Entwurfsbeschreibung

Zum Prototypen 1:

Der Prototyp 1 ist ein mit Hilfe von GMF generierter Editor. Er dient der Modellierung von Zustandsübergangsdiagrammen bzw. Zustandsautomaten, wie sie aus der angewandten und theoretischen Informatik bekannt sind. Dabei ist es möglich, sowohl deterministische als auch nicht-deterministische Übergänge zu konstruieren. Damit wird dem Benutzer ein größtmögliches Maß an Flexibilität bei der Modellierung geboten.

Der Editor stellt eine *Toolbar*, in der Zustände sowie Zustandsübergänge ausgewählt werden können, bereit, sowie eine *Zeichenfläche*, wo diese Elemente dargestellt werden können. Zustände werden durch Kreise bzw. Ellipsen repräsentiert, wobei die Darstellung noch zwischen *StartZustand*, *InnererZustand* und *EndZustand* differenziert wird. Für den Startzustand wird eine breite *LINE_DOT* verwendet, für einen inneren Zustand eine dünne *LINE_SOLID* und für den Finalzustand eine breite *LINE_SOLID*, als Kreislinie. Diese Unterscheidung der grafischen Darstellung erlaubt ein leichtes Überblicken des modellierten Automaten bzw. Diagramms und ermöglicht es dem Benutzer auf das Verwenden des *Property-View* zur Feststellung der verschiedenen Zustandstypen zu verzichten. Für weitere Benutzerfreundlichkeit, sorgt das Design der *ZustandsÜbergänge*. Sie werden intuitiverweise durch eine Linie mit Pfeil dargestellt, die einfach aus der *Toolbar* ausgewählt und per Drag&Drop zwischen zwei Zustände gesetzt werden können. Dabei können diese Linien direkt in der *Zeichenfläche* benannt werden, das heißt, dass die notwendige Eingabe für den Zustandsübergang angegeben wird. Analog können auch die Zustände direkt in der *Zeichenfläche* benannt werden. Damit kann bei der Modellierung gänzlich auf das *Property-View* verzichtet werden und ausschließlich mit der grafischen Oberfläche (*Zeichenfläche*, *Toolbar*) gearbeitet werden.

Zum Prototypen 2:

Der Prototyp 2 ist ein mit Hilfe von GMF generierter Editor, der der Erzeugung und Darstellung von *Rectangle EClasses* der Klasse *gmfgraph.ecore* aus dem Paket *org.eclipse.gmf.graphdef* dient. Diese *Rectangle EClasses* werden in beiden Editoren (2 & 3) durch Rechtecke dargestellt, die in der *Toolbar* ausgewählt und auf der *Zeichenfläche* gezeichnet werden können. Beim Erstellen eines solchen Rechteckes wird in einer dazugehörigen *gmfgraph-Datei* eine *Rectangle EClass* instanziiert. Die so entstanden Datei, kann anschließend zum Entwickeln neuer Editoren mittels GMF weiterverwendet werden, da die enthaltenen Komponenten dem durch *gmfgraph.ecore* definierten Standard entsprechen.

Für die Umsetzung des Problems wurde zunächst *gmfgraph.ecore* als *ecore-Modell* verwendet (Dabei musste das *Property „abstract"* bei allen Interfaces auf *„true"* gesetzt werden, damit das Modell fehlerfrei wurde). Dadurch wird es später möglich sein, die im Modell enthaltenen *Rectangle EClasses* zu instanziieren. Anschließend wurde eine *gmfgraph-Datei* erzeugt, in der ein durch ein *Rectangle* dargestelltes *Node* eingefügt wurde. Dieses *Node* stellt die grafische Repräsentation der *Rectangle EClass* dar. Im Weiteren wurde eine *gmftool-Datei* erstellt, in der ein *Creation Tool* für das zu zeichnende Rechteck eingefügt wurde. Die *EClass,* das *Node* und das *Creation Tool* wurden anschließend im *gfmmap-Modell* abgebildet und das generierte *Label-Mapping* entfernt, da im Editor kein *Label* benötigt wird. Als *RootElement* wurde die *EClass FigureGallery* gewählt, da sie üblicherweise in *gmfgraph-Dateien* Figuren wie etwa *Rectangles* enthält.

Aus den *gmfgraph/gmftool/ecore/gfmmap-Dateien* kann nun der Editor für den Prototyp 2 generiert werden. Der entstandene Code dient gleichzeitig als Grundlage für den auf Prototyp 2 aufbauenden dritten Prototyp.

Zum Prototypen 3:

Der Prototyp 3 baut auf dem zweiten Prototypen auf und erlaubt es zusätzlich, die Farbe der instanziierten *Rectangle EClass* auszuwählen, die in der dazugehörigen *gmfgraph-Datei* abgespeichert werden. Außerdem soll das dargestellte Rechteck (die Linie) die gewählte Farbe annehmen. Als Format wurde hier die *EClass RGBColor* gewählt, mit der es möglich ist, die Farbanteile von Rot, Grün und Blau separat festzulegen. Dazu wurde für diese *EClass* ein neues *Creation Tool* sowie ein neuer *Node* (ebenfalls als *Rectangle* dargestellt) inklusive 3 *Labels* (die die Farbwerte enthalten) erstellt. Im *Mapping-Modell* wurden nun wiederum *EClass, Node* und *Creation Tool* gemapped, allerdings als *ChildReference* vom *Mapping* des Rechteckes. Damit wird es im Editor nur möglich sein, ein *RGBColor*-Objekt innerhalb eines Rechteckes zu erzeugen, was

die Intuitivität der Bedienung erhöht. Weiterhin wurde die *Property „Containment Feature*“ der „*ChildReference*“ auf „*foregroundColor*“ gesetzt, sodass sich die Farbänderungen auf die Linie des *Rectangles* auswirken. Zusätzlich werden nun noch drei *Label-Mappings* eingeführt, die die Farb-Attribute von *RGBColor* auf die Labels aus dem *gmfgraph-Modell* abbilden. Damit ist nun gewährleistet, dass im Editor beim Erstellen eines *RGBColor*-Objektes in einem Rechteck auch ein *foregroundColor*-Objekt als *Child* von *Rectangle* in der *gmfgraph-Datei* erstellt wird. Die Farbwerte von *foregroundColor* entsprechen dabei den Werten des in den *Labeln* enthaltenen Label-Textes.
Für die Realisierung der Farbänderung des im Editor dargestellten Rechteckes, wird nun im *gmfgraph-Modell* dem *RGBColor-Node* eine *foreGroundColor* hinzugefügt. Im Editor-Quellcode sind nun noch einige Änderungen notwendig, um die Werte der einzelnen *Labels*, auf diese *foreGroundColor* abbilden zu können. Dazu wird die Methode „getWrapLabel()“ von

- gmfgraph.diagram.edit.parts
    - MygmfgraphEditPartFactory
        - TextCellEditorLocator

abgeändert. (Details, siehe Quellcode)
Um die Bedienung des Editors noch benutzerfreundlicher zu gestalten, wird nun noch die Schriftfarbe der *Labels* so abgeändert, dass sie jeweils dem dazugehörigen Farbanteil im *RGBColor*-Objekt entsprechen. Dafür wurde die Methode „setFontColor(Color color)“ der Klassen RGBColorBlueEditPart, RGBColorGreenEditPart, RGBColorRedEditPart aus dem Package „gmfgraph.diagram.edit.parts“ so abgeändert, das jeweils die passende Schriftfarbe gesetzt wird (Details, siehe Quellcode).
Nun ist leicht zu sehen, ob der Rot-,Grün- oder Blauanteil geändert wird, was das Arbeiten mit dem Editor erheblich vereinfacht.