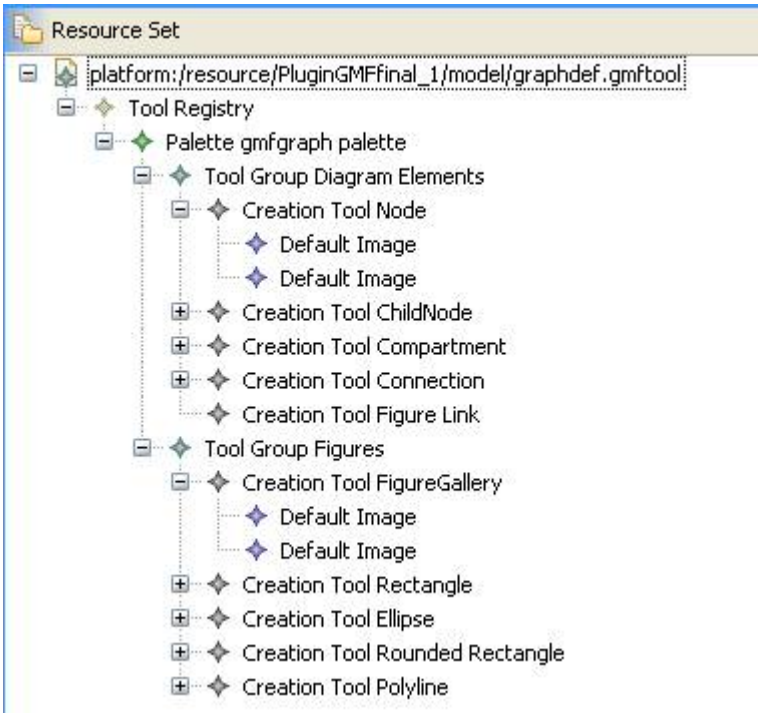
3.1 Modell des Editors auf Basis von GMF

Für die eCore-Datei wird eine grapfdef.ecore verwendet. Es wurde nur die Einschränkung der in der Canvas-EClass gemacht, das nur eine FigureGallery gezeichnet werden darf. Weiterhin haben wir uns bei dem Prototypen auf 2 Unterebenen innerhalb der Figuregallery beschränkt.

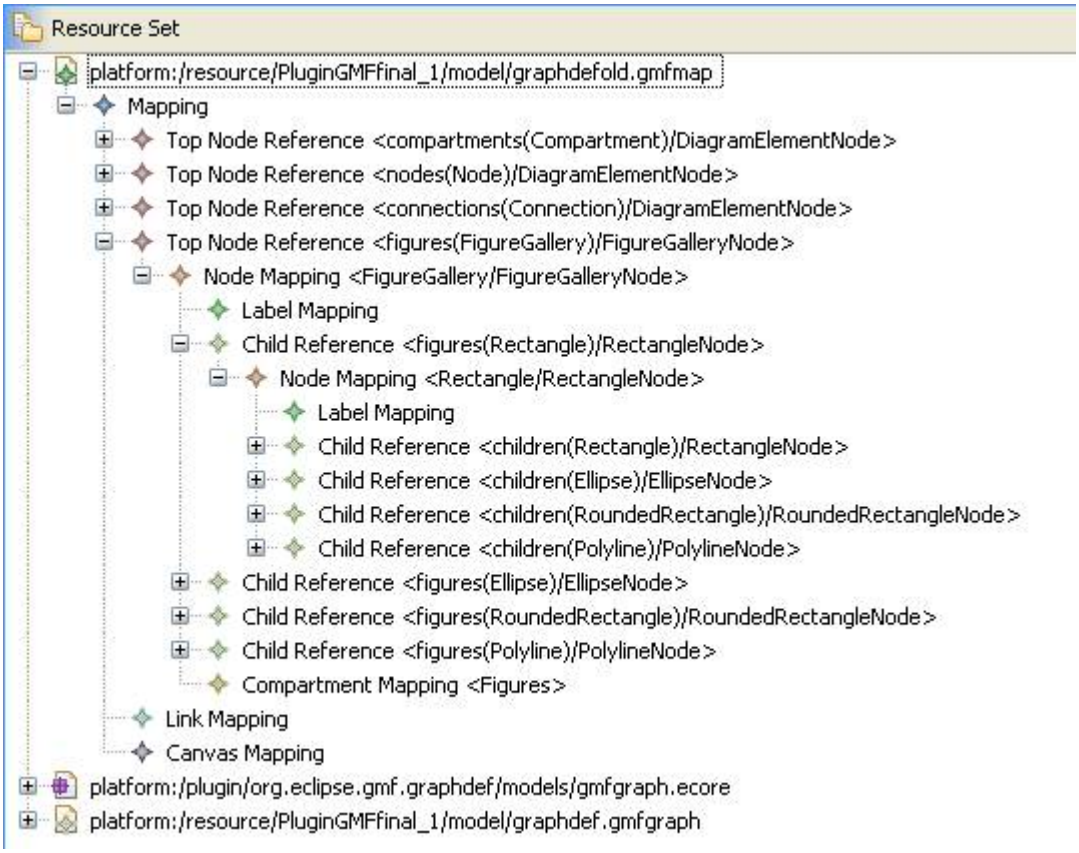


Das Modell enthält insbesondere Grafiken (Rechteck, Ellipse), FigureGallery, Nodes und FigureLinks. Die FigureGallery wird als Rechteck auf der Zeichenoberfläche dargestellt. Die als Rechtecke zeichenbare Nodes repräsentieren die Referenz Nodes für die grafischen Repräsentation in der *.gmfgraph. Das Mapping zwischen den Nodes und den grafischen Repräsentation wird durch den FigureLink grafisch auf der Zeichenoberfläche sowie im Modell realisiert.

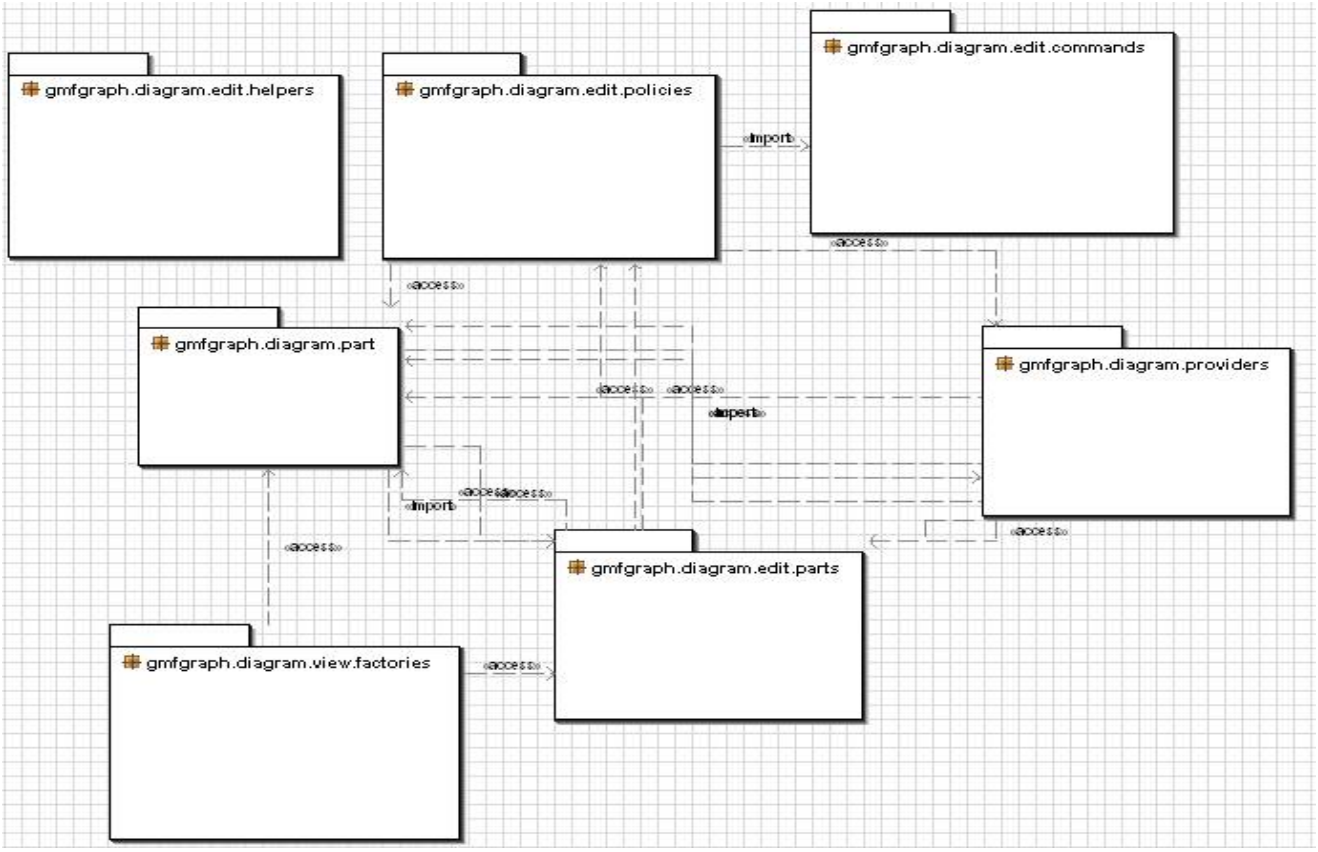
- *.gmftool



- *.gmfmap



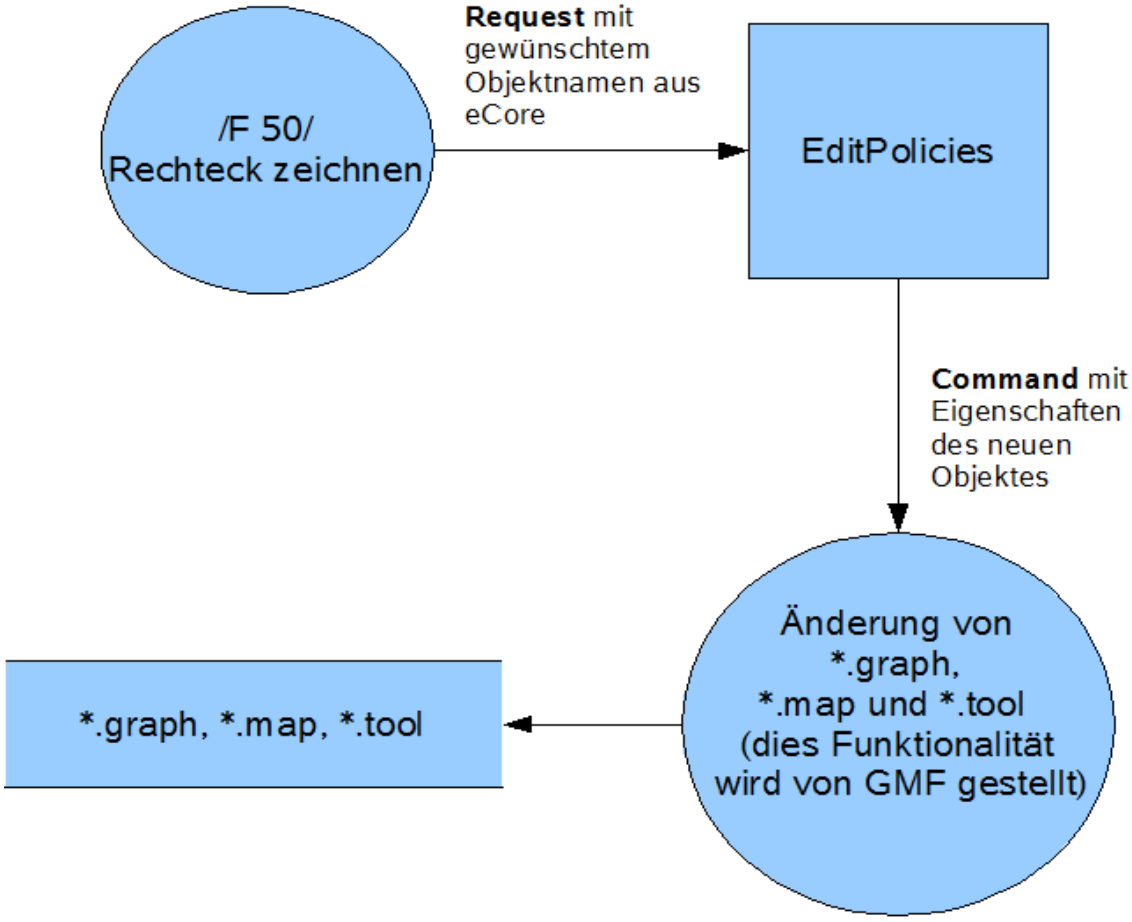
3.2 Struktur und Funktionen der Editor Packages



edit.commands: enthält eine Sammlung von Commands zur Editierung von Modell Elementen
edit.helpers: In diesem Paket sind die EditHelpers untergebracht, diese implementieren den DefaultEditCommand Algorithmus. Hier werden auch die Requests empfangen und die zugehörigen Commands zurückgegeben.
edit.parts: EditParts implementieren konkrete Funktionen des Elements, installieren/ initialisieren die DefaultEditPolicies, die Methoden zur Änderung des Labels, der Größe, sowie die Rückgabe der graphischen Darstellung.
edit.policies: Die Editpolicies ergänzen/überschreiben die bestehende Funktionalität der zugehörigen EditParts und werden auch von diesen aufgerufen.
part: In diesem Package werden der Aufbau der Wizards, die Zusammensetzung der Toolbars und ähnliches definiert und die zugehörigen EditParts zu diesen hinzugefügt.
providers: Providers sind verantwortlich für die Rückgabe der Klassen, welche eine konkrete Funktionalität implementieren.
view.factory: In diesem Paket werden die ViewFactorys definiert. In diesen wird das Aussehen und die Darstellung der einzelnen Objekte (EClasses, EPackages, ...) festgelegt.

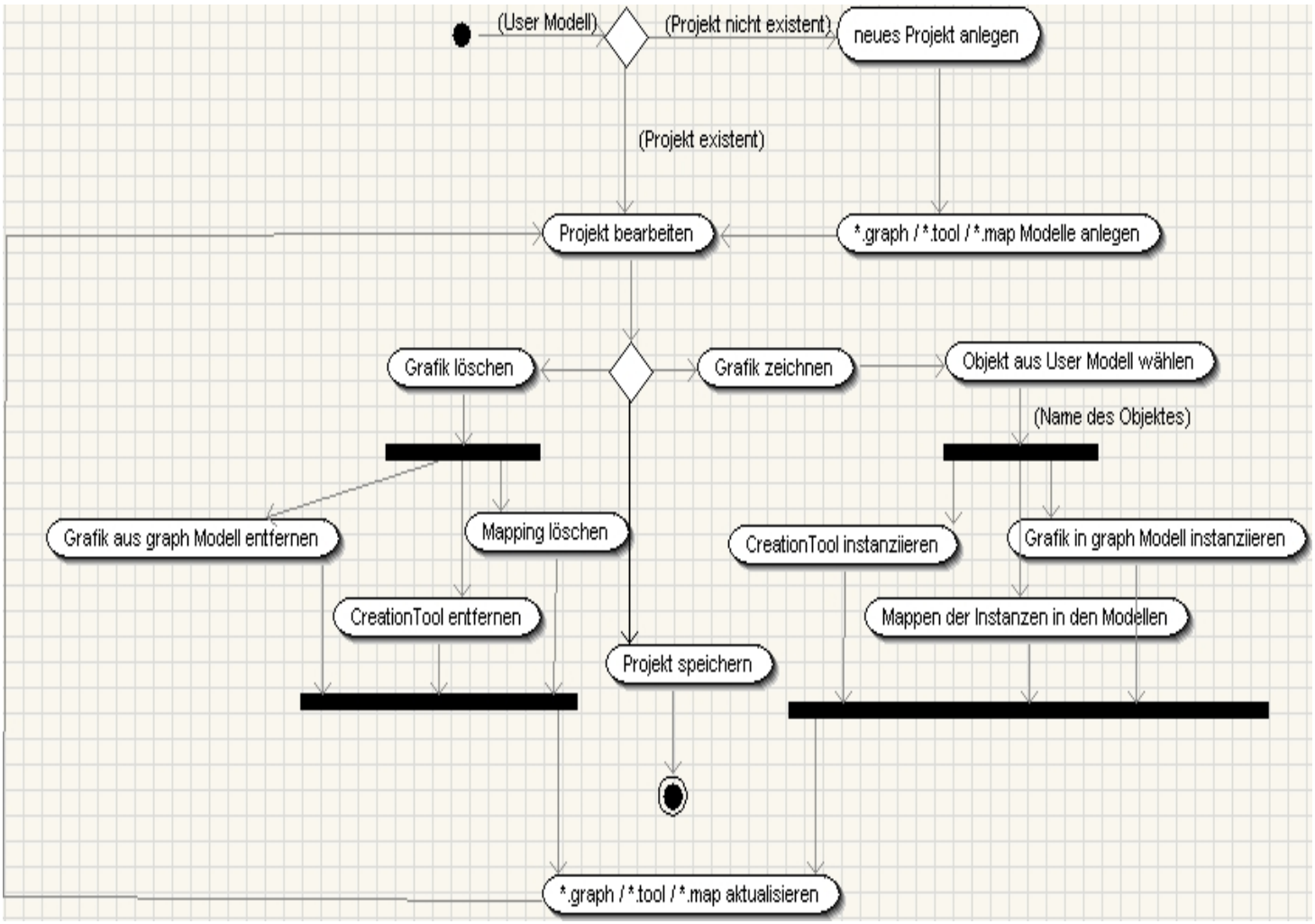
4 Dynamisches Modell – Funktionalität

4.1 Übersicht



Dieses Diagramm zeigt beispielhaft den groben Ablauf beim Zeichnen eines Rechteckes auf der Zeichenfläche. Dabei wird durch die EditPolicies das semantische Objekt erzeugt und geändert. Außerdem werden hier auch die Änderungen, via Command, in den *.tool bzw. *.map Modellen ausgelöst.

4.2 Aktivitätsdiagram



Dieses Diagramm zeigt mögliche Benutzeraktivitäten im Editor. Um das Diagramm nicht unnötig zu überladen, sind mit „Grafiken“ implizit auch deren referenzierte Nodes gemeint. Die Aktivitäten „CreationTool“ , „Grafik im graph Modell instanzieren“, „Mapping“ sind keine eigentlichen Benutzeraktivitäten, da sie automatisch ausgeführt werden. Darin liegt dann auch die eigentliche Funktionalität des Editors.

4.3 Benutzung des Editors

Um das Plugin zu starten, legt der Benutzer ein neues Projekt (New -> General -> Project) an. Darin legt er ein neues Example (New -> Examples .> „GMFGraph Diagramm“) an.

Die nun erzeugten Modelldateien erhalten den Namen der im Dialog angegeben wurde. Als nächstes öffnet sich ein Dialog mit dem der Benutzer die Datei die das ecore Modell enthält auswählt. Diese wird in den Projektordner kopiert und erhält den selben Dateinamen wie alle anderen Modelldateien.

Danach öffnet sich ein Dialog in dem der Benutzer das *DomainMetaElement* auswählt.

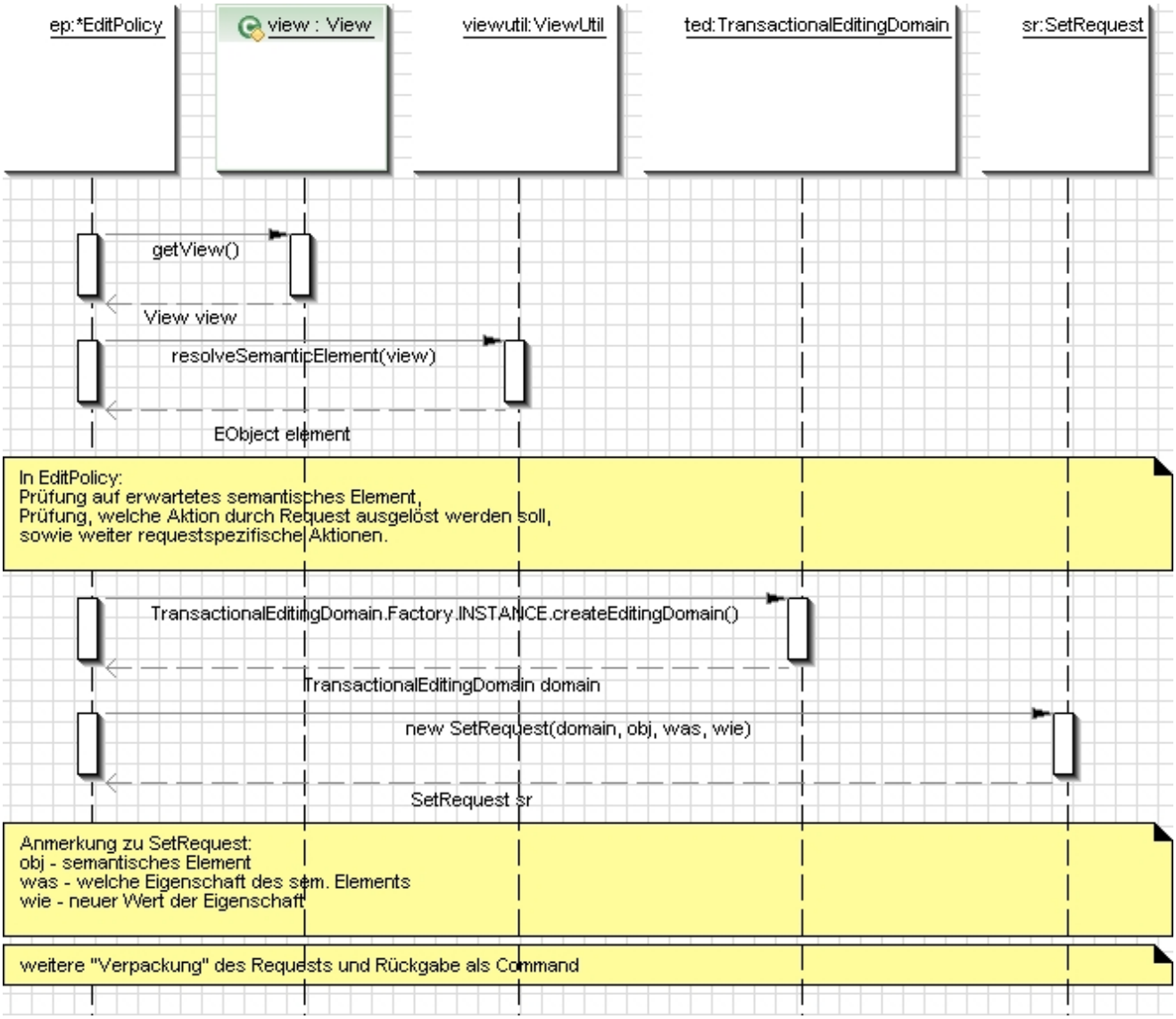
Jetzt kann der Benutzer das Plugin nutzen um Nodes und deren grafische Repräsentation auf der Zeichenoberfläche zu erstellen. Um die Grafischen Repräsentation zu zeichnen, muss der Benutzer zuerst die FigureGallery aus der Toolbar selektieren und dann die FigureGallery auf die Zeichenfläche zeichnen. Darin kann er jetzt verschiedene Grafiken (Rechtecke,Ellipsen, etc) aus der Toolbar wählen und in die FigureGallery zeichnen.

Im Bereich um die FigureGallery herum, kann der Benutzer Nodes zeichnen. Dazu muss der Benutzer den Eintrag Nodes in der Toolbar auswählen und kann dann Nodes zeichnen. Dabei wird für jedes neu erstellte Node ein Auswahlmenu angezeigt, aus dem der Benutzer die gewünschte Eclass (aus dem ecore Modell) auswählen muss, die durch das Node repräsentiert werden soll. Wurde eine eclass ausgewählt, muss jetzt das *Containment Feature* gesetzt werde. Auch hierzu wird automatisch ein Auswahlmenu angeboten. Um den Node mit einer grafischen Repräsentation zu verbinden, muss der Benutzer das NodeLink Tool auswählen und kann dann eine Verbindung zwischen Node und Grafischer Repräsentation ziehen.

Zuletzt kann der Benutzer die Änderungen speichern [Strg + S] womit diese auch in den Modelldateien persistent gespeichert werden.

Bei der Erstellung von Nodes auf der Zeichenfläche werden die davon ausgelösten Requests abgefangen und ein PopUp Menu geöffnet. In diesem Menu sind sämtliche Eclasses aus dem ecore Modell aufgelistet. Der Name der gewählten Eclass wird zurückgegeben und dem CreationTool sowie dem Node in der gmfgraph als Name übergeben. Der Command zur Erstellung des CreationTools wird mit dem zugehörigen Namen auf den Commandstack gelegt und ausgeführt.

4.4 Funktionsweise der EditPolicies



4.5 Anpassung der EditPolicies

Beim Start des Plugins mit New Project -> New Example (GMFGraph Diagramm) werden die gmfgraph, gmftool, gmfmap – Modelle bzw. Dateien erstellt bzw. (wenn existent) geladen. Dies ist in der *GMFGraphEditorUtil* implementiert. Die Modelle werden in entsprechende Ressourcen geladen. Im Tool Modell wird sofort eine Tool Registry und darunter eine Pallette angelegt. Auf der Zeichenoberfläche kann man mittels des FigureGallery Tools eine FigureGallery zeichnen, worauf diese im gmfgraph Modell erzeugt wird.

Neue Elemente (CreationTools, NodeReferences) lassen sich aus den eInstances der entsprechenden Factory Klassen erzeugen.
`GMFToolFactory.eINSTANCE.createCreationTool()`

Um die Modelle zu lesen und zu ändern, wird mittels `ressource.getContents()` auf deren Inhalt zugegriffen und per *add()* vorher erstellte Elemente (z.B. ToolRegistry) dem Modell hinzugefügt.

Die Befehle werden in einer `EditingDomain` erstellt und in ein `AbstractTransactionalCommand` gepackt und per `execute()` ausgeführt.

Die *NodeItemSemanticEditPolicy* wurde so angepasst bzw. erweitert das sie beim Erstellen einer neue Node auf der Zeichenfläche, die Änderungen in den Modellen bzw. Projektdateien vollzieht.

Für die grafischen Repräsentation wurden die **GraphicalNodeEditPolicy's* erweitert. In diesen Dateien werden die Request abgefangen, die bei Änderungen an den grafischen Repräsentation ausgelöst werden. D.h. Änderungen an der Größe bzw. Position auf der Zeichenfläche oder Farbänderungen per Kontextmenu. Die neuen Werte werden ausgelesen und in requests verpackt. Diese werden dann in commands verpackt und ausgeführt.