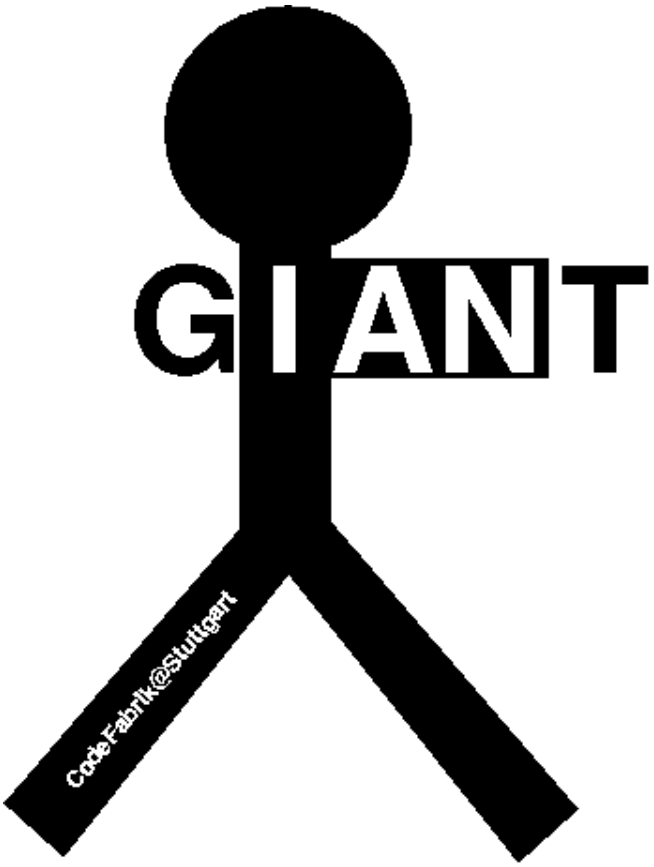
Universität Stuttgart
Studienprojekt A – IML Browser
CodeFabrik@Stuttgart

Spezifikation

Version 1.0



Versionsgeschichte

- Version 1.0 (08.04.2003)

  Zur Prüfung im Kundenreview am 14.04.2003 - 16.04.2003.

Inhaltsverzeichnis

1 Einleitung

1.1 Über dieses Dokument

Diese Spezifikation beschreibt die funktionalen und die nicht-funktionalen Anforderungen an den IML-Browser GIANT sowie die Rahmenbedingungen, unter denen GIANT lauffähig sein muss.
Dieses Dokument ist Grundlage für die Entwicklung aller weiteren Dokumente innerhalb dieses Projekts, das heißt im besonderen für das Benutzerhandbuch, den Entwurf, die Implementierung und den Test des Softwaresystems. Das Dokument ist an die Mitglieder der Projektgruppe sowie an den Kunden und dessen technische Berater gerichtet.
Dieses Dokument ist Vertragsbestandteil für die weitere Entwicklung von GIANT und somit auch Grundlage für die Abnahme des fertigen Produktes.

1.2 Über GIANT – Graphical IML Analysis Navigation Tool

GIANT ist ein Werkzeug, welches an der Universität Stuttgart von der CodeFabrik@Stuttgart im Rahmen des Studienprojektes A IML-Browser des Studiengangs Softwaretechnik entwickelt wird.
Ziel dieser Entwicklung ist es, die bereits bestehende HTML-basierte Lösung der Bauhaus Reengineering GmbH für IML-Graphen durch ein komfortables graphisches Werkzeug zu ergänzen. GIANT soll es dem Kunden ermöglichen, Teile von großen IML-Graphen zu visualisieren. Durch die Unterscheidung der verschiedenen Kanten- und Knotenklassen sollen darüber hinaus auch im IML-Graphen enthaltene Analyseergebnisse übersichtlich dargestellt werden können.
Das Produkt soll sich durch einen hohen Grad an Wartbarkeit auszeichnen. Dies soll insbesondere dadurch erreicht werden, dass die Anbindung an die Bauhaus-IML-Graph-Bibliothek derart vorgenommen wird, dass Änderungen in der Spezifikation des IML-Graphen, wie z.B. das Einbringen neuer Attribute, möglichst keine Wartungsarbeiten an GIANT selbst zur Folge haben.

2 Produktübersicht

In diesem Kapitel werden grundlegende Konzepte von GIANT vorgestellt und kurz beschrieben. Eine ausführliche Beschreibung dieser Konzepte erfolgt weiter hinten in der Spezifikation.

2.1 GIANT Projekte

Innerhalb eines Projektes fasst GIANT Informationen, wie Anzeigefenster und IML-Teilgraphen für einen vorgegebenen IML-Graphen, zusammen und speichert diese persistent. Die Zusammenfassung zu Projekten soll der Übersicht dienen und den Austausch von Teilergebnissen, wie z.B. einzelner Anzeigefenster, erleichtern.
Für weitere Informationen zu Projekten siehe Kapitel 12.

2.2 IML-Teilgraphen und Selektionen

Hinsichtlich der logischen Zusammenfassung von Knoten und Kanten unterscheidet GIANT IML-Teilgraphen und Selektionen.

2.2.1 IML-Teilgraphen

Ein IML-Teilgraph ist eine Knoten- und Kantenmenge aus dem IML-Graphen mit der Bedingung, daß die Start- und Zielknoten jeder Kante Teil der Menge sind. Diese sogenannten Graph-Knoten und Graph-Kanten können in Anzeigefenstern hervorgehoben werden. Desweiteren können die Graph-Knoten und Graph-Kanten von IML-Teilgraphen als Fenster-Knoten und Fenster-Kanten in Anzeigefenster eingefügt und layoutet werden. Die IML-Teilgraphen sind global in der Anwendung verfügbar, aber völlig unabhängig von den Anzeigefenstern und enthalten insbesondere keine Layoutinformationen.

2.2.2 Selektionen

Selektionen stellen die lokale Variante von Knoten- und Kantenmengen dar. Sie sind immer einem festen Anzeigefenster zugeordnet. Selektionen können beliebige Knoten und Kanten umfassen, ohne dass diese einen Teilgraphen bilden müssen. Jeder Knoten und jede Kante einer Selektion muss auch im Anzeigefenster als Fenster-Knoten oder Fenster-Kante vorhanden sein.

2.3 Anzeigefenster

Anzeigefenster sind die Fenster von GIANT, in denen eine benutzerdefinierte Auswahl von Knoten und Kanten visualisiert wird. Es kann beliebig viele Anzeigefenster geben und jedes Anzeigefenster kann beliebig viele Selektionen haben.

2.3.1 Pins

Da bei großen Graphen selten alle zu einem Anzeigeinhalt gehörenden Fenster-Knoten und Fenster-Kanten gemeinsam auf dem Bildschirm sichtbar dargestellt werden können, kann sich der Benutzer zu jedem Anzeigefenster eine Liste von Pins anlegen. In den Pins wird jeweils die Position des sichtbaren Anzeigeinhaltes und die Zoomstufe gespeichert, so dass zu beliebigen Zeitpunkten die Position des sichtbaren Anzeigeinhalts rekonstruiert werden kann.

2.3.2 Visualisierungsstile

Mittels sogenannter Visualisierungsstile kann der Benutzer die Darstellung von Fenster-Kanten und Fenster-Knoten auch während der Laufzeit von GIANT beeinflussen. Da über entsprechende XML-Dateien verschiedene Visualisierungsstile definiert werden können, kann GIANT so an spezifische Problemstellungen angepasst werden.
Weitere Informationen sind in Abschnitt 13.3 verfügbar.

2.4 Knoten-Annotationen

Jeder Knoten kann mit einer textuellen Annotation versehen werden. Diese Annotation kann in einem Fenster außerhalb des Anzeigefensters zur Anzeige gebracht und bearbeitet werden.
Weitere Informationen sind in Abschnitt 12.2.4 verfügbar.

2.5 Anfragen

Eine vielseitige Anfragesprache – die GIANT Query and Scripting Language GQSL – stellt nahezu die gesamte Funktionalität von GIANT zur Verfügung und kann insbesondere auch zum Aufruf via Kommandozeile genutzt werden. Die drei Schritte des anschließend beschriebenen „Drei-Stufen-Konzeptes“ können über diese Anfragesprache auch „auf einen Schlag“ erledigt werden.
Die GQSL ist unter Kapitel 10 im Detail spezifiziert.

2.6 Drei-Stufen-Konzept

Die im Folgenden beschriebenen drei Schritte sollen dem Benutzer die Möglichkeit bieten, von den Knoten und Kanten des IML-Graphen ausgehend geeignete Teilgraphen in Anzeigefenstern zu visua-

lisieren. Von der Gestaltung der UseCases her, werden diese drei Schritte sequentiell nacheinander ausgeführt, dies ist aber nicht zwingend nötig.

1. Erzeugen geeigneter IML-Teilgraphen, d.h. Auswahl geeigneter Knoten und Kanten aus dem IML-Graphen mittels der Anfragesprache.
2. Einfügen dieser IML-Teilgraphen in ein Anzeigefenster unter Anwendung eines Layoutalgorithmus.
3. Weitere Bearbeitung der Fenster-Knoten und Fenster-Kanten (wie z.B. Verschieben, Annotieren und Erzeugen von Selektionen).

3 Allgemeine Funktionale Anforderungen

Dieses Kapitel beschreibt die funktionalen Anforderungen an GIANT, die das System als Ganzes und nicht nur einzelne UseCases betreffen. Dies sind z.B. allgemeine Bedienkonzepte zur Selektion von Knoten und Kanten, sowie das grundlegende Fehlerverhalten von GIANT.

3.1 Fehlerverhalten

Beim Auftreten von Fehlern während des Betriebs von GIANT werden wann immer möglich die folgenden Konventionen eingehalten.

1. Fehlermeldungen werden grundsätzlich über einen allgemeinen Fehlerdialog ausgegeben (siehe auch 14.11). Ist dies nicht möglich, wird die Fehlermeldung über die Kommandozeile ausgegeben.

2. GIANT verhält sich bezüglich der Fehlerbehandlung konsistent. Das im Folgenden beschriebene Fehlerverhalten gilt daher für das gesamte System.

3. Abweichungen von diesem Verhalten werden gegebenenfalls an der entsprechenden Stelle explizit spezifiziert.

4. Sollten mehrere Fehler, wie z.B. Eingabefehler, gleichzeitig auftreten, so werden die entsprechenden Fehlermeldungen falls möglich in dem allgemeinen Fehlerdialog „auf einen Schlag“ an den Benutzer ausgegeben.

3.1.1 Ungültige oder unvollständige Eingaben bei Dialogen

1. Eingaben in Dialogen werden immer dann geprüft, wenn der Benutzer alle Eingaben in dem Dialogfenster durch klicken auf einen „OK-Button“ bestätigt.

2. Stellt das System dann eine unzulässige oder fehlende Eingabe fest, so erscheint der allgemeine Fehlerdialog mit einer Mitteilung zur Fehlerursache.

3. Nach Schließen des Fehlerdialoges kehrt das System zu der Stelle, die im Ablauf unmittelbar vor der Fehlerüberprüfung lag, zurück. Also typischerweise zu dem Dialog, in dem die fehlerhafte Eingabe getätigt wurde. Die zuvor in den jeweiligen Dialogen getätigten Eingaben des Benutzers bleiben erhalten.

3.1.2 Ungültige oder unvollständige Eingaben bei Kommandozeilenaufruf

Der Start von GIANT wird abgebrochen. Auf der Kommandozeile wird eine Meldung mit Hinweisen zur Fehlerursache ausgegeben.

3.1.3 Fehler beim Einlesen von Dateien

1. Können zum Betrieb von GIANT nötige Dateien nicht an der dafür vorgesehenen Position gefunden werden, so wird das System mit einer entsprechenden Fehlermeldung beendet. In dieser Fehlermeldung werden die nicht gefundenen Dateien und der Pfad, der von GIANT durchsucht wurde, angegeben.
2. Das Verhalten bei allen anderen möglichen Fehlern, die im Zusammenhang mit dem Zugriff auf Dateien auftreten (fehlende Zugriffsrechte, defekte Sektoren etc.), ist undefiniert.

3.1.4 Einlesen unzulässiger Daten aus Dateien

1. GIANT verfügt über keine Mechanismen, die speziell darauf ausgelegt sind, die Korrektheit von Daten, die aus Dateien gelesen werden, zu prüfen.
2. Stellt das System dennoch fest, dass eine einzulesende Datei ungültige Daten enthält, so wird das System mit einer entsprechenden Fehlermeldung beendet.

3.2 Grundlegendes Feedbackverhalten

Hier wird beschrieben, wie GIANT dem Benutzer über den aktuellen Systemzustand informiert.

3.2.1 „I am busy Feedback“

Während der Berechnung durch Layoutalgorithmen und Anfragen informiert GIANT den Benutzer laufend, dass es gerade mit Berechnungen beschäftigt ist. Dies keine über eine der beiden hier beschriebenen Möglichkeiten geschehen:

1. Anzeige eines Dialoges mit einem sich ständig leicht ändernden Inhalt, so dass erkennbar ist, dass das System noch läuft.
2. Anzeige einer Progressbar, der über den aktuellen Fortschritt einer Berechnung informiert, ohne dabei den Gesamtaufwand zu kennen.

3.3 Zulässige Eingaben

Hier wird beschrieben, welche Eingaben für Namen etc. innerhalb von GIANT (z.B. bei den entsprechenden Texteingabedialogen) zulässig sind. Unzulässige Namen werden generell abgelehnt und führen zu einer Fehlermeldung.

3.3.1 Bezeichnertypen

Hier werden Bezeichnertypen, also vom GIANT als zulässig akzeptierte Folgen von Ziffern, Buchstaben und Sonderzeichen definiert.

3.3.1.1 Der String „STANDARD NAME“

Ein String, der nur die im Folgenden beschriebenen Zeichen enthalten darf.
Zulässige Zeichen sind:

1. Groß- und Kleinbuchstaben des deutschen Alphabets. Nicht zulässig sind Umlaute und das Zeichen ”ß”.
2. Die Ziffern ”0” bis ”9”.
3. Das Zeichen ”_”.

3.3.2 Zulässige Namen

Für Namen, die der Benutzer z.B. für Projekte vergeben darf, wird hier der jeweils zulässige Bezeichnertyp (siehe 3.3.1) spezifiziert.

1. Name eines Projektes: STANDARD NAME
2. Name eines IML-Teilgraphen: STANDARD NAME
   Dieser Name muss für jeden IML-Teilgraphen innerhalb eines Projektes eindeutig sein.
3. Name eines Anzeigefensters: STANDARD NAME
   Dieser Name muss für jedes Anzeigefenster innerhalb des Projektes eindeutig sein.
4. Name einer Selektion: STANDARD NAME
   Dieser Name muss für jede Selektion innerhalb eines Anzeigefensters eindeutig sein.
5. Name eines Pins: STANDARD NAME
   Dieser Name muss für jeden Pin innerhalb eines Anzeigefensters eindeutig sein.

3.4 Allgemeine Bedienkonzepte

Hier werden grundlegende Konzepte der Bedienung von GIANT beschrieben.

3.4.1 Selektieren von Fenster-Knoten und Fenster-Kanten in Anzeigefenstern

Hier werden die Abläufe zur Selektion und Deselektion einzelner Fenster-Knoten und Fenster-Kanten spezifiziert.

1. Selektieren durch Anklicken mit linker Maustaste.
   Nur der angeklickte Fenster-Knoten oder die angeklickte Fenster-Kante wird selektiert. Andere bis dahin selektierte Fenster-Knoten und Fenster-Kanten werden deselektiert.

2. Selektieren durch Anklicken mittels Shift + linke Maustaste.
   Falls noch nicht selektiert wird der angeklickte Fenster-Knoten oder die angeklickte Fenster-Kante zu der aktuellen Selektion hinzugefügt. Falls bereits selektiert wird der Fenster-Knoten oder die Fenster-Kante aus der aktuellen Selektion entfernt.
3. Selektieren durch Aufziehen eines Rahmens mittels gedrückter linker Maustaste.
   Alle Fenster-Knoten und Fenster-Kanten innerhalb des Rahmens werden selektiert, alle Fenster-Knoten und Fenster-Kanten außerhalb des Rahmens werden deselektiert.
4. Selektieren durch Aufziehen eines Rahmens mittels Shift + gedrückte linke Maustaste.
   Alle Fenster-Knoten und Kanten innerhalb des Rahmens, die noch nicht selektiert sind, werden der aktuellen Selektion hinzugefügt.
   Alle Fenster-Knoten und Kanten innerhalb des Rahmens, die bereits selektiert sind, werden aus der aktuellen Selektion entfernt.

3.4.2 Standard-Selektion

Jedes Anzeigefenster hat genau eine Standard-Selektion. Diese Selektion wird bei der Erzeugung des Fensters angelegt und ist zu Anfang leer. Die Standard-Selektion kann nicht gelöscht werden. Abgesehen davon verhält sich die Standard-Selektion wie vom Benutzer angelegte Selektionen.

3.4.3 Aktuelle Selektion

1. Es gibt zu jedem Anzeigefenster beliebig viele Selektionen, davon ist immer eine die aktuelle Selektion (siehe 3.4.1).
2. Das oben beschriebene Selektieren mittels linker Maustaste etc. betrifft immer nur die aktuelle Selektion.
   Alle anderen Selektionen können zwar hervorgehoben werden, aber nicht durch das Selektieren mittels linker Maustaste etc. geändert werden.
3. Der Benutzer kann jederzeit bestimmen, welche der Selektion die aktuelle Selektion ist.
4. Die aktuelle Selektion wird immer hervorgehoben.

3.5 Verhalten beim Einfügen von IML-Teilgraphen und Selektionen in Anzeigefenster

In diesem Abschnitt wir das Einfügen von IML-Teilgraphen und Selektionen in Anzeigefenster spezifiziert.

3.5.1 Vorgabe der Zielposition

Für bestimmte UseCases (siehe z.B. 4.14) muss der Benutzer manuell Zielpositionen innerhalb eines zu bestimmenden Anzeigefensters auswählen. An dieser Zielposition werden dann z.B. neue Fenster-

Knoten etc. eingefügt.
Ablauf:

1. Der Benutzer wählt das entsprechende Anzeigefenster aus (dadurch dass er ihm je nach Betriebssystemkonvention den Fokus gibt). Hierbei können nur bereits geöffnete Anzeigefenster des Projektes ausgewählt werden.
2. Befindet sich die Maus über dem Bereich des Anzeigefensters, wo der IML-Graph visualisiert wird, so wird anstatt des Mauszeigers ein Fadenkreuz angezeigt.
3. Durch Klicken mit der linken Maustaste wird dann über das Fadenkreuz die Zielposition vorgegeben.

3.5.2 Automatisches Erzeugen neuer Selektionen im Zielfenster

Sämtliche Fenster-Knoten und Fenster-Kanten, die in einem Schritt in ein Anzeigefenster eingefügt werden, werden dort automatisch zur aktuellen Selektion und können dann vom Benutzer mit einem Namen versehen werden und als Selektion gespeichert werden.

3.5.3 Einfügen von Selektionen in Anzeigefenster

Hier wird das grundlegende Verhalten beim Einfügen von Selektionen in Anzeigefenster (Kopieren einer Selektion aus einem Anzeigefenster in ein anderes) beschrieben.

1. Das Layout der Selektion wird beim Einfügen in ein Anzeigefenster übernommen.
2. Die Position von Knoten der einzufügenden Selektion, die bereits in dem Anzeigefenster visualisiert sind, bleibt falls nicht an entsprechender Stelle anders spezifiziert unverändert.
3. Kanten der Selektion werden nur eingefügt, falls ihr Start- und Zielknoten ebenfalls Bestandteil der Selektion ist oder bereits im Anzeigefenster visualisiert ist.
4. Die neu eingefügte Selektion ist nun auch dem entsprechenden Anzeigefenster (dort wo sie eingefügt wurde) als Selektion bekannt.

3.5.4 Einfügen von IML-Teilgraphen in Anzeigefenster

Hier wird das grundlegende Verhalten des Systems beim Einfügen von IML-Teilgraphen in Anzeigefenster (Kopieren einer Selektion aus einem Anzeigefenster in ein anderes) beschrieben.

1. Das Layout für den IML-Teilgraphen wird vor dem Einfügen gemäß eines vom Benutzer vorgegebenen Layoutalgorithmus berechnet.
2. Die Graph-Knoten und Graph-Kanten des IML-Teilgraphen werden dann genau wie eine Selektion in das Anzeigefenster eingefügt, dies geschieht dann zu den gleichen Bedingungen, wie oben unter 3.5.3 beschrieben.

3.6 Verhalten bei Umwandeln von Selektionen und IML-Teilgraphen

Dieser Abschnitt beschreibt grundlegende Konventionen für die Umwandlung von IML-Teilgraphen in Selektionen und anders herum.

3.6.1 Selektion aus IML-Teilgraphen ableiten

1. Der IML-Teilgraph bleibt unverändert.
2. Im zu bestimmenden Anzeigefenster wird eine neue Selektion erzeugt, die alle Knoten und Kanten des IML-Teilgraphen umfasst, welche bereits als Fenster-Knoten und Fenster-Kanten im Anzeigefenster vorhanden sind.
3. Knoten und Kanten des IML-Teilgraphen, die nicht als Fenster-Knoten und Fenster-Kanten im entsprechenden Anzeigefenster vorhanden sind, werden ignoriert.

3.6.2 IML-Teilgraph aus Selektion ableiten

1. Die Selektion bleibt unverändert.
2. Es wird ein neuer IML-Teilgraph erzeugt. Dieser umfasst alle Knoten und Kanten der Selektion, welche einen gültigen Teilgraphen bilden. Kanten der Selektion, deren Start- und Zielknoten nicht ebenfalls Bestandteil der Selektion sind, werden also ignoriert.

3.7 Verhalten beim Entfernen von Fenster-Knoten und Fenster-Kanten

Beim „Entfernen“ von Fenster-Knoten und Fenster-Kanten aus einem Anzeigefenster werden nur die graphischen Repräsentationen der Knoten und Kanten des Bauhaus-IML-Graphen in dem betroffenen Anzeigefenster gelöscht. Die IML-Knoten und IML-Kanten des IML-Graphen bleiben davon unberührt. Grundsätzlich gelten beim „Entfernen“ die im Folgenden beschriebenen Konventionen.

3.7.1 Entfernen aller Knoten und Kanten einer Selektion

1. Alle Fenster-Knoten und Fenster-Kanten der Selektion werden aus dem Anzeigefenster entfernt.
2. Fenster-Kanten im Anzeigefenster, deren Start- oder Zielknoten entfernt wird, werden ebenfalls entfernt (auch wenn sie nicht zur entsprechenden Selektion gehören).
3. Andere Selektionen des Anzeigefensters werden entsprechend aktualisiert, die betroffenen Fenster-Knoten und Fenster-Kanten werden also auch aus diesen Selektionen entfernt.

3.8 Auseinanderschieben von Fenster-Knoten

Sollen Fenster-Knoten auf dem Anzeigeinhalt automatisch auseinandergeschoben werden, so geschieht dies nach dem hier beschriebenen Modus. GIANT stellt diese Funktionalität dem Benutzer über einen UseCase zur manuellen Ausführung zur Verfügung. Ein automatisches „Auseinanderschieben von Fenster-Knoten“ durch GIANT selbst ist nicht vorgesehen, der Entwurf wird aber eine einfache Erweiterbarkeit von GIANT diesbezüglich unterstützen.

1. Es gibt einen fixen Punkt von dem aus alle Fenster-Knoten weggeschoben werden.
2. Die Fenster-Knoten werden immer entlang einer Strecke von diesem Punkt durch den jeweiligen Fensterknoten verschoben.
3. Jeder Fenster-Knoten wird um einen konstanten Betrag (kann vom Benutzer eingegeben werden) von dem Punkt weggeschoben.

4 Funktionale Anforderungen

Dieses Kapitel beschreibt die funktionalen Anforderungen an GIANT in Form von UseCases, d.h. alle Aktionsmöglichkeiten, die dem Benutzer über die GUI von GIANT zugänglich sind, werden in diesem Kapitel durch einen UseCase repräsentiert.

4.1 Der Aktor „Benutzer“

Aktor im Sinne der anschließend spezifizierten UseCases ist immer die Person, die den IML-Browser GIANT gerade bedient, also der „Benutzer“. Neben dem „Benutzer“ sind keine weiteren Aktoren vorgesehen.

4.1.1 Anforderungen an den Aktor „Benutzer“

Zur Bedienung von GIANT muss der Benutzer über die folgenden Kenntnisse verfügen:

1. Erfahrung im Umgang mit graphischen Benutzeroberflächen und dem entsprechenden Betriebssystem.
2. Kenntnisse über Struktur und Aufbau des IML-Graphen.
3. Kenntnisse in der Skriptsprache GQSL von GIANT.

4.2 Starten von GIANT

Der Benutzer startet das Programm GIANT durch Ausführen der entsprechenden Programmdatei. Hierbei hat er bezüglich eventueller Kommandozeilenparameter die folgenden Optionen.

4.2.1 Mögliche Parameter

GIANT kann folgendermaßen gestartet werden:

```
giant [(-n file -i file|-l file) [-e file]|-h]
```

Hier werden die Parameter im einzelnen erklärt:

1. `--newproject=Pfad` oder `-n Pfad`

   Angabe des Pfades für die neu zu erstellenden Projektdatei. Die Datei darf nicht vorhanden sein. Das Verzeichnis der Projektdatei ist automatisch Projektverzeichnis und muss bereits existieren.

   `--iml=Pfad` oder `-i Pfad`

   Angabe des Pfades zu einer existierenden IML-Graph Datei.

2. `--loadproject=Pfad` oder `-l Pfad`

   Angabe des Pfades zu einer existierenden Projektdatei.

3. `--execute=Pfad` oder `-e Pfad`

   Pfad zu einer Datei in der ein GQSL Skript gespeichert ist.

4. `--help` oder `-h`

   Zeigt eine Hilfe zu den möglichen Kommandozeilenparametern an.

4.2.2 Mögliche Aufrufe

Hier werden mögliche Parameterfolgen zum Start von GIANT via Kommandozeile erläutert.

1. Eingabe von: `giant`
   GIANT startet und zeigt das Main Window an. Es ist noch kein Projekt geladen.
2. Eingabe von: `giant --newproject=/home/projekt/sample.giant --iml=sample.iml`
   Ein neues Projekt wird geöffnet und erhält den Namen der Projektdatei.
3. Eingabe von: `giant --loadproject=sample.giant`
   Das angegebene Projekt wird geöffnet.
4. Bei 2 und 3 kann zusätzlich der Parameter `--execute=file.script` eingegeben werden. In diesem Falle wird das übergebene GQSL Skript ausgeführt.

4.3 Beenden von GIANT

Ablauf:

1. GIANT wählt „Quit“ im Hauptmenü
2. Ist ein Projekt geöffnet, so fragt GIANT nach, ob das Projekt gespeichert werden soll. Falls der Benutzer das Speichern ablehnt, so gehen alle nicht gespeicherten Informationen verloren. Entscheidet er sich für Speichern, so wird die unter 12.2.1 beschriebene Funktionalität ausgeführt.
3. GIANT wird beendet.

4.4 UC: Neues Projekt

Erstellt ein neues GIANT Projekt. Ein eventuell bereits geöffnetes Projekt wird dabei geschlossen, wobei Änderungen auf Nachfrage vorher gespeichert werden. Weitere wichtige Informationen, die in engem Zusammenhang mit diesem UseCase stehen, sind unter dem Abschnitt 12.1 zu finden.
Zulässige Namen für Projekte sind unter Abschnitt 3.3.2 spezifiziert.

4.4.1 Vorbedingung

1. Das Programm ist gestartet.

4.4.2 Nachbedingung Erfolg

1. Ein neues GIANT Projekt mit dem eingegebenen Namen ist erstellt und geladen.
2. Eine IML-Datei ist geladen.
3. Das angegebene Projektverzeichnis ist gegebenenfalls erstellt worden (siehe 12.1.1).
4. Die Projektdatei (siehe 12.1.2) liegt im Projektverzeichnis.
5. Ein eventuell zuvor geöffnetes Projekt ist geschlossen.
6. Änderungen an einem zuvor geöffneten Projekt sind gespeichert oder verworfen.

4.4.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

4.4.4 Beschreibung

1. Der Benutzer startet den UseCase über das Hauptmenü (siehe 14.3.2.1) durch Auswahl von „New Project“.
2. Der Benutzer wählt im Standard-Filechooser-Dialog („Please Select IML File“) eine IML-Datei aus und bestätigt seine Eingabe mit OK (siehe 14.13.1).
3. Der Benutzer gibt im Standard-Filechooser-Dialog den Pfad und Namen der Projektdatei („Please select Project Path and Project File Name“) ein. Der Name der Projektdatei ist automatisch auch der Name für das Projekt.
4. Dann bestätigt er seine Eingabe mit OK.
   Existiert die eingegebene Projektdatei bereits, erscheint eine Fehlermeldung gemäß den unter Abschnitt 3.1 beschriebenen Konventionen.
   Existiert in dem Projektverzeichnis bereits eine andere Projektdatei, so erscheint ebenfalls eine Fehlermeldung.
   Das Verzeichnis in dem die Projektdatei liegt, wird automatisch zum Projektverzeichnis.

5. Falls bereits ein Projekt geöffnet ist, erscheint eine Sicherheitsabfrage (siehe 14.12) ob dieses gespeichert werden soll. Lehnt der Benutzer dies ab, gehen alle nicht gespeicherten Informationen verloren.
   Entscheidet der Benutzer sich für Speichern, so wird die unter 12.2.1 beschriebene Funktionalität ausgeführt.
6. GIANT schließt das aktuell geöffnete Projekt, erstellt ein neues Projekt und öffnet dieses.

4.4.5 Alternativen

- Bei Punkt 2:
  Der Benutzer bricht die Verarbeitung mit Cancel ab.
- Bei Punkt 3:
  Der Benutzer bricht die Verarbeitung mit Cancel ab.

4.5 UC: Projekt öffnen

Öffnet ein GIANT Projekt. Ein eventuell bereits vorhandenes Projekt wird dabei geschlossen, wobei Änderungen auf Nachfrage vorher gespeichert werden.
Sollte der Benutzer die XML-Dateien innerhalb des Projektverzeichnisses (siehe 12.1.1) manuell modifiziert haben, so dass diese von den durch GIANT automatisch erstellten Dateien abweichen, so wird keinerlei Garantie für das korrekte Öffnen des Projektes übernommen. Das Verhalten bezüglich eventuell auftretender Fehler ist undefiniert.

4.5.1 Vorbedingung

1. Das Programm ist gestartet.

4.5.2 Nachbedingung Erfolg

1. Das gewünschte GIANT Projekt ist geladen.
2. Die zugehörige IML-Datei ist geladen.
3. Änderungen an einem zuvor geöffneten Projekt sind gespeichert oder verworfen.

4.5.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

4.5.4 Beschreibung

1. Der Benutzer startet den UseCase über das Hauptmenü (siehe 14.3.2.1 Menü) durch Auswahl von „Load Project“.
2. Der Benutzer wählt aus dem Standard-Filechooser-Dialog (siehe 14.13.1) eine vorhandene GIANT Projektdatei (siehe 12.1.2) aus und bestätigt mit OK.
3. Falls bereits ein Projekt geöffnet ist, erscheint eine Sicherheitsabfrage (siehe 14.12) ob dieses gespeichert werden soll. Lehnt der Benutzer dies ab, gehen alle nicht gespeicherten Informationen verloren.
   Entscheidet der Benutzer sich für Speichern, so wird die unter 12.2.1 beschriebene Funktionalität ausgeführt.
4. GIANT schließt das alte Projekt und lädt das angegebene Projekt.

4.5.5 Alternativen

- Bei Punkt 2:
  Der Benutzer bricht die Verarbeitung mit Cancel ab.

4.6 UC: Projekt speichern

Speichert alle Änderungen an einem Projekt. Der Zustand der entsprechenden Verwaltungsdateien im Projektverzeichnis entspricht nach erfolgreicher Ausführung dieses UseCases exakt dem aktuellen Zustand des geöffneten Projektes. Alle Konventionen zur Persistenz von Projekten sind im Abschnitt 12.1 exakt spezifiziert.

4.6.1 Vorbedingung

1. Das Programm ist gestartet.
2. Ein Projekt ist geöffnet.

4.6.2 Nachbedingung Erfolg

1. Die Informationen des Projekts (einschließlich aller möglichen Änderungen) sind persistent in die Verwaltungsdateien geschrieben.

4.6.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

4.6.4 Beschreibung

1. Der Benutzer startet den UseCase über das Hauptmenü (siehe 14.3.2.1) durch Auswahl von „Save Project“.
2. GIANT führt die unter 12.2.1 beschriebene Funktionalität aus und speichert alle Informationen zu dem Projekt in der zugehörigen Projektdatei.

4.6.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.7 UC: Projekt speichern unter

Speichert alle Informationen zu einem Projekt in eine neue Projektdatei (entsprechende Verwaltungsdateien werden ebenfalls dupliziert).

4.7.1 Vorbedingung

1. Das Programm ist gestartet.
2. Ein Projekt ist geöffnet (entweder ein neu erzeugtes oder ein geladenes).

4.7.2 Nachbedingung Erfolg

1. Eine neue Projektdatei ist erzeugt.
2. Das angegebene Projektverzeichnis ist gegebenenfalls erstellt worden.
3. Die Informationen des Projekts sind persistent in die neuen Verwaltungsdateien im Projektverzeichnis der neuen Projektdatei geschrieben (siehe auch 12.1).
4. Das Projekt bleibt in GIANT geöffnet, zukünftiges Speichern (siehe 4.6) betrifft nur die Dateien im neu erzeugten Projekt.
5. Die alte Projektdatei und alle Verwaltungsdateien bleiben unverändert.

4.7.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

4.7.4 Beschreibung

1. Der Benutzer startet den UseCase über das Hauptmenü (siehe 14.3.2.1) durch Auswahl von „Save Project As...“.
2. Der Benutzer gibt im Standard-Filechooser-Dialog (siehe 14.13.1) den Pfad, das neue Projektverzeichnis (siehe 12.1.1) und den Namen für die neue Projektdatei (siehe 12.1.2) vor.
   Der Name der Projektdatei ist automatisch auch der Name für das Projekt. Zulässige Namen sind unter Abschnitt 3.3.2 spezifiziert.
   Das Verzeichnis in dem die Projektdatei liegt, wird automatisch zum Projektverzeichnis.
3. Der Benutzer bestätigt seine Eingabe mit OK.
   Existiert die eingegebene Datei schon, erscheint eine Fehlermeldung.
   Existiert in dem Projektverzeichnis bereits eine andere Projektdatei, so erscheint eine entsprechende Fehlermeldung.

4.7.5 Alternativen

- Bei Punkt 2:
  Der Benutzer bricht die Verarbeitung mit Cancel ab.

4.8 UC: Leeres Anzeigefenster erzeugen

Über diesen UseCase kann der Benutzer neue Anzeigefenster innerhalb eines Projektes anlegen.

4.8.1 Vorbedingung

1. Ein Projekt ist geladen.

4.8.2 Nachbedingung Erfolg

1. Das neue, leere Anzeigefenster ist geöffnet.
2. Das neue Anzeigefenster ist Bestandteil des Projektes.

4.8.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

4.8.4 Beschreibung

1. Der Benutzer startet den UseCase über das Popup-Menü (siehe 14.3.4.2) durch Auswahl von „New Window“.
2. GIANT erzeugt ein neues Anzeigefenster (siehe 14.4) mit einem Standard-Namen und öffnet dies.

4.8.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.9 UC: Anzeigefenster öffnen

Dient zum Öffnen eines Anzeigefensters des Projektes.

4.9.1 Vorbedingung

1. Ein Projekt mit mindestens einem Anzeigefenster ist geladen.
2. Es gibt mindestens ein nicht geöffnetes Anzeigefenster.

4.9.2 Nachbedingung Erfolg

1. Das Anzeigefenster ist geöffnet.

4.9.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.9.4 Beschreibung

1. Der Benutzer führt einen Doppelklick auf ein nicht geöffnetes Anzeigefenster in der Liste über die Anzeigefenster (siehe 14.3.4.1) durch, oder wählt im zugehöhrigen Popup-Menü (siehe 14.3.4.2) „Open Window“ aus.
2. GIANT öffnet das entsprechende Anzeigefenster.

4.9.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.10 UC: Anzeigefenster umbenennen

Dient zum Umbenennen eines Anzeigefensters des Projektes.

4.10.1 Vorbedingung

1. Ein Projekt mit mindestens einem Anzeigefenster ist geladen.

4.10.2 Nachbedingung Erfolg

1. Das Anzeigefenster hat einen neuen Namen.

4.10.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.10.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf ein Anzeigefenster in der Liste über die Anzeigefenster (siehe 14.3.4.1) durch und wählt im Popup-Menü (siehe 14.3.4.2) „Rename Window“ aus.
2. GIANT öffnet den allgemeinen Texteingabedialog (siehe 14.7) „Enter Name For Window“.
3. Der Benutzer gibt dort einen zulässigen Namen für das Anzeigefenster ein und bestätigt seine Eingabe mit OK.
4. GIANT benennt das Anzeigefenster (siehe 14.4) um.

4.10.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.11 UC: Anzeigefenster speichern

Mit diesem UseCase wird ein Anzeigefenster gespeichert. Näheres zur Persistenz von Anzeigefenstern ist unter Abschnitt 12.1 spezifiziert.

4.11.1 Vorbedingung

1. Ein Projekt mit mindestens einem Anzeigefenster ist geladen.

4.11.2 Nachbedingung Erfolg

1. Nach dem letzten Speichern am Anzeigefenster vorgenommene Modifikationen sind in der Verwaltungsdatei (siehe 12.1.4) gespeichert.

4.11.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.11.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf ein Anzeigefenster in der Liste über die Anzeigefenster (siehe 14.3.4.1) durch und wählt im Popup-Menü (siehe 14.3.4.2) „Save Window“ aus.
2. GIANT schreibt alle Änderungen in die Verwaltungsdatei des Anzeigefensters.

4.11.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.12 UC: Anzeigefenster schließen

Mit diesem UseCase wird ein geöffnetes Anzeigefenster geschlossen. Näheres zur Persistenz von Anzeigefenstern ist unter Abschnitt 12.2.2 spezifiziert.

4.12.1 Vorbedingung

1. Ein Projekt mit mindestens einem Anzeigefenster ist geladen.
2. Es gibt mindestens ein geöffnetes Anzeigefenster.

4.12.2 Nachbedingung Erfolg

1. Das Anzeigefenster ist geschlossen.
2. Nach dem letzten Speichern am Anzeigefenster vorgenommene Modifikationen sind in der Verwaltungsdatei (siehe 12.1.4) gespeichert.

4.12.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.12.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf ein Anzeigefenster in der Liste über die Anzeigefenster (siehe 14.3.4.1) durch und wählt im Popup-Menü (siehe 14.3.4.2) „Close Window“ aus.
2. GIANT zeigt die allgemeine Sicherheitsabfrage (siehe 14.12) und fragt nach, ob eventuelle Änderungen im Anzeigefenster gespeichert werden sollen oder nicht „The content has changed. Do you want to save the changes?“.
3. Bestätigt der Benutzer mit Yes, werden die Änderungen in die Verwaltungsdatei geschrieben. Anderenfalls gehen sämtliche nicht gespeicherten Änderungen am Anzeigefenster verloren.
4. GIANT schließt das Anzeigefenster.

4.12.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.13 UC: Anzeigefenster löschen

Mit diesem UseCase werden bestehende Anzeigefenster aus dem Projekt entfernt und gelöscht. Alle Informationen zu dem Anzeigefenster gehen hierbei unwiederbringlich verloren.

4.13.1 Vorbedingung

1. Ein Projekt mit mindestens einem Anzeigefenster ist geladen.

4.13.2 Nachbedingung Erfolg

1. Das gelöschte Anzeigefenster ist nicht mehr Bestandteil des Projektes.
2. Die Verwaltungsdatei für das Anzeigefenster (siehe 12.1.4) wurde ebenfalls gelöscht.

4.13.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

4.13.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf ein Anzeigefenster in der Liste über die Anzeigefenster (siehe 14.3.4.1) durch und wählt im Popup-Menü (siehe 14.3.4.2) „Delete Window“ aus.
2. GIANT zeigt die allgemeine Sicherheitsabfrage (siehe 14.12) an und fragt nach, ob das Anzeigefenster wirklich gelöscht werden soll: „Do you really want to delete the selected window?“
3. Der Benutzer bestätigt mit Yes.
4. GIANT entfernt das Anzeigefenster aus dem Projekt und löscht die zugehörige Verwaltungsdatei (siehe auch 12.1.4).

4.13.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht die Verarbeitung mit No ab.

4.14 UC: IML-Teilgraph in Anzeigefenster einfügen

Mit diesem UseCase können die Graph-Kanten und Graph-Knoten von IML-Teilgraphen in Anzeigefenster eingefügt werden. Siehe auch 3.5 und insbesondere 3.5.4.

4.14.1 Vorbedingung

1. Ein Projekt mit mindestens einem geöffneten Anzeigefenster ist geladen.
2. Es gibt mindestens einen IML-Teilgraphen.

4.14.2 Nachbedingung Erfolg

1. Alle Knoten und Kanten des IML-Teilgraphen sind in das Anzeigefenster entsprechend dem gewählten Layout an der vorgegebenen Position eingefügt.
2. In dem Anzeigefenster gibt es eine neue aktuelle Selektion, die die neu eingefügten Knoten und Kanten umfasst.

4.14.3 Nachbedingung Fehlschlag

1. Hat der Benutzer den UseCase an irgendeinem Punkt abgebrochen, kehrt das System zu dem Zustand zurück, in dem es vor dem Start des UseCase war.

4.14.4 Beschreibung

1. Der Benutzer startet den UseCase über das Popup-Menü (siehe 14.3.5.2) im Hauptfenster „Insert IML Subgraph“. Hierdurch wird der einzufügende IML-Teilgraph bestimmt (immer der IML-Teilgraph, auf dem der Rechtsklick ausgeführt wurde).
2. GIANT zeigt in der Statuszeile im Hauptfenster „Select Position In Display Window For Insertion Of IML Subgraph“ an. Der Benutzer wählt das entsprechende Anzeigefenster aus und gibt über das Fadenkreuz (siehe 14.14) die Position vor, an der die neuen Fenster-Knoten und Fenster-Kanten eingefügt werden sollen. Die Statuszeile im Hauptfenster schaltet auf Normalmodus zurück.
3. GIANT zeigt den Dialog zur Auswahl von Layoutalgorithmen (siehe 14.9).
4. Der Benutzer wählt einen der vorgegebenen Layoutalgorithmen aus. Bei semantischen Layouts gibt er über den Layoutalgorithmen Dialog (siehe 14.9) auch die Kantenklassen vor, die für das Layout berücksichtigt werden sollen (siehe Kapitel 11 für Details zu Layoutalgorithmen).
5. Der Benutzer bestätigt mit OK.
6. GIANT berechnet das entsprechende Layout und zeigt einen Dialog an, der den Benutzer über den Fortschritt der Berechnung informiert (siehe 14.15.2).

Während der Berechnung des Layouts kann das System GIANT nicht bedient werden. Zugänglich ist nur der Button zum Abbruch des Algorithmus (siehe 14.15.2).

7. Nach Abschluss der Berechnung fügt GIANT die Fenster-Knoten und Fenster-Kanten in das entsprechende Anzeigefenster ein.

4.14.5 Alternativen

- Bei Punkt 4:
  Der Benutzer bricht den UseCase mit Cancel ab.
- Bei Punkt 6:
  Der Benutzer bricht die Berechnung des Layouts ab.

4.15 UC: Selektion in Anzeigefenster einfügen

Mit diesem UseCase kann eine Selektion aus einem Quell-Anzeigefenster in ein Ziel-Anzeigefenster unter Beibehaltung des Layouts kopiert werden (siehe 3.5 und 3.5.3).

4.15.1 Vorbedingung

1. Ein Projekt mit mindestens zwei geöffneten Anzeigefenstern ist geladen.
2. Es gibt mindestens eine Selektion.

4.15.2 Nachbedingung Erfolg

1. Das Position von Fenster-Knoten, die vor dem Einfügen bereits im Ziel-Anzeigefenster vorhanden waren, bleibt je nach Wahl des Benutzers unverändert oder wird ebenfalls geändert.
2. Die kopierte Selektion existiert auch im Ziel-Anzeigefenster als Ziel-Selektion und trägt dort ebenfalls den Namen der Selektion.
3. Im Ziel-Anzeigefenster gibt es keine Knoten mit der selben ID mehrfach.

4.15.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.15.4 Beschreibung

1. Der Benutzer startet den UseCase über das Popup-Menü (siehe 14.4.4) in der Selektionsauswahlliste. Durch Auswahl eines der beiden Menüeinträge „Copy SelEction Keep Existing Layout“ oder „Copy Selection Change Existing Layout“. Hierdurch wird automatisch die Selektion bestimmt (immer die Selektion, auf der der Rechtsklick ausgeführt wurde).
2. Der Benutzer wählt das entsprechende Ziel-Anzeigefenster aus. Das Ziel-Anzeigefenster darf nicht das Quell-Anzeigefenster sein, sonst wird eine Fehlermeldung ausgegeben. GIANT zeigt in der Statuszeile im Hauptfenster „Select Position in Display Window for Insertion of copied IML Subgraph“. Der Benutzer gibt über das Fadenkreuz die Position vor, an der die neuen Fenster-Knoten und Fenster-Kanten eingefügt werden sollen (siehe auch 14.14).
3. GIANT kopiert die Knoten und Kanten der Selektion in das Ziel-Anzeigefenster. Je nachdem, welchen Eintrag der Benutzer im Popup-Menü ausgewählt hat, geschieht mit den bereits im Ziel-Anzeigefenster vorhandenen Fenster-Knoten der Selektion folgendes:
   a) Falls „Copy Selection Keeping Existing Layout“ gewählt wurde, wird ihre Position im Ziel-Anzeigefenster nicht verändert.
   b) Falls „Copy Selection Changing Existing Layout“ gewählt wurde, wird ihre Position im Zielanzeigefenster gemäß dem Layout der Selektion verändert.

4.15.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.16 UC: Fenster-Knoten und Fenster-Kanten einer Selektion aus Anzeigefenster löschen

Mittels dieses UseCases können alle Fenster-Knoten und Fenster-Kanten einer Selektion aus einem Anzeigefenster gelöscht werden (siehe auch 3.7).

4.16.1 Vorbedingung

1. Ein Projekt mit mindestens einem geöffneten Anzeigefenster ist geladen.
2. Es gibt eine Selektion.

4.16.2 Nachbedingung Erfolg

1. Die Selektion ist aus dem Anzeigefenster gelöscht.
2. Alle betroffenen Fenster-Knoten und Fenster-Kanten sind gemäß der unter 3.7 beschriebenen Konvention aus dem Anzeigefenster entfernt.
3. Alle anderen Selektionen des Anzeigefensters wurden aktualisiert.

4.16.3 Nachbedingung Fehlschlag

1. Hat der Benutzer den UseCase abgebrochen, kehrt das System zu dem Zustand zurück, in dem es vor dem Start des UseCase war.

4.16.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf die entsprechende Selektion in der Selektionsauswahlliste durch (siehe 14.4.4) und wählt im entsprechenden Popup-Menü „Delete Selection“ aus.
2. GIANT zeigt die Sicherheitsabfrage (siehe 14.12) und fragt nach, ob es die Selektion samt ihrer Knoten und Kanten wirklich löschen soll („Really delete Selection from its window including Nodes and Edges?“).
3. Der Benutzer bestätigt mit Yes.
4. GIANT löscht die Selektion samt allen zugehörigen Fenster-Knoten und Fenster-Kanten aus dem entsprechenden Anzeigefenster.

4.16.5 Alternativen

- Bei Punkt 2:
  Der Benutzer bricht den UseCase mit No ab.

4.17 UC: Den Visualisierungsstil eines Anzeigefensters ändern

Mittels dieses UseCase kann der Benutzer die Visualisierung von Fenster-Knoten und Fenster-Kanten innerhalb eines Anzeigefensters dynamisch durch Auswahl verschiedener frei definierbarer Visualisierungsstile ändern (siehe auch 13.3).

4.17.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster.
2. Es gibt mindestens einen definierten Visualisierungsstil.

4.17.2 Nachbedingung Erfolg

1. Die Darstellung der Fenster-Knoten und Fenster-Kanten in dem Anzeigefenster entspricht den Vorgaben des gewählten Visualisierungsstils.
2. Alle anderen Zustände und Eigenschaften des Anzeigefensters, wie z.B. die hervorgehobenen Selektionen, bleiben unverändert.

4.17.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.17.4 Beschreibung

1. Der Benutzer ändert den Visualisierungsstil eines Anzeigefensters dadurch, dass er in der Stilauswahl-Combobox des Anzeigefensters (siehe 14.4) einen anderen Visualisierungsstil einstellt.
2. GIANT ändert die Darstellung von Fenster-Knoten und Fenster-Kanten entsprechend ab (näheres zur Visualisierung von Fenster-Knoten und Fenster-Kanten ist in Kapitel 15 spezifiziert).

4.17.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.18 UC: Anzeigefenster scrollen

Verändert die Position des sichtbaren Anzeigeinhaltes.

4.18.1 Vorbedingung

1. Ein Projekt mit mindestens einem geöffneten Anzeigefenster ist geladen.

4.18.2 Nachbedingung Erfolg

1. Die Position des sichtbaren Anzeigeinhalts wurde entsprechend abgeändert.

4.18.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.18.4 Beschreibung

1. a) Der Benutzer scrollt den sichtbaren Anzeigefokus mittels der horizontalen oder vertikalen Bildlaufleisten des Anzeigefensters (siehe 14.4.7). Dies geschieht mittels der Maus gemäß der Konventionen von GTK/Ada für Bildlaufleisten.

   b) Es kann auch mittels der Cursortasten gescrollt werden.
   Das Drücken der linken Cursortaste führt z.B. dazu, dass der sichtbare Anzeigeinhalt des aktiven Anzeigefensters nach links verschoben wird.

4.18.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.19 UC: Anzeigefenster zoomen

Verändert den Maßstab der Darstellung von Knoten und Kanten.

4.19.1 Vorbedingung

1. Ein Projekt mit mindestens einem geöffneten Anzeigefenster ist geladen.

4.19.2 Nachbedingung Erfolg

1. Der angezeigte Bereich des sichtbaren Anzeigeinhalts wurde entsprechend vergrößert oder verkleinert. Die Detailstufe (siehe 15.4) wurde ggf. automatisch angepasst.

4.19.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.19.4 Beschreibung

1. Der Benutzer gibt in der Zoom-Kontrolle-Combobox (siehe 14.4.6) des Anzeigefensters einen neuen Zoomwert ein, wählt darin einen der vorgefertigten Werte aus oder ändert den Zoomwert in festgelegten Schritten mit den „+“ oder „-“ Buttons.
2. GIANT berechnet den neuen sichtbaren Anzeigeinhalt anhand der neuen Zoomstufe.

4.19.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.20 UC: Zoomen auf eine gesamte Selektion

Wählt die passende Zoomstufe und scrollt den sichtbaren Anzeigeinhalt so, dass eine Selektion im Anzeigefenster vollständig sichtbar ist.

4.20.1 Vorbedingung

1. Ein Projekt mit mindestens einem geöffneten Anzeigefenster ist geladen.

4.20.2 Nachbedingung Erfolg

1. Der sichtbare Anzeigeinhalts wurde mittels Zoomen und Scrollen so verändert, dass die ausgewählte Selektion vollständig sichtbar ist. Die Detailstufe (siehe 15.4) wurde ggf. automatisch angepasst.

4.20.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.20.4 Beschreibung

1. Der Benutzer klickt auf „Zoom To Make Selection Fill Window“ im Popup-Menü der Selektionsauswahlliste (siehe 14.4.4).
2. GIANT scrollt und zoomt automatisch so, dass die gesamte Selektion im Anzeigefenster sichtbar wird.

4.20.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.21 UC: Zoomen auf gesamten Inhalt eines Anzeigefensters

Wählt die passende Zoomstufe und scrollt den sichtbaren Anzeigeinhalt so, dass der Anzeigeinhalt vollständig im sichtbaren Anzeigeinhalt dargestellt wird.

4.21.1 Vorbedingung

1. Ein Projekt mit mindestens einem geöffneten Anzeigefenster ist geladen.

4.21.2 Nachbedingung Erfolg

1. Der sichtbare Anzeigeinhalts wurde mittels Zoomen und Scrollen so verändert, dass der Anzeigeinhalt vollständig sichtbar ist. Die Detailstufe (siehe 15.4) wurde ggf. automatisch angepasst.

4.21.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.21.4 Beschreibung

1. Der Benutzer klickt auf „Fit in window“ in der Zoomkontrolle des Fensters (siehe 14.4.6).
2. GIANT berechnet die neue Zoomstufe für das Anzeigefenster.

4.21.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.22 UC: Zoomen auf eine Kante

Wählt die passende Zoomstufe und scrollt den sichtbaren Anzeigeinhalt so, dass eine Kante mit ihren Start– und Ziel-Fenster-Knoten komplett im sichtbaren Anzeigeinhalt liegt.

4.22.1 Vorbedingung

1. Ein Projekt mit mindestens einem geöffneten Anzeigefenster mit mindestens einer Kante ist geladen.

4.22.2 Nachbedingung Erfolg

1. Der sichtbare Anzeigeinhalts wurde mittels Zoomen und Scrollen so verändert, dass die ausgewählte Kante und ihre Start- und Ziel-Fenster-Knoten vollständig sichtbar ist. Die Detailstufe (siehe 15.4) wurde ggf. automatisch angepasst.

4.22.3 Nachbedingung Fehlschlag

1. Hat der Benutzer den UseCase abgebrochen, kehrt das System zu dem Zustand zurück, in dem es vor dem Start des UseCase war.

4.22.4 Beschreibung

1. Der Benutzer wählt den Button „Pick Edge“ in der Zoomkontrolle (siehe 14.4.6) aus.
2. Daraufhin erscheint in der Statuszeile von GIANT im Hauptfenster der Text „Select edge to be zoomed onto“ und der Fadenkreuz-Cursor (siehe 14.14) wird angezeigt, wenn der Mauscursor über den sichtbaren Anzeigeinhalt eines Anzeigefensters bewegt wird.
3. Der Benutzer klickt mit der linken Maustaste auf die gewünschte Kante im gewünschten Anzeigefenster.
4. GIANT berechnet für das gewünschte Anzeigefenster die Zoomstufe so, dass die gesamte Fenster-Kante im Anzeigeinhalt sichtbar dargestellt wird.

4.22.5 Alternativen

- Bei Punkt 3:
  Der Benutzer kann den UseCase abbrechen, indem er auf eine einen Rechtsklick mit der Maus ausführt.

4.23 UC: Verschieben von Fenster-Knoten und Selektionen mittels Cut and Paste

Mit diesem UseCase können Fenster-Knoten und Selektionen auf dem Anzeigeinhalt verschoben werden. Dieses Verschieben geschieht mittels „Cut and Paste“.

4.23.1 Vorbedingung

1. Ein Projekt mit mindestens einem geöffneten Anzeigefenster und mit mindestens einem Fenster-Knoten oder mindestens einer Selektion ist geladen.

4.23.2 Nachbedingung Erfolg

1. Der einzelne Fensterknoten oder alle Fenster-Knoten der Selektion wurden im Anzeigeinhalt verschoben.

4.23.3 Nachbedingung Fehlschlag

1. Hat der Benutzer den UseCase abgebrochen, so werden keine Fenster-Knoten verschoben.

4.23.4 Beschreibung

1. Der Benutzer wählt die zu verschiebende Selektion oder den zu verschiebenden Fenster-Knoten aus („Cut“), indem er:
   a) Einen Rechtsklick auf eine Selektion in der Selektionsauswahlliste durchführt (siehe 14.4.4) und im Popup-Menü „Move Selection“ auswählt,
   b) oder einen Rechtsklick auf einen Fenster-Knoten durchführt und im Popup Menü (siehe 14.4.2.1) „Move Node“ auswählt.
2. GIANT geht in den den „Paste Modus“ über und zeigt dies in der Statusleiste (siehe 14.3.3) des Hauptfensters an. Der Cursor wird, falls er über den sichtbaren Anzeigeinhalt eines Anzeigefensters bewegt wird, zum Fadenkreuz (siehe 14.14). Die Funktionalität zum Zoomen und Scrollen des Anzeigefensters mittels der beiden UseCases 4.18 und 4.19 bleibt weiterhin verfügbar, die übrige Funktionalität von GIANT wird gesperrt.
3. Der Benutzer klickt mit der linken Maustaste an eine beliebige Stelle innerhalb des sichtbaren Anzeigeinhaltes des Anzeigefensters.
4. GIANT verschiebt die ausgewählten Fenster-Knoten an die gewünschte Stelle.

4.23.5 Alternativen

- Bei Punkt 3:
  Der Benutzer kann den UseCase abbrechen, indem er einen Rechtsklick mit der Maus im sichtbaren Anzeigeinhalt durchführt.

4.24 UC: Verschieben einzelner Fenster-Knoten mittels Drag and Drop

Mit diesem UseCase können einzelne Fenster-Knoten mittels Drag and Drop auf dem sichtbaren Anzeigeinhalt verschoben werden.

4.24.1 Vorbedingung

1. Ein Projekt mit mindestens einem geöffneten Anzeigefenster und mit mindestens einem Fenster-Knoten ist geladen.

4.24.2 Nachbedingung Erfolg

1. Der einzelne Fensterknoten wurde auf dem Anzeigeinhalt entsprechend verschoben.

4.24.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.24.4 Beschreibung

1. Der Benutzer bewegt den Mauscursor über den zu verschiebenden Fenster-Knoten und drückt die linke Maustaste („Drag“), dann bewegt er den Fenster-Knoten an eine beliebige andere Stelle innerhalb des Anzeigeinhaltes und lässt die linke Maustaste los („Drop“).

4.24.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.25 UC: Platz schaffen

Dieser UseCase wird benötigt, um Fenster-Knoten auseinander schieben zu können. So kann der Benutzer an einer beliebigen Stelle des Anzeigefensters genügend Platz zum Einfügen neuer Fenster-Knoten und Fenster-Kanten schaffen (siehe auch 3.8).

4.25.1 Vorbedingung

1. Ein Projekt mit mindestens einem geöffneten Anzeigefenster ist geladen.

4.25.2 Nachbedingung Erfolg

1. Alle Fenster-Knoten und Fenster-Kanten des Anzeigefensters sind um den entsprechenden Betrag vom vorgegebenen Punkt innerhalb des Anzeigeinhaltes weggeschoben worden. An der entsprechenden Stelle im Anzeigeinhalt ist eine freie Fläche ohne Fenster-Knoten geschaffen worden. Diese Fläche kann aber von Fenster-Kanten gekreuzt werden.
2. Das Layout aller Fenster-Knoten des Anzeigeinhaltes bleibt ansonsten weitgehend unverändert.

4.25.3 Nachbedingung Fehlschlag

1. Hat der Benutzer den UseCase abgebrochen, so wird der Anzeigeinhalt nicht verändert.

4.25.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf eine beliebige Stelle des sichtbaren Anzeigeinhaltes durch und wählt im daraufhin erscheinenden Popup-Menü (siehe 14.4.2.1) dem Eintrag „Make Room“ aus.
2. GIANT zeigt in der Statuszeile im Hauptfenster „Select position in display window“. Der Benutzer gibt den Punkt um den herum die Fenster-Knoten (und damit automatisch auch die Fenster-Kanten) auseinander geschoben werden sollen über das Fadenkreuz vor (siehe 14.14). Hierbei kann er mit dem Fadenkreuz den aktuellen sichtbaren Anzeigeinhalt auf dem Anzeigefenster nicht verlassen.
3. GIANT zeigt einen Dialog an, in dem der Benutzer auswählt, um welchen Betrag die Fenster-Knoten auseinander geschoben werden sollen (siehe 14.10.2).
4. Der Benutzer wählt einen geeigneten Betrag aus und bestätigt mit OK.
5. GIANT schiebt die Knoten entsprechend auseinander (siehe auch 3.8).

4.25.5 Alternativen

- Bei Punkt 2:
  Der Benutzer kann den UseCase abbrechen, indem er mit der Maus einen Rechtsklick auf eine beliebige Stelle durchführt.
- Bei Punkt 3:
  Der Benutzer kann den UseCase mit Cancel abbrechen.

4.26 UC: Pin anlegen

Mit diesem UseCase kann ein neuer Pin erzeugt werden (siehe auch 2.3.1).

4.26.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster.

4.26.2 Nachbedingung Erfolg

1. In der Liste über die Pins des Anzeigefensters (siehe 14.4.3) befindet sich ein neuer Pin mit dem vom Benutzer definierten Namen.

4.26.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

4.26.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf den Anzeigeinhalt eines Anzeigefensters durch und wählt aus dem Popup-Menü (siehe 14.4.2.1) „New Pin“ aus. Der später erstellte Pin verweist dann auf die Stelle im Anzeigeinhalt, auf die der Rechtklick durchgeführt wurde.
2. GIANT öffnet den allgemeinen Texteingabedialog (siehe 14.7).
3. Der Benutzer gibt dort einen zulässigen Namen für den neuen Pin ein und bestätigt mit OK (zulässige Namen für sind in Abschnitt 3.3.2 spezifiziert).
4. GIANT speichert die aktuelle Zoomstufe und die Position des sichtbaren Anzeigeinhaltes in einem neuen Pin.

4.26.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht die Verarbeitung mit Cancel ab.

4.27 UC: Pin anspringen

Stellt die im Pin gespeicherte Position des sichtbaren Anzeigeinhaltes wieder her (siehe auch 2.3.1).

4.27.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit mindestens einem Pin.

4.27.2 Nachbedingung Erfolg

1. Der sichtbare Anzeigefokus des Anzeigefensters ist auf die entsprechenden Koordinaten und die entsprechende Zoomstufe, wie sie im ausgewählten Pin hinterlegt waren, gesetzt.

4.27.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.27.4 Beschreibung

1. Das Anspringen des Pins kann über die folgenden beiden Möglichkeiten geschehen:
   a) Der Benutzer führt einen Doppelklick auf den entsprechenden Pin in der Pinliste (siehe 14.4.3) aus.
   b) Der Benutzer öffnet ein Popup-Menü durch Rechtsklick auf den anzuspringenden Pin in der Pinliste (siehe 14.4.3) und wählt „Focus Pin“ aus.
2. GIANT setzt den sichtbaren Anzeigeinhalt gemäß den im Pin gespeicherten Informationen.

4.27.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.28 UC: Pin löschen

Löscht einen Pin (siehe auch 2.3.1).

4.28.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit mindestens einem Pin.

4.28.2 Nachbedingung Erfolg

1. Der Pin ist gelöscht und nicht mehr in der Pinliste (siehe 14.4.3) des entsprechenden Anzeigefensters auswählbar.

4.28.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.28.4 Beschreibung

1. Der Benutzer öffnet das Popup-Menü der Pinliste (siehe 14.4.3) durch Rechtsklick auf den Pin und wählt „Delete Pin“ aus.
2. GIANT löscht den Pin.

4.28.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.29 UC: Layout auf Selektion anwenden

Mittels dieses UseCase können Selektionen innerhalb eines Anzeigefensters layoutet werden. Die verfügbaren Layoutalgorithmen sind im Kapitel 11 beschrieben.

4.29.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit einer Selektion.

4.29.2 Nachbedingung Erfolg

1. Die Position der Fenster-Knoten der Selektion wurde gemäß den Vorgaben des Layoutalgorithmus geändert.
2. Alle anderen Fenster-Knoten bleiben unverändert.

4.29.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.
2. Hat der Benutzer einen Layoutalgorithmus während der Berechnung des Layouts abgebrochen, so bleiben die Positionen aller Fenster-Knoten des Anzeigefensters unverändert.

4.29.4 Beschreibung

1. Der Benutzer wählt aus der Selektionsauswahlliste (siehe 14.4.4) die Selektion aus, deren Fenster-Knoten layoutet werden sollen. Hierzu führt er einen Rechtsklick auf die Selektion durch und wählt im Popup-Menü den Eintrag „Layout Selection“ aus.
2. GIANT zeigt einen Dialog zur Auswahl und Konfiguration des gewünschten Layoutalgorithmus (siehe 14.9).
3. Der Benutzer wählt in diesem Dialog den gewünschten Layoutalgorithmus aus.
4. Falls der gewählte Layoutalgorithmus dies unterstützt, kann der Benutzer in diesem Dialog die Typen von IML-Kanten, die beim Layout berücksichtigt werden sollen, vorgeben.
5. Der Benutzer bestätigt seine Eingaben mit OK.
6. GIANT berechnet das Layout und zeigt einen Progress-Dialog (siehe 14.15.2) an. Während der Berechnung des Layouts ist die GUI mit Ausnahme des Progress-Dialogs gesperrt.

4.29.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht die Verarbeitung mit Cancel ab.

- Bei Punkt 4:
  Der Benutzer bricht die Verarbeitung mit Cancel ab.
- Bei Punkt 6:
  Der Benutzer kann die Berechnung eines Layouts jederzeit mit Cancel (siehe 14.15.2) abbrechen.

4.30 UC: Details zu Knoten in einem neuen Informationsfenster anzeigen

Dieser UseCase ermöglicht es dem Kunden, sich alle verfügbaren Informationen zu einem Fenster-Knoten (Kanten und Attribute des zugehörigen IML-Knoten) anzeigen zu lassen. Mit diesem UseCase kann der Benutzer beliebig viele Knoten-Informationsfenster (siehe 14.5) öffnen.

4.30.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit einem Fenster-Knoten.

4.30.2 Nachbedingung Erfolg

1. Die Daten über den Fenster-Knoten werden in einem neuen Knoten-Informationsfenster angezeigt.

4.30.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

4.30.4 Beschreibung

1. Der Benutzer wählt im entsprechenden Anzeigefenster den gewünschten Fenster-Knoten aus und führt einen Linksklick auf diesen Fenster-Knoten durch. Im darauf angezeigten Popup-Menü (siehe 14.4.2.1) wählt der Benutzer dann den Eintrag „Show Node Info Window“ aus.
2. GIANT öffnet ein neues Knoten-Informationsfenster (siehe 14.5) und zeigt alle Verfügbaren Informationen zu dem IML-Knoten an.

4.30.5 Alternativen

- Bei Punkt 1:
  Der Benutzer bricht die Verarbeitung mit einem Rechtsklick ab.

4.31 UC: Details zu Knoten in einem bestehenden Informationsfenster anzeigen

Dieser UseCase ermöglicht es dem Kunden, sich alle verfügbaren Informationen zu einem Fenster-Knoten (Kanten und Attribute des zugehörigen IML-Knoten) innerhalb eines bereits geöffneten Knoten-Informationsfensters (siehe 14.5) anzeigen zu lassen.

4.31.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit einem Fenster-Knoten.
2. Es gibt ein offenes Knoten-Informationsfenster.

4.31.2 Nachbedingung Erfolg

1. Anstatt der Daten, die zuvor in dem Knoten-Informationsfenster angezeigt wurden, werden dort nun die Informationen zu dem gewählten Fenster-Knoten angezeigt.

4.31.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.31.4 Beschreibung

1. Der Benutzer betätigt in einem geöffneten Knoten-Informationsfenster den Button „Pick“ (siehe 14.5).
2. Daraufhin fordert GIANT den Benutzer auf einen Fenster-Knoten auszuwählen.
3. Der Benutzer wählt den gewünschten Fensterknoten dadurch aus, dass er innerhalb des Anzeigeinhaltes eines Anzeigefensters mittels eines Fadenkreuzes den Fenster-Knoten als Zielposition vorgibt. Das Verfahren zur Vorgabe der Zielposition ist im Detail unter 3.5.1 beschrieben.
4. GIANT stellt die verfügbaren Informationen zu dem ausgewählten Fenster-Knoten im Knoten-Informationsfenster dar.

4.31.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.32 UC: Anzeige des Quellcodes eines Knotens in externem Editor

Mittels dieses UseCases kann der zu einem IML-Knoten korrespondierende Quellcode innerhalb eines Editors zur Anzeige gebracht werden.

4.32.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit einem Fenster-Knoten.

4.32.2 Nachbedingung Erfolg

1. Der zu dem IML-Knoten korrespondierende Quellcode wird in dem über die Konfigurationsdatei vorgegebenen Editor angezeigt (siehe auch 13.2.3). Der Cursor des Editors ist auf die dem IML-Knoten entsprechende Position innerhalb des Quelltextes gesetzt, also z.B. auf einen Bezeichnernamen, den der IML-Knoten repräsentiert.

4.32.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.32.4 Beschreibung

1. Der Benutzer wählt in einem Anzeigefenster den gewünschten Fenster-Knoten aus und führt einen Rechtsklick auf diesen Knoten durch. Im darauf angezeigten Popup-Menü (siehe 14.4.2.1) wählt der Benutzer dann den Eintrag „Show Corresponding Source Code“ aus.
2. GIANT öffnet die Quellcode Datei für den IML-Knoten in einem Editor.

4.32.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

4.33 UC: Verschieben des sichtbaren Anzeigeinhaltes über die Minimap

Der Benutzer kann mittels der Minimap (siehe auch 14.4.1) den sichtbaren Anzeigeinhalt scrollen.

4.33.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster.

4.33.2 Nachbedingung Erfolg

1. Der sichtbare Anzeigeinhalt ist auf den Bereich des Anzeigefensters gesetzt, der dem Punkt welcher auf der Minimap angeklickt wurde, entspricht.

4.33.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

4.33.4 Beschreibung

1. Der Benutzer klickt mit der linken Maustaste auf einen Punkt auf der Minimap.
2. GIANT zentriert den sichtbaren Anzeigeinhalt des Anzeigefensters auf den vorgegebenen Punkt.

4.33.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

5 Knoten-Annotationen

Die folgenden UseCases beschreiben die Funktionalität zum Erzeugen, Ändern und Löschen von Knoten-Annotationen.

5.1 UC: Fenster-Knoten annotieren

Dieser UseCase dient zum Erzeugen von Knoten-Annotationen (siehe auch 12.2.4).

5.1.1 Vorbedingung

1. Es gibt mindestens ein Anzeigefenster mit Fenster-Knoten.

5.1.2 Nachbedingung Erfolg

1. Die neue Annotation ist noch nicht in der Verwaltungsdatei für Knoten-Annotationen (siehe 12.1.6) eingetragen. Dies geschieht erst beim nächsten Speichern des gesamten Projektes.
2. Der annotierte Knoten wird in allen Anzeigefenstern mittels eines Icons als annotiert gekennzeichnet (siehe 15.1.3).

5.1.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

5.1.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf den Knoten, der annotiert werden soll, durch und wählt im Popup Menü für Knoten den Eintrag „Annotate Node“ aus (siehe 14.4.2.1).
2. GIANT zeigt nun den Knoten Annotations-Dialog (siehe 14.10.1).
3. Der Benutzer gibt dort den gewünschten Text für die Knoten-Annotation ein.
4. Nach Abschluss der Eingabe wählt der Benutzer OK aus.
5. GIANT übernimmt die neue Annotation. Die eingegebene Annotation muss allerdings mindestens ein Zeichen haben, sonst erscheint eine Fehlermeldung.

5.1.5 Alternativen

- Bei Punkt 3:
  Der Benutzer kann die Eingabe der neuen Knoten-Annotation jeder Zeit mit Cancel abbrechen.

5.2 UC: Knoten-Annotation ändern

Dient zum Ändern bestehender Knoten-Annotationen (siehe auch 12.2.4). Dieser UseCase ist auch für das Anzeigen von Knoten-Annotationen vorgesehen.

5.2.1 Vorbedingung

1. Es gibt mindestens einen annotierten Fenster-Knoten (siehe auch 15.1.3).

5.2.2 Nachbedingung Erfolg

1. Die Änderung an der Annotation ist noch nicht in der Verwaltungsdatei für Knoten-Annotationen eingetragen (siehe 12.1.6). Dies geschieht erst beim nächsten Speichern des gesamten Projektes.
2. Die Änderung der Annotation ist dem System bekannt und wird bei der nächsten Ausführung dieses UseCases angezeigt.

5.2.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

5.2.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf den Knoten, dessen Annotation geändert werden soll, durch und wählt im Popup Menü für Knoten den Eintrag „Change Node Annotation“ aus (siehe 14.4.2.1).
2. GIANT zeigt nun den Knoten Annotations-Dialog (siehe 14.10.1). Hier wird der Text für die bisherige Annotation dargestellt.
3. Der Benutzer ändert den Text für die Annotation entsprechend ab. Der Benutzer wird aber nicht gezwungen die bestehende Knoten-Annotation zu ändern.
4. Nach Abschluss der Eingabe wählt der Benutzer den OK aus.
5. GIANT übernimmt die vorgenommenen Änderungen an der Annotation. Die neue Annotation muss allerdings mindestens ein Zeichen haben, ansonsten erscheint eine Fehlermeldung.

5.2.5 Alternativen

- Generell:
  Der Benutzer kann das Ändern der Annotation jeder Zeit mit Cancel abbrechen.

5.3 UC: Knoten-Annotation löschen

Löscht eine bestehende Knoten-Annotationen (siehe auch 12.2.4).

5.3.1 Vorbedingung

1. Es gibt mindestens einen annotierten Fenster-Knoten (siehe 15.1.3).

5.3.2 Nachbedingung Erfolg

1. Die Annotation ist noch nicht aus der Verwaltungsdatei für Knoten-Annotationen entfernt. Dies geschieht erst beim nächsten Speichern des gesamten Projektes.
2. Das Entfernen der Annotation ist dem System bekannt und wird entsprechend angezeigt.

5.3.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

5.3.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf den Fenster-Knoten, dessen Annotation gelöscht werden soll, durch und wählt im Popup Menü für Knoten den Eintrag „Delete Node Annotation“ aus (siehe 14.4.2.1).
2. GIANT löscht die entsprechende Annotation.

5.3.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

5.4 UC: Unreferenzierte Knoten-Annotationen löschen

Dieser UseCase soll es ermöglichen, nicht mehr benötigte Annotationen automatisch entfernen zu lassen. Es werden alle bestehenden Knoten-Annotationen des Projektes für die es in Anzeigefenstern und IML-Teilgraphen keine Fenster-Knoten und Graph-Knoten gibt gelöscht.

5.4.1 Vorbedingung

1. Ein Projekt ist geladen.

5.4.2 Nachbedingung Erfolg

1. Die Annotationen sind noch nicht aus der Verwaltungsdatei für Knoten-Annotationen (siehe 12.1.6) entfernt. Dies geschieht erst beim nächsten Speichern des Projekts.

5.4.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

5.4.4 Beschreibung

1. Der Benutzer wählt im Hauptmenü den Eintag „Delete Annotations Having No Visible Node“ (siehe 14.3.2.2) aus.
2. GIANT zeigt eine Sicherheitsabfrage und fordert den Benutzer zur Bestätigung auf (siehe 14.12).
3. Der Benutzer bestätigt die Sicherheitsabfrage.
4. GIANT löscht alle Annotationen für die es keine Fenster-Knoten und Graph-Knoten gibt, d.h. jede Knoten-Annotation deren IML-Knoten weder in einem Anzeigefenster als Fenster-Knoten visualisiert noch als Graph-Knoten Bestandteil eines IML-Teilgraphen ist, wird entfernt.

5.4.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht mit Cancel ab.

6 Selektionen

Alle UseCases, die im Zusammenhang mit Selektionen stehen werden hier beschrieben.

6.1 UC: Selektion zur aktuellen Selektion machen

Dieser UseCase dient dazu, eine Selektion zur aktuellen Selektion zu machen. Dies ist nötig, da nur die aktuelle Selektion mittels der Maus modifiziert werden kann (siehe auch 3.4.3).

6.1.1 Vorbedingung

1. Es gibt ein Anzeigefenster mit mindestens einer Selektion.

6.1.2 Nachbedingung Erfolg

1. Die vorherige aktuelle Selektion ist nicht mehr aktuell.
2. Die entsprechende Selektion ist nun die aktuelle Selektion und als solche erkennbar angezeigt und hervorgehoben, auch mittels einer Hervorhebung in der Selektionsauswahlliste des Fensters.

6.1.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

6.1.4 Beschreibung

1. Der Benutzer führt mit der linken Maustaste einen Doppelklick auf eine nicht aktuelle Selektion in der Selektionsauswahlliste (siehe 14.4.4) durch.
2. GIANT macht die entsprechende Selektion zur aktuellen Selektion.

6.1.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

6.2 UC: Selektion graphisch hervorheben

Dieser UseCase dient zum Hervorheben von Selektionen innerhalb eines Anzeigefensters (siehe auch 15.3).

6.2.1 Vorbedingung

1. Es gibt ein Anzeigefenster mit mindestens einer Selektion.

6.2.2 Nachbedingung Erfolg

1. Die Selektion ist im entsprechenden Anzeigefenster hervorgehoben.
2. Die Selektion, welche vorher mit der gleichen Farbe hervorgehoben war, ist nicht mehr hervorgehoben.

6.2.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

6.2.4 Beschreibung

1. Der Benutzer startet den UseCase mit einem Rechtsklick auf die gewünschte Selektion in der Selektionsauswahlliste (siehe 14.4.4) und wählt im zugehörigen Popup-Menü das Untermenü „Highlight Selection“ und dort die gewünschte Farbe aus.
2. GIANT hebt die Selektion mit der ausgewählten Farbe hervor.

6.2.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

6.3 UC: Graphische Hervorhebung einer Selektion aufheben

Dieser UseCase dient dazu, die Hervorhebung von Selektionen innerhalb eines Anzeigefensters aufzuheben.

6.3.1 Vorbedingung

1. Es gibt ein Anzeigefenster mit mindestens einer hervorgehobenen Selektion.

6.3.2 Nachbedingung Erfolg

1. Die Selektion ist im entsprechenden Anzeigefenster nicht mehr hervorgehoben.

6.3.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

6.3.4 Beschreibung

1. Der Benutzer startet den UseCase mit einem Rechtsklick auf die gewünschte Selektion in der Selektionsauswahlliste (siehe 14.4.4) und wählt im zugehörigen Popup-Menü den Eintrag „Unhighlight Selection“ aus.
2. GIANT setzt die Hervorhebung der Selektion zurück.

6.3.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

6.4 UC: Neue Selektion anlegen

Mit diesem UseCase können neue, leere Selektionen angelegt werden.

6.4.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster.

6.4.2 Nachbedingung Erfolg

1. Eine neue Selektion mit entsprechendem Namen ist angelegt und erscheint in der Liste der Selektionen (siehe 14.4.4).
2. Diese neue Selektion hat keinen Inhalt (selektierte Knoten und Kanten).

6.4.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

6.4.4 Beschreibung

1. Der Benutzer startet den UseCase mit Rechtsklick auf die Selektionsauswahlliste (siehe 14.4.4) und wählt im Popup-Menü den Eintrag „New Selection“ aus.
2. GIANT öffnet den allgemeinen Texteingabedialog (siehe14.7).
3. Der Benutzer gibt dort einen zulässigen Namen für die neue Selektion ein und bestätigt mit OK (siehe 3.3.2).
   Hat bereits eine andere Selektion innerhalb des Anzeigefensters den selben Namen, erscheint eine Fehlermeldung.
4. GIANT erzeugt die neue Selektion.

6.4.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht die Verarbeitung mit Cancel ab.

6.5 UC: Selektion kopieren

Dieser UseCase dient zum Kopieren von Selektionen innerhalb eines Anzeigefensters.

6.5.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit einer Selektion.

6.5.2 Nachbedingung Erfolg

1. Eine neue Selektion mit entsprechendem Namen ist angelegt und erscheint in der Liste der Selektionen (siehe 14.4.4).
2. Die neue Selektion umfasst die gleichen Knoten und Kanten wie die Selektion, von der kopiert wurde.

6.5.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

6.5.4 Beschreibung

1. Der Benutzer startet den UseCase durch Rechtsklick auf die zu kopierende Selektion in der Selektionsauswahlliste (siehe 14.4.4) und wählt aus dem Popup-Menü den Eintrag „Copy Selection“.
2. GIANT öffnet den allgemeinen Texteingabedialog (siehe 14.7).
3. Der Benutzer gibt dort einen zulässigen Namen für die neue Selektion ein und bestätigt mit OK (siehe 3.3.2).
   Hat bereits eine andere Selektion innerhalb des Anzeigefensters den selben Namen erscheint eine Fehlermeldung.
4. GIANT kopiert die Quellselektion und legt eine neue Selektion an.

6.5.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht die Verarbeitung mit Cancel ab.

6.6 UC: Selektion löschen

Dieser UseCase dient zum Löschen von Selektionen innerhalb eines Anzeigefensters. Hierdurch bleiben die Fenster-Knoten und Fenster-Kanten unverändert.

6.6.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit einer Selektion.

6.6.2 Nachbedingung Erfolg

1. Die entsprechende Selektion ist gelöscht.
2. Die Fenster-Knoten und Fenster-Kanten, die zu dieser Selektion gehörten, werden nicht gelöscht.
3. War die Selektion hervorgehoben, so wird die entsprechende Hervorhebung der Fenster-Knoten und Fenster-Kanten aufgehoben.

6.6.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

6.6.4 Beschreibung

1. Der Benutzer startet den UseCase durch Rechtsklick auf die zu löschende Selektion in der Selektionsauswahlliste 14.4.4 und wählt aus dem Popup-Menü den Eintrag „Delete Selection“ aus. Falls der Benutzer die Standard-Selektion ausgewählt hat ist der Eintrag deaktiviert (siehe 3.4.2).
2. GIANT löscht die entsprechende Selektion.

6.6.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

6.7 UC: Selektionen manuell modifizieren

Jeweils die aktuelle Selektion kann mittels der Maus modifiziert werden (siehe auch 3.4.3).

6.7.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit mindestens einer Selektion.

6.7.2 Nachbedingung Erfolg

1. Die entsprechenden Änderungen an der Selektion werden von GIANT sofort durchgeführt und übernommen.

6.7.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

6.7.4 Beschreibung

1. Falls noch nicht der Fall, macht der Benutzer die zu modifizierende Selektion zur aktuellen Selektion (siehe 6.1).
2. Mittels der unter 3.4.1 beschriebenen Möglichkeiten fügt der Benutzer der Selektion neue Fenster-Knoten und Fenster-Kanten hinzu oder entfernt bestehende Fenster-Knoten und Fenster-Kanten aus der Selektion.

6.7.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

6.8 UC: Selektion aus IML-Teilgraph erzeugen

Leitet eine Selektion aus einem IML-Teilgraphen ab (siehe 3.6.1).

6.8.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster.
2. Es gibt mindestens einen IML-Teilgraphen.

6.8.2 Nachbedingung Erfolg

1. Im Ziel-Anzeigefenster wurde eine neue Selektion mit dem entsprechenden Namen erzeugt.

6.8.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

6.8.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf den Quell-IML-Teilgraphen in der entsprechenden Liste im Hauptfenster (siehe 14.3.5) aus und wählt im dazugehörigen Popup-Menü (siehe 14.3.5.2) den Eintrag „Create Window Selection from IML Subgraph“ aus.
2. GIANT zeigt den allgemeinen Texteingabedialog (siehe 14.7).
3. Der Benutzer gibt einen Namen (siehe auch 3.3.2) für die neu zu erstellende Selektion ein und bestätigt mit OK.
4. Die Statuszeile im Hauptfenster zeigt an „Please select window for Insertion of new window selection“, der Mauszeiger verwandelt sich in ein Fadenkreuz.
5. Der Nutzer klickt auf den sichtbaren Anzeigeinhalt des Anzeigefensters, in dem er die neue Selektion erstellen will (Ziel-Anzeigefenster).
6. Die Statuszeile im Hauptfenster wechselt wieder zur normalen Anzeige. GIANT erzeugt gemäß der unter Abschnitt 3.6.1 beschriebenen Konvention im Ziel-Anzeigefenster eine neue Selektion als Ableitung aus dem Quell-IML-Teilgraphen.

6.8.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht den UseCase mit Cancel ab.
- Bei Punkt 4:
  Der Benutzer bricht den UseCase durch einen Rechtsklick ab.

6.9 UC: Mengenoperationen auf 2 Selektionen

Zusätzlich zu den Möglichkeiten der Anfragesprache GQSL (siehe 10) kann der Benutzer die gängigen Mengenoperationen, wie Mengenvereinigung, Schnitt und Differenz, auch direkt über einen entsprechenden Dialog (siehe 14.8) ausführen. Das Vorgehen ist analog zu dem UseCase 8.6.

6.9.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit mindestens zwei Selektionen.

6.9.2 Nachbedingung Erfolg

1. Eine neue Selektion mit entsprechendem Namen (eingegeben unter TARGET) ist angelegt und erscheint in der Liste der Selektionen (siehe 14.4.4).
2. Im Falle einer Mengenvereinigung umfasst, die neue Selektion TARGET alle Knoten und Kanten aus der LEFT_SOURCE Selektion und der RIGHT_SOURCE Selektion.
3. Im Falle einer Mengendifferenz umfasst, die neue Selektion TARGET alle Knoten und Kanten aus der LEFT_SOURCE Selektion, die nicht Bestandteil der RIGHT_SOURCE Selektion sind.
4. Im Falle eines Mengenschnitts umfasst, die neue Selektion TARGET alle Knoten und Kanten, die der LEFT_SOURCE Selektion und der RIGHT_SOURCE Selektion gemeinsam angehören.

6.9.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

6.9.4 Beschreibung

1. Der Benutzer startet den UseCase über durch Auswahl des Menüpunktes Selection Set Operation (Union/Difference/Intersection) aus dem Popup-Menü in der Selektionsauswahlliste (siehe 14.4.4).
2. GIANT öffnet den Set-Operation-Dialog (siehe 14.8).
3. Der Benutzer wählt dort die beiden Quell-Selektionen (LEFT_SOURCE und RIGHT_SOURCE) aus, bestimmt die durchzuführende Mengenoperation und gibt unter TARGET den Namen der neue zu erzeugenden Selektion ein (gültige Namen siehe 3.3.2).
   Dann bestätigt er die Eingabe mit OK.
4. GIANT führt die Mengenoperation aus.

6.9.5 Alternativen

- Generell:
  Der Benutzer bricht die Eingabe der Daten mit Cancel ab.

7 Filter

Hier werden die Filter zum Aus- und Einblenden von Fenster-Knoten und Fenster-Kanten beschrieben,

7.1 UC: Selektionen ausblenden

Mit diesem UseCase können Selektionen innerhalb eines Anzeigefensters ausgeblendet werden.

7.1.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit einer Selektion.

7.1.2 Nachbedingung Erfolg

1. Alle zu der Selektion gehörende Fenster-Knoten und Fenster-Kanten sind ausgeblendet, d.h. sie sind im Anzeigefenster nicht mehr sichtbar. Dies trifft auch für Fenster-Knoten und Fenster-Kanten, die noch zu weiteren Selektionen, gehören zu.
2. Die ausgeblendeten Selektionen können nicht zur aktuellen Selektion gemacht werden (siehe 6.1) und damit nicht mehr direkt bearbeitet werden, d.h. die Menge der selektierten Fenster-Knoten und Fenster-Kanten kann nicht mehr abgeändert werden (siehe 3.4.1).
3. Die Fenster-Knoten und Fenster-Kanten sind aber immer noch Bestandteil des Anzeigefensters und können über den folgenden UseCase (siehe 7.2) wieder zur Anzeige gebracht werden.

7.1.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

7.1.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick mit der Maus auf die auszublendende Selektion in der Selektionsauswahlliste (siehe 14.4.4) durch und wählt im Popup-Menü den Eintrag „Hide Selection“ aus. Diese Funktionalität kann nicht auf die aktuelle Selektion angewendet werden (siehe 3.4.3).
2. GIANT blendet die Selektion aus.

7.1.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

7.2 UC: Selektionen einblenden

Mit diesem UseCase können ausgeblendete Selektionen wieder eingeblendet werden.

7.2.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster mit einer ausgeblendeten Selektion.

7.2.2 Nachbedingung Erfolg

1. Die Selektion ist wieder eingeblendet, alle zu ihr gehörenden Fenster-Knoten und Fenster-Kanten sind wieder im Anzeigeinhalt sichtbar dargestellt.

7.2.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

7.2.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick mit der Maus auf eine ausgeblendete Selektion in der Selektionsauswahlliste (siehe 14.4.4) durch und wählt im Popup-Menü den Eintrag „Show Selection“ aus.
2. GIANT blendet die Selektion ein.

7.2.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

8 Teilgraphen

Alle UseCases, die sich mit IML-Teilgraphen befassen, sind in diesem Abschnitt zusammen gefasst.

8.1 UC: IML-Teilgraph graphisch hervorheben

Dieser UseCase dient zum Hervorheben von IML-Teilgraphen innerhalb der Anzeigefenster (siehe auch 15.3).

8.1.1 Vorbedingung

1. Es gibt mindestens einen IML-Teilgraphen.

8.1.2 Nachbedingung Erfolg

1. Der IML-Teilgraph ist in jedem geöffneten Anzeigefenster entsprechend hervorgehoben.
2. Der IML-Teilgraph, welcher vorher mit der gleichen Farbe hervorgehoben war, ist nicht mehr hervorgehoben.

8.1.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

8.1.4 Beschreibung

1. Der Benutzer startet den UseCase über das Popup-Menü der Liste über die IML-Teilgraphen (siehe 14.3.5) durch Rechtsklick auf den gewünschten IML-Teilgraphen. In dem Popup Menü (siehe 14.3.5.2) wählt er das Untermenü „Highlight“ und dort die gewünschte Farbe aus.
2. GIANT hebt den IML-Teilgraphen in allen geöffneten Anzeigefenstern mit der ausgewählten Farbe hervor.

8.1.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

8.2 UC: Graphische Hervorhebung von IML-Teilgraphen aufheben

Mit diesem UseCase kann die graphische Hervorhebung von IML-Teilgraphen aufgehoben werden.

8.2.1 Vorbedingung

1. Es gibt einen hervorgehobenen IML-Teilgraphen.

8.2.2 Nachbedingung Erfolg

1. Der IML-Teilgraph ist nicht mehr hervorgehoben.

8.2.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

8.2.4 Beschreibung

1. Der Benutzer startet den UseCase durch Rechtsklick auf den hervorgehobenen IML-Teilgraphen in der Liste über die IML-Teilgraphen (siehe 14.3.5) In dem zugehörigen Popup-Menü (siehe 14.3.5.2) wählt er den Eintrag „Unhighlight In All Windows“ aus.
2. GIANT setzt die Hervorhebung des IML-Teilgraphen zurück.

8.2.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

8.3 UC: IML-Teilgraph aus einer Selektion erzeugen

Leitet einen neuen IML-Teilgraphen aus einer Quell-Selektion ab (siehe auch Abschnitt 3.6.2.

8.3.1 Vorbedingung

1. Es gibt ein geöffnetes Anzeigefenster.
2. Es gibt mindestens eine Selektion.

8.3.2 Nachbedingung Erfolg

1. Es wurde ein neuer IML-Teilgraph mit entsprechendem Namen erzeugt.

8.3.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

8.3.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf die Quell-Selektion in der Selektionsauswahlliste (siehe 14.4.4) aus und wählt im Popup-Menü den Eintrag „Create New IML Subgraph from This Selection“ aus.
2. GIANT zeigt den allgemeinen Texteingabedialog (siehe 14.7) an.
3. Der Benutzer gibt einen Namen für den neu zu erstellenden IML-Teilgraphen ein und bestätigt mit OK (gültige Namen für IML-Teilgraphen sind unter Abschnitt 3.3.2 spezifiziert).
   Gibt der Benutzer hier keinen Namen ein, so vergibt GIANT automatisch einen Namen.
4. GIANT erzeugt gemäß der unter Abschnitt 3.6.2 beschriebenen Konvention eine neuen IML-Teilgraphen aus der Quell-Selektion.

8.3.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht den UseCase mit Cancel ab.

8.4 UC: IML-Teilgraph kopieren

Kopiert einen Quell-IML-Teilgraphen in einen neuen IML-Teilgraphen. Bestehende IML-Teilgraphen können nicht überschrieben werden.

8.4.1 Vorbedingung

1. Es gibt mindestens einen IML-Teilgraphen.

8.4.2 Nachbedingung Erfolg

1. Es wurde ein neuer IML-Teilgraph mit entsprechendem Namen erzeugt.
2. Der neue IML-Teilgraph hat alle Graph-Knoten und Graph-Kanten des Quell-IML-Teilgraphen.

8.4.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

8.4.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf den zu kopierenden Quell-IML-Teilgraphen in der Liste über die IML-Teilgraphen im Hauptfenster aus (siehe 14.3.5) und wählt im zugehörigen Popup-Menü (siehe 14.3.5.2) den Eintrag „Copy IML Subgraph“ aus.
2. GIANT zeigt den allgemeinen Texteingabedialog (siehe 14.7).
3. Der Benutzer gibt einen Namen für den neu zu erstellenden IML-Teilgraphen ein und bestätigt mit OK (gültige Namen siehe 3.3.2).
   Gibt der Benutzer hier keinen Namen ein, so vergibt GIANT automatisch einen Namen.
   Gibt der Benutzer den Namen eines bereits vorhandenen IML-Teilgraphen ein, so erscheint eine Fehlermeldung.
4. GIANT kopiert den Quell-IML-Teilgraphen in einen neuen Teilgraphen.

8.4.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht den UseCase mit Cancel ab.

8.5 UC: IML-Teilgraph löschen

Dieser UseCase löscht einen IML-Teilgraphen. Die zugehörigen Fenster-Knoten und Fenster-Kanten in den Anzeigefenstern bleiben davon unberührt.

8.5.1 Vorbedingung

1. Es gibt mindestens einen IML-Teilgraphen.

8.5.2 Nachbedingung Erfolg

1. Der IML-Teilgraph wurde aus der Liste über die IML-Teilgraphen gelöscht (siehe 14.3.5).

8.5.3 Nachbedingung Fehlschlag

1. Es sind keine Fehlerfälle vorgesehen.

8.5.4 Beschreibung

1. Der Benutzer führt einen Rechtsklick auf den zu löschenden IML-Teilgraphen aus und wählt im Popup-Menü der Liste über die IML-Teilgraphen den Eintrag „Delete IML Subgraph“ aus (siehe 14.3.5.2).
2. GIANT löscht den gewählten IML-Teilgraphen.

8.5.5 Alternativen

- Bei Punkt **??**:
  s sind keine alternativen Abläufe vorgesehen.

8.6 UC: Mengenoperationen auf 2 IML-Teilgraphen

Ergänzend zu den Möglichkeiten der Anfragesprache (siehe Kapitel 10) kann der Benutzer die gängigen Mengenoperationen, wie Mengenvereinigung, Schnitt und Differenz, auch direkt über einen Dialog ausführen. Für eine genaue Beschreibung des Dialoges siehe auch Abschnitt 14.8. Das Vorgehen innerhalb dieses UseCases ist im Wesentlichen analog zu dem UseCase 6.9.

8.6.1 Vorbedingung

1. Es gibt mindestens zwei IML-Teilgraphen.

8.6.2 Nachbedingung Erfolg

1. Eine neuer IML-Teilgraph mit entsprechendem Namen (im Dialog eingegeben unter TARGET) ist angelegt und erscheint in der Liste über alle IML-Teilgraphen des Projektes (siehe 14.3.5).
2. Im Falle einer Mengenvereinigung umfasst der neue IML-Teilgraph TARGET alle Knoten und Kanten aus dem LEFT_SOURCE IML-Teilgraphen und dem RIGHT_SOURCE IML-Teilgraphen.
3. Im Falle einer Mengendifferenz umfasst der neue IML-Teilgraph TARGET alle Knoten und Kanten aus dem LEFT_SOURCE IML-Teilgraphen, die nicht Bestandteil des RIGHT_SOURCE IML-Teilgraphen sind.
4. Im Falle eines Mengenschnitts umfasst der neue IML-Teilgraph TARGET alle Knoten und Kanten, die sowohl dem LEFT_SOURCE IML-Teilgraphen als auch dem RIGHT_SOURCE IML-Teilgraphen angehören.

8.6.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.

8.6.4 Beschreibung

1. Der Benutzer startet den UseCase über das Popup-Menü der Liste über die IML-Teilgraphen (siehe 14.3.5.2) indem er einen Rechtsklick innerhalb der Liste ausführt und im Popup Menü den Eintrag „IML Subgraph Set Operation“ auswählt.
2. GIANT öffnet den Set-Operation-Dialog (siehe 14.8).
3. Der Benutzer wählt dort die beiden Quell-IML-Teilgraphen (LEFT_SOURCE und RIGHT_SOURCE) aus, bestimmt die auszuführende Mengenoperation und gibt unter TARGET den Namen des neu zu erzeugenden IML-Teilgraphen ein (gültige Namen für IML-Teilgraphen siehe 3.3.2).
   Er bestätigt mit OK.
   Existiert bereits ein IML-Teilgraph mit dem unter TARGET eingegebenen Namen, so erscheint eine Fehlermeldung.

4. GIANT führt die Mengenoperation aus und erzeugt den neuen IML-Teilgraphen.

8.6.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht die Eingabe der Daten mit Cancel ab.

9 Skripte

Hier werden UseCases zur Eingabe und Ausführung von GQSL Skripten beschrieben.

9.1 UC: Neues Skript ausführen

Mit diesem UseCase kann eine Skript über den Skriptdialog (siehe 14.6) eingegeben werden. Die Möglichkeiten der GQSL sind im Detail in Kapitel 10 beschrieben.

9.1.1 Vorbedingung

1. Ein Projekt ist geladen.

9.1.2 Nachbedingung Erfolg

1. Das Skript wurde ausgeführt. Alle Ergebnisse liegen vor.

9.1.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.
2. Wurde der UseCase während der Berechnung des abgebrochen, so gehen sämtliche bereits fertig gestellten Teilergebnisse verloren.

9.1.4 Beschreibung

1. Der Benutzer startet den UseCase durch Auswahl des Eintrags „Execute GQSL Script“ im Hauptmenü (siehe 14.3.2.2).
2. GIANT öffnet den Skriptdialog (siehe 14.6).
3. Der Benutzer gibt dort im dafür vorgesehenen Textfeld das GQSL Skript (siehe auch Kapitel 10) ein und bestätigt mit „Start Query“.
4. GIANT prüft das eingegebene GQSL Skript. Sollte das Skript nicht den Vorgaben der Grammatik (siehe Kapitel 10) entsprechen, erscheint eine Fehlermeldung (siehe 3.1) und das System kehrt zu Schritt 3 des UseCase zurück.

5. GIANT führt das Skript aus und teilt dem Benutzer den Fortschritt mittels eines Progress-Dialogs (siehe 14.15.2) mit. Während der Ausführung des Skripts ist das System mit Ausnahme des Progress-Dialogs gesperrt.

9.1.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht mit Cancel ab.
- Bei Punkt 5:
  Die laufende Ausführung des Skripts wird vom Benutzer mit Cancel abgebrochen werden.

9.2 UC: Skript laden

Der Benutzer kann zusätzlich zur manuellen Eingabe von GQSL Skripten (siehe 9.1) auch gespeicherte Skripte aus einer Datei (siehe 13.4) laden.

9.2.1 Vorbedingung

1. Ein Projekt ist geladen.

9.2.2 Nachbedingung Erfolg

1. Das Skript wurde ausgeführt. Alle Ergebnisse liegen vor.

9.2.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.
2. Wurde der UseCase während der Berechnung abgebrochen, so gehen sämtliche bereits fertig gestellten Teilergebnisse verloren.

9.2.4 Beschreibung

1. Der Benutzer startet den UseCase durch Auswahl des Eintrags „Execute GQSL Script“ im Hauptmenü (siehe 14.3.2.2).
2. GIANT öffnet den Skriptdialog (siehe 14.6).
3. Der Benutzer betätigt im Dialog den Button „Open...“.
4. Daraufhin zeigt GIANT den Standard-Filechooser-Dialog (siehe 14.13.1).
5. Der Benutzer wählt die Datei (siehe 13.4) aus.
6. GIANT zeigt das aus der Datei geladene GQSL Skript (siehe Kapitel 10) im Textfeld des Skriptdialogs an.
7. Falls gewünscht kann der Benutzer das GQSL Skript im Textfeld noch manuell weiter modifizieren.
8. Der Benutzer startet die Berechnung des Skripts durch Betätigung des „Start Query“ im Skriptdialog (siehe 14.6).
9. GIANT prüft das geladene und eventuell modifizierte GQSL Skript. Sollte das Skript nicht den Vorgaben der Grammatik (siehe Kapitel 10) entsprechen, erscheint eine Fehlermeldung (siehe 3.1) und das System kehrt zu Schritt 7 des UseCase zurück.

10. GIANT führt das Skript aus und teilt dem Benutzer den Fortschritt mittels eines Progress-Dialogs (siehe 14.15.2) mit. Während der Ausführung des Skripts ist das System mit Ausnahme des Progress-Dialogs gesperrt.

9.2.5 Alternativen

- Bei Punkt 3:
  Der Benutzer bricht den UseCase mit Cancel ab.
- Bei Punkt 4:
  Der Benutzer bricht die Auswahl der Datei mit Cancel ab. Das System kehrt dann zu Schritt 2 bei der Abarbeitung des UseCase zurück.
- Bei Punkt 6:
  Der Benutzer bricht den UseCase mit Cancel ab.
- Bei Punkt 10:
  Die laufende Ausführung des Skripts wird vom Benutzer mit Cancel abgebrochen werden.

9.3 UC: Skript speichern

Mit diesem UseCase kann der Benutzer GQSL Skripte aus dem Skriptdialog (siehe 14.6) in Dateien (siehe 13.4) speichern.

9.3.1 Vorbedingung

1. Der Skriptdialog (siehe 14.6) ist geöffnet und enthält in dem dafür vorgesehenen Textfeld entweder ein manuell eingegebenes oder ein aus einer Datei geladenes und eventuell modifiziertes GQSL Skript.

9.3.2 Nachbedingung Erfolg

1. Eine Datei, welche das GQSL Skript enthält, wurde angelegt.
2. GIANT zeigt den Skriptdialog, das gespeicherte GQSL Skript ist weiterhin in dem Textfeld vorhanden.

9.3.3 Nachbedingung Fehlschlag

1. Das System bleibt im bisherigen Zustand.
2. Es wurde keine Datei erzeugt
3. GIANT zeigt weiterhin den Skriptdialog an (siehe 14.6) und alle dort getätigten Eingaben (insbesondere das GQSL Skript im Textfeld des Dialoges) bleiben erhalten.

9.3.4 Beschreibung

1. Der Benutzer betätigt im Skriptdialog (siehe 14.6) den Button „Save As...“.
2. GIANT prüft ob das GQSL Skript im Textfeld des Skriptdialogs den Vorgaben der Grammatik (siehe Kapitel 10) entspricht. Falls nicht, erscheint eine Fehlermeldung (siehe 3.1) und das System kehrt zu Schritt 1 zurück.
3. GIANT öffnet den Standard-Filechooser-Dialog (siehe 14.13.1).
4. Der Benutzer gibt den Pfad und die Datei, in der das GQSL Skript gespeichert werden soll, vor und bestätigt mit OK.
5. GIANT speichert das GQSL Skript in der vorgegebenen Datei.

9.3.5 Alternativen

- Bei Punkt 4:
  Der Benutzer bricht den UseCase mit Cancel ab.

10 GIANT Query & Skripting Language

In diesem Kapitel wird die GIANT Query and Scripting Language (GQSL) sowie Interpreter für die GQSL spezifiziert. GQSL ist eine Sprache mit deren Hilfe Anfragen an die IML-Bibliothek gestellt werden und auf den Resultaten Aktionen ausgeführt werden können.

Hier wird zunächst die Struktur von IML-Graphen aus der Sicht von GIANT beschrieben und danach Syntax und Sematik von GQSL definiert.

10.1 IML-Graphen

Der Kunde stellt die Reflektion zur Verfügung. GIANT verwendet diese Bibliothek, um auf IML-Graph Dateien zuzugreifen.

10.1.1 IML-Programme

In jeder IML-Graph Datei ist ein IML-Programm enthalten. Das IML-Programm besteht aus

1. IML-Knoten
2. Attributen von IML-Knoten
3. nichts sonst.

Es gibt in jedem IML-Programm einen besonderen IML-Knoten, der Wurzelknoten genannt wird.

10.1.1.1 Attribute von IML-Knoten

Jeder IML-Knoten hat einen Typ. Für jeden Typ von IML-Knoten existiert ein String, der den Typ eineindeutig identifiziert.

Jeder IML-Knoten verfügt über eine Folge von Attributen. Diese Folge definiert eine endliche Anzahl einzelner Attribute, die der IML-Knoten besitzt sowie eine Reihenfolge dieser Attribute. Weder die Folge von Attributen noch die in der Folge enthaltenen Attribute eines IML-Knoten können sich während der Laufzeit von GIANT verändern.

Jedes Attribut besitzt einen Attribut-Namen. Kein IML-Knoten besitzt zwei verschiedene Attribute, die einen gleichen Namen besitzen.

Einige Attribute sind Verweise. Sie verweisen auf höchstens einen IML-Knoten. Es werden unterschieden:

genutzte Verweise Verweise, die auf genau einen IML-Knoten verweisen.

ungenutzte Verweise Verweise, die auf nichts verweisen.

Jedes Attribut eines IML-Knoten gehört genau einer der folgenden Klassen an:

Einfache Attribute : Haben einen bestimmten Wert. Mögliche Typen dieses Werts sind:

1. Source Location
   a) Zeilennummer (Natural)
   b) Spaltennummer (Natural)
   c) Filename (String)
   d) Pfadname (String)
2. Boolean
3. Natural
4. String
5. Folge von Strings

Bezeichner-Attribut Haben als Wert einen String

Genutztes Verweis-Attribut Ein genutzter Verweis

Ungenutztes Verweis-Attribut Ein ungenutzter Verweis

Verweisfolgen-Attribut Eine endliche Folge von genutzten Verweisen

Verweismengen-Attribut Eine endliche Menge von genutzten Verweisen

10.1.2 IML-Graphen

Jedes IML-Programm definiert einen zugehörigen IML-Graph. Sprechweise: Das IML-Programm liegt dem IML-Graph zugrunde. Ein IML-Graph ist ein gerichteter knoten- und kantenannotierter Graph mit Schlingen und Mehrfachkanten. Er ist definiert durch die folgende Vorschrift:

1. Ein IML-Graph besitzt als Knotenmenge die Menge aller IML-Knoten des zugrunde liegenden IML-Programms. Der IML-Graph besitzt keine weiteren Knoten.
2. Annotationen eines IML-Knoten sind alle Attribute des IML-Knoten.
3. Eine Kante im IML-Graph heißt IML-Kante. Eine IML-Kante e von einem IML-Knoten v zu einem IML-Knoten w des selben IML-Graph existiert genau dann, wenn eine der folgenden Bedingungen erfüllt ist:
   a) v besitzt ein Genutztes Verweis-Attribut, das auf w verweist. e besitzt als Annotation den Attribut-Namen des Genutzten Verweis-Attributs. Dieser Attribut-Name ist der Typ von e
   b) v besitzt ein Verweisfolgen-Attribut f. Ein genutzter Verweis aus f verweist auf w. e besitzt als Annotation den Attribut-Namen von f sowie die Nummer innerhalb der Folge f des genutzten Verweises auf w. Der Attribut-Name von f ist der Typ von e.

c) v besitzt ein Verweismengen-Attribut m. Ein genutzter Verweis aus m verweist auf w. e besitzt als Annotation den Attribut-Namen von m. Der Attribut-Name von m ist der Typ von e.

4. Falls eine IML-Kante e existiert, so hat e keine Annotationen, außer den in Punkt 3 genannten.

10.2 GQSL

Die GIANT Query & Scripting Language dient dazu, IML-Teilgraphen und Selektionen aus einem IML-Graph anzufragen und auf diesen Aktionen auszuführen.

Die Sprache lässt sich in zwei zentrale Konstrukte gliedern:

Queries dienen der Anfrage eines Werts.

Scripts werden verwendet, um Aktionen auszuführen.

Als Aktion wird die Ausführung eines vordefinierten Scripts bezeichnet (siehe Kapitel 10.2.3). Aktionen können GIANT zu Reaktionen veranlassen, die für den Benutzer wahrnehmbar werden und für seine Arbeit nützlich sein können (z.B. Anzeigen eines IML-Teilgraphen in einem Anzeigefenster).

GQSL Queries können nicht direkt ausgeführt, sondern nur von GQSL Scripts aufgerufen werden. GQSL Scripts werden von einem GQSL Interpreter ausgeführt. Scripts können Queries aufrufen, um Daten zu gewinnen. Mit diesen Daten sollen Aktionen nach den Wünschen des Benutzers konfiguriert werden. Queries und Scripts können in Variablen Werte zwischenspeichern.

10.2.1 GQSL Interpreter

Ein GQSL Interpreter ist ein System, das GQSL Scripts ausführt, gemäß der Sprachdefinition in Kapitel 10.2.2. Jeder GQSL Interpreter muss die vordefinierte Sprachumgebung gemäß Kapitel 10.2.3 unterstützen.

10.2.1.1 Kontext des Interpreters

Der GQSL Interpreter kann im Kontext eines Anzeigefensters ausgeführt werden. Ist dies der Fall, so sollen die ausgeführten GQSL Scripts und GQSL Queries Zugriff auf das Anzeigefenster haben, ohne den Namen des Anzeigefensters zu kennen.

Der Zugriff auf das Anzeigefenster in dessen Kontext der GQSL Interpreter ausgeführt wird, soll über vordefinierte Queries durchgeführt werden.

10.2.1.2 Verhalten im Fehlerfall

Der GQSL Interpreter muss ein Script bzw. eine Query zurückweisen, falls eine Forderung der Sprachdefinition von diesem Script bzw. dieser Query verletzt wird. In diesem Fall muss der Interpreter dem Benutzer eine Fehlermeldung anzeigen, die folgende Anforderungen erfüllt:

1. Der Benutzer wird informiert, dass der Text des Scripts bzw. der Query fehlerhaft ist.
2. Dem Benutzer wird die Art des Fehlers präzise erläutert.
3. Dem Benutzer wird angezeigt an welcher Stelle in welchem Script bzw. in welcher Query der Fehler erkannt oder verursacht wurde.

Andernfalls startet der GQSL Interpreter die Ausführung.

Der GQSL Interpreter muss die Ausführung sofort stoppen, sobald die erste expression fehlerhaft ausgewertet wird oder sobald ein statement oder complex_statement fehlerhaft ausgeführt wird. Diese Ereignisse werden als Fehler angesehen. Der GQSL Interpreter darf die Ausführung bereits nach einem früheren Ausführungsschritt abbrechen, wenn sichergestellt ist, dass nach einem späteren Ausführungsschritt einer der genannten Fälle eintreten wird.

Der GQSL Interpreter muss dem Benutzer unverzüglich eine Meldung anzeigen, die folgende Anforderungen erfüllt:

1. Der Benutzer muss informiert werden, dass ein Fehler aufgetreten ist.
2. Dem Benutzer muss präzise und für ihn leicht verständlich erläutert werden, welcher Fehler aufgetreten ist.
3. Dem Benutzer kann eine Möglichkeit aufgezeigt werden, wie er den Fehler beseitigen kann um eine fehlerfreie Ausführung des selben Scripts zu ermöglichen.

Der GQSL Interpreter muss nach einem Fehler entweder alle seit Beginn der Ausführung bereits ausgeführten Aktionen rückgängig machen, oder er muss dem Benutzer anzeigen, welche Aktionen bereits durchgeführt wurden, bevor der Fehler passierte. Anhand dieser Information soll es dem Benutzer möglich sein, die Aktionen manuell rückgängig zu machen.

10.2.2 GQSL Sprachdefinition

Die Syntax von GQSL wird beschrieben in Anlehnung an die Beschreibungsmethode des Ada Reference Manual. Siehe ARM Kapitel 1.1.4 für Details.

10.2.2.1 White Space

Leerzeichen, Tabulatoren, Zeilenumbrüche und Kommentare können als Trennzeichen verwendet werden.

Kommentare werden durch // eingeleitet und gehen immer bis zum Zeilenende.

10.2.2.2 Schlüsselwörter

10.2.2.3 Bezeichner

identifier ::= letter { letter | digit }

1. letter steht für einen der Buchstaben a-z oder A-Z.

2. digit steht für eine der Ziffern 0-9.
3. Der gesamte Text des identifier darf nicht ein Schlüsselwort von GQSL sein.

10.2.2.4 Literale

string_literal ::= **"** { string_char } **"**

Wobei string_char für ein von **"** verschiedenes Zeichen steht. Der Typ eines string_literal ist **string**.

int_literal ::= [+ | -] digit { digit }

Der Typ eines int_literal ist **integer**

regexp ::= < { regexp_char } >

Wobei regexp_char für ein von < und > und von White Space verschiedenes Zeichen steht. { regexp_char } muss ein gültiger regulärer Ausdruck sein, nach der Definition des Ada95-Pakets GNAT.Regpat.

10.2.2.5 Typen

Jede expression und jede Variable hat in GQSL einen Typ, der ihren Wertebereich bestimmt.

type ::= **integer** | **string** | **node_set** | **edge_set** | **subgraph** | **selection**

integer Ganze Zahlen in einem implemetierungsabhängigen Bereich $min..max$ mit $min < -16.000$ und $max > +16.000$. Ein GQSL-Interpreter muss den genauen Wertebereich dokumentieren.

string Zeichenkette beliebiger Länge

node_set Endliche Menge von IML-Knoten

edge_set Endliche Menge von IML-Kanten

subgraph Endliche Menge von IML-Knoten zusammen mit endlicher Menge von IML-Kanten. Für jede enthaltene IML-Kante e mit Quellknoten v und Zielknoten w sind stets v und w ebenfalls in dem **subgraph** enthalten. Variablen dieses Typs können IML-Teilgraphen speichern.

selection Endliche Menge von IML-Knoten zusammen mit endlicher Menge von IML-Kanten. Variablen dieses Typs können Selektionen speichern.

10.2.2.6 Variablen

Eine query oder ein script kann in ihrem/seinem declaration_part Variablen deklarieren. Jede dieser Variablen hat einen definierten Typ (siehe Kapitel 10.2.2.5).

declaration_part ::= { declaration **;** }

declaration ::= identifier **:** type

1. Wird eine query oder ein script aufgerufen, so werden alle Variablen des zugehörigen declaration_part erzeugt.

2. Alle diese Variablen werden zerstört wenn die Ausführung der query bzw. des script endet.
3. Zu jedem Zeitpunkt der Ausführung hat jede dieser Variablen einen Wert innerhalb des Wertebereichs ihres Typs. Der Wert ändert sich nur durch ein assignment an die Variable.
4. Jede declaration deklariert genau eine Variable, die über ihren identifier innerhalb des query_body bzw. des script_body referenziert werden kann. Sie ist vom Typ type.
5. Keine zwei verschiedenen declaration dürfen den gleichen identifier innerhalb des selben declaration_part verwenden.
6. Keine declaration darf den gleichen identifier verwenden wie eine declaration innerhalb einer parameter_list, die der selben query bzw. dem selben script angehört (siehe Kapitel 10.2.2.7).
7. Zum Zeitpunkt der Erzeugung nimmt eine Variable ihren Initialwert an. Der Initialwert einer Variable ist implementierungsabhängig und kann nicht explizit angegeben werden. Es dürfen keine Annahmen über den Initialwert getroffen werden.

10.2.2.7 Parameter

Eine query oder ein script kann Parameter deklarieren. Jeder dieser Parameter hat einen definierten Typ (siehe Kapitel 10.2.2.5).

parameter_list ::= declaration { **;** declaration }

1. Wird eine query oder ein script aufgerufen, so werden alle Parameter erzeugt und nehmen den durch den Aufruf spezifizierten Initialwert an (siehe Kapitel 10.2.2.20 bzw. 10.2.2.13).
2. Alle Parameter werden zerstört, wenn die Ausführung der query bzw. des script endet.
3. Zu jedem Zeitpunkt der Ausführung hat jeder Parameter einen Wert innerhalb des Wertebereichs seines Typs. Der Wert eines Parameters ändert sich nur duch ein assignment an den Parameter.
4. Jede declaration deklariert genau einen Parameter, der über seinen identifier innerhalb des query_body bzw. des script_body referenziert werden kann.
5. Keine zwei verschiedenen declaration dürfen den gleichen identifier innerhalb der selben parameter_list verwenden.
6. Keine declaration darf den gleichen identifier verwenden wie eine declaration innerhalb eines declaration_part, der zu der selben query bzw. dem selben script gehört (siehe Kapitel 10.2.2.6).

10.2.2.8 Query

query ::=
query identifier
[**(** parameter_list **)**]
return type **is**
declaration_part

```
begin
  query_body
  return expression ;
end ;
```

1. Definiert eine GQSL Query, die aus jeder anderen GQSL Query sowie aus jedem GQSL Script durch den identifier aufgerufen werden kann.
2. In dem query_body und in der expression sind alle Variablen aus dem declaration_part, alle Parameter aus der parameter_list, sowie alle dem Interpreter bekannten GQSL Queries sichtbar. Sonst ist in dem query_body nichts sichtbar.
3. Nach einem Aufruf werden die Parameter der Query erzeugt und nehmen ihre Initialwerte an (siehe Kapitel 10.2.2.7), die Variablen werden erzeugt (siehe Kapitel 10.2.2.6) und schließlich wird der query_body ausgeführt (siehe Kapitel 10.2.2.10).
4. Falls der query_body erfolgreich ausgeführt wurde, dann wird die expression (siehe Kapitel 10.2.2.14) ausgewertet. Andernfalls ist die Ausführung der Query fehlerhaft.
5. Falls die expression erfolgreich ausgewertet wurde, so wird ihr Ergebnis zum Rückgabewert. Die Ausführung der Query ist erfolgreich. Andernfalls ist die Ausführung der Query fehlerhaft.
6. Falls die Ausführung der Query erfolgreich ist, so werden ihre Variablen und Parameter zerstört und die Ausführung der Query endet.

10.2.2.9 Script

```
script ::=
    script identifier
      [ ( parameter_list ) ]
      declaration_part
    begin
      script_body
    end ;
```

1. Definiert ein GQSL Script, das von jedem GQSL Script durch den identifier aufgerufen werden kann. Das Script kann auch von einem GQSL Interpreter direkt ausgeführt werden.
2. In dem script_body sind alle Variablen aus dem declaration_part, alle Parameter aus der parameter_list, sowie alle dem Interpreter bekannten GQSL Queries und GQSL Scripts sichtbar. Sonst ist in dem script_body nichts sichtbar.
3. Nach einem Aufruf werden die Parameter des Scripts erzeugt und nehmen ihren Initialwert an (siehe Kapitel 10.2.2.7), die Variablen werden erzeugt (siehe Kapitel 10.2.2.6) und schließlich wird der script_body ausgeführt (siehe Kapitel 10.2.2.11).
4. Falls der script_body erfolgreich ausgeführt wurde, dann ist die Ausführung des Scripts erfolgreich. Andernfalls ist die Ausführung des Scripts fehlerhaft.
5. Falls die Ausführung des Scripts erfolgreich ist, so werden seine Variablen und Parameter zerstört und die Ausführung des Scripts endet.

10.2.2.10 Query Body

query_body ::= { statement }

1. Ein query_body hat eine bestimmte *Sicht*. Die Sicht ist die Menge aller Objekte (genauer: Variablen, Parameter, Queries und Scripts), die in dem query_body sichtbar sind (vgl. Kapitel 10.2.2.8). Ein Objekt ist in dem query_body sichtbar, genau dann wenn es in dessen Sicht enthalten ist.
2. Alle Nicht-Terminale, die aus dem query_body abgeleitet sind (die in einem Ableitungsbaum unter dem Symbol query_body angeordnet sind), haben die selbe Sicht wie der query_body.
3. Ein query_body wird ausgeführt, indem alle statement in der Reihenfolge einer Linksableitung der Grammatik ausgeführt werden. Sobald ein statement fehlerhaft ausgeführt wird, stoppt die Ausführung, es wird keines der nachfogenden statement ausgeführt und der query_body ist fehlerhaft ausgeführt.
4. Der query_body ist erfolgreich ausgeführt, falls er entweder kein statement enthält oder falls das letzte statement erfolgreich ausgeführt wurde.
5. Die Ausführung des query_body endet sobald er entweder erfolgreich ausgeführt ist, oder fehlerhaft ausgeführt ist.

10.2.2.11 Script Body

script_body ::= { complex_statement }

1. Ein script_body hat eine bestimmte *Sicht*. Die Sicht ist die Menge aller Objekte (genauer Variablen, Parameter, Queries und Scripts), die in dem script_body sichtbar sind (vgl. Kapitel 10.2.2.9). Ein Objekt ist in dem script_body sichtbar, genau dann wenn es in der Sicht enthalten ist.
2. Alle Nicht-Terminale, die aus dem script_body abgeleitet sind (die in einem Ableitungsbaum unter dem Symbol script_body angeordnet sind), haben die selbe Sicht wie der script_body.
3. Ein script_body wird ausgeführt, indem alle complex_statement in der Reihenfolge einer Linksableitung der Grammatik ausgeführt werden. Sobald ein complex_statement fehlerhaft ausgeführt wird, stoppt die Ausführung, es wird keines der nachfolgenden complex_statement ausgeführt und der script_body ist fehlerhaft ausgeführt.
4. Der script_body ist erfolgreich ausgeführt, falls er entweder kein complex_statement enthält oder falls das letzte complex_statement erfolgreich ausgeführt wurde.
5. Die Ausführung des script_body endet, sobald er entweder erfolgreich ausgeführt ist, oder fehlerhaft ausgeführt ist.

10.2.2.12 Statement

statement ::= assignment **;**

assignment ::= identifier = expression

1. Der identifier muss entweder eine sichtbare Variable oder einen sichtbaren Parameter bezeichnen. Diese/r Variable/Parameter wird lhs genannt.
2. Der Typ der expression muss der selbe Typ sein wie der Typ von lhs
3. Zur Ausführung des statement wird die expression ausgewertet. Falls die expression erfolgreich ausgewertet wurde, so nimmt lhs den Wert der expression an. In diesem Fall ist die Ausführung des statement erfolgreich.
4. Falls expression fehlerhaft ausgewertet wurde, dann ist die Ausführung des statement fehlerhaft.

10.2.2.13 Complex Statement

complex_statement ::= statement | script_call **;**
script_call ::= identifier [(argument_list)]
argument_list ::= expression { **,** expression }

statement Jedes complex_statement enthält alle Möglichkeiten des statement.

1. Zum Ausführen des complex_statement wird das statement ausgeführt.
2. Das complex_statement ist erfogreich ausgeführt, falls das statement erfolgreich ausgeführt ist.
3. Das complex_statement ist fehlerhaft ausgeführt, falls das statement fehlerhaft ausgeführt ist.

script_call Aufruf eines GQSL Scripts.

1. identifier muss ein sichtbares GQSL Script S bezeichnen.
2. Falls S keine parameter_list besitzt, so darf keine argument_list angegeben sein.
3. Falls S eine parameter_list besitzt, so muss eine parameter_list angegeben sein und es müssen in der argument_list genau so viele expression angegeben sein, wie in der parameter_list declaration angegeben sind.
4. Der Typ der i-ten expression der argument_list muss der selbe sein wie der Typ der i-ten declaration der parameter_list für alle i.
5. Zum Ausführen des complex_statement werden alle expression ausgewertet. Die Reihenfolge in der diese Auswirkung geschieht hat auf das Ergebnis keinen Einfluss. Die Reihenfolge wird durch den Interpreter gewählt.
6. Falls eine der expression fehlerhaft ausgewertet wird, so ist die Ausführung des complex_statement fehlerhaft. Andernfalls wird S aufgerufen. Der i-te Parameter von S nimmt den Wert der i-ten expression an für alle i.
7. Falls S erfolgreich ausgeführt wurde, so ist das complex_statement erfolgreich ausgeführt. Falls S fehlerhaft ausgeführt wurde, so ist das complex_statement fehlerhaft ausgeführt.

10.2.2.14 Expression

expression ::=
node_set_expression
| edge_set_expression
| subgraph_expression
| selection_expression
| variable_inspection
| query_call
| calculation
| literal_expression

Annahme: Aus den Alternativen sei die Regel mit der rechten Seite R gewählt (R besteht aus genau einem Nicht-Terminal).

1. Der Typ der expression ist der Typ des Nicht-Terminals R (siehe die folgenden Kapitel für Details).
2. Die expression wird erfolgreich ausgewertet, falls R erfolgreich ausgewertet wird. Andernfalls ist die expression fehlerhaft ausgewertet.
3. Falls R erfolgreich ausgewertet wird, so ist der Wert der expression gleich dem Wert der Auswertung von R.

10.2.2.15 Node Set Expression

node_set_expression ::= **select nodes from** expression [node_predicate]
| **build nodes from** expression [**accept nodes** node_predicate]
| **all_nodes**
| **root_node**

Der Typ einer node_set_expression ist **node_set**.

select Trifft eine Auswahl bestimmter IML-Knoten aus einer Menge.

1. Der Typ von expression muss **node_set**, **subgraph** oder **selection** sein.
2. Die Auswertung ist erfolgreich genau dann, wenn die Auswertung der expression erfolgreich ist und die Vorbereitung des node_predicate erfolgreich ist falls dieses angegeben ist. Andernfalls ist die Auswertung fehlerhaft.
3. Ist die Auswertung erfolgreich, dann ist das Ergebnis die größte Menge M von IML-Knoten, so dass $\forall v \in M$
   a) v ist in expression enthalten.
   b) Falls ein node_predicate angegeben ist, dann erfüllt v es (siehe Kapitel 10.2.2.23).

build Bildet eine Menge von IML-Knoten, die inzident sind zu einer gegebenen Menge von IML-Kanten.

1. Der Typ von expression muss **edge_set** sein.

2. Die Auswertung ist erfolgreich genau dann, wenn die Auswertung von expression erfolgreich ist und die Vorbereitung von node_predicate erfolgreich ist, falls ein node_predicate angegeben ist. Andernfalls ist die Auswertung fehlerhaft.
3. Ist die Auswertung erfolgreich, dann ist das Ergebnis die größte Menge M von IML-Knoten, so dass $\forall v \in M$
   a) v ist inzident zu einer Kante e aus expression
   b) v erfüllt node_predicate, falls dieses angegeben ist.

all_nodes Die Menge aller IML-Knoten des IML-Graphen.

1. Die Auswertung ist stets erfolgreich.
2. Das Ergebnis ist die Menge aller IML-Knoten des IML-Graphen.

root_node Die Wurzel des IML-Graphen.

1. Die Auswertung ist stets erfolgreich.
2. Das Ergebnis ist die Menge, die den Wurzelknoten des dem IML-Graphen zugrundeliegenden IML-Programms enthält und sonst nichts.

10.2.2.16 Edge Set Expression

edge_set_expression ::= **select edges from** expression [edge_predicate]
| **build edges** [**from** expression] [**to** expression] [edge_predicate]

Der Typ einer edge_set_expression ist **edge_set**.

select Trifft eine Auswahl von bestimmten IML-Kanten aus einer Menge.

1. Der Typ von expression muss **edge_set**, **subgraph** oder **selection** sein.
2. Die Auswertung ist erfolgreich genau dann, wenn die Auswertung von expression erfolgreich ist und das edge_predicate entweder nicht angegeben ist oder seine Vorbereitung erfolgreich ist. Andernfalls ist die Auswertung fehlerhaft.
3. Falls die Auswertung erfolgreich ist, dann ist das Ergebnis die größte Menge von IML-Kanten M mit $\forall e \in M$
   a) e ist in der Menge der IML-Kanten aus expression enthalten.
   b) Falls das edge_predicate angegebn ist, so wird es von e erfüllt (siehe Kapitel 10.2.2.23)

build Bildet eine Menge von IML-Kanten, die zwischen zwei Mengen von IML-Knoten verlaufen.

1. Beide expression müssen – sofern sie angegeben sind – den Typ **node_set** haben.
2. Die Auswertung ist erfolgreich, falls alle angegebenen expression erfolgreich ausgewertet werden und das edge_predicate entweder nicht angegebn ist oder erfolgreich vorbereitet wird. Andernfalls ist die Auswertung fehlerhaft.
3. Das Ergebnis ist die Menge M aller IML-Kanten mit $\forall e \in M$
   a) e ist im IML-Graph enthalten.

b) Der Start-Knoten von e ist in der **from**-expression enthalten, falls **from**-expression angegeben ist.

c) Der Ziel-Knoten von e ist in der **to**-expression enthalten, falls **to**-expression angegeben ist.

d) e erfüllt das edge_predicate falls dieses angegeben ist.

10.2.2.17 Subgraph Expression

subgraph_expression ::= **make graph (** expression **,** expression **)**
| **build graph based on** expression **follow edges** edge_predicate [**accept nodes** node_predicate] [**until depth** expression]

Der Typ einer subgraph_expression ist immer **subgraph**.

make Erstellt einen IML-Teilgraph aus einer Menge von Knoten und einer Menge von Kanten.

1. Der Typ der ersten expression muss **node_set** sein.
2. Der Typ der zweiten expression muss **edge_set** sein.
3. Die Auswertung ist erfolgreich genau dann, wenn die Auswertung beider expression erfolgreich ist. Andernfalls ist die Auswertung fehlerhaft.
4. Falls die Auswertung erfolgreich ist, so ist das Ergebnis ein IML-Teilgraph mit allen IML-Knoten der ersten expression sowie denjenigen IML-Kanten die in der zweiten expression enthalten sind und deren Start- und Zielknoten in der ersten expression enthalten sind.

build Fragt einen Teilgraph an, ausgehend von einer Grundmenge von IML-Knoten.

1. Der Typ der ersten expression muss **node_set** sein.
2. Falls die zweite expression angegeben ist, so muss ihr Typ **integer** sein.
3. Die Auswertung ist erfolgreich falls die folgenden Bedingungen erfüllt sind:
   a) Alle expression werden erfolgreich ausgewertet.
   b) Die Vorbereitung des edge_predicate muss erfolgreich sein.
   c) Die Vorbereitung des node_predicate muss erfolgreich sein, falls es angegeben ist.
   d) Falls die zweite expression angegeben ist, so muss ihr Wert größer oder gleich 0 sein.

   andernfalls ist die Auswertung fehlerhaft.
4. Der Wert ergibt sich durch eine Breitensuche auf dem IML-Graph. Begonnen wird mit der IML-Knotenmenge, die durch die erste expression gegeben ist (Ebene 0). Dann werden alle IML-Kanten verfolgt für die
   a) Das edge_predicate erfüllt ist.

b) Der über diese IML-Kante erreichte IML-Knoten das node_predicate erfüllt, sofern dieses angegeben ist.

c) Die Nummer der Ebene kleiner ist als die zweite expression, sofern diese angegeben ist.

Wurde eine IML-Kante verfolgt, so wird sie zum Ergebnis hinzugefügt. Der erreichte IML-Knoten wird ebenfalls zum Ergebnis hinzugefügt. Die Menge aller so erreichten IML-Knoten bildet die Ausgangsmenge für den nächsten Schritt. Sie ist eine Ebene tiefer. Die Suche bricht ab, wenn keine neuen Kanten mehr aufgenommen werden.

10.2.2.18 Selection Expression

selection_expression ::= **make selection (** expression **,** expression **)**

1. die erste expression muss den Typ **node_set** haben.
2. die zweite expression muss den Typ **edge_set** haben.
3. Der Typ der selection_expression ist **selection**.
4. Die Auswertung ist erfolgreich genau dann, wenn die Auswertung beider expression erfolgreich ist. Andernfalls ist die Auswertung fehlerhaft.
5. Falls die Auswertung erfolgreich ist, dann ist der Wert die Selektion, die alle IML-Knoten und alle IML-Kanten aus den beiden expression und sonst nichts enthält.

10.2.2.19 Variable Inspection

variable_inspection ::= identifier

1. Der identifier muss entweder eine sichtbare Variable v oder einen sichtbaren Parameter p bezeichnen.
2. Der Typ der variable_inspection ist der Typ von v bzw. p.
3. Eine variable_inspection wird stets erfolgreich ausgewertet.
4. Der Wert der variable_inspection ist der Wert, den v bzw. p zum Zeitpunkt der Auswertung hat.

Anmerkung Eine variable_inspection kann nur während der Lebenszeit einer Variable bzw. eines Parameters geschehen. Eine Variable hat zu jedem Zeitpunkt ihrer Lebenszeit einen Wert. (vgl. Kapitel 10.2.2.6, 10.2.2.7)

Anmerkung Der Wert von v verändert sich durch die variable_inspection nicht.

10.2.2.20 Query Call

query_call ::= identifier [**(** argument_list **)**]

Anmerkung: argument_list siehe 10.2.2.13

1. identifier muss eine sichtbare GQSL Query Q bezeichnen.
2. Falls Q keine parameter_list besitzt, so darf keine argument_list angegeben sein.
3. Falls Q eine parameter_list besitzt, so muss eine argument_list angegeben sein und darin müssen genau so viele expression angegeben sein, wie in der parameter_list declaration angegeben sind.
4. Der Typ der i-ten expression der argument_list muss der selbe sein wie der Typ der i-ten declaration der parameter_list für alle i.
5. Der Typ des query_call ist der Typ des Rückgabewerts von Q.
6. Zum Auswerten des query_call werden alle expression in durch den Interpreter willkürlich bestimmter Reihenfolge ausgewertet.
7. Falls eine der expression fehlerhaft ausgewertet wird, so ist die Auswertung des query_call fehlerhaft. Andernfalls wird Q aufgerufen. Der i-te Parameter nimmt den Wert der i-ten expression an für alle i.
8. Falls Q erfolgreich ausgeführt wurde, so ist der query_call erfolgreich ausgewertet. Falls Q fehlerhaft ausgeführt wurde, so ist der query_call fehlerhaft ausgewertet.
9. Falls der query_call erfolgreich ausgewertet wurde, so ist sein Wert gleich dem Rückgabewert von Q.

10.2.2.21 Calculation

calculation ::= **(** expression **)**
| expression + expression
| expression - expression
| expression **intersect** expression

1. Der Typ aller expression muss der selbe sein.
2. **Hinweis**: nicht alle Typen sind für jede rechte Seite der Regel legal. Details werden in Punkt 7 dieser Aufzählung spezifiziert.
3. Der Typ der calculation ist der selbe Typ wie der der expression.
4. Die calculation wird erfolgreich ausgewertet genau dann alle expression erfolgreich ausgewertet werden und keine Einschränkung aus Punkt 7 zutrifft. Sonst ist sie fehlerhaft ausgewertet.
5. Die Operatoren +, -, **intersect** haben alle eine gleich hohe Priorität. In einer Kette mehrerer calculation dieser Operatoren erfolgt die Auswertung von links nach rechts.
6. Die rechte Seite **(**... **)** wird zur Veränderung der Auswertungsreihenfolge verwendet.
7. Gültigkeit und Wert in Abhängigkeit von der gewählten Regel:

   ( ) a) Es ist jeder Typ zulässig.

   b) Der Wert ist der Wert der expression.

   + Es ist jeder Typ zulässig. Der Wert ist abhängig vom verwendeten Typ:

integer Summe der expression. Falls die Summe den (implementationsabhängigen) Wertebereich des Typs **integer** verlässt, so ist die calculation fehlerhaft ausgewertet.

string Konkatenation der expression.

node_set, edge_set Vereinigung der expression.

subgraph, selection Alle IML-Knoten aus beiden expression mit allen IML-Kanten aus beiden expression.

- Es sind alle Typen außer **string** zugelassen. Der Wert ist abhängig vom verwendeten Typ:

integer Differenz der expression. Falls die Differenz den (implementationsabhängigen) Wertebereich verlässt, so ist die calculation fehlerhaft ausgewertet.

string Illegal.

node_set, edge_set Mengendifferenz der expression.

subgraph Menge der IML-Knoten ist die Differenz der IML-Knoten-Mengen der expression. Menge der IML-Kanten ist die größte Teilmenge der Differenz der IML-Kanten-Mengen, so dass der Start- und Zielknoten jeder Kante in der Differenz der IML-Knoten-Mengen enthalten ist.

selection Differenz der IML-Knoten-Mengen und Differenz der IML-Kanten-Mengen.

intersect Die Typen **integer** und **string** sind nicht zugelassen. Der Wert ist abhängig vom angegebenen Typ:

integer, string Illegal.

node_set, edge_set Mengenschnitt der expression

node_set, edge_set Menge der IML-Knoten ist Schnittmenge der IML-Knoten-Mengen der expression. Menge der IML-Kanten ist Schnittmenge der expression.

10.2.2.22 Literal Expression

literal_expression ::= int_literal
| string_literal

Eine literal_expression wird stets erfolgreich ausgewertet.

int_literal Der Typ der literal_expression ist **integer**. Der Wert der literal_expression ist die Interpretation der Ableitung des int_literal im 10er Zahlensystem. Falls dieser Wert den (implementationsabhängigen) Wertebereich überschreitet, so ist die literal_expression nicht zulässig.

string_literal Der Typ der literal_expression ist **string**. Der Wert der literal_expression ist die Ableitung des string_literal.

10.2.2.23 Logik

node_predicate ::= node_choosing_product [**or** node_predicate]

edge_predicate ::= edge_choosing_product [**or** edge_predicate]

attrib_expr ::= attrib_product [**or** attrib_expr]

1. Das node_predicate bzw. das edge_predicate bzw. die attrib_expr wird erfolgreich vorbereitet, falls auf der rechten Seite der Regel sowohl das node_choosing_product bzw. das edge_choosing_product bzw. das attrib_product und (falls angegeben) das node_predicate bzw. das edge_predicate bzw. die attrib_expr erfolgreich vorbereitet werden. Andernfalls wird fehlerhaft vorbereitet.

2. Das node_predicate bzw. das edge_predicate bzw. die attrib_expr wird erfüllt dann und nur dann, falls das node_choosing_product bzw. das edge_choosing_product bzw. das attrib_product oder auf der rechten Seite der Regel das node_predicate bzw. das edge_predicate bzw. die attrib_expr angegeben ist und erfüllt wird.

node_choosing_product ::= node_choosing_statement [**and** node_choosing_product]

edge_choosing_product ::= edge_choosing_statement [**and** edge_choosing_product]

attrib_product ::= attrib_statement [**and** attrib_product]

1. Das node_choosing_product bzw. das edge_choosing_product bzw. das attrib_product wird erfolgreich vorbereitet, falls auf der rechten Seite das node_choosing_statement bzw. das edge_choosing_statement bzw. das attrib_statement und (falls angegeben) das node_choosing_product bzw. das edge_choosing_product bzw. das attrib_product erfolgreich vorbereitet werden. Andernfalls ist die Vorbereitung fehlerhaft.

2. Das node_choosing_product bzw. das edge_choosing_product bzw. das attrib_product wird erfüllt genau dann, falls das node_choosing_statement bzw. das edge_choosing_statement bzw. das attrib_statement und auf der rechten Seite das node_choosing_product bzw. das edge_choosing_product bzw. das attrib_product nicht angegeben ist oder erfüllt wird.

node_choosing_statement ::= **(** node_predicate **)**
| **not** node_choosing_statement
| simple_node_choice

edge_choosing_statement ::= **(** edge_predicate **)**
| **not** edge_choosing_statement
| simple_edge_choice

attrib_statement ::= **(** attrib_expr **)**
| **not** attrib_statement
| simple_attrib_statement

Klammerung Wird verwendet, um die Auswertungsreihenfolge zu beeinflussen.

1. Das node_choosing_statement bzw. das edge_choosing_statement bzw. das attrib_statement wird erfolgreich vorbereitet, falls auf der rechten Seite das node_predicate bzw. das edge_predicate bzw. die attrib_expr erfolgreich vorbereitet wird. Andernfalls ist die Vorbereitung fehlerhaft.

2. Das node_choosing_statement bzw. das edge_choosing_statement bzw. das attrib_statement wird erfüllt genau dann, falls das node_predicate bzw. das edge_predicate bzw. die attrib_expr erfüllt wird.

Negation Logische Negation.

1. Das node_choosing_statement bzw. das edge_choosing_statement bzw. das attrib_statement wird erfolgreich vorbereitet, falls auf der rechten Seite das node_choosing_statement bzw. das edge_choosing_statement bzw. das attrib_statement erfolgreich vorbereitet wird. Andernfalls ist die Vorbereitung fehlerhaft.

2. Das node_choosing_statement bzw. das edge_choosing_statement bzw. das attrib_statement wird erfüllt genau dann, falls auf der rechten Seite das node_choosing_statement bzw. das edge_choosing_statement bzw. das attrib_statement nicht erfüllt wird.

Simple Statement Logische Aussage.

1. Das node_choosing_statement bzw. das edge_choosing_statement bzw. das attrib_statement wird erfolgreich vorbereitet, falls auf der rechten Seite der simple_node_choice bzw. der simple_edge_choice bzw. das simple_attrib_statement erfolgreich vorbereitet wird. Andernfalls ist die Vorbereitung fehlerhaft.

2. Das node_choosing_statement bzw. das edge_choosing_statement bzw. das attrib_statement wird erfüllt genau dann, falls der simple_node_choice bzw. der simple_edge_choice bzw. das simple_attrib_statement erfüllt wird.

10.2.2.24 Simple Node Choice

simple_node_choice ::= **having (** attrib_expr **)** | **type** regexp

having Testet ob die Attribute eines IML-Knoten einen bestimmten logischen Ausdruck erfüllen.

1. Die Vorbereitung ist erfolgreich genau dann, wenn die Vorbereitung der attrib_expr erfolgreich ist. Andernfalls ist die Vorbereitung fehlerhaft.

2. Der simple_node_choice wird erfüllt genau dann, wenn die attrib_expr erfüllt wird. Andernfalls wird der simple_node_choice nicht erfüllt.

type Testet ob ein IML-Knoten einen bestimmten Typ hat.

1. Die Vorbereitung ist erfolgreich.

2. Der simple_node_choice wird erfüllt von einem IML-Knoten N, falls der Name des Typs von N in der Sprache von regexp enthalten ist. Andernfalls wird der simple_node_choice nicht erfüllt.

10.2.2.25 Simple Edge Choice

simple_edge_choice ::= **type** regexp

1. Die Vorbereitung ist erfolgreich.

2. Der simple_edge_choice wird erfüllt von einer IML-Kante E, falls der Name des Typs von E in der Sprache von regexp enthalten ist. Andernfalls wird der simple_edge_choice nicht erfüllt.

10.2.2.26 Atomic Attribute Values

Die Eine Attributinspektion wird auf einen einzelnen IML-Knoten N angewendet. Sie hat als Ergebnis den Wert eines bestimmten Attributs dieses IML-Knotens. Der Typ einer Attributinspektion kann nicht statisch bestimmt werden.

attrib_atomic_value ::= atomic_value | atomic_string_list | atomic_sloc |
atomic_target_type | atomic_link_target | atomic_size | atomic_list_choice

Das Nicht-Terminal attrib_atomic_value ist eine Zusammenfassung aller möglichen Attributinspektionen. Das rechts stehende Nicht-Terminal bestimmt die Semantik der Inspektion.

atomic_value ::= identifier

1. Die Auswertung ist erfolgreich, falls N ein einfaches Attribut a mit Namen identifier besitzt und der Typ dieses Attributs Boolean oder Natural ist. Andernfalls ist die Auswertung fehlerhaft.
2. Falls die Auswertung erfolgreich ist, dann ist der Typ des atomic_value gleich dem Typ von a.
3. Falls die Auswertung erfolgreich ist, dann ist der Wert des atomic_value der Wert des Attributs a.

atomic_string_list ::= identifier **(** expression **)**

1. Die Auswertung ist erfolgreich, falls
   a) N ein Einfaches Attribut a mit Namen identifier besitzt.
   b) Falls der Typ von a Folge von Strings ist.
   c) Falls der Typ der expression **integer** ist.
   d) Falls die expression erfolgreich ausgewertet wird.
   e) Falls der Wert i der expression im Bereich $1 \leq i \leq max$ ist, wenn max die Anzahl der Folgenglieder in a ist.

   Andernfalls ist die Auswertung fehlerhaft.
2. Der Typ ist String.
3. Falls die Auswertung erfolgreich ist, so ist der Wert der expression-te String von a.

atomic_sloc ::= identifier **.** (**line** | **column** | **file_name** | **path**)

1. Die Auswertung ist erfolgreich genau dann, wenn N ein Attribut a besitzt mit Namen identifier und Typ Source Location. Sonst ist die Auswertung fehlerhaft.
2. Falls die Auswertung erfolgreich ist, so ist der Typ Natural, falls die Regel nach **line** oder **column** abgeleitet wurde. Andernfalls ist der Typ String.
3. Falls die Auswertung erfolgreich war, dann ist der Wert:

   line Die Zeilennummer von a

   column Die Spaltennummer von a

file_name Der Dateiname von a

path Der Pfad von a

atomic_target_type ::= identifier **. target_type**

1. Die Auswertung ist erfolgreich, falls N ein Genutztes Verweis-Attribut a_g, ein Ungenutztes Verweis-Attribut a_u, ein Verweismengen-Attribut a_m oder ein Verweisfolgen-Attribut a_f mit Namen identifier besitzt. Sonst ist die Auswertung fehlerhaft.
2. Der Typ ist String.
3. Falls die Auswertung erfolgreich ist, so ist der Wert der Name des Typs von IML-Knoten auf den a_g verweist bzw. auf den a_u verweisen könnte bzw. auf den ein Genutzter Verweis in a_m verweist bzw. auf den ein Genutzter Verweis in a_f verweist.

atomic_link_target ::= identifier **.** attrib_atomic_value

1. Besitzt N ein Genutztes Verweis-Attribut a mit Namen identifier, und verweist a auf den IML-Knoten M, so wird attrib_atomic_value auf M ausgewertet.
2. Die Auswertung ist erfolgreich, falls a existiert und falls die Auswertung von attrib_atomic_value auf M erfolgreich ist. Sonst ist die Auswertung fehlerhaft.
3. Falls a existiert, so ist der Typ der Typ der Auswertung von attrib_atomic_value auf M.
4. Falls die Auswertung erfolgreich ist, so ist der Wert der Wert der Auswertung von attrib_atomic_value auf M.

atomic_size ::= identifier **. size**

1. Die Auswertung ist erfolgreich, falls N ein Verweisfolgen-Attribut a_f oder ein Verweismengen-Attribut a_m mit Namen identifier besitzt. Andernfalls ist die Auswertung fehlerhaft.
2. Der Typ ist Natural.
3. Falls die Auswertung erfolgreich ist, so ist der Wert die Kardinalität von a_m bzw. die Anzahl der Folgenglieder von a_f.

atomic_list_choice ::= identifier **(** expression **) .** attrib_atomic_value

1. Die Auswertung ist erfolgreich, falls
   a) N ein Verweisfolgen-Attribut a mit Namen identifier besitzt.
   b) Falls der Typ der expression **integer** ist.
   c) Falls die expression erfolgreich ausgewertet wird.
   d) Falls der Wert i der expression im Bereich $1 \leq i \leq max$ ist, wenn max die Anzahl der Folgenglieder in a ist.

   Andernfalls ist die Auswertung fehlerhaft.
2. Falls die Auswertung erfolgreich ist, so sei M der IML-Knoten auf den das i-te Folgenglied in a verweist. Es wird attrib_atomic_value auf M ausgewertet.

3. Falls die Auswertung erfolgreich ist, so ist der Typ der Typ von attrib_atomic_value ausgewertet auf M.

4. Falls die Auswertung erfolgreich ist, so ist der Wert der Wert der Auswertung von attrib_atomic_value ausgewertet auf M.

10.2.2.27 Simple Attribute Statement

simple_attrib_statement ::= attrib_atomic_value [compare_operator expression]

compare_operator ::= < | = | > | <= | >= | **!=**

1. Das simple_attrib_statement wird erfolgreich vorbereitet, falls die expression entweder nicht angegeben ist oder erfolgreich ausgewertet wird. Andernfalls wird das simple_attrib_statement fehlerhaft vorbereitet.

2. Es sind nur die nachfolgend genannten Kombinationen der Typen von attrib_atomic_value, expression und des compare_operator zulässig:

   Boolean, — Es darf kein compare_operator und keine expression angegeben sein. Das simple_attrib_statement wird erfüllt, falls das attrib_atomic_value erfolgreich ausgewertet wird und den Wert true hat. Sonst wird das simple_attrib_statement nicht erfüllt.

   Natural, integer Zulässig sind alle Möglichkeiten für compare_operator. Das simple_attrib_statement wird erfüllt, falls der attrib_atomic_value erfolgreich ausgewertet wird und der entstehende Ausdruck nach der mathematischen Definition korrekt ist.

   String, string Zulässig sind alle Möglichkeiten für compare_operator. Das simple_attrib_statement wird erfüllt, falls das attrib_atomic_value erfolgreich ausgewertet wird und der entstehende Ausdruck korrekt ist. Eine Zeichenfolge a gilt als kleiner als eine Zeichenfolge b, falls a lexikographisch vor b kommt.

simple_attrib_statement ::= attrib_atomic_value **in** regexp

1. Das simple_attrib_statement wird stets erfolgreich vorbereitet.

2. Das simple_attrib_statement wird erfüllt, falls das attrib_atomic_value erfolgreich ausgewertet wird und in der durch regexp gegebenen Sprache enthalten ist. Andernfalls wird simple_attrib_statement nicht erfüllt.

10.2.3 GQSL vordefinierte Sprachumgebung

Vordefinierte Queries

```
Get_Current_Window
  return string is
// ... implementation defined ...
```

Liefert als Ergebnis den Namen des Anzeigefensters, in dessen Kontext der GQSL Interpreter sich befindet. Befindet sich der GQSL Interpreter nicht im Kontext eines Anzeigefensters, so wird die Query fehlerhaft ausgeführt.

```
Get_Current_Selection
  (Window_Name : string)
  return selection is
// ... implementation defined ...
```

Erhält als Argument den Namen eines Anzeigefensters und liefert als Ergebnis die aktuelle Selektion in diesem Anzeigefenster. Existiert kein Anzeigefenster des übergebenen Namens, so wird die Query fehlerhaft ausgeführt.

```
Get_Selection
  (Window_Name    : string;
   Selection_Name : string)
  return selection is
// ... implementation defined ...
```

Erhält als erstes Argument den Namen eines Anzeigefensters, als zweites Argument den Namen einer Selektion in diesem Anzeigefenster. Liefert als Ergebnis die durch diese Namen gewählte Selektion. Existiert kein Anzeigefenster des gewählten Namens oder existiert in dem Anzeigefenster keine Selektion des gewählten Namens, so wird die Query fehlerhaft ausgeführt.

```
Get_Subgraph
  (Subgraph_Name : string)
  return subgraph is
// ... implementation defined ...
```

Erhält als Argument den Namen eines IML-Teilgraphen. Liefert als Ergebnis den IML-Teilgraphen des gewählten Namens. Existiert kein IML-Teilgraph dieses Namens, so wird die Query fehlerhaft ausgeführt.

Vordefinierte Scripts

```
Create_Window
  (Window_Name : string;
   Layout_Algo : string;
   Content     : subgraph) is
// ... implementation defined ...
```

Erzeugt ein neues Anzeigefenster mit dem Namen `Window_Name`. Wendet den Layout-Algorithmus `Layout_Algo` auf `Content` an und fügt die so erzeugte neue Selektion in das neue Anzeigefenster ein.

Falls `Window_Name` keinen gültigen Namen für ein Anzeigefenster enthält oder falls bereits ein Anzeigefenster dieses Namens existiert, so ist die Ausführung dieses Scripts fehlerhaft. Falls `Layout_Algo` kein gültiger Layout-Algorithmus ist, so ist die Ausführung dieses Scripts fehlerhaft.

```
Insert_Into_Window
  (Window_Name : string;
   Layout_Algo : string;
   New_Content : subgraph) is
// ... implementation defined ...
```

Wendet den Layout-Algorithmus `Layout_Algo` auf `Content` an und fügt die so erzeugte neue Selektion in das Anzeigefenster mit dem Namen `Window_Name` ein.

Falls kein Anzeigefenster des Namens `Window_Name` existiert, so ist die Ausführung dieses Scripts fehlerhaft. Falls `Layout_Algo` kein gültiger Layout-Algorithmus ist, so ist die Ausführung dieses Scripts fehlerhaft.

```
Set_Selection
  (Window_Name    : string;
   Selection_Name : string;
   Value          : selection) is
// ... implementation definded ...
```

Falls in dem Anzeigefenster `Window_Name` noch keine Selektion mit Name `Selection_Name` existiert, dann erzeugt diese Selektion. Setzt den Inhalt der Selektion mit Name `Selection_Name` im Anzeigefenster `Window_Name` auf `Value`.

Falls kein Anzeigefenster mit Name `Window_Name` existiert oder falls `Selection_Name` kein gültiger Name für Selektionen ist, so ist die Ausführung dieses Scripts fehlerhaft.

```
Set_Subgraph
  (Subgraph_Name : string;
   Value         : subgraph) is
// ... implementation defined ...
```

Falls noch kein IML-Teilgraph mit Name `Subgraph_Name` existiert, dann erzeugt diesen IML-Teilgraphen. Setzt den Inhalt des IML-Teilgraphen mit Name `Subgraph_Name` auf `Value`.

Falls `Subgraph_Name` kein gültiger Name für IML-Teilgraphen ist, so ist die Ausführung dieses Scripts fehlerhaft.

11 Layoutalgorithmen

Hier werden die Layoutalgorithmen von GIANT beschrieben, mittels derer das Layout von Fenster-Knoten innerhalb von Anzeigefenstern automatisch berechnet werden kann.

GIANT bietet zwei verschiedene Layoutalgorithmen an:

1. Treelayout
2. Matrixlayout

11.1 Treelayout

11.1.1 Parameter

1. **IML-Teilgraph**
   Der IML-Teilgraph, der layoutet werden soll.
2. **Kantenklassen**
   Zu beachtete Kantenklassen können angegeben werden. Ist dies nicht der Fall, werden alle berücksichtigt.
3. **Zielposition**
   Die Position, an die der Mittelpunkt des Wurzelknotens des Baums positioniert werden soll.

11.1.2 Beschreibung

Das Treelayout basiert auf Walkers Algorithmus. Dabei werden folgende Eigenschaften erreicht:

1. Alle Kinder eines Knotens sind auf der gleichen Ebene
2. Alle Kinder eines Knotens haben den gleichen horizontalen Abstand zueinander

Als Wurzelknoten wird der erste Knoten des übergebenen IML-Teilgraphen gewählt. Die Reihenfolge der Kinder von der `IML_Reflection` übernommen. In der Grundfunktionalität werden alle Kanten betrachtet.

Der vertikale Abstand von Ebene zu Ebene richtet sich nach der Höhe des höchsten Knotens (siehe 15.1.1).

Falls der übergebene IML-Teilgraph keinen Baum darstellt, werden die Knoten in die Ebene eingegliedert, die bei einer Tiefensuche als erstes erreicht wird.

Zerfällt der IML-Teilgraph in nicht-zusammenhängede Komponenten, so wird das Layout für jede dieser Komponente durchgeführt.

Als Erweiterung kann der Algorithmus nur bestimmte Kanten betrachten. Dabei werden im ersten Lauf nur die Knoten betrachtet, die durch die gegebenen Kanten erreicht werden können. Im zweiten Lauf werden Knoten, die durch die gegebenen Kanten nicht erreicht werden konnten, im Matrixlayout (siehe 11.2) rechts von dem letzten Baum angeordnet.

11.2 Matrixlayout

11.2.1 Parameter

1. **IML-Teilgraph**
   Der IML-Teilgraph, der layoutet werden soll
2. **Zielposition**
   Die Position, an der die linke obere Ecke des Quadrats liegen soll.

11.2.2 Beschreibung

Im Matrixlayout werden die Knoten in einem Rechteck ausgerichtet. Dabei ist die Zahl der Knoten pro Zeile und Spalte gleich. Die Breite des Rechtecks wird mittels der abgerundeten Wurzel der Anzahl der zu layouteten Knoten bestimmt; die Höhe ergibt sich automatisch. Die Knoten werden mittels einer Breitensuche besucht und in dieser Besuchsreihenfolge in das Rechteck zeilenweise eingefügt. Dabei hat die obere Begrenzung jedes Knotens einer Zeile die selbe Y-Koordinate. Der vertikale Abstand von Zeile zu Zeile richtet sich nach der Höhe des höchsten Knotens (siehe 15.1.1).

12 GIANT Projektverwaltung

In diesem Kapitel wird beschrieben, wie persistente Arbeitsergebnisse von GIANT strukturiert und gespeichert werden. Arbeitsergebnisse sind z.B., vom Benutzer erzeugte Anzeigefenster mit visualisierten Knoten eines IML-Graphen.

12.1 Persistenz der Projekte

GIANT speichert persistente Informationen in sogenannten Projekten. Ein Projekt fasst alle Arbeitsergebnisse, die der Benutzer mittels GIANT für einen IML-Graphen erzeugen kann, wie z.B. Anzeigefenster und IML-Teilgraphen, logisch zusammen.
Hierbei sind die anschließend beschriebenen Konventionen zu beachten:

1. Die gesamte in dieser Spezifikation beschriebene Funktionalität, die sich auf das Laden bestehender und das Öffnen neuer Projekte während der Laufzeit von GIANT bezieht, kann nur implementiert werden, falls die Reflektion zur Bauhaus IML-Graph Bibliothek dies unterstützt. Der Kunde muss hierzu in der Reflektion die Möglichkeit vorsehen, IML-Graphen zur Laufzeit laden und bereits geladene IML-Graphen zur Laufzeit aus dem Speicher entfernen zu können.
2. Zur sicheren Identifikation der IML-Graph Datei, die zu einem Projekt gehört, muss über die Reflektion eine eindeutige Prüfsumme für jeden IML-Graphen berechnet werden können.
3. Wird während des Betriebs von GIANT ein neues Projekt angelegt, so erhält es automatisch eine Projektdatei.
4. Ein Projekt besteht aus einem Verweis auf eine IML-Graph-Datei, auf die sich die gespeicherten Informationen beziehen, sowie aus den gespeicherten Informationen für IML-Teilgraphen, Anzeigefenster und Knoten-Annotationen.
5. Der Name eines bereits angelegten Projektes kann mit den Mitteln von GIANT nicht geändert werden (außer dadurch, dass man das Projekt unter neuem Namen neu speichert).
6. Der Benutzer kann beliebig viele Projekte anlegen.
7. In GIANT darf immer nur ein Projekt gleichzeitig geöffnet sein.
8. Während der Arbeit mit GIANT kann jederzeit ein vorhandenes Projekt geladen oder ein neues Projekt angelegt werden.
9. Ein nicht gerade in GIANT geöffnetes Projekt kann vom Benutzer beliebig modifiziert werden. GIANT unterstützt dies durch den Einsatz von XML. Insbesondere kann der Benutzer die Projektdatei (siehe 12.1.2) modifizieren und so dem Projekt z.B. neue Verwaltungsdateien (siehe 12.1.6, 12.1.4 und 12.1.5) hinzufügen oder die zugehörige IML-Graph Datei ändern.

Das Fehlerverhalten von GIANT gegenüber Fehlern aufgrund manueller Änderungen des Benutzers an einem Projekt (Editieren der Projektdatei und Ändern der Verwaltungsdateien) ist undefiniert.

10. Um Inkonsistenzen und Konflikte zu vermeiden sollten alle Dateien, die zu einem Projekt gehören, im Projektverzeichnis liegen (GIANT macht dies automatisch immer so). Man sollte in die Projektdatei also manuell keine Referenzen auf Verwaltungsdateien außerhalb der Projektverzeichnisses eintragen; verboten ist dies allerdings nicht. Das Verhalten von GIANT im Fehlerfall bleibt aber diesbezüglich undefiniert.

12.1.1 Das Projektverzeichnis

1. Sämtliche Dateien, die zu einem Projekt gehörige Informationen enthalten, befinden sich in diesem Verzeichnis. GIANT speichert alle Verwaltungsdateien (siehe 12.1.6, 12.1.4 und 12.1.5) automatisch im Projektverzeichnis.
2. In einem Projektverzeichnis darf nur ein Projekt abgelegt werden.

12.1.2 Die Projektdatei

1. Die Projektdatei liegt als XML-Datei vor.
2. Die Projektdatei befindet sich im Projektverzeichnis und enthält Informationen, die zur Identifikation des zu einem Projekt gehörenden IML-Graphen nötig sind.
3. Der Name der Projektdatei entspricht dem Namen des Projektes.
4. Die Projektdatei enthält Referenzen zu allen Dateien, die Bestandteil des Projektes sind (Verwaltungsdateien für IML-Teilgraphen und Anzeigefenster, sowie die Verwaltungsdatei für Knoten-Annotationen). Verwaltungsdateien, die zwar im Projektverzeichnis liegen, zu denen aber keine Referenz in der Projektdatei existiert, gehören nicht zum Projekt.
5. Der Pfad zu der Datei, die den IML-Graphen enthält, ist in der Projektdatei gespeichert.

12.1.3 Prüfung der IML-Graph Datei

Beim Laden eines Projektes wird überprüft, ob die in der Projektdatei gespeicherte Prüfsumme der Prüfsumme der zu ladenden IML-Graph Datei entspricht. Das Verhalten von GIANT für den Fall, dass eine IML-Graph Datei geladen wird, die zwar die passende Prüfsumme hat, aber nicht den IML-Graphen enthält, der dem Projekt eigentlich zu Grunde liegt, ist undefiniert.

12.1.4 Verwaltungsdateien für Anzeigefenster

1. Diese Datei ist eine Binärdatei.
2. Zu jedem Anzeigefenster gibt es eine Verwaltungsdatei.

3. Diese Verwaltungsdatei enthält alle Informationen zur kompletten Rekonstruktion eines Anzeigefensters. Alle Informationen werden in binärer Form gespeichert.

Insbesondere werden folgende Informationen gespeichert:

1. Der komplette Anzeigeinhalt (alle visualisierten Knoten und Kanten mit Position).
2. Alle Pins (gespeicherte sichtbare Anzeigeinhalte).
3. Alle Selektionen des Anzeigefensters.

Der für das Anzeigefenster gewählte Visualisierungsstil wird zwar mit gespeichert, da die Visualisierungsstile völlig unabhängig von den Projekten über Konfigurationsdateien realisiert werden, kann nicht garantiert werden, dass der gewünschte Stiel beim Laden des Projektes auch wieder gefunden wird, in diesem Fall wird dann ein „Standard-Stil“ verwendet.

12.1.5 Verwaltungsdateien für IML-Teilgraphen

1. Diese Datei ist eine Binärdatei.
2. Für jeden IML-Teilgraphen gibt es eine eigene Verwaltungsdatei.
3. Sämtliche erzeugten IML-Teilgraphen werden in Verwaltungsdateien in binärer Form gespeichert.

12.1.6 Die Verwaltungsdatei für Knoten-Annotationen

1. Diese Datei liegt als XML Datei vor.
2. Alle Knoten-Annotationen werden in einer Verwaltungsdatei gespeichert (von dieser Datei darf es immer nur eine im Projekt geben).

12.2 Grundlegendes Verhalten von GIANT beim Speichern von Projekten

12.2.1 „Alles Speichern“

Diese Funktionalität wird von entsprechenden UseCases genutzt. Hierbei werden alle Anzeigefenster, IML-Teilgraphen und Knoten-Annotationen in die Verwaltungsdateien geschrieben, wobei auch alle Änderungen an noch offenen Anzeigefenstern berücksichtigt werden; d.h. der aktuelle Zustand (z.B. die Position der IML-Knoten) der offenen Anzeigefenster wird ohne explizite Rückfrage beim Benutzer in die Verwaltungsdateien geschrieben. Der Zustand der persistenten Informationen in den Verwaltungsdateien entspricht nach Ausführung von „Alles Speichern“ exakt dem Zustand des Projektes innerhalb von GIANT.

12.2.2 Persistenz von Anzeigefenstern

1. Beim Speichern von Anzeigefenstern in Verwaltungsdateien werden für jedes Anzeigefenster alle Informationen gespeichert, die nötig sind um den Zustand des Anzeigefensters zu rekonstruieren, der möglichst exakt dem Zustand bei der Speicherung des Anzeigefensters entspricht. Insbesondere werden hierbei die folgenden Bestandteile des Anzeigefensters gespeichert:

   Alle Selektionen und Pins des Anzeigefensters.

   Der exakte Zustand jeder Selektion; d.h. ob und wie die Selektion hervorgehoben ist.

   Die Position der Fenster-Knoten und Fenster-Kanten des Anzeigefensters.

2. Für den gewählten Visualisierungsstil des Anzeigefensters wird nur der Name gespeichert, da die Visualisierungsstile über die Konfigurationsdateien beliebig editiert und auch gelöscht werden können. Wird beim Laden eines Anzeigefensters aus der Verwaltungsdatei kein Visualisierungstil gefunden, dessen Name dem gespeicherten entspricht, so wird für das Anzeigefenster statt dessen ein Standard-Visualisierungsstil verwendet.

3. Sämtliche dem Projekt bekannten Anzeigefenster (alle Anzeigefenster zu denen es eine entsprechende Verwaltungsdatei gibt) werden in der Liste über die Anzeigefenster angezeigt.

4. Zu jedem Anzeigefenster eines Projektes gibt es eine Verwaltungsdatei. Bei neu erzeugten Anzeigefenstern wird diese Verwaltungsdatei beim ersten Speichern angelegt.

5. Wird ein geöffnetes Anzeigefenster geschlossen, so fragt GIANT nach, ob es eventuelle Änderungen speichern soll oder nicht. Falls ja, werden eventuelle Änderungen in die für das Anzeigefenster vorhandene Verwaltungsdatei geschrieben, anderenfalls bleibt der Zustand des Anzeigefensters nach der letzten Speicherung vorhanden (die zugehörige Verwaltungsdatei wird nicht verändert).

6. Modifikationen (z.B. das Verschieben von Knoten) auf einem Anzeigefenster werden nicht automatisch nach deren Durchführung gespeichert (so kann notfalls ein Rückgängig durchgeführt werden).

12.2.3 Persistenz von IML-Teilgraphen

1. Alle in einem Projekt bereits vorhandenen IML-Teilgraphen werden in einer entsprechenden Liste angezeigt.

2. Neu erzeugte IML-Teilgraphen und Änderungen an bestehenden IML-Teilgraphen können nur über „Alles Speichern“ (siehe oben) gespeichert werden.

3. Wird das Programm beendet, ohne das zuvor „Alles Speichern“ ausgeführt worden ist, so gehen alle nicht gespeicherten Informationen zu den IML-Teilgraphen verloren (alle zwischenzeitlich ausgeführten Modifikationen und alle zwischenzeitlich neu erzeugten IML-Teilgraphen). Der Zustand nach dem letzten Speichern bleibt dann erhalten.

4. Modifikationen an bestehenden IML-Teilgraphen werden nicht automatisch gespeichert. Zu neu erzeugten IML-Teilgraphen wird nicht automatisch eine Verwaltungsdatei erzeugt.

5. IML-Teilgraphen können gelöscht werden. Falls vorhanden, wird dann auch die entsprechende Verwaltungsdatei ebenfalls sofort gelöscht.

12.2.4 Persistenz von Knoten-Annotationen

Hier wird beschrieben, wie Knoten-Annotationen von GIANT persistent verwaltet werden.

1. Änderungen bestehender oder neu erzeugte Knoten-Annotationen werden nur über die Funktionalität „Alles Speichern“ (siehe 12.2.1) in die Verwaltungsdatei für Knoten-Annotationen (siehe 12.1.6) geschrieben.
2. Einmal erzeugte Knoten-Annotationen werden jedem IML-Knoten mit der entsprechenden ID zugeordnet, unabhängig davon in welchem Anzeigefenster diese visualisiert sind. Ein Knoten kann auch annotiert sein, wenn er in keinem Anzeigefenster visualisiert ist. Wird ein annotierter Knoten gelöscht (aus einem Anzeigefenster entfernt), so wird der dazu vorhandene Eintrag für die Annotation in der Verwaltungsdatei nicht automatisch mit gelöscht.

13 Konfiguration von GIANT

Hier wird beschrieben, wie benutzerdefinierbare Einstellungen von GIANT gespeichert und verwaltet werden. In diesem Kapitel wird insbesondere spezifiziert, welche Einstellungen der Benutzer in welchen Konfigurationsdateien vornehmen kann.

13.1 Allgemeines

1. Sämtliche konfigurierbaren Einstellungen werden in XML-Dateien vorgenommen.
2. Es gibt genau eine „globale Konfigurationsdatei“. Zudem kann es beliebig viele weitere XML-Dateien zur Konfiguration der weiter unten beschriebenen Visualisierungsstile geben.
   Diese Dateien liegen in einem von GIANT fest vorgegebenen Verzeichnis.
3. Die unter 2 beschriebenen Dateien zu Konfiguration können vom Benutzer auch in einem separaten Unterverzeichnis seines Home-Verzeichnisses abgelegt werden. Findet GIANT beim Start ein derartiges Verzeichnis, so werden die Einstellungen aus der dort vorgefundenen Konfigurationsdatei, sowie die dort vorgefundenen Visualisierungsstile geladen.

13.2 Die globale Konfigurationsdatei

Diese Datei ist in XML verfasst. In Ihr können die anschließend beschriebenen Einstellungen vorgenommen werden.

13.2.1 Verweise auf GQSL-Skript-Makros

Es können GQSL-Skript-Makros definiert werden. Jeder Makro wird durch einen eindeutigen Namen und durch einen Verweis auf die Datei, welche das GQSL-Skript enthält, spezifiziert. Der Makro kann dann aus einem Popup-Menü heraus aufgerufen werden, wobei dann das in der Datei stehende GQSL Skript ausgeführt wird.

13.2.2 Farbe von Hervorhebungen

Einstellung der Farben, mittels derer Selektionen oder IML-Teilgraphen hervorgehoben werden.

1. Eine beliebige Farbe für die aktuelle Selektion.
2. Beliebige Farben für das Hervorheben weiterer Selektionen.

3. Beliebige Farben für das Hervorheben von IML-Teilgraphen.

13.2.3 Editor zur Anzeige des Quellcodes

Der Benutzer legt in der globalen Konfigurationsdatei fest, mit welchem Editor der zu IML-Knoten korrespondierende Quellcode automatisch angezeigt wird.

13.3 Visualisierungsstile

Der Benutzer kann beliebig viele Visualisierungsstile definieren. Mittels dieser Visualisierungsstile kann die Darstellung von Fenster-Knoten und Fenster-Kanten dynamisch zur Laufzeit geändert werden (siehe auch 4.17). Weitere Informationen zur Visualisierung von Fenster-Knoten und Fenster-Kanten sind unter den Abschnitten 15.2 und 15.1 zu finden.
Für die Visualisierungsstile gelten die folgenden Konventionen:

1. Für jeden Visualisierungsstil muss es eine entsprechende XML-Datei geben.
2. Ein Visualisierungsstil beschreibt, wie die Knoten und Kanten des IML-Graphen innerhalb eines Anzeigefensters graphisch dargestellt werden können.
3. In einem Visualisierungsstil können klassenspezifische Einstellungen vorgenommen werden, die nur für Knoten und Kanten gelten, die zu den entsprechenden Klassen gehören. Bei jeder klassenspezifischen Einstellung für Knoten und Kanten kann daher eine Liste der betroffenen Knoten- und Kantenklassen angegeben werden, für die diese Einstellungen gelten sollen.
4. Wird eine Kanten- oder eine Knotenklasse innerhalb eines Visualisierungsstils mehreren klassenspezifischen Einstellungen zugeordnet, führt dies zu keinem Fehler, es bleibt aber unspezifiziert, welche Einstellung tatsächlich genommen wird.
5. Es gibt jeweils eine Standard-Einstellung, die für alle Knoten- und Kantenklassen genommen wird, für die keine extra klassenspezifischen Einstellungen vorgenommen wurden.
6. Es gibt immer einen von GIANT fest vorgegeben Standard-Visualisierungsstil. Dieser Visualisierungsstil wird für neu erzeugte Anzeigefenster genommen. Kann GIANT den einem Anzeigefenster zugewiesenen Visualisierungsstil nicht finden (falls die zugehörige XML-Datei fehlt) wird ebenfalls der Standard-Visualisierungsstil genommen.

13.3.1 Name des Visualisierungsstils

Jeder Visualisierungsstil erhält einen Namen. Unter diesem Namen ist der Visualisierungsstil für jedes Anzeigefenster einzeln auswählbar.

13.3.2 Einstellungen innerhalb des Visualisierungsstils

1. Die Hintergrundfarbe im Anzeigefenster.

13.3.3 Klassenspezifische Einstellungen für Knoten

Folgende Einstellungen können für Knotenklassen vorgenommen werden. Es muss eine Standard-Einstellung erstellt werden, die für alle Knotenklassen angewendet wird, für die nichts anderes definiert ist.

1. Ein Icon für die Knotenklasse (Verweis auf eine entsprechende Bilddatei). Das Icon muss im Pixmap Format bei 16*16 Pixel vorliegen.
2. Eine Liste der Attribute der Knotenklasse, welche direkt innerhalb des Anzeigefensters in dem „Rechteck“ für den Knoten dargestellt werden sollen.
3. Die Füllfarbe des „Rechtecks“ in welchem die Attribute zu dem Knoten dargestellt werden.
4. Die Rahmenfarbe des „Rechtecks“.

13.3.4 Klassenspezifische Einstellungen für Kanten

Folgende Einstellungen können für Kantenklassen vorgenommen werden. Es muss eine Standard-Einstellung erstellt werden, die für alle Kantenklassen angewendet wird, für die nichts anderes definiert ist.

1. Die Farbe der Kante.
2. Die Art der Linie der Kante (normal, gestrichelt).
3. Ob die Kante mit ihrer Kantenklasse beschriftet werden soll oder nicht.

13.4 Skript-Dateien

Eine Textdatei, die genau ein GQSL Skript enthält – also genau ein gemäß der Grammatik zulässiges Wort. Auf diese Art und Weise können komplexe Skripte gespeichert und wiederverwendet werden. Geladen werden können solche Dateien beim Kommandozeilenaufruf (siehe 4.2) oder im Dialog zur Eingabe eine Skripts (vgl. UseCase 9.2). Für eine exakte Spezifikation der GQSL siehe Kapitel 10.

14 Beschreibung der GUI

In diesem Kapitel werden die Dialoge und Menüs der graphischen Benutzerschnittstelle von GIANT beschrieben. Die Funktionalität dieser Dialoge wird im Detail bei den jeweiligen UseCases beschrieben.

14.1 Über die Benutzeroberfläche

1. GIANT besitzt eine graphische Benutzeroberfläche, die per Maus zu bedienen ist. Ausgewählte Funktionen können auch per Tastatur ausgelöst werden.
2. Die Interaktionssprache mit dem Benutzer ist Englisch.

14.2 Über dieses Kapitel

Hier werden alle Elemente der GUI beschrieben. Sofern nicht anders angegeben, sind hintereinander angegebene Elemente (z.B. Listeneinträge) immer von links nach rechts oder von oben nach unten beschrieben. Alle Ausgaben sind immer linksbündig formatiert, sofern nicht anders vermerkt.

14.3 Main Window

Jede Instanz von GIANT hat genau ein Hauptfenster. Im Hauptfenster (MAIN_WINDOW) werden vorhandene Anzeigefenster und IML-Teilgraphen angezeigt. Über Popup-Menüs können IML-Teilgraphen und Anzeigefenster manipuliert werden.

Im Fenster befinden sich zwei Listen, WINDOW_LIST (2 Spalten), und SUBGRAPH_LIST (4 Spalten).

14.3.1 Kopfzeile

In der Kopfzeile wird „GIANT - <Projektname>“ dargestellt.

14.3.2 Menüleiste (Main Window)

In der Menüleiste befinden sich folgende Einträge:

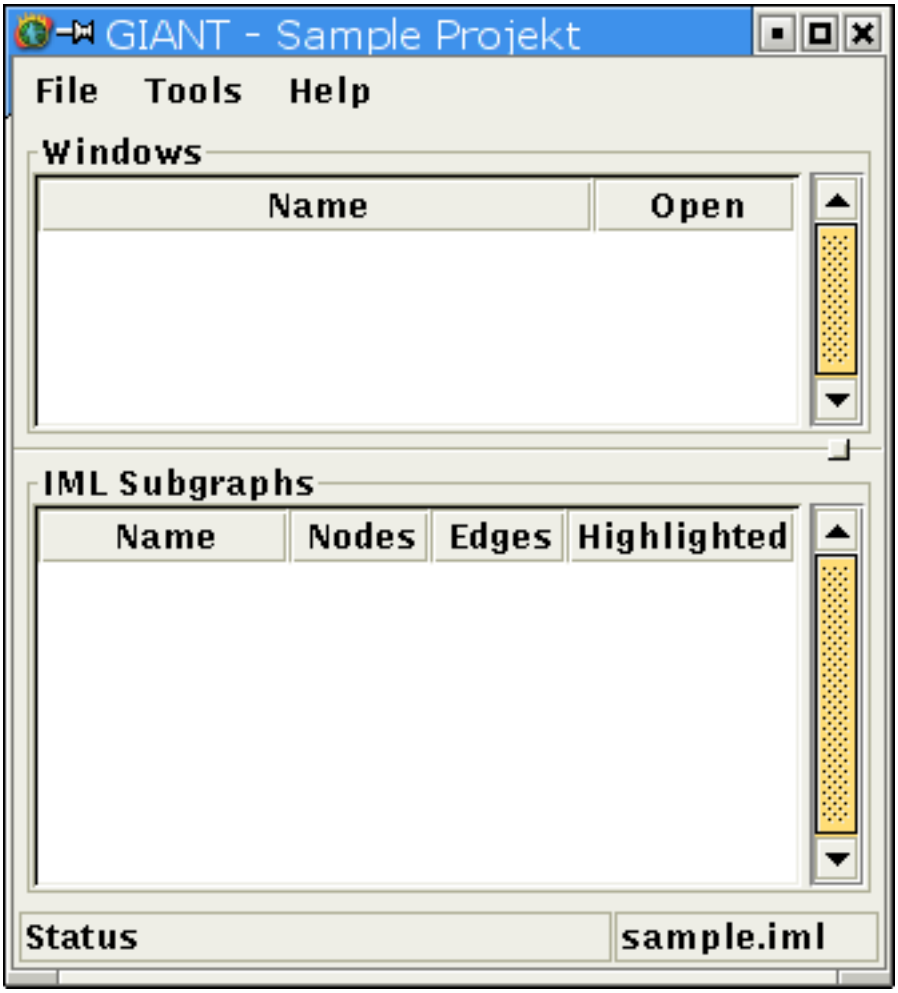


Abbildung 14.1: Main-Window

14.3.2.1 Untermenü File

1. New Project
2. Load Project
3. Save Project
4. Save Project As...
5. Quit

14.3.2.2 Untermenü Tools

1. Delete Unreferenced Annotations
2. Execute GQSL Query

14.3.2.3 Untermenü Info

1. About GIANT

14.3.3 Statuszeile

Die Statuszeile befindet sich ganz unten im Fenster. In der Statustzeile Informationen zum aktuellen Zustand von GIANT dargestellt.

14.3.4 Window List

Im Fenster befindet sich zuoberst eine Liste WINDOW_LIST, welche alle Fenster des Projektes (offen und geschlossen anzeigt).

14.3.4.1 Inhalt Window List

WINDOW_LIST hat die Spalten WINDOW_NAME „Name“ und WINDOW_OPEN „Open“. In WINDOW_NAME wird der Name aller Anzeigefenster dargestellt, in WINDOW_OPEN befindet sich entweder ein Strich-Symbol (Fenster nicht geöffnet) oder ein Häkchensymbol (Fenster geöffnet).

14.3.4.2 Popup-Menü Window List

Beim Rechtsklick auf WINDOW_LIST öffnet sich ein Popup-Menü mit folgenden Menüpunkten:

1. New Window
2. Open Window
3. Close Window
4. Save Window
5. Delete Window
6. Rename Window

14.3.5 Subgraph List

Unter der WINDOW_LIST befindet sich eine Liste SUBGRAPH_LIST.

14.3.5.1 Inhalt Subgraph List

SUBGRAPH_LIST hat die Spalten

1. Name (SUBGRAPH_NAME)

2. Nodes (SUBGRAPH_NODES)
3. Edges (SUBGRAPH_EDGES)
4. Highlighted (SUBGRAPH_HIGHLIGHT_COL).

In Subgraph Name stehen die Namen der existierenden Subgraphen, in Nodes die Anzahl der Knoten des Subgraphen, in Edges die Anzahl der Kanten und in Highlight Color ein Kästchen mit der Farbe, die verwendet wird, wenn dieser Subgraph markiert wird.

14.3.5.2 Popup-Menü Subgraph List

Beim Rechtsklick auf SUBGRAPH_LIST öffnet sich ein Popup-Menü mit folgenden Menüpunkten:

1. Highlight (Menü)
    a) Color 1
    b) Color 2
    c) Color 3
2. Unhighlight In All Windows
3. Copy IML Subgraph
4. Create Window Selection
5. Delete IML Subgraph
6. IML Subgraph Set Operation

14.4 Anzeigefenster

Im Programm kann es mehrere Anzeigefenster geben. Ein Anzeigefenster besteht aus folgenden Elementen:

In Anzeigefenstern (VISUALIZATION_WINDOW) werden Graphen visualisiert. Sie erscheinen im großen quadratischen rechten Teil (VIS_PANE) des Fensters. Die VIS_PANE enthält den Anzeigeinhalt des Fensters. Im linken Viertel des Fensters, der Toolbar (VISUALIZATION_WINDOW_TOOLBAR) befinden sich untereinander die MiniMap (MINIMAP), Pinliste (PIN_LIST), Selektionsliste (SELECTION_LIST), die Stilauswahl-Combobox (STYLE_COMBO) und die Zoom-Kontrolle.

14.4.1 MiniMap

In der MiniMap wird der Anzeigeinhalt verkleinert dargestellt. Die MiniMap wird von einem Rahmen umschlossen. Der sichtbare Anzeigeinhalt wird durch einen kleineren Rahmen repräsentiert, der innerhalb der MiniMap durch Mausklicks versetzt werden kann. Je nach Zoomstufe ist dieser Rahmen größer oder kleiner. Wird der Rahmen versetzt, der sichtbare Anzeigeinhalt im VIS_PANE entsprechend angepasst.

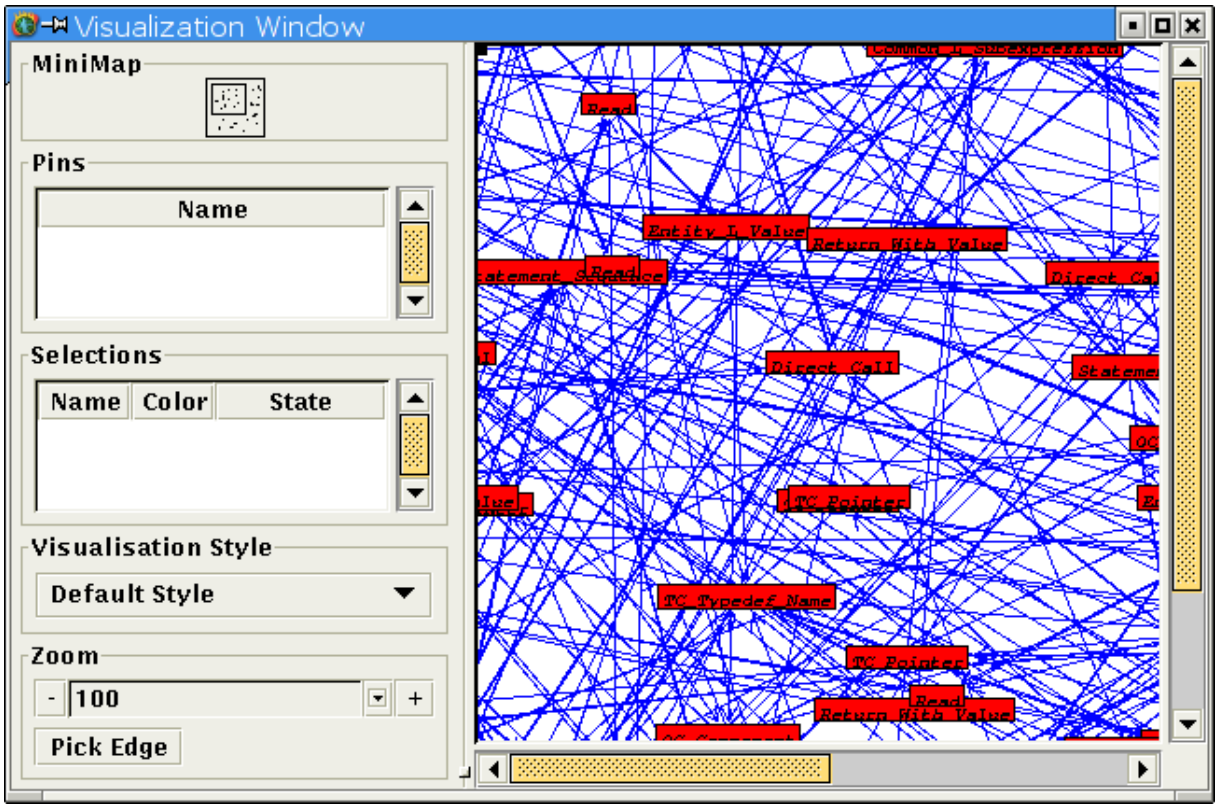


Abbildung 14.2: Graph-Window

14.4.2 Visualisierung der Knoten

In der VIS_PANE werden die Fenster-Knoten und Fenster-Kanten angezeigt.

Wenn auf einem Knoten die rechte Maustaste geklickt wird, öffnet sich folgendes Popup-Menü:

14.4.2.1 Node-Popup-Menü

1. Show Node Info Window
2. Show Corresponding Source Code
3. Annotation (Menü)
    a) Add
    b) Change
    c) Delete
4. Show Node Annotation

Wenn in der VIS_PANE mit der rechten Maustaste auf eine Stelle geklickt wird an der kein Knoten liegt, öffnet sich das folgende Popup-Menü. Die Einträge dieses Menüs sind auch im Node-Popup-Menü (siehe 14.4.2.1) sichtbar.

1. Make Room
2. New Pin

14.4.3 Pins

In der Pinliste (PIN_LIST) können Pins festgelegt werden.

Im Popup-Menü der Pinliste befinden sich folgende Menüpunkte:

1. Focus Pin (PIN_JUMP)
2. Delete Pin (PIN_DELETE)

14.4.4 Selektionsauswahlliste

Die Selektionsauswahlliste hat die Spalten „Name“ mit dem Namen der Selektion, „Color“ mit der Farbe der Selektion und „State“ mit dem Status der Selektion (hervorgehoben oder nicht hervorgehoben).

Im Popup-Menü der Selektionsauswahlliste (SELECTION_LIST) befinden sich folgende Einträge:

1. New Selection
2. Move Selection
3. Show Selection

4. Hide Selection
5. Layout Selection
6. Create New IML Subgraph From This Selection
7. Highlight Selection (Menü)
    a) Color 1
    b) Color 2
    c) Color 3
8. Zoom To Make Selection Fill Window
9. Copy Selection
10. Unhighlight Selection
11. Copy Selection Keeping Existing Layout
12. Copy Selection Changing Existing Layout
13. Delete Selection
14. Selection Set Operation (Union/Difference/Intersection)
15. Set Layout Algorithm For Selection

14.4.5 Stilauswahl-Combobox

In der Stilauswahl-Combobox (STYLE_CHOOSER) kann ein Anzeigestil festgelegt werden.

14.4.6 Zoom-Kontrolle

Die Zoom-Kontrolle besteht aus folgenden Elementen:

1. Button „-“ (ZOOM_MINUS)
2. Combobox ZOOM_CONTROL mit vorgefertigten Werten und Eingabemöglichkeit
3. Button „+“ (ZOOM_PLUS)
4. Button „Pick Edge“ (ZOOM_EDGE)

In der ComboBox ist neben den vorgegebenen Zoom Stufen noch der Eintrag „Fit in Window“.

14.4.7 Scrollbars

Mit den beiden Scrollbars am unteren und rechten Rand des Anzeigefensters kann der sichtbare Anzeigeinhalt des verändert werden.

14.5 Knoten-Informationsfenster

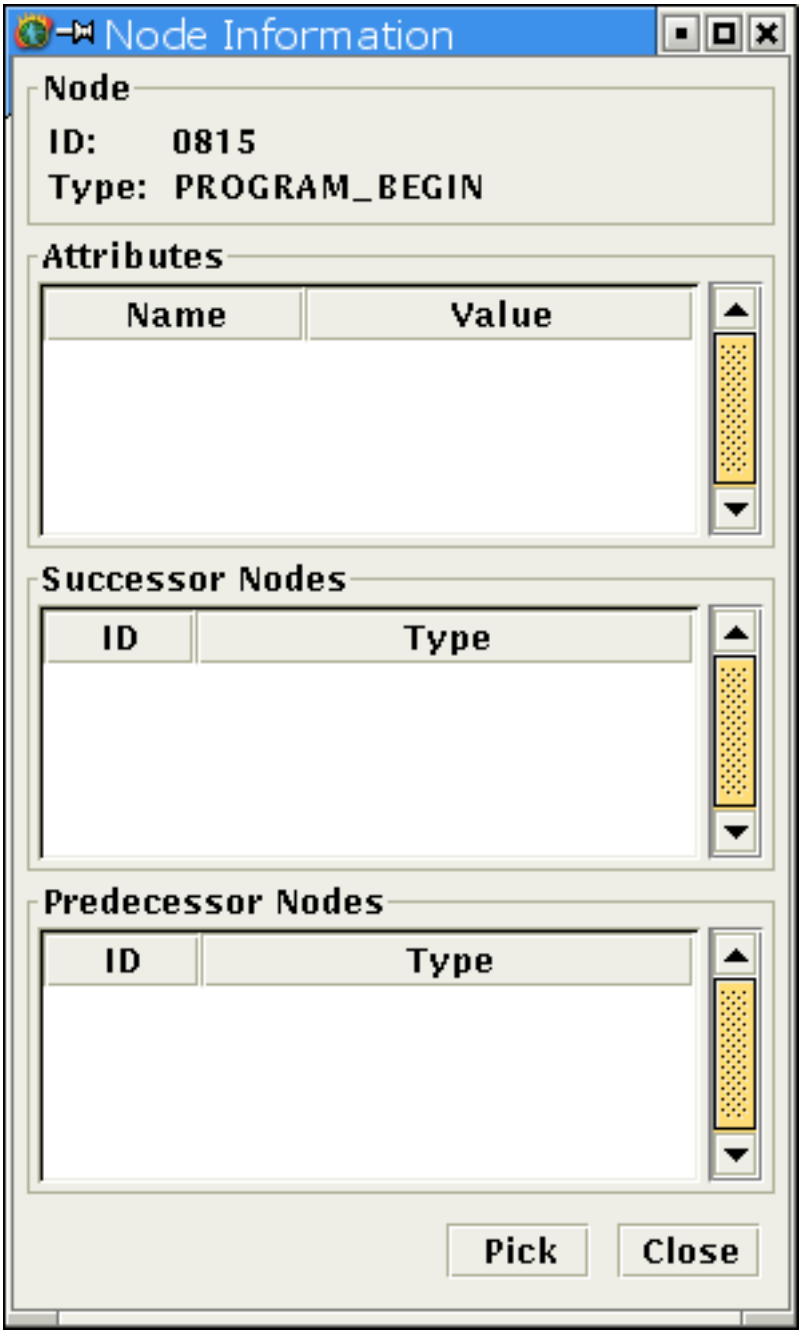


Abbildung 14.3: Node-Info-Window

In Knoten-Informationsfenstern (NODE_INFO) können nähere Informationen zu Knoten angezeigt werden.

Im Fenster wird die ID und der Typ des Knotens dargestellt, darunter befinden sich unteinander drei getrennt scrollbare Listen, ATTRIBUTES, SUCCESSOR_NODES und PREDECESSOR_NODES.

Unter den drei Listen befinden sich nebeneinander mittig zwei Buttons PICK und CLOSE. PICK dient zum Auswählen eines anderen Knotens zur Anzeige im dem Knoten-Informationsfenster.

14.5.1 Liste ATTRIBUTES

Die Liste ATTRIBUTES enthält die Attribute des Knotens und hat zwei Spalten:

1. Name (ATTRIBUTE)
2. Value (VALUE)

14.5.2 Liste SUCCESSOR_NODES

Die Liste SUCCESSOR_EDGES hat eine Zeile für jede vom Knoten abgehende Kante mit zwei Spalten:

1. ID (NODE_ID, Typ der abgehenden Kante)
2. Type (EDGE_TYPE, ID des Knotens, zu dem sie führt)

14.5.3 Liste PREDECESSOR_NODES

Die Liste PREDECESSOR_EDGES hat eine Zeile für jede vom Knoten abgehende Kante mit zwei Spalten:

1. ID (NODE_ID, ID des Knotens, von dem sie kommt)
2. Type (EDGE_TYPE, Typ der ankommenden Kante)

14.6 Skriptdialog

Der Skriptdialog QUERY_DIALOG hat zuoberst ein Textfeld QUERY_TEXT, in dem der Text des Skripts eingegeben werden kann.

Unter QUERY_TEXT befinden sich folgende Buttons:

1. Open... (QUERY_LOAD)
2. Save As... (QUERY_SAVE)
3. OK (QUERY_START)
4. Cancel (QUERY_CANCEL)

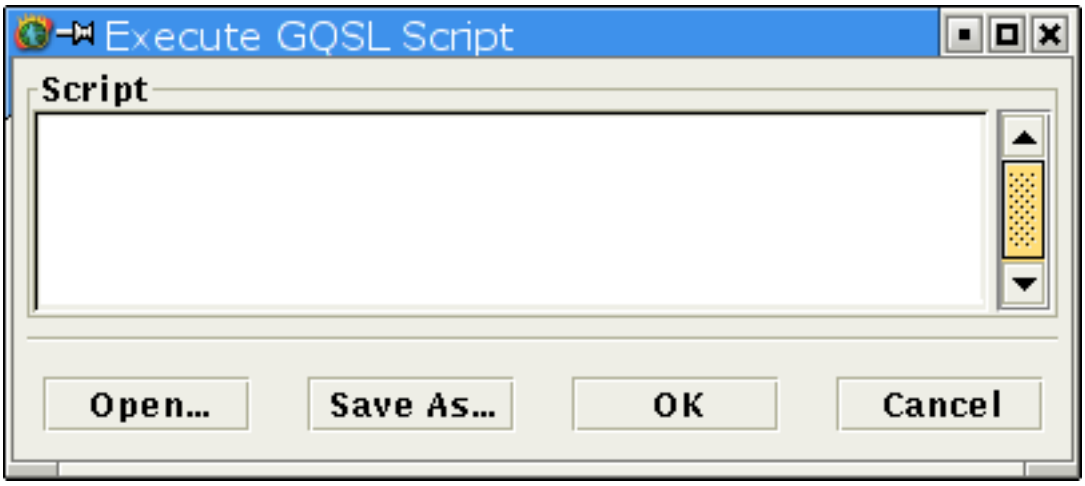


Abbildung 14.4: Query-Dialog

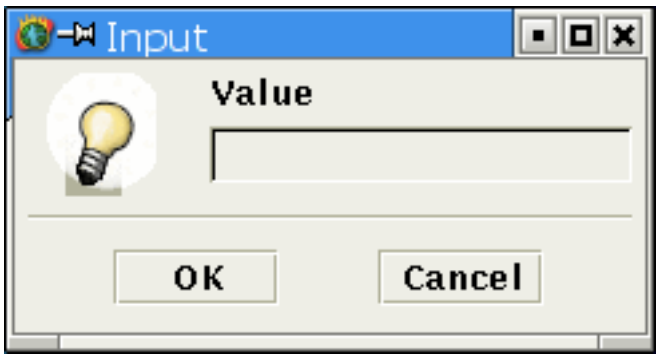


Abbildung 14.5: Dialog-Window

14.7 Allgemeiner Texteingabedialog

Ein allgemeiner Texteingabedialog ist ein Fenster DIALOG_WINDOW mit dem Titel „Input“. Im Fenster ist ein einzeiliges Textfeld TEXT_FIELD mit einem Prompt-Text TEXT_LABEL, zwei Buttons OK_BUTTON und CANCEL_BUTTON mit den Beschriftungen „OK“ und „Cancel“.

14.8 Set-Operation-Dialog

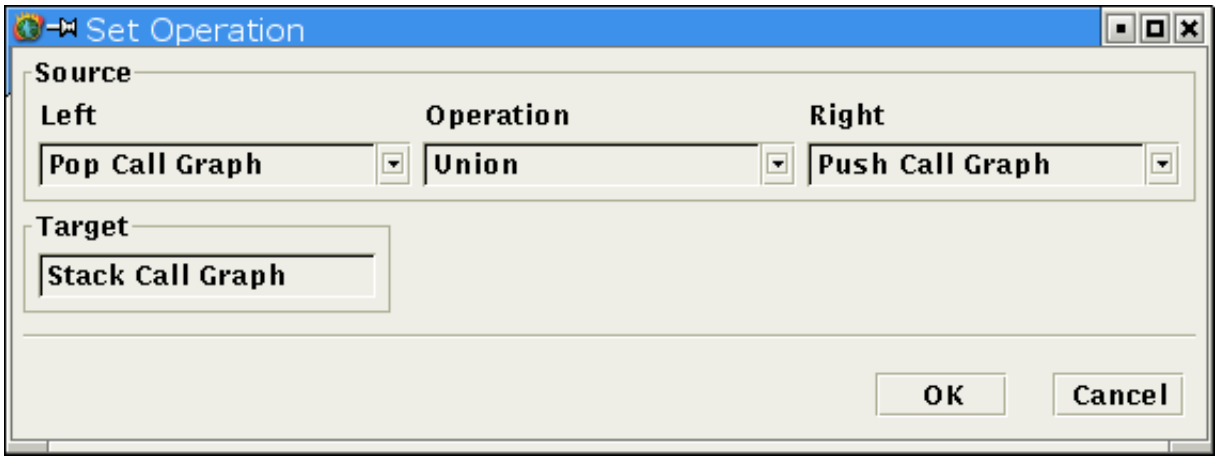


Abbildung 14.6: Set-Operation-Dialog

Dieser Dialog dient dazu, Mengenoperationen über Selektionen und IML-Teilgraphen durchzuführen. Die Mengenoperation lässt sich dann wie folgt beschreiben: LEFT_SOURCE <op> RIGHT_SOURCE := TARGET

Bestandteile des Dialoges sind:

1. LEFT_SOURCE Combobox
   Soll eine Mengenoperation für Selektionen ausgeführt werden, so können hier alle Selektionen des entsprechenden Anzeigefensters ausgewählt werden.
   Bei einer Mengenoperation über IML-Teilgraphen, werden alle IML-Teilgraphen des Projektes angezeigt.

2. <OP> Combobox
   Mengenoperation, die ausgeführt werden soll: Union, Difference oder Intersection.

3. RIGHT_SOURCE Combobox
   Soll eine Mengenoperation für Selektionen ausgeführt werden, so können hier alle Selektionen des entsprechenden Anzeigefensters ausgewählt werden.
   Bei einer Mengenoperation über IML-Teilgraphen, werden alle IML-Teilgraphen des Projektes angezeigt.

4. TARGET Textfield
   Name der neuen Selektion oder des neuen IML-Teilgraphen als Ergebnis der Mengenoperation.

14.9 Layoutalgorithmen Dialog

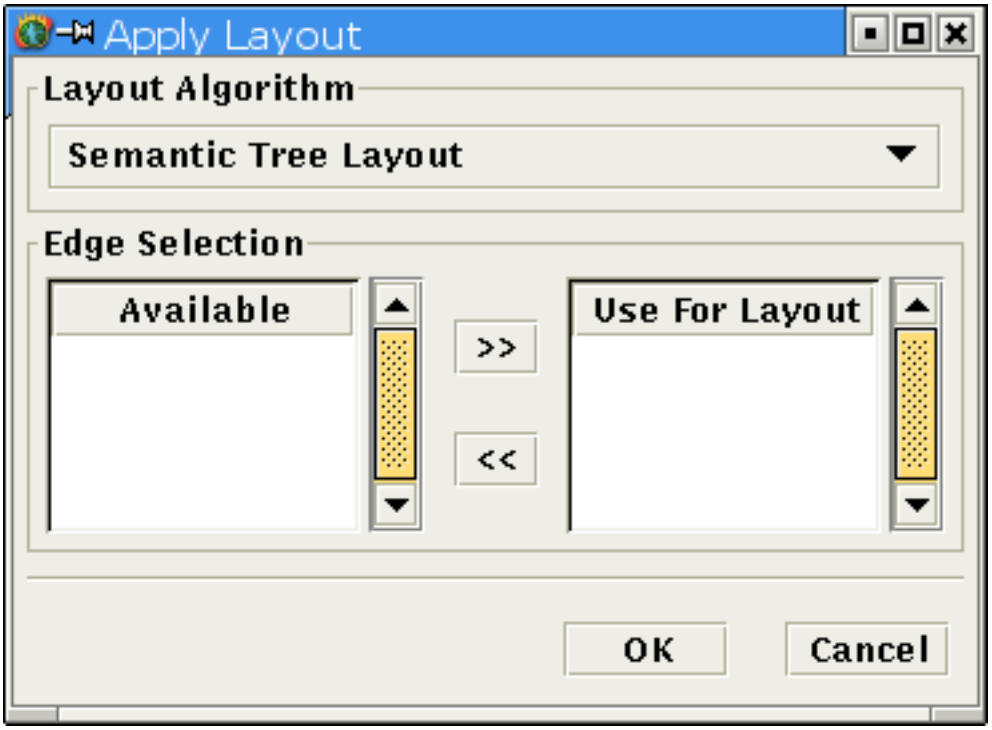


Abbildung 14.7: Layout-Algorithm-Dialog

Der Dialog Layoutalgorithmen bietet folgende Möglichkeiten:

1. Auswahl des Layoutalgorithmuses
2. Eingabe der zu berücksichtigenden Kanten bei semantischen Layouts
3. OK Button
4. Cancel Button

14.10 Dateneingabe

Beschreibung von verschiedenen GUI Elementen zur Eingabe von Daten.

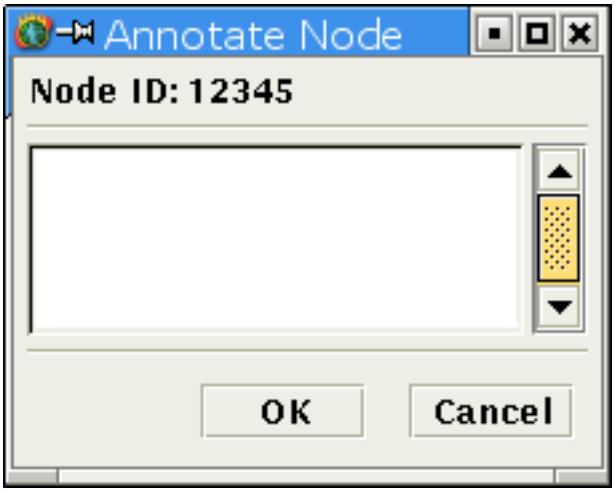


Abbildung 14.8: Node-Annotation-Dialog

14.10.1 Knoten-Annotations-Dialog

Der Knoten-Annotations-Dialog besteht aus einem Fenster mit einem Label mit der Knoten-ID und einem Textfeld für den Annotationstext. Mittels der beiden Buttons OK und Cancel können die Änderungen im Textfeld übernommen oder verworfen werden.

14.10.2 Platz Schaffen-Dialog

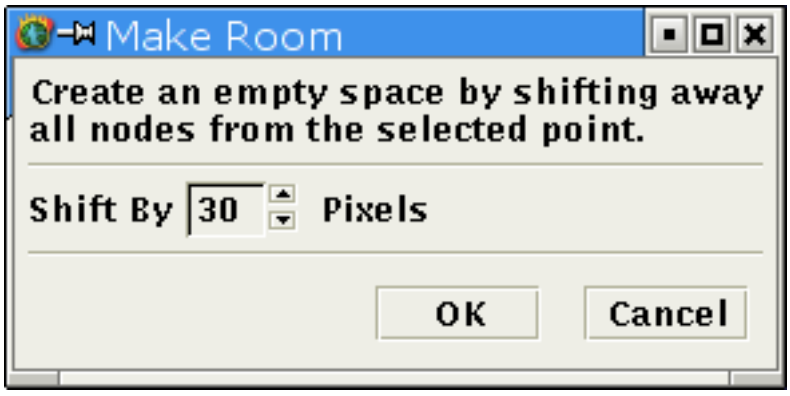


Abbildung 14.9: Make-Room-Dialog

Im Dialogfenster MAKE_ROOM wird in einem Textfeld als Zahl angegeben, um wieviel Pixel die Knoten an der vorher markierten Stelle auseinandergeschoben werden sollen. Mit dem Button OK kann bestätigt werden, mit Button Cancel abgebrochen werden.

Abbildung 14.10: Error-Window

14.11 Ausgabe von Fehlermeldungen

Ein allgemeiner Fehlerdialog ist ein Fenster ERROR_WINDOW mit dem Titel „Error“. Im Fenster ist ein Fehlertext Text ERROR_LABEL und ein Button OK_BUTTON mit der Beschriftung OK_LABEL. Standardmäßig ist der Button mit „OK“ beschriftet.

14.12 Sicherheitsabfrage

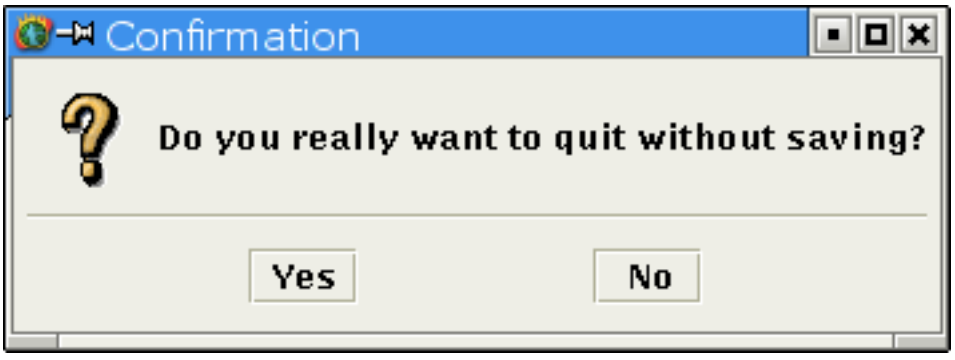


Abbildung 14.11: Confirmation-Window

Eine allgemeine Sicherheitsabfrage ist ein Fenster CONFIRMATION_WINDOW mit dem Titel „Confirmation“. Im Fenster ist ein Fragetext Text CONFIRMATION_LABEL und ein Button YES_BUTTON mit der Beschriftung YES_LABEL, sowie ein NO_BUTTON mit der Beschriftung NO_LABEL. In dem Fragetext steht jeweils kontextabhängig die Information, die der Benutzer zur Beantwortung der Sicherheitsabfrage benötigt. Standardmäßig ist der YES_BUTTON mit „Yes“ beschriftet, NO_BUTTON mit „No“.

14.13 Auswahl von Dateien

14.13.1 Der „Standard-Filechooser-Dialog“

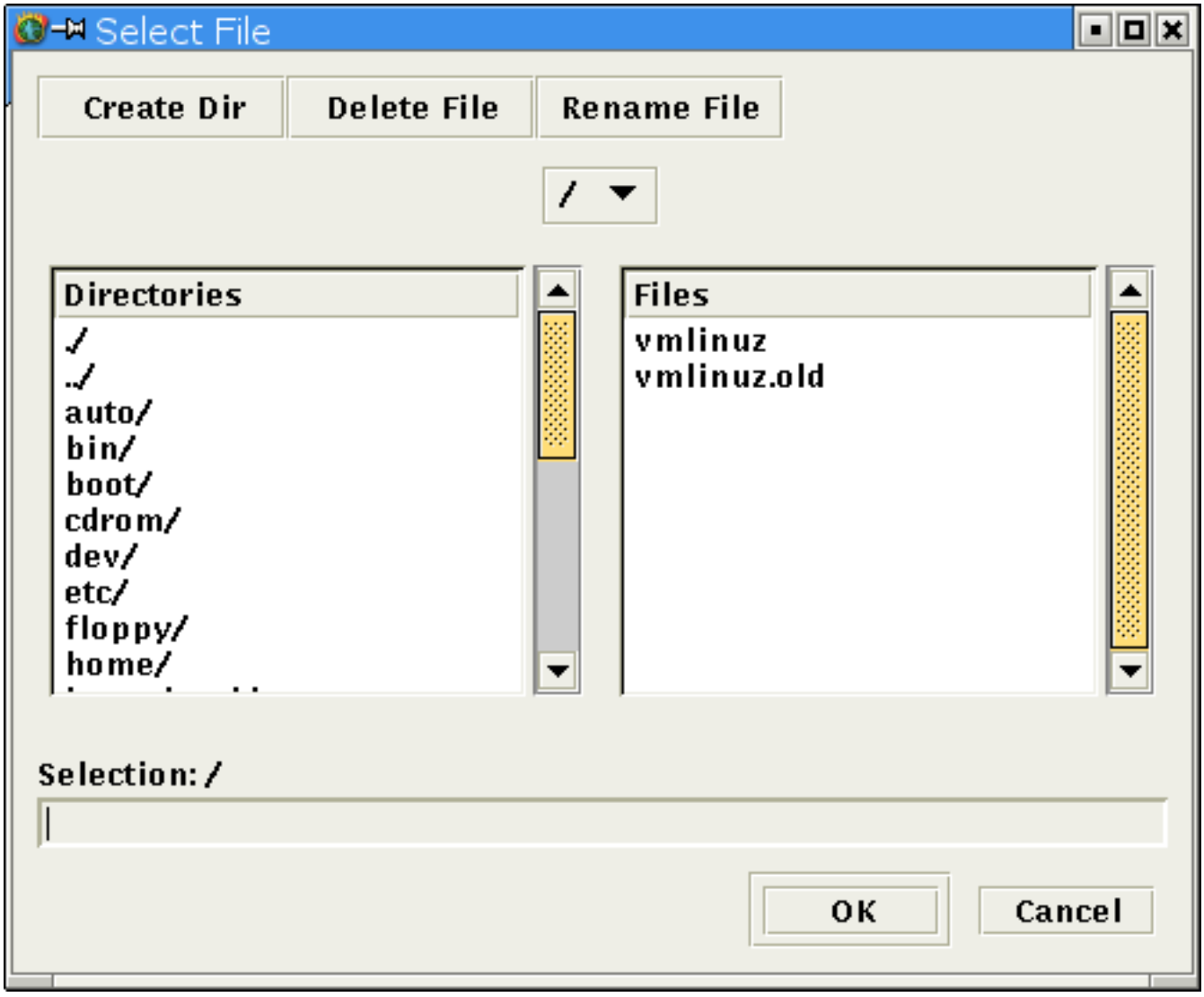


Abbildung 14.12: Filechooser-Window

Die Auswahl von Dateien (Laden/Speichern) passiert bei GIANT mittels des Standard Filechooser-Dialogs von GTK.

14.14 Fadenkreuz-Cursor

Der Cursor verwandelt sich nach Auswahl bestimmter Funktionen in ein Fadenkreuz. Durch Linksklick auf einen Knoten, eine Kante oder leere Position in einem Fenster wird diese(r) für die jewei-

lige Funktion ausgewählt. Der Cursor verwandelt sich dann wieder in den Standard-Cursor. Durch Rechtsklick läßt sich die ausgewählte Funktion abbrechen.

14.15 Fortschrittsanzeige

Es gibt in GIANT zwei Möglichkeiten zur Fortschrittsanzeige während laufender Berechnungen. Jede Progressbar ist mit „The system is busy. Please be patient.“ beschriftet.

14.15.1 Progressbar-Runs

Fenster mit Anzeige eines Dialoges mit einem sich ständig leicht ändernden Inhalt, so dass erkennbar ist, dass das System noch arbeitet.

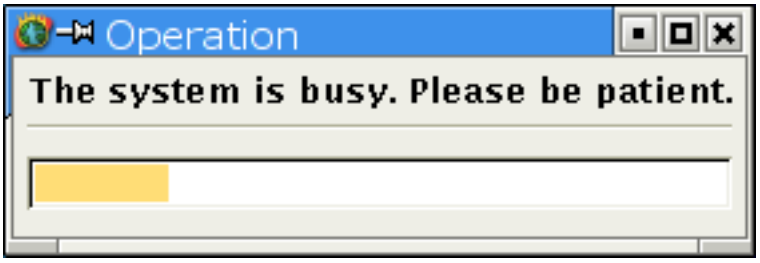


Abbildung 14.13: Progressbar-Runs

14.15.2 Progressbar-Modal

Fenster mit Anzeige eines Progressbars, der über den aktuellen Fortschritt einer Berechnung informiert, ohne dabei den Gesamtaufwand zu kennen. Dabei wird kontinuierlich die Anzahl der schon bearbeiteten Datensätze ausgegeben, und, sofern bekannt, die Prozentzahl der schon bearbeiteten Datensätze als Text. Während dieser Dialog sichtbar ist, ist der Rest der GUI von GIANT gesperrt. Im Fenster existiert ein Button „Cancel“ , der die momentane Berechnung abbricht.

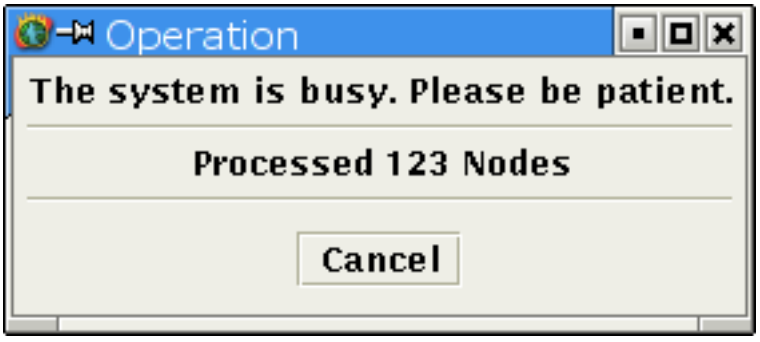


Abbildung 14.14: Progressbar-Modal

15 Visualisierung des IML-Graphen

Dieses Kapitel beschreibt, wie Knoten und Kanten des IML-Graphen innerhalb von Anzeigefenstern als Fenster-Knoten und Fenster-Kanten dargestellt werden. Spezifiziert wird hier das Ergebnis der Visualisierung, also das Aussehen der Fenster-Knoten und Fenster-Kanten, nicht aber die Art und Weise, wie GIANT diese Visualisierung erzeugt bzw. berechnet.

15.1 Visualisierung von Knoten

Hier wird die Visualisierung eines Fenster-Knoten innerhalb eines Anzeigefensters beschrieben. Die tatsächliche Darstellung hängt stark von der aktuellen Zoomstufe und dem gewählten Visualisierungsstil ab.

15.1.1 Knoten-Rechteck

1. Grundlage der Darstellung eines Fenster-Knotens ist immer ein Rechteck mit fester, automatisch berechneter Breite.
2. Die Höhe des Rechtecks ist proportional zur Anzahl der Attribute des Knotens, die direkt im Anzeigefenster dargestellt werden.
3. Jedes Knoten-Rechteck hat eine konfigurierbare Füllfarbe und einen Rahmen, dessen Farbe ebenfalls konfiguriert werden kann.

15.1.2 Knoten-Icon

Jeder Knoten kann über ein vordefiniertes Icon verfügen, welches im linken oberen Eck des Knoten-Rechtecks dargestellt wird. Alle Knoten für die kein Icon definiert wurde, haben ein Standard-Icon.

15.1.3 Icon für Knoten-Annotationen

Der Benutzer kann Knoten annotieren. Jeder Knoten, für den so eine Annotation verfasst wurde, zeigt dies über ein entsprechendes Icon im Knoten-Rechteck an.
Für weitere Informationen zu Knoten-Annotationen siehe auch Abschnitt 12.2.4.

15.1.4 Klassenname und ID

Die ID eines Knotens wird innerhalb des Knoten-Rechtecks rechts oben als Hexadezimalzahl angezeigt.
Der Name der Knotenklasse wird bei den Attributen angezeigt.

15.1.5 Attribute des Knoten

Innerhalb des Knoten-Rechtecks können auch Attribute dargestellt werden (Attributname und Wert), da das Knoten-Rechteck eine fixe Breite hat, wird der zugehörige Text abgeschnitten, falls er zu lang ist. Dieses Abschneiden wird dem Benutzer durch das Anfügen von „...“ an das Ende des Textes dargestellt.

15.2 Visualisierung von Kanten

1. Kanten sind immer gerade Linien von einem Start- zu einem Zielknoten.
2. Kanten können sich hinsichtlich ihrer Linienfarbe und der Art der Linie (z.B. gestrichelt oder durchgezogen) unterscheiden.
3. Selbstkanten werden als kreisförmige Schleifen dargestellt.

15.3 Hervorheben von Knoten und Kanten

Hier ist spezifiziert, wie Fenster-Knoten und Fenster-Kanten hervorgehoben werden. Die genaue Art der Hervorhebung kann sich aber u.U. ändern, wenn die Implementierung zeigen sollte, dass ein anderes Vorgehen sinnvoller ist.

Innerhalb eines Anzeigefensters können

1. die „aktuelle Selektion“,
2. bis zu drei weitere Selektionen,
3. und bis zu drei IML-Teilgraphen

unterscheidbar hervorgehoben werden.

15.3.1 Hervorheben von Selektionen und der aktuelle Selektion

Beim Hervorheben von Knoten und Kanten von Selektionen werden diese mittels einer konfigurierbaren Farbe eingefärbt.

1. Jeder Knoten einer Selektion wird durch entsprechendes Einfärben des Knoten-Rechtecks hervorgehoben.

2. Gehört ein Knoten zu mehreren Selektionen, die hervorgehoben sind, so wird das Knoten-Rechteck entsprechend mehrfarbig eingefärbt.
3. Jede Kante wird dadurch hervorgehoben, dass ihre Linie entsprechend dicker gezeichnet wird, wobei der Teil, um den die Kante verbreitert wird jeweils entsprechend eingefärbt wird.
4. Gehört eine Fenster-Kante zu mehreren Selektionen, die hervorgehoben werden, so wird sie entsprechend weiter verbreitert und mehrfarbig eingefärbt.

15.3.2 Hervorheben von IML-Teilgraphen

1. Jeder Fenster-Knoten eines IML-Teilgraphen wird dadurch hervorgehoben, dass er mit einem weiteren rechteckigen Rahmen versehen wird, der entsprechend eingefärbt ist.
2. Gehört ein Fenster-Knoten zu mehreren hervorgehobenen IML-Teilgraphen kommen entsprechend weitere Rahmen dazu.
3. Jede Fenster-Kante wird dadurch hervorgehoben, dass ihre Linie entsprechend dicker gezeichnet wird, wobei die der Teil, um den die Kante verbreitert wird entsprechend eingefärbt wird.
4. Gehört eine Fenster-Kante zu mehreren IML-Teilgraphen, die hervorgehoben werden, so wird sie entsprechend weiter verbreitert und mehrfarbig eingefärbt.

15.4 Detailstufen beim Zoomen

1. Beim Herauszoomen müssen Details abstrahiert werden, dies geschieht bei GIANT in mehreren Stufen.
2. Eine exakte Spezifikation der Detailstufen, insbesondere ab welcher Zoomstufe welche Detailstufe gewählt wird, ist an dieser Stelle nicht sinnvoll. Optimale Parameter hierfür werden im Laufe der Implementierung ermittelt.
3. Beim Herauszoomen wird GIANT zuerst versuchen, die Größe der Knoten-Rechtecke durch Verringerung der Schriftgröße zu verringern. Erst wenn die Schrift klein genug ist (aber noch gut lesbar) wird GIANT Elemente der Darstellung der Fenster-Knoten entfernen.

Es wird in etwa die folgenden Detailstufen geben.

15.4.1 Maximale Detailstufe

Bei maximaler bis hoher Zoomstufe (nahe ans Bild herangezoomt).
Alle oben beschriebenen Bestandteile von Knoten und Kanten werden angezeigt.
Alle Beschriftungen in maximaler Schriftgröße.

15.4.2 Sehr hohe Detailstufe

Nur die Schriftgröße für Knoten- und Kantenbeschriftungen wird verringert.

15.4.3 Hohe Detailstufe

Alle oben beschriebenen Bestandteile von Kanten werden angezeigt.
Bei Knoten werden nur noch das Icon, der Klassenname und die ID im Knoten-Rechteck angezeigt, die anderen Attribute aber nicht mehr.

15.4.4 Durchschnittliche Detailstufe

Die Beschriftungen von Kanten werden nicht mehr angezeigt.
Bei Knoten enthält das Knoten-Rechteck nur noch das Icon.

15.4.5 Geringe Detailstufe

Bei sehr geringer Zoomstufe.
Bei Knoten wird nur noch das reine Knoten-Rechteck mit Rahmen und Füllfarbe angezeigt (kein Icon mehr).

15.4.6 Minimale Detailstufe

Bei sehr geringer bis hin zur minimalen Zoomstufe.
Die Knoten-Rechtecke werden immer kleiner gezeichnet, bis sie ihre minimale Größe erreicht haben.

15.5 Minimap

Zu jedem Anzeigefenster gibt es genau eine Minimap, welche den gesamten Anzeigeinhalt stark abstrahiert darstellt.
Die Minimap bietet die folgende Funktionalität:

1. Der aktuell sichtbare Anzeigeinhalt wird auf der Minimap mit seiner relativen Position und Größe im Verhältnis zum abstrahierten Anzeigeinhalt dargestellt.
2. Über die Minimap kann auch gescrollt werden. Klickt der Benutzer auf einen Punkt innerhalb der Minimap, so wird der sichtbare Anzeigeinhalt automatisch auf den diesem Punkte entsprechenden Bereich des Anzeigeinhaltes gesetzt.

16 Nichtfunktionale Anforderungen

Hier werden die wesentlichen nichtfunktionalen Anforderungen an GIANT spezifiziert. Insbesondere sind dies Anforderungen aus dem Bereich des Software-Engineering, wie z.B. Wartbarkeit und Erweiterbarkeit.

16.1 Plattformunabhängigkeit

Das Produkt soll ein hohes Maß an Plattformunabhängigkeit, wie sie sich aus der eingesetzten Entwicklungsumgebung ergibt, verfügen (also alle Systeme auf denen GTK/ADA 1.2.12 lauffähig ist). Explizit während der Entwicklung getestet und damit garantiert werden kann dies aber nur für Sun Solaris und Linux.

16.2 Wartbarkeit

Das System soll sich durch ein hohes Maß an Wartbarkeit und Erweiterbarkeit auszeichnen. Erreicht werden soll dies durch die folgenden Grundsätze.

1. Wartbarkeit und Erweiterbarkeit gehen über Performanz.
2. Strukturierung des Entwurfes nach Kriterien des Software-Engineerings.
3. Strikte Trennung von GUI und Funktionalität.
4. Einsatz von XML für alle editierbaren Konfigurationsdateien.
5. Konfigurierbarkeit der Menüeinträge über eine XML Datei. Vorbehaltlich der Machbarkeit regelt der Entwurf hierzu näheres.

16.3 Portabilität

1. Generell soll GIANT Betriebssystem unabhängig entworfen und Implementiert werden.
2. Weitere Maßnahmen zur expliziten Unterstützung der Portabilität sind nicht vorgesehen.

16.4 Mengengerüst

1. Ein explizites Mengengerüst, z.B. hinsichtlich der minimalen und maximalen Anzahl von darstellbaren Knoten, gibt es nicht.
2. Größenbeschränkungen (z.B. Größe der zu ladenden IML-Graph-Datei, Anzahl Knoten in einem Anzeigefenster ...) sollen sich alleine aus dem verfügbaren Arbeitsspeicher ergeben.
3. Falls der verfügbare Arbeitsspeicher erschöpft ist, soll das System nicht abstürzen, sondern vielmehr die entsprechende Aktion kontrolliert abbrechen und dem Benutzer ein Weiterarbeiten ermöglichen.

16.5 Robustheit

1. GIANT soll möglichst auf alle Fehler mit einer qualifizierenden Fehlermeldung reagieren und nicht unkontrolliert abstürzen.
2. Wo immer möglich soll dem Benutzer nach Auftreten eines Fehlers ein sinnvolles Weiterarbeiten ermöglicht werden.
3. Insbesondere bei Fehlern, die mit dem Überschreiten der verfügbaren Hauptspeicherkapazität in Zusammenhang stehen, kann solch ein robustes Verhalten nicht garantiert werden.
4. Speicherlecks können im Falle Abbruchs einer Aktion im Fehlerfalle nicht ausgeschlossen werden.

16.6 Robustheit gegenüber Änderungen der IML-Graph-Spezifikation

Das System soll so an Bauhaus angebunden werden, dass Änderungen in der Spezifikation des IML-Graphen möglichst keine Wartungsarbeiten erfordern. Besonders soll dies für die verschiedenen Attribute, Knoten- und Kantenklassen des Bauhaus-IML-Graphen gelten, da diese sich oft ändern können. Die Klassen und Attribute der Bauhaus-IML-Graph Bibliothek werden durch den IML-Browser nur so unterschieden, dass neu hinzukommende Attribute automatisch mit erfasst und damit angezeigt werden können.

16.7 Leistungsanforderungen

1. Bei GIANT handelt es sich um ein striktes Einbenutzersystem.
2. Die selbe Installation von GIANT darf nicht gleichzeitig mehrfach gestartet werden.
3. Das Systemverhalten für den Fall, dass zwei Benutzer mittels verschiedener laufender Instanzen von GIANT (als Prozess im Betriebssystem) gleichzeitig auf denselben Daten (Projekte und entsprechende Verwaltungsdateien) arbeiten, ist undefiniert.

16.8 Antwortverhalten

1. Bei Anfragen und Layoutalgorithmen kann keine maximale Rechenzeit garantiert werden.
2. Der Benutzer hat aber zu jeder Zeit die Möglichkeit, entsprechende Aktionen abzubrechen.

16.9 Sicherheit

1. Ein automatisches Schreiben von Änderungen etc. in die Projektdateien ist nicht vorgesehen, der Benutzer muss dies explizit veranlassen.
   Bevor Informationen verloren gehen (z.B. beim Beenden von GIANT) erscheint aber eine Sicherheitsabfrage.
2. Muss GIANT auf Grund eines Fehlers beendet werden, so gehen alle bis dahin nicht persistent für das Projekt gespeicherten Informationen unwiederbringlich verloren.

16.10 Erweiterbarkeit

Eine spätere Erweiterbarkeit von GIANT ist besonders hinsichtlich der im Folgenden beschriebenen Funktionalität vorgesehen. Dies wird im Rahmen des Entwurfes berücksichtigt.

1. Erweiterung um neue Layoutalgorithmen.
2. Erweiterung zur graphisch unterstützten Eingabe von GQSL-Skripten.
3. Erweiterung um Kantenknickpunkte.

17 Technische Produktumgebung

Hier werden die zum Betrieb von GIANT nötige Hardware und Software spezifiziert.

17.1 Software

Das Programm stellt an die installierte Software folgende Anforderungen:

- Sun Solaris oder Linux Betriebssystem
- Bauhaus Tools
- Emacs oder vi Texteditor

17.2 Hardware

Das Programm läuft auf SPARC Workstations und x86 kompatiblen PCs. Im Folgenden sind die minimalen Hardwareanforderungen zur Arbeit mit kleinen und mittleren Projekten beschrieben. Bei großen Projekten ist ein Speicherausbau von 2 GB und mehr empfehlenswert.

17.2.1 Hardwareanforderungen SPARC

- UltraSPARC-II 300 MHz
- 512 MB Hauptspeicher
- 8 Bit Grafik mit einer min. Auflösung von 1024*786
- Maus mit mindestens 3 Tasten

17.2.2 Hardwareanforderungen x86

- Pentium III 600 MHz
- 512 MB Hauptspeicher
- 8 Bit Grafik mit einer min. Auflösung von 1024*786
- Maus mit mindestens 3 Tasten

17.3 Produkt-Schnittstellen

Das Programm soll in die Bauhaus Suite integriert werden können. Als Schnittstelle dient dabei die Bauhaus IML Bibliothek. Weiterhin kann die Verwendung von Kommandozeilenparametern und der GQSL zur Integration in das vorhandene System genutzt werden.

18 Anforderungen an die Umgebung

Hier werden die wesentlichen Anforderungen an die Entwicklungsumgebung von GIANT spezifiziert. Dies sind Werkzeuge, Programmiersprachen und Bibliotheken, die bei der Entwicklung eingesetzt werden müssen, und Standards, die beachtet werden sollen. Spezifiziert wird hier auch die Sprache aller zum Lieferumfang von GIANT gehörender Dokumente.

18.1 Compiler und Bibliotheken

Das System soll in der Sprache Ada95 mit GNAT 3.14 entwickelt werden, wobei GTK/ADA 1.2.12 als graphische GUI-Bibliothek eingesetzte werden soll. Das System baut auf der vom Kunden bereit gestellten IML-Graph-Bibliothek auf und soll des weiteren zur Unterstützung der Wartbarkeit möglichst auch die vom Kunden zur Verfügung gestellten Datenstrukturen (wie z.B. Hashtables aus Bauhaus/reuse/src) nutzen.

18.1.1 Lizenzrechtliches zu den Paketen des Kunden

Folgendes gilt nicht für die IML-Graph-Bibliothek.
Die vom Kunden zur Verfügung gestellten Datenstrukturen werden den Entwicklern von GIANT ohne lizenzrechtliche Bedingungen überlassen. Die Nutzungsrechte der Entwickler am Produkt GIANT werden durch Einsatz dieser Datenstrukturen in keinster Weise berührt.

18.2 Einlesen und Schreiben von XML Dateien

Auf XML-Dateien soll mittels des DOM (Document Object Model) Parsers aus XML/Ada 0.7.1 zugegriffen werden. XML/Ada 0.7.1 unterliegt lizenzrechtlich der „GNAT Modified GNU Public License“ (GMGPL).
Als Alternative ist der XML-Parser aus GTK/Ada vorgesehen – Paket Glib.XML.

18.3 Sprache

Hier wird beschrieben, in welcher Sprache die einzelnen Dokumente des Systems GIANT verfasst werden.

18.3.1 Spezifikation

Die Spezifikation – dieses Dokument – wird in deutscher Sprache verfasst.

18.3.2 Benutzerhandbuch

Das Benutzerhandbuch wird in deutscher Sprache verfasst.

18.3.3 Entwurf

Der Entwurf wird in Englisch verfasst.

18.3.4 Interne Dokumentation

Die interne Dokumentation von GIANT – Kommentare im Quellcode – erfolgt in englischer Sprache.

18.3.5 Interaktion mit dem Benutzer

Die GUI von GIANT interagiert mit dem Benutzer ausschließlich in englischer Sprache.

18.3.6 Konfigurationsdateien

Die Knoten und Attribute der XML-Konfigurationsdateien werden mit englischen Begriffen benannt.

19 Begriffslexikon

Das Begriffslexikon nennt Begriffe aus der „realen Welt“ und definiert ihre besondere Bedeutung innerhalb dieser Spezifikation und darauf aufbauender Dokumente. Des weiteren wird hier die englische Übersetzung dieser Begriffe für alle englischsprachigen Dokumente von GIANT vorgegeben.

- **Anfrage** (Query)
  Eine Anfrage beschreibt einen Vorgang, bei dem über geeignete Kriterien Konten und Kanten aus dem IML-Graphen oder aus IML-Teilgraphen ausgewählt werden.
- **Anzeigefenster** (Visualization Window)
  Ein Fenster in dem ein Teilgraph des IML-Graphen nach bestimmten Kriterien visualisiert ist. Jedem Anzeigefenster ist ein Anzeigeinhalt zugeordnet.
- **Anzeigeinhalt** (Window Content)
  Eine „virtuelle“ Oberfläche auf der die Objekte des visualisierten Teilgraphen (also Fenster-Knoten und Fenster-Kanten) angeordnet sind, d.h. räumliche Layoutinformation zu allen Objekten des entsprechenden Anzeigefensters. Abhängig von der Zoomstufe ist jeweils nur ein bestimmter Teil des Anzeigeinhaltes sichtbar - der sichtbare Anzeigeinhalt. Die Größe des Anzeigeinhaltes ist theoretisch unbegrenzt.
- **Fenster-Kante** (Window Edge)
  Die graphische Repräsentation einer IML-Kante innerhalb eines Anzeigefensters.
- **Fenster-Knoten** (Window Node)
  Die graphische Repräsentation eines IML-Knoten innerhalb eines Anzeigefensters.
- **Graph-Kante** (Graph Edge)
  Eine Kante des IML-Graphen welche Bestandteil eines IML-Teilgraphen ist.
- **Graph-Knoten** (Graph Node)
  Ein Knoten des IML-Graphen welcher Bestandteil eines IML-Teilgraphen ist.
- **IML-Graph** (IML Graph)
  Der IML-Graph, wie er von der Bauhaus Reengineering GmbH gestellt wird. Auf diesen Graphen wird über das sogenannte Reflection Model zugegriffen.
- **IML-Kante** (IML Edge)
  Eine Kante des IML-Graphen.
- **IML-Knoten** (IML Node)
  Ein Knoten des IML-Graphen.
- **IML-Teilgraph** (IML Subgraph)
  Eine Menge über Knoten und Kanten des IML-Graphen, die so gestaltet ist, dass sie einen Teilgraphen des IML-Graphen darstellt.

- **Kantenklasse** (Edge Class)
  Die Einteilung der Kanten des IML-Graphen in verschiedene Klassen, wie sie sich aus der IML-Graph-Bibliothek von Bauhaus ergibt.
- **Knoten-Annotationen** (Node Annotation)
  Eine textuelle Beschreibung zu einem bestimmten Knoten des IML-Graphen.
- **Knotenklasse** (Node Class)
  Die Einteilung der Knoten des IML-Graphen in verschiedene Klassen, wie sie sich aus der IML-Graph-Bibliothek von Bauhaus ergibt.
- **Layout** (Layout)
  Die zweidimensionale räumliche Anordnung von Fenster-Knoten und Fenster-Kanten innerhalb eines Anzeigefensters auf dem sogenannten Anzeigeinhalt.
- **Reflektion** (Reflection Model)
  Die Schnittstelle zum Zugriff auf den IML-Graphen.
- **Schleife** (Loop)
  Eine Kante mit identischem Start- und Zielknoten. Wird oft auch als Selbstkante bezeichnet.
- **Selektion** (Selection)
  Eine Auswahl von Knoten und Kanten eines visualisierten Teilgraphen des IML-Graphen innerhalb eines Anzeigefensters.
- **Sichtbarer Anzeigeinhalt** (Visible Window Content)
  Der Bereich des Anzeigeinhaltes eines Anzeigefensters, der zur Zeit sichtbar dargestellt wird.
- **Zoomstufe** (Zoom Level)
  Dieser Faktor beschreibt die Größe des sichtbaren Anzeigeinhaltes. Bei einer sehr niedrigen Zoomstufe (auch: weit weg gezoomt) ist ein größerer Teil des im Anzeigefenster visualisierten IML-Graphen sichtbar als bei einer hohen Zoomstufe (auch: sehr nach heran gezoomt).
- **hervorheben** (to highlight)
  Hervorheben bedeutet, dass in einem Anzeigefenster visualisierte Fenster-Knoten oder Fenster-Kanten z.B. durch eine farbige Umrahmung von anderen Fenster-Knoten oder Fenster-Kanten unterscheidbar gemacht werden.
- **selektieren** (to select)
  Selektieren beschreibt einen Vorgang über den der Benutzer, z.B. durch Anklicken von Fenster-Knoten oder Fenster-Kanten mit der Maus, eine Selektion aufbaut.

20 Index