Designbeschreibung

Softwaretechnik-Praktikum SS 2003

Gruppe: Geo01

Version	Autor	Datum	Status	Kommentar
1.0	Güpner	21.05.2003	draft	Allgemeines, Produktübersicht, Paket CAS
1.1	Reusche, Rose	23.05.2003	draft	Reverse Engeneering
	Reusche	23.05.2003	draft	.mdl Datei
1.2	Seifarth, Hartmann	23.05.2003	draft	Paket Controller
1.3	Kretschmann	24.05.2003	draft	Paket View
1.4	Reusche, Kretschmann	25.05.2003	draft	Grundsätzliche Designentscheidungen, Paket Parser
1.5	Hartmann	25.05.03	Final	Entfassung

GEO01 - SOFTWARE SOLUTIONS

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

1. Allgemeines

1.1 Kurzcharakterisierung

GeoViewer ist eine menügesteuerte graphische Java-Applikation, mit der sich geometrische Konfigurationen darstellen lassen, die in Form von GEO-Records vorliegen.
Die Berechnung zur Bestimmung der Koordinaten der geometrischen Objekte erfolgt mit dem GeoProver-Paket und dem Mathematikprogramm Maple.

1.2 Systemvoraussetzungen

GeoViewer ist als eigenständige Applikation konzipiert, die über eine grafische Nutzerschnittstelle mit gängigen Fenster-Techniken bedient wird.
GeoViewer benötigt als Voraussetzungen zum Starten die Java JRE 1.4, Maple (ab Version 5), das GeoProver-Paket für Maple sowie GEO-Records zur Eingabe. Die genaue Lage der einzelnen Ressourcen kann in der Datei *GeoViewer.rc* spezifiziert werden.

1.3 Beschreibung der Produktumgebung

GEO-Records enthalten Beschreibungen geometrischer Beweisschemata, die in einem speziellen XML-artigen Format abgelegt sind, das auf der GeoCode-Spezifikation aufsetzt. Beides ist näher in der Dokumentation des SymbolicData-Projekts (http://www.symbolicdata.org) beschrieben.
Ein GEO-Record enthält neben der Beschreibung der geometrischen Konfiguration eine Liste von unabhängigen Parametern, die für die konkrete Darstellung auf der Zeichnoberfläche mit geeigneten Zahlenwerten zu belegen sind. Die Parameterwerte haben Einfluss auf die Lage einzelner geometrischer Objekte (freier Punkte und Gleiter) und damit auch auf die Lage abgeleiteter Objekte. Verschiedene Parameterwerte ergeben also verschiedene Bilder derselben Konstruktion. In diesem Sinne kann die Darstellung „dynamisiert“ werden. Zur Interpretation dieser Beweisschemata und Behandlung der algebraischen Fragen wird eine Implementierung des GeoCode-Standards im GeoProver-Paket für das Mathematikprogramm Maple herangezogen.

1.4. Abgrenzung

Ein GEO-Record kann auch abhängige Parameter enthalten, deren Werte sich als Lösungen eines Gleichungssystems ergeben, dessen Koeffizienten mit den unabhängigen Parametern variieren.
Da die Behandlung von (nichtlinearen) Gleichungssystemen mit variierenden reellen Koeffizienten auch für Maple schwierig ist, werden derartige GEO-Records (vom Gleichungstyp) nicht auf der Zeichnoberfläche dargestellt. GEORecords ohne abhängige Parameter bezeichnet man als Records vom konstruktiven Typ. Eine „Dynamisierung“ der Darstellungen durch direkte Mausmanipulation ist nicht implementiert, da hierfür weitergehendes Event-Handling erforderlich ist.
Parameterwerte einer Visualisierung können nur über ein Dialogfeld geändert werden.

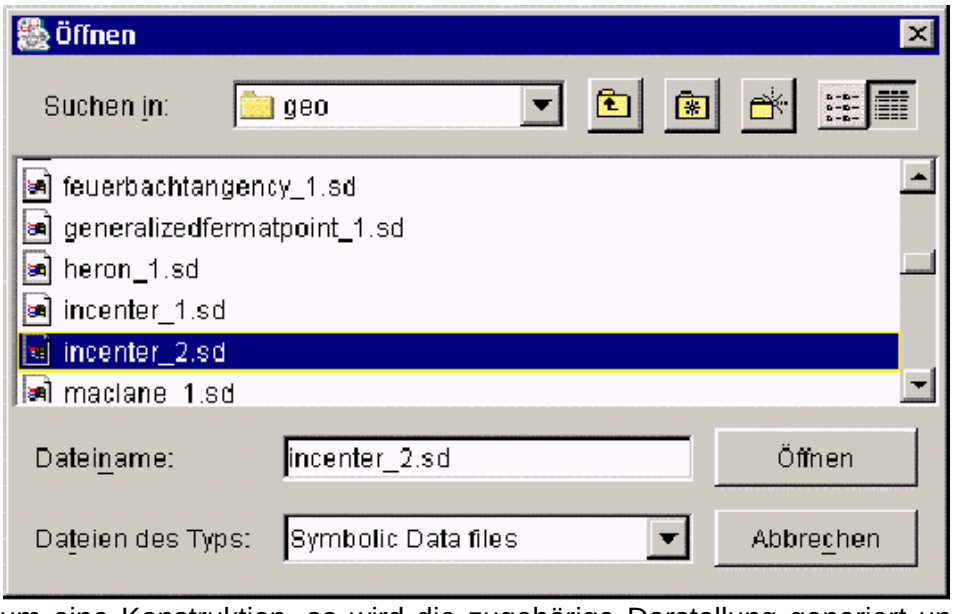
2. Produktübersicht

Die Applikation wird durch den Kommandozeilen-Aufruf *java geometry.GeoViewer*
im Wurzelverzeichnis der Pakethierarchie gestartet. Während der *Initialisierung* wird die Verbindung zu Maple hergestellt und im Erfolgsfall das Hauptfenster aufgebaut. Das *Hauptfenster* enthält am oberen Rand einen Menü- und Toolbalken, am unteren Rand einen Statusbalken und in der Mitte eine Desktopfläche, in der im weiteren Verlauf die verschiedenen Zeichenflächen sowie ein spezielles Fenster für Fehlermeldungen geöffnet werden können. Über den Menüeintrag *File* oder das linke Icon wird das *Laden eines GEO-Records* gestartet.

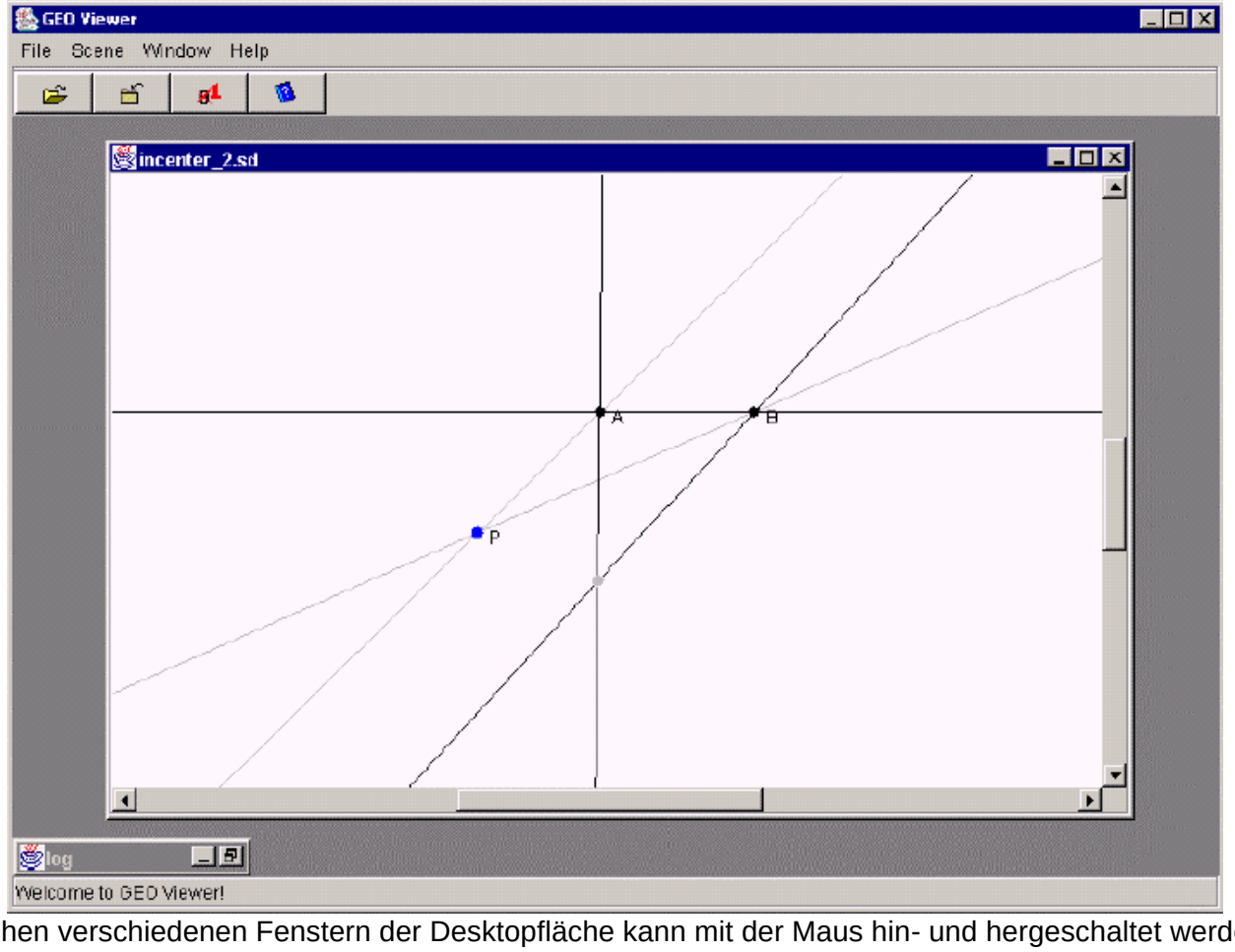
GEO01 - SOFTWARE SOLUTIONS

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann



Handelt es sich um eine Konstruktion, so wird die zugehörige Darstellung generiert und in einer neuen Zeichenoberfläche angezeigt und aktiviert. Die Parameterwerte werden dabei zunächst mit Zufallszahlen initialisiert.

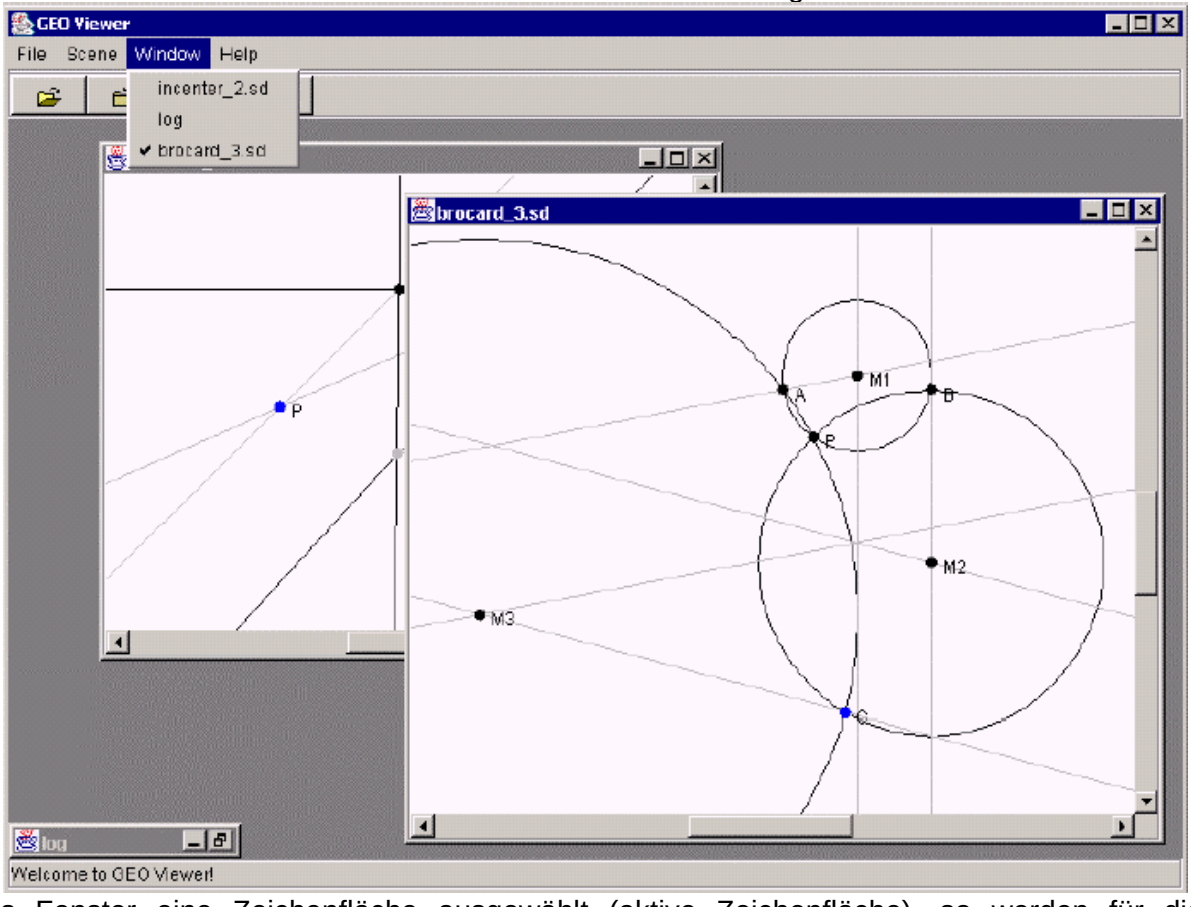


Zwischen verschiedenen Fenstern der Desktopfläche kann mit der Maus hin- und hergeschaltet werden.

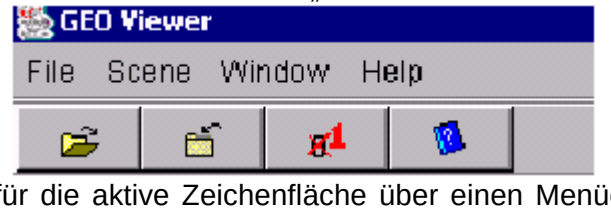
04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

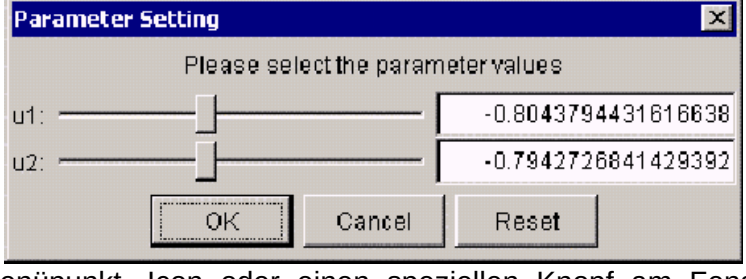
Alternativ kann das aktuelle Fenster über ein RadioButton-Menü eingestellt werden.



Wird als Fenster eine Zeichenfläche ausgewählt (aktive Zeichenfläche), so werden für diese die Menüpunkte und Icons „Parameterwerte ändern“ und „Zeichenfläche schließen“ aktiviert.



Die Parameterwerte können für die aktive Zeichenfläche über einen Menüeintrag geändert werden, der ein Dialogfenster mit Schiebereglern öffnet. Danach werden die Koordinaten der einzelnen Objekte von Maple neu berechnet und die Darstellung in der aktiven Zeichenfläche neu dargestellt.



Über einen weiteren Menüpunkt, Icon oder einen speziellen Knopf am Fensterrand wird die aktive Zeichenfläche geschlossen. Die Applikation wird über den Exit-Eintrag im File -Menü beendet.

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

3. Grundsätzliche Design Entscheidungen

Es gibt mehrere grundlegende Möglichkeiten eine dynamische Geometriesoftware zu verwirklichen. Diese beziehen sich hauptsächlich auf die Berechnungs- und Verarbeitungsweise eines Programmes.

Zum Einen gibt es die Möglichkeit das das Programm alle Berechnungen und geometrische Objekte selbst vornimmt und verwaltet. Es verwendet also eine eigene Konstruktion.
Die zweite Möglichkeit ist, das das Programm alle Berechnungen (Abstände, Winkel etc.) von einem anderen Programm durchführen lässt. Die Ansteuerung dieses Programmes erfolgt über Schnittstellen, d.h. das Programm verfügt über eine eigene Konstruktion, die auch vom Programm gespeichert wird. Die Änderungen an dieser Konstruktion hingegen werden von dem anderen Programm berechnet.
Die dritte Möglichkeit der Programmumsetzung ist, das das Programm die komplette Konstruktion und alle durchzuführenden Berechnungen und Änderungen durch ein anderes Programm ausführen lässt. Somit währe die entwickelte dynamische Geometriesoftware nur eine grafische Ein- und Ausgabeoberfläche für das benutzte berechnende Programm.
Bei dem vorliegenden Programm GeoViewer wurde die zweite Möglichkeit der Programmumsetzung verwendet. Das zur Berechnung verwendete Programm ist das CAS (Computer Algebra System) Maple.
Der Autor hat diese Art der Programmstruktur gewählt, weil dies einem angemessenen Zeit und Kostenaufwand entspricht. Eine vollständige Neuentwicklung ist um einiges zeit- und kostenintensiver und wird vom Kunden meist nicht gewünscht. Des weiteren sind die mathematischen Lösungsmöglichkeiten bereits sehr umfangreich und qualitativ gut in Maple implementiert.

Zum Anderen ist natürlich die Programmstruktur von entscheidender Bedeutung für die Funktionsweise des Programmes. Der Autor hatte hier mehrere Möglichkeiten:
Man kann das MVC vollständig umsetzen und die Klassenstruktur des Programmes streng in Modell, View und Controller aufteilen.
Eine andere Möglichkeit währe das MVC Konzept nur zum Teil umzusetzen, das heißt sich für einen Kompromiss zu entscheiden, in dem man Model, View und Controller in verschiedenen Ausprägungsstufen implementiert. Bei GeoViewer wurde letzteres angewandt, und besonders der Modellteil des MVC-Konzeptes wurde nur zum Teil umgesetzt:

- es besteht zum einen aus dem package CAS, welches die Kommunikation mit Maple ermöglicht
- des weiteren wurde es teilweise im package Controller integriert

Umsetzung der Architektur des Programmes:
Aufteilung des Programmes in mehrere Pakete

- geometrie.CAS
- geometrie.Controller
- geometrie.View
- geometrie.Parser

Die Klassen des Programmes sind in den verschiedenen packages folgendermaßen angeordnet:

GEO01 - SOFTWARE SOLUTIONS

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

Paket: geometrie.CAS

Das CAS stellt die Klassen zur Verfügung, die die Schnittstellen zwischen dem GeoViewer und dem CAS „Maple“ realisieren. Demnächst kann statt Maple auch das CAS Mupad verwendet werden.

Zwei Klassen dienen der Vermeidung von Fehlern durch den Nutzer:
`CASDivisionByZeroException.java` dient der Meldung des Fehlers, wenn bei der Berechnung eine Division durch 0 durchgeführt werden muss.
Diese Klasse ist eine Unterklasse der Klasse `CASException.java`. Diese Klasse meldet, wenn Ausnahmefälle im CAS eingetreten sind.
Sie ist die Unterklasse der Klasse IOException.
Diese beiden genannten Oberklassen sind in der Java Bibliothek bereitgestellt.
Und dann gibt es drei Schnittstellen. Zum einen die Schnittstelle zum CAS `CASInterface.java`, dann die Schnittstelle zum Maple `MapleInterface.java` und die Schnittstelle zum Mupad `MupadInterface.java`.
Die Mupad Schnittstelle ist noch in Bearbeitung, deswegen ist sie noch nicht einsatzbereit, d.h. der GeoViewer arbeitet vorerst nur in Verbindung mit dem CAS Maple.
Bei der CAS Schnittstelle handelt es sich um Funktionen die allgemeine Dinge lösen, die mit dem CAS zu tun haben, so z.B. Öffnen und Schließen des CAS, Auswertung von geometrischen Ausdrücken und das Übermitteln der aktuellen Ergebnisse vom CAS zum GeoViewer. Die anderen beiden Klassen sind dann die Schnittstellen, die direkt für ein CAS zugeschnitten sind, eine für Maple und eine für Mupad, wobei das für Mupad, wie schon erwähnt, noch in Arbeit ist. Die Klasse `MapleInterface.java` hat viele Funktionen um die von GeoViewer erzeugten geometrischen Ausdrücke umzuwandeln, um sie mit Maple dann berechnen zu können. Das Schließen ist auch noch einmal enthalten, da das Programm testen muss, ob das CAS noch Berechnungen durchführt oder ob der letzte Rechenvorgang abgeschlossen ist, also ob Maple terminiert hat. Es wird die Übermittlung der kompletten Daten realisiert und immer wenn Fehler bei der Übermittlung auftreten, gibt es entsprechende Hinweise. Die Maple Schnittstelle öffnet immer dann das CAS Maple, wenn es gebraucht wird, d.h. wenn im GeoViewer geometrische Ausdrücke berechnet werden müssen.

Paket: geometrie.Controller

Die eigentliche Aufgabe des Controllers ist es, die vom Benutzer übergebenen Daten zu verwalten und zu verarbeiten.
Der Controller hat die Aufgabe die Eingabe und das Anlegen der Objekte zu sichern und zu übernehmen. Über ihn werden geometrische Elemente angelegt oder verändert. Es stehen hierbei verschiedene Methoden und Klassen zur Verfügung.

Die Klasse `GeoElement` ist eine Oberklasse für alle geometrischen Elemente welche das Programm benutzt.
Diese Klasse implementiert das Interface Comparable, diese Schnittstelle wird von allen Klassen implementiert, deren Objekte paarweise miteinander vergleichbar sind (enthält die Funktion `int compareTo(Objekt o)`).
Die Klasse enthält weiterhin eine Methode für die Koordinaten des Elements, eine Methode die die Priorität für jeden Typ festlegt (die Priorität wird benutzt um die Abfolge beim Zeichnen festzulegen), sowie eine `compareTo` Methode zum Vergleichen.
Von dieser Klasse `GeoElement` werden weitere Klassen abgeleitet, welche die Funktionen und Attribute von `GeoElement` erben und sie spezifizieren.

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

So wird in den abgeleiteten Klassen `GeoAngle`, `GeoPoint`, `GeoDistance`, `GeoLine` und `GeoCircle` jeweils eine Funktion `expr` beim Erstellen eines Objektes dieser Klasse mit gegeben und dieses wird direkt an die Oberklasse `GeoElement` weitergegeben. Dort wird diese Funktion `expr` auf einer Funktion `coordinates` gespeichert und somit festgehalten.
Diese 5 abgeleiteten Klassen unterscheiden sich in ihren verschiedenen Prioritäten, welche durch die Funktion `getPriority( )` festgelegt werden. Diese Prioritäten werden vom View benutzt um die Reihenfolge des Zeichnens festzulegen.

Die Klasse `Expression` hat einen besonderen Stellenwert im Controller.
Sie beinhaltet neben einigen finalen Attributen wie `CONCLUSION` oder `MAIN` auch ein `double-buffered value array`, sowie einen optionalen Namen.
Beides wird benötigt um andere Objekte anzulegen. Man kann zum einen ein Constant Objekt, Variable Object oder einen Parameter anlegen. `Constant` legt ein `double array` an, welches eine feste Größe hat. `Parameter` legt ein Objekt mit einem eindeutigen Namen an.
Somit sind Funktion, Constant, Parameter und Variable geometrische Ausdrücke mit denen die verschiedenen Sichten (Model und View) arbeiten können. Sie können hier angelegt werden und unterscheiden sich, wie es bei Java üblich ist, durch die verschiedenen Reingabewerte.

Die Klassen `Funktion`, `Constant`, `Parameter` und `Variable` sind abgeleitet von der Klasse `Expression`.
In den Klassen `Function`, `Parameter` und `Variable` wird jeweils `ein identifier` an den Konstruktor der Klasse `Expression` übergeben, dieser erzeugt dann ein neues Objekt. Für die Klasse `Function` ein neues Objekt vom Typ Function, für die Klasse `Parameter` ein neues Objekt vom Typ Parameter und für die Klasse `Variable` ein neues Objekt von Typ Variable.
Die Methode `Constant` der Klasse `Constant` bekommt den numerischen Wert einer Konstante als String übergeben und übergibt diesen ebenfalls dem Konstruktor der Oberklasse `Expression`, der dann ein neues Objekt anlegt.
Die Klasse `Function` modelliert geometrische und algebraische Funktionen (Koordinaten speichern, sowie Operationen deklarieren) Sie modelliert also geometrische Operationen sowie eine geometrische Funktionen.
In dieser Klasse bzw. in Objekten von dieser, wird der Befehl direkt in einem String gespeichert. Der Oberklasse `Expression` wird ein `identifier` übergeben. Dies ist nicht anderes als ein String, durch den man später im CAS festlegen kann was wem zugehört.
Die Klasse `Operation` enthält eine Methode `Operation`, die die Parameter `intentifier`, `command` und `arguments` an den Konstruktor der Klasse `Function`, der dann ein neues Objekt anlegt.

Die Klasse `SceneController` enthält die Methoden um die Darstellung zu kontrollieren.
Wird der Konstruktor der Klasse aufgerufen benötigt er dazu das CAS Interface.
Die Klasse dient zum Öffnen und Schließen der Zeichenoberfläche.
Es gibt Methoden für Parameter, Elemente usw.

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

Paket: geometrie.View

Klasse Application Frame:

Die Klasse `ApplicationFrame` stellt das Hauptfenster des Programmes dar. Die Klasse ist ein erweitertes JFrame, und enthält alle Buttons und Menüs, die man im Hauptfenster des Programmes sehen kann. Ebenso werden hier die Ereignisverarbeitenden Klassen für die Schaltelemente instanziert.
Dies sind die Klassen, die für die Ereignisverarbeitung verantwortlich sind:

Klasse DesktopListener:
Diese Klasse ist eine Interface-Klasse für die Klasse DesktopAdapter.

Klasse DesktopAdapter:
Die Klasse DesktopAdapter aktivert und deaktiviert die zum aktuellen SceneFrame gehörenden AktionListener. Sie dient dem Wechsel zwischen den einzelnen Darstellungsfenstern.

Klasse OpenAction:
Wenn dieser Klasse ein Ereignis zugewiesen wird, wird eine Instanz der Klasse FileSelectionFlield angelegt, die eine Referenz auf die Datei angibt die dann einer neuen SceneFrame Instanz übergeben wird.

Klasse CloseAction:
Wenn dieser Klasse ein Ereignis zugewiesen wird, dann schließt die Klasse das Aktuelle Fenster Unterfenster des Programmes.

Klasse ExitAction:
Wenn dieser Klasse ein Ereignis zugewiesen wird, dann wird das geamte Programm beendet, indem das Hauptfentser geschlossen wird.

Klasse AboutAction:
Wenn dieser Klasse ein Ereignis zugewiesen wird, dann wird ein Instanz der Klasse AboutBox angelegt und angezigt.

Klasse ParameterAction:
Wenn dieser Klasse ein Ereignis zugewiesen wird, dann wird eine Instanz der Klasse ParameterBox angelegt und angezeigt.

Klasse DesktopEvent:
Diese Klasse erkennt, wenn sich das aktive Darstellungsfenster des Programmes ändert, und kann das nun aktive Unterfenster des Programmes ermitteln.

Klasse Window Menu:
In dieser Klasse wird das Menu, das zum Umschalten zwischen den einzelnen SceneFrames (Beschreibung siehe bei der entsprechenden Klasse) notwendig ist, verwaltet.

Klasse GeoDesktopPane:
Die Klasse Geo DesktopPane stellt ein erweitertes JDesktopPane dar, das es ermöglicht auf das aktuell aktive Frame zuzugreifen, und Referenzen aller zugeordneten Frames in einer Liste speichert.

Klasse StatusBar:
Diese Klasse iste eine Statusanzeige, in der die jeweiligen Änderungen und Aktionen, die ausgeführt werden angezeigt werden.

Klasse GeoToolBar:
Diese Klasse ist eine Erweiterung von JToolBar, in der ein Button (durch ein entsprechendes Icon) darstellt, welche Aktion in dem Programm gerade ausgeführt wird.

Klasse Panel1:
Bei der Klasse Panel1 handelt es sich um den Zeichenbereich, den es in jedem Darstellungsfenster SceneFrame gibt.

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

Klasse ElementGraphics:
Die Klasse ElementGraphics enthält alle Routinen um die GeoElemente und alle Unterklassen Objekte, die zur Konstruktion gehören, im SceneFrame darzustellen

Klasse ElementGraphicsException:
Diese Klasse ist eine Exception-Klasse die aufgerufen wird, wenn ein GeoElement in ElementGraphics mittels draw nicht dargestellt werden kann. Sie erledigt die Fehlermeldung, das das Objekt nicht dargestellt werden konnte.
Klasse SceneFrame:
Dies ist die Klasse, die die Zeichenfenster in dem die Konstruktion dargestellt wird, enthält. Sie enthält sowohl ein dynamisches Feld an GeoElementen, die ja in dem entsprechenden Fenster dargestellt werden müssten, als auch einen Controller für dieses Fenster (SceneController).
Klasse logFrame:
Diese Klasse ist ein Unterfenster des Programmes, in dem wichtige Meldungen für den Benutzer geloggt werden.
Klasse FileSelectionField:
Die Klasse FileSelectionField ist einDialogfeld, das durch die OpenAction-Klasse aufgerufen wird. Darin wird eine Datei ausgewählt und eine Referenz auf die Datei an OpenAction zurückgegeben. Hier wird die Klasse ExtensionFileFIlter instanziert.
Klasse ExtensionFileFilter:
Diese Klasse stellt einen Filter für die Dateianzeige im FileSelectionField dar, so das nur die Dateien angezeigt werden, die die Dataiendung haben, die vorher der Instanz dieser Klasse übergeben wurde.
Klasse ParameterBox:
Diese Klasse ist ein Dialogfeld in dem die Parameter, die für die Darstellung der GeoElemente notwendig sind (siehe Abbildung bei 2.), geändert werden können.
Klasse AboutBox:
Die Klasse About Box ist ein Dialogfeld, in dem die wichtigsten Daten des Programmes niedergeschrieben sind (Versionsnummer, Autor, etc.).

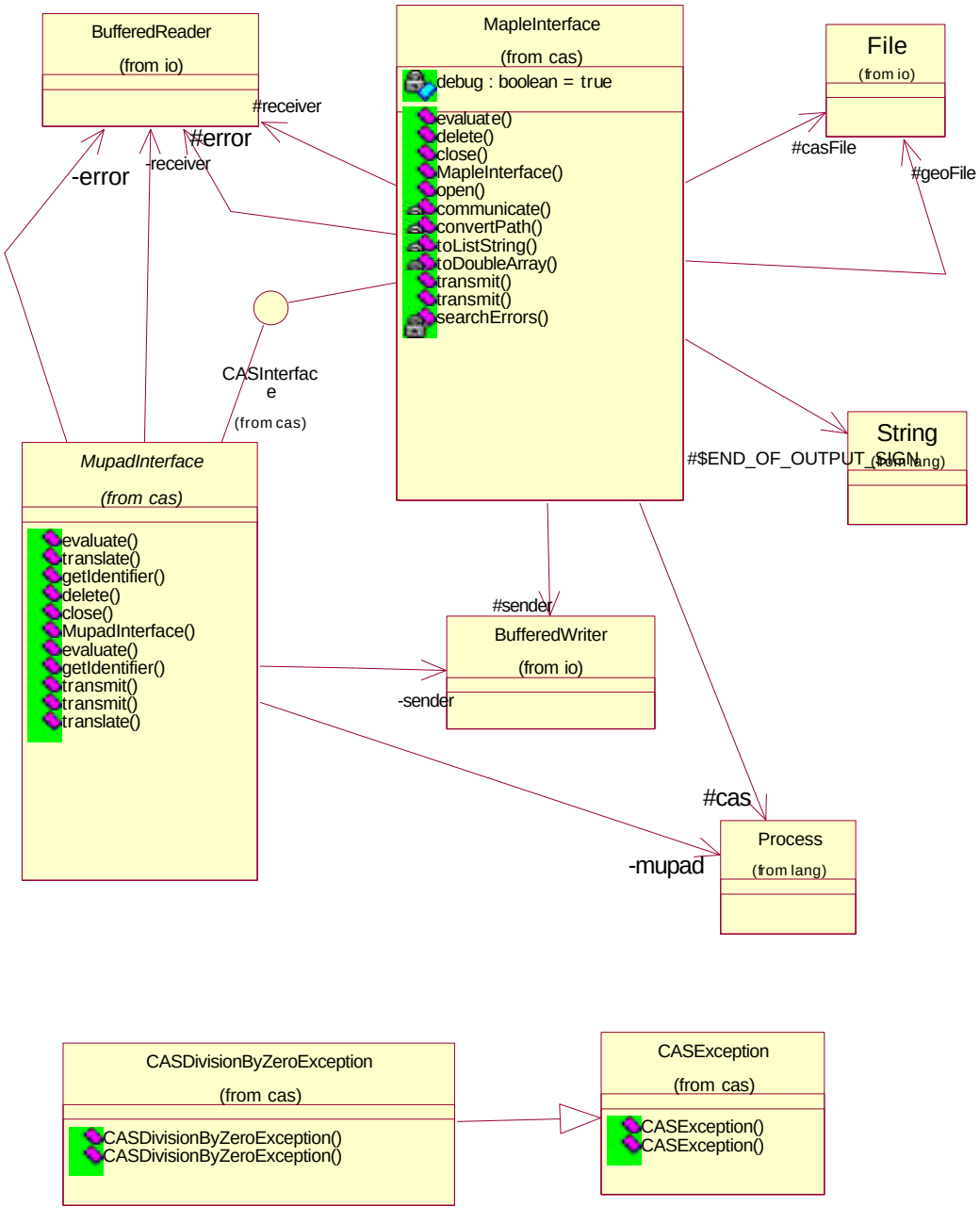
Paket: geometrie.Parser
Das Paket Parser ist dafür verantwortlich, das die Anweisungen die über die Klassen des Paketes CAS an Maple geschickt werden, in Befehle konvertiert werden, mit denen dieses arbeitet.
Also werden alle Anweisungen die vom Programm über das Interface des packages CAS an Maple sendet, vorher noch einmal durch die Klassen des packages Parser modifiziert.

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

4. Paket- und Klassenstrukturen

Darstellung der verwendeten Klassen als Klassendiagramm mit Hilfe des CASE Tools Rational Rose:

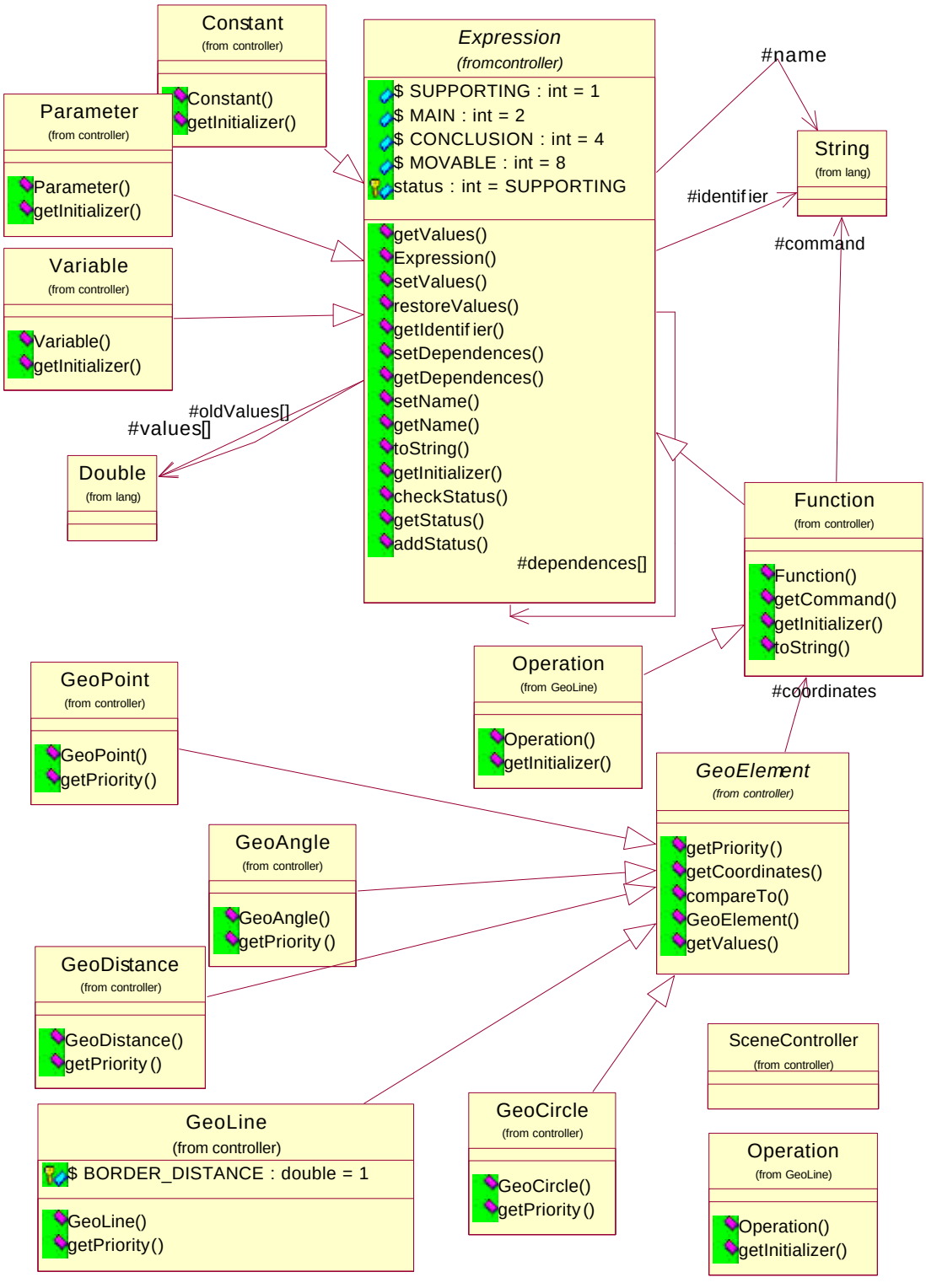


GEO01 - SOFTWARE SOLUTIONS

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

Controller

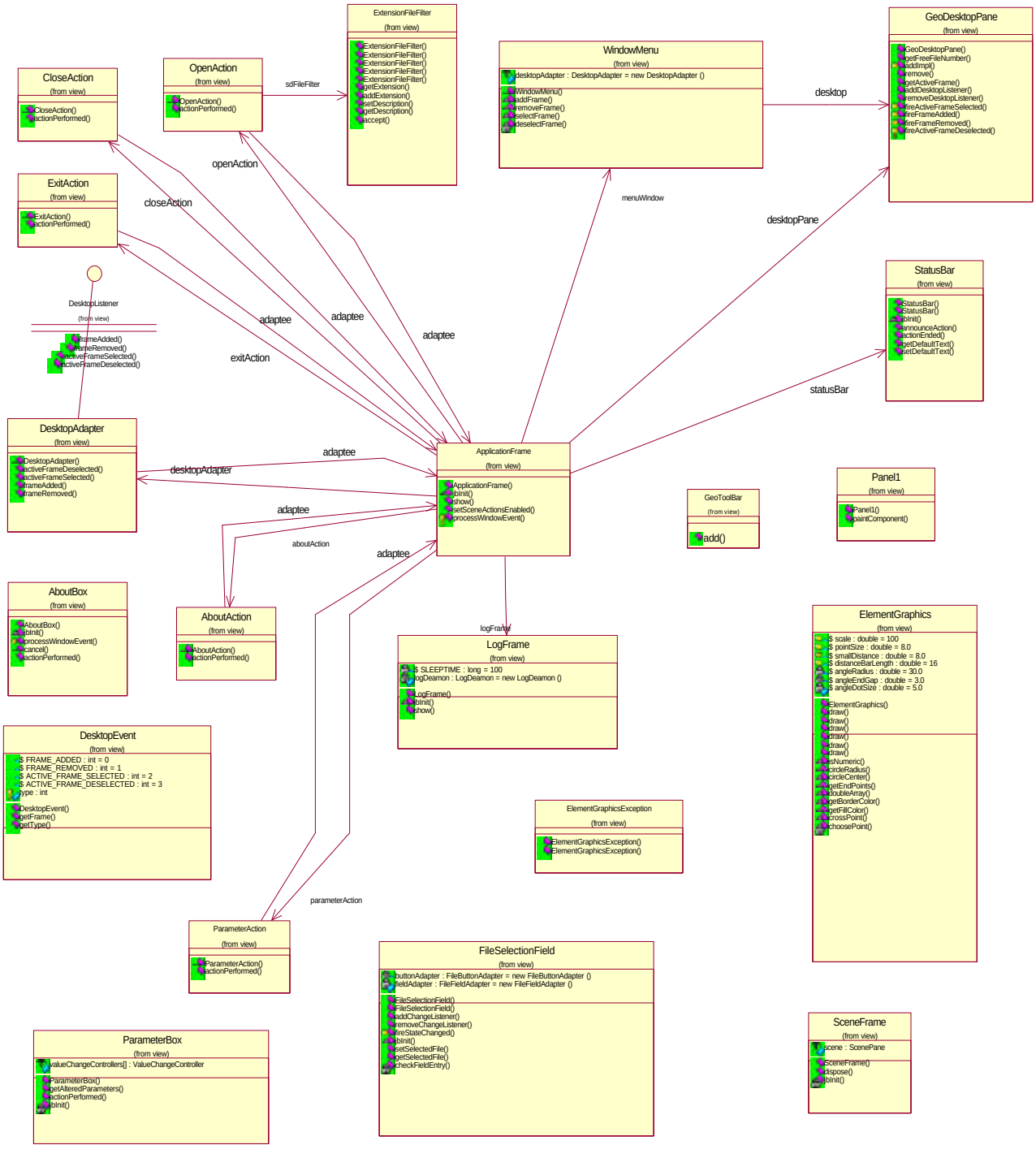


GEO01 - SOFTWARE SOLUTIONS

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

View



GEO01 - SOFTWARE SOLUTIONS

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Anne Güpner, Marcel Kretschmann, Daniel Seifarth
Matthias Reusche, Madlen Hartmann

Parser