Implementierungsbericht

Simon Bischof, Jan Haag, Adrian Herrmann, Lin Jin, Tobias Schlumberger, Matthias Schnetz

Praxis der Softwareentwicklung
Projekt 3:
Automatisches Prüfen der Korrektheit von Programmen
Gruppe 1

WS 2011/2012

Inhaltsverzeichnis

1 Änderungen gegenüber dem Entwurf

1.1 Paketänderungen

- Das Interface ASTVisitor wurde vom interpreter- in das ast-Paket verschoben, da das Interface logisch unabhängig vom Interpreter ist.
- Ebenso wurden der TypeChecker und die hiervon verwendete IllegalTypeException in das parser-Paket verschoben, da beide Klassen hauptsächlich beim Parse-Vorgang verwendet werden.
- Der SMTLib-Translator wurde vom interpreter- in das verifier-Paket verschoben, da dieser wesentlicher Bestandteil der Verifikation ist und unabhängig vom Interpreter ist.
- Die Klasse MessageSystem befindet sich nicht mehr im interpreter-Paket, sondern im misc-Paket, da sie als Modell dient.

1.2 Paket ast

- Assignment hat ein zusätzliches Attribut `depth` zur Represäntation der Scopetiefe der Variablendeklaration.
- ArrayType hat eine zusätzliche Assoziation zu Type mit der Rolle `basetype`. Hiermit werden unterschiedliche Arraytypen, wie z.B. `bool[]`, `int[]` und `int[][]`, unterschieden.
- Die Klasse Length wurde gelöscht, da diese nun in den Visitorklassen als spezieller Funktionsaufruf behandelt wird.
- FunctionCall hat eine zusätzliche Assoziation zu Identifier, welche vom TypeChecker zur Referenzauflösung verwendet wird. Hierdurch ist auch die Methode `setFunction` zur Klasse hinzugefügt worden.
- Loops können nun neben Invarianten auch Ensures beinhalten, die als Nachbedingungen nach Schleifenende geprüft werden und für eine Verifikation bewiesen werden müssen.
- Functions haben die Möglichkeit Assumptions und Ensures zu besitzen. Assumptions stellen hierbei Vorbedingungen, Ensures Nachbedingungen dar. Beide Zusicherungen sind für einen Korrektheitsbeweis nötig. Verwendet man Assumptions in der `main`-Funktion, so kann man den Parameterbereich für den Beweiser einschränken.
- Die Klasse Range wurde neu eingeführt, welche zwei Assoziationen `lower` und `upper` zu Expression besitzt. Die Klasse dient zur Einschränkung des Bereichs in quantifizierten Ausdrücken. Infolgedessen haben QuantifiedExpressions eine Assoziation zu Range.
- BooleanLiteral und IntegerLiteral speichern ihre Werte im neuen `value`-Attribut vom Typ Value.
- Das Interface ASTVisitor kann nun auch StatementBlocks mit der `visit`-Methode besuchen.
- Die Klasse QuantifiedExpression erbt nun von LogicalExpression, da jede QuantifiedExpression ein Boolean zurück gibt.
- Die Indizes aller Array-Klassen wurden vom Typ ArithmeticExpression zu Expression geändert, die Bedingung von Loops und Conditionals ebenfalls von LogicalExpression zu Expression. Dies ermöglicht es nun auch Variablen bzw. Arraykomponenten und gegebenenfalls auch Funktionsaufrufe in diesen Fällen zuzulassen. Die Typsicherheit wird durch den TypeChecker sichergestellt.
- Die Klassen LogicalOperator und ArithmeticOperator wurden zu Interfaces, die von den bisherigen Kindklassen implementiert werden. Zusätzlich erben die Kindklassen von einem der beiden neuen Klassen UnaryOperator bzw. BinaryOperator.
- Die Methode `getNextStatement` in StatementBlock wird ersetzt durch eine neue Methode `getIterator`, die einen Iterator für die Statements in diesem StatementBlock zurückgibt. Dies entkoppelt die Klasse des Syntaxbaums von einer konkreten Abarbeitung.
- Die Klasse UnaryPlus wurde als unnötig identifiziert und somit entfernt.

1.3 Paket parser

- Neue Exception FunctionCallNotAllowedException, die geworfen wird, wenn Funktionsaufrufe an einer Stelle stehen, an der sie nicht erlaubt sind. Infolgedessen hat der TypeChecker ein neues Attribut `functionCallAllowed` mit Getter- und Setter-Methode.
- Die Klasse TypeChecker bekommt ein neues Attribut `currentScope` vom Typ Scope und eine dazugehörige Setter-Methode, die bei der Auswertung von GlobalBreakpoints benutzt wird.
- Die Klasse IllegalTypeException ist zur besseren Anzeige mit einer Position assoziiert.

1.4 Paket interpreter

- Die neuen Klassen BooleanValue, IntegerValue und ArrayValue erben von der Klasse Value und repräsentieren einen Boolean-Wert durch ein Attribut vom Typ `boolean`, eine Ganzzahl durch ein Attribut vom Typ `BigInteger` bzw. ein Array durch ein Array vom Typ `Value[]`.
- Die Klasse GlobalBreakpoint hat nun eine Assoziation zur Klasse Expression statt zu LogicalExpression. Dies ermöglicht es nun auch Variablen bzw. Arraykomponenten und gegebenenfalls auch Funktionsaufrufe in globalen Breakpoints zuzulassen. Die Typsicherheit wird durch den TypeChecker sichergestellt.
- Die Klasse StatementBreakpoint ist nicht mehr einem Statement, sondern eine Zeile zugeordnet, da hierdurch die Breakpointbehandlung vereinfacht wird.
- Die Klasse Scope:
  - hat ein neues Attribut `returnValues` vom Typ IdentityHashMap zur Speicherung von Zwischenergebnissen von Funktionsaufrufen und eine zugehörige Getter-Methode.
  - hat ein neues Attribut `currentFunction` vom Typ Function zur Identifizierung der zum Scope zugehörigen Funktion, falls vorhanden.
  - hat ein neues Attribut `statements` vom Typ Iterator¡Statements¿ zum Iterieren über Statements unabhägig von anderen Scopes, was insbesondere bei Rekursionen wichtig ist.
  - hat ein neues Attribut `currentStatement` vom Typ Statement zum Speichern des momentanen Statements im Fall eines Funktionsaufrufs und eine zugehörige Getter-Methode.
  - hat aufgrund der neuen Attribute einen geänderten Konstruktor.
  - hat eine neue Methode `isFunctionScope` zur Feststellung, ob ein Funktionsscope am Ende einer Funktion abgebaut wird.
  - hat eine neue Methode `getCurrentFunction`, die die Funktion zurückgibt, in der der Scope bzw. ein übergeordneter Scope ist.
  - hat eine neue Methode `createVar` zum Erzeugen von Variablen und eine neue Methode `createArray` zum Erzeugen von Arrays. Infolgedessen gibt es getrennte Methoden `setVar` und `setArray` zur Änderung der Werte der Variablen bzw. Arrays.
  - hat eine neue Methode `createFunctionResult` zur Berechnung neuer Ergebnisse für `returnValues`.
  - hat eine neue Methode `clearFunctionResult` zum Löschen der Zwischenergebnisse in `returnValues`.
- Infolge der Änderungen der Klasse Scope wurde in der Klasse State die Parameter der Methoden `createScope`, `setVar` und `createVar` angepasst und die Methoden `createArray`, `setArray`, `getCurrentFunction`, `isFunctionScope` und `getReturnValues` hinzugefügt. Außerdem gibt es eine neue Methode `adjustStatement`, die das nächste Statement bestimmt und setzt.
- Die Klasse ProgramExecution:
  - die Methode `setBreakpoint` wurde entfernt. Zur besseren Behandlung der beiden Breakpoint-Typen existieren nun Assoziationen zu GlobalBreakpoint und StatementBreakpoint. Für die Überprüfung von globalen Breakpoints besitzt die Klasse nun ein Attribut vom Typ TypeChecker. Aufgrund der Änderungen wurde der Konstruktor der Klasse angepasst.

  - die Methode `checkBreakpoint` liefert nun im Fall eines getroffenen Breakpoints den jeweiligen Breakpoint zur Anzeige zurück.
  - hat neue Methoden `initParams` und `initArray` zur Initialisierung von Parametern der `main`-Funktion.

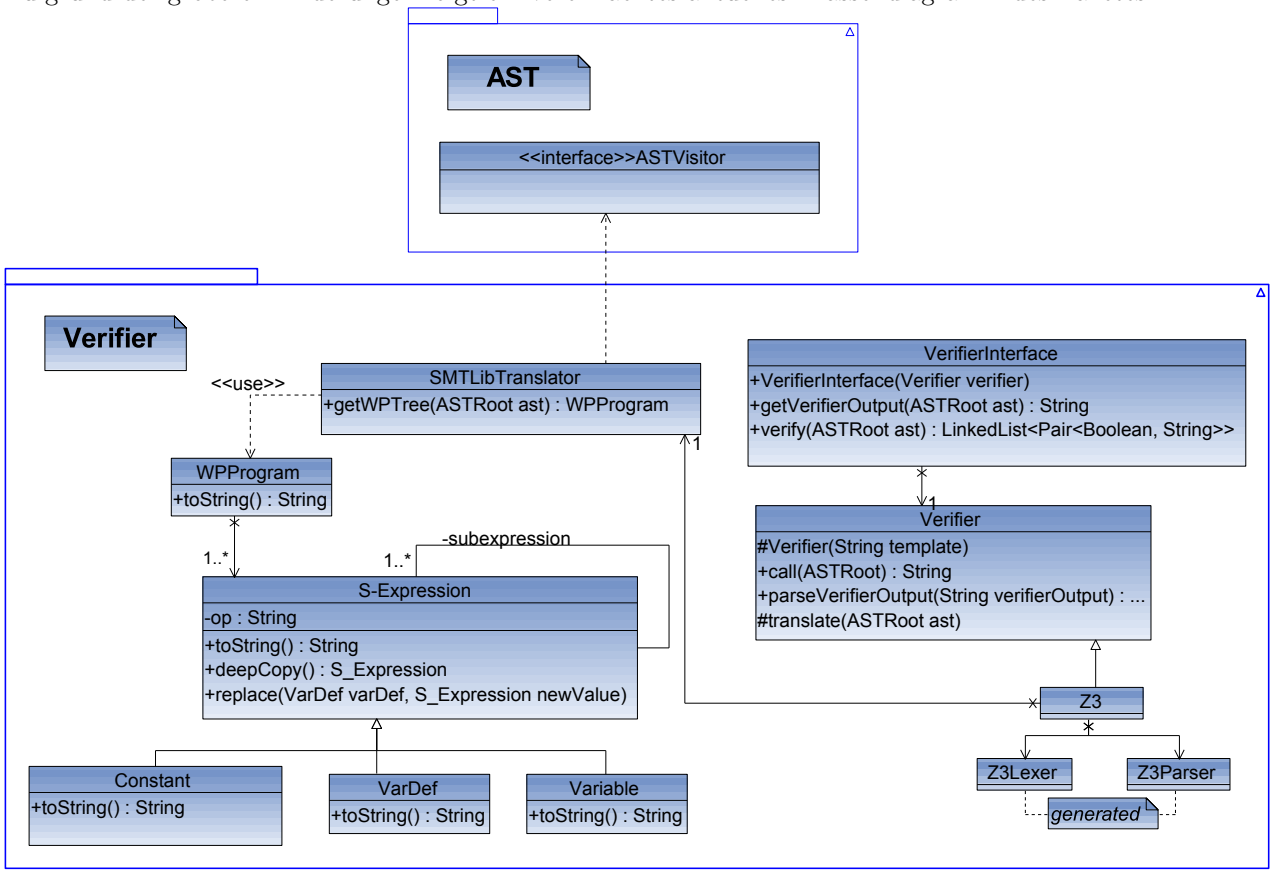
- Die Klasse Interpreter hat eine innere Klasse StopStatementExecution, die zum Abbrechen des aktuellen Statements bei Funktionsaufrufen benutzt wird.
- Die Klasse Interpreter hat neue Methoden `adjustStatement` zur Anpassung des momentanen Statements und `checkAssumptions` zur Spezialbehandlung der Assumptions der `main`-Methode.
- Die Klasse AssertionFailureException ist zur besseren Anzeige mit einer Position assoziiert.

1.5 Paket verifier

Die Schnittstelle und Arbeitsweise des Beweiserpakets wurde während der Implementierungsphase geändert, um eine höhere Flexibilität und Erweiterbarkeit erreichen.

- Neue Klasse Verifier als Basisklasse und einheitliche Schnittstelle für einen Beweiser. Die Klasse Z3 erbt von dieser Klasse.
- Neue Klasse VerifierInterface als Fassade des verifier-Pakets nach außen.
- Neue Klassen VarDef und Variable, die von S_Expression erben und zur Ersetzung im wp-Kalkül dienen.
- Die Klasse S_Expression hat eine neue Methode `deepCopy`, die referenzungleiche Kopien einer S_Expression und ihrer Subexpressions erzeugt.
- Die Klasse SMTLibTranslator hat neue Methoden `prepareFinalProgram` und `createBlock` zum Abschluss des Übersetzungsvorgangs.

Aufgrund der größeren Änderungen folgt ein vereinfachtes aktuelles Klassendiagramm des Paketes:

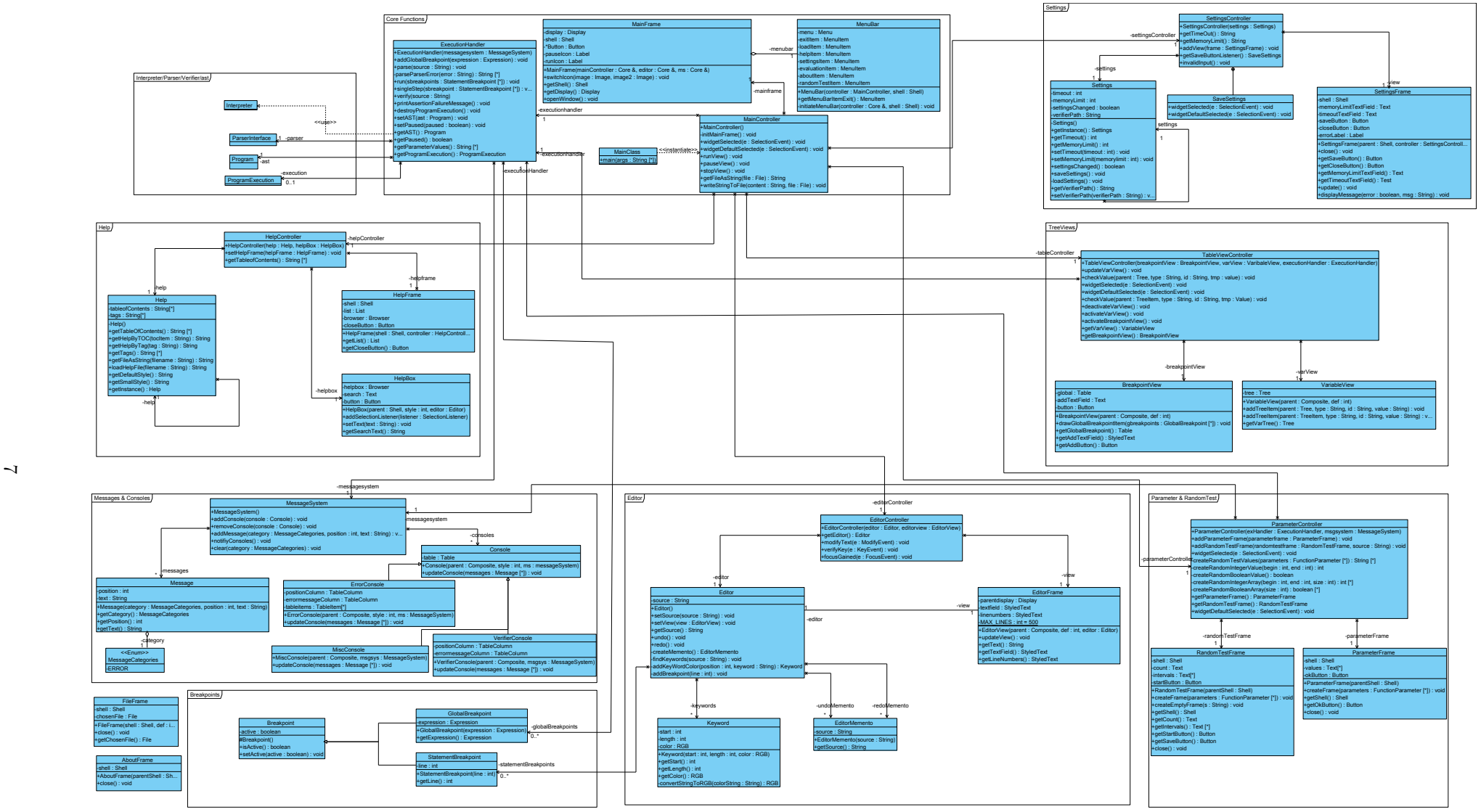


1.6 Pakete gui und misc

Die Pakete der GUI-Komponente wurden während der Implementierungsphase nochmals überarbeitet. Der ursprüngliche Entwurf war nicht auf das verwendete Toolkit SWT ausgelegt, deswegen wurden die Attribute und Methoden der meisten Klassen des gui-Pakets angepasst. Auch an der gesamten Klassenstruktur mussten einige Veränderungen vorgenommen werden. Die Bestandteile des MVC-Patterns wurden zum Teil nicht klar getrennt und der Entwurf erschien dadurch inkonsistent. Die wichigsten Änderungen sind im Folgenden aufgelistet.

- Die Klasse ExecutionController wurde zu ExecutionHandler umbenannt, da der alte Name leicht mit einem Controller des MVC-Patterns verwechselt werden konnte und die Klasse in Wirklichkeit nicht die Rolle des Controllers spielt. Auf Grund ihrer Teilfunktion als Modell wurde sie in das misc-Paket verschoben.
- Die Funktionen Parsen, Interpretieren und Validieren werden nun alle vom ExecutionHandler aufgerufen und nicht, wie im Entwurf, zum Teil vom MainController. Diese Änderung war wichtig, da wir nun die Funktionen einheitlich behandeln können. So ist der MainController nur noch für die Inizialiserung des MainFrames und aller anderen Controller zuständig.
- Assoziationen zwischen diversen Controllern und der ProgramExecution des interpreter-Pakets wurden aufgelöst und durch eine ExecutionHandler-ProgramExecution-Assoziation ersetzt. So konnte die Verbindung zwischen der GUI und dem Rest des Systems gelockert werden.
- VariableViewController und BreakpointViewController wurden zusammen gefasst zu TableViewController, da sie das selbe Modell benutzen.
- Die ursprüngliche Editor-Klasse war View, Controller und Modell zugleich. Dies wurde verbessert und dadurch sind die Klassen Editor (als Modell), EditorController und EditorView entstanden.
- Der MiscController hat an Bedeutung verloren und entsprach nicht mehr dem aktuellen Entwurf, deshalb wurde die Klasse entfernt. Stattdessen haben die Frames, die gesteuert werden müssen, ihren eigenen Controller bekommen. Deswegen wurde das gui-Paket durch die Klassen HelpController, ParameterController und SettingsController ergänzt.
- Es wurden die Klassen ParameterFrame und ParameterController hinzugefügt, da nicht an die Übergabe der Parameter an die Main-Funktion gedacht wurde. Dies ist nun durch das Öffnen eines ParameterFrames möglich.

Aufgrund der größeren Änderungen folgt ein aktuelles Klassendiagramm des Paketes:


Core Functions
ExecutionHandler
+ExecutionHandler(messagesystem : MessageSystem)
+addGlobalBreakpoint(expression : Expression) : void
+parse(source : String) : void
-parseParserError(error : String) : String [*]
+run(sbreakpoints : StatementBreakpoint [*]) : void
+singleStep(sbreakpoint : StatementBreakpoint [*]) : v...
+verify(source : String)
+printAssertionFailureMessage() : void
+destroyProgramExecution() : void
+setAST(ast : Program) : void
+setPaused(paused : boolean) : void
+getAST() : Program
+getPaused() : boolean
+getParameterValues() : String [*]
+getProgramExecution() : ProgramExecution
MainFrame
-display : Display
-shell : Shell
-*Button : Button
-pauseIcon : Label
-runIcon : Label
+MainFrame(mainController : Core &, editor : Core &, ms : Core &)
+switchIcon(image : Image, image2 : Image) : void
+getShell() : Shell
+getDisplay() : Display
+openWindow() : void
MenuBar
-menu : Menu
-exitItem : MenuItem
-loadItem : MenuItem
-helpItem : MenuItem
-settingsItem : MenuItem
-evaluationItem : MenuItem
-aboutItem : MenuItem
-randomTestItem : MenuItem
+MenuBar(controller : MainController, shell : Shell)
+getMenuBarItemExit() : MenuItem
-initiateMenuBar(controller : Core &, shell : Shell) : void
MainController
+MainController()
-initMainFrame() : void
+widgetSelected(e : SelectionEvent) : void
+widgetDefaultSelected(e : SelectionEvent) : void
+runView() : void
+pauseView() : void
+stopView() : void
+getFileAsString(file : File) : String
+writeStringToFile(content : String, file : File) : void
MainClass
+main(args : String [*])
Interpreter/Parser/Verifier/ast
Interpreter
ParserInterface
Program
ProgramExecution
Settings
SettingsController
+SettingsController(settings : Settings)
+getTimeOut() : String
+getMemoryLimit() : String
+addView(frame : SettingsFrame) : void
+getSaveButtonListener() : SaveSettings
+invalidInput() : void
Settings
-timeout : int
-memoryLimit : int
-settingsChanged : boolean
-verifierPath : String
-Settings()
+getInstance() : Settings
+getTimeout() : int
+getMemoryLimit() : int
+setTimeout(timeout : int) : void
+setMemoryLimit(memorylimit : int) : void
+settingsChanged() : boolean
+saveSettings() : void
-loadSettings() : void
+getVerifierPath() : String
+setVerifierPath(verifierPath : String) : v...
SaveSettings
+widgetSelected(e : SelectionEvent) : void
+widgetDefaultSelected(e : SelectionEvent) : void
SettingsFrame
-shell : Shell
-memoryLimitTextField : Text
-timeoutTextField : Text
-saveButton : Button
-closeButton : Button
-errorLabel : Label
+SettingsFrame(parent : Shell, controller : SettingsControll...
+close() : void
+getSaveButton() : Button
+getCloseButton() : Button
+getMemoryLimitTextField() : Text
+getTimeoutTextField() : Text
+update() : void
+displayMessage(error : boolean, msg : String) : void
Help
HelpController
+HelpController(help : Help, helpBox : HelpBox)
+setHelpFrame(helpFrame : HelpFrame) : void
+getTableofContents() : String [*]
Help
-tableofContents : String[*]
-tags : String[*]
-Help()
+getTableOfContents() : String [*]
+getHelpByTOC(tocItem : String) : String
+getHelpByTag(tag : String) : String
+getTags() : String [*]
+getFileAsString(filename : String) : String
+loadHelpFile(filename : String) : String
+getDefaultStyle() : String
+getSmallStyle() : String
+getInstance() : Help
HelpFrame
-shell : Shell
-list : List
-browser : Browser
-closeButton : Button
+HelpFrame(shell : Shell, controller : HelpControll...
+getList() : List
+getCloseButton() : Button
HelpBox
-helpbox : Browser
-search : Text
-button : Button
+HelpBox(parent : Shell, style : int, editor : Editor)
+addSelectionListener(listener : SelectionListener)
+setText(text : String) : void
+getSearchText() : String
TreeViews
TableViewController
+TableViewController(breakpointView : BreakpointView, varView : VariableView, executionHandler : ExecutionHandler)
+updateVarView() : void
+checkValue(parent : Tree, type : String, id : String, tmp : value) : void
+widgetSelected(e : SelectionEvent) : void
+widgetDefaultSelected(e : SelectionEvent) : void
+checkValue(parent : TreeItem, type : String, id : String, tmp : Value) : void
+deactivateVarView() : void
+activateVarView() : void
+activateBreakpointView() : void
+getVarView() : VariableView
+getBreakpointView() : BreakpointView
BreakpointView
-global : Table
-addTextField : Text
-button : Button
+BreakpointView(parent : Composite, def : int)
+drawGlobalBreakpointItem(gbreakpoints : GlobalBreakpoint [*]) : void
+getGlobalBreakpoint() : Table
+getAddTextField() : StyledText
+getAddButton() : Button
VariableView
-tree : Tree
+VariableView(parent : Composite, def : int)
+addTreeItem(parent : Tree, type : String, id : String, value : String) : void
+addTreeItem(parent : TreeItem, type : String, id : String, value : String) : v...
+getVarTree() : Tree
Messages & Consoles
MessageSystem
+MessageSystem()
+addConsole(console : Console) : void
+removeConsole(console : Console) : void
+addMessage(category : MessageCategories, position : int, text : String) : v...
+notifyConsoles() : void
+clear(category : MessageCategories) : void
Message
-position : int
-text : String
+Message(category : MessageCategories, position : int, text : String)
+getCategory() : MessageCategories
+getPosition() : int
+getText() : String
<<Enum>>
MessageCategories
-ERROR
Console
-table : Table
+Console(parent : Composite, style : int, ms : messageSystem)
+updateConsole(messages : Message [*]) : void
ErrorConsole
-positionColumn : TableColumn
-errormessageColumn : TableColumn
-tableItems : TableItem[*]
+ErrorConsole(parent : Composite, style : int, ms : MessageSystem)
+updateConsole(messages : Message [*]) : void
MiscConsole
+MiscConsole(parent : Composite, msgsys : MessageSystem)
+updateConsole(messages : Message [*]) : void
VerifierConsole
-positionColumn : TableColumn
-errormessageColumn : TableColumn
+VerifierConsole(parent : Composite, msgsys : MessageSystem)
+updateConsole(messages : Message [*]) : void
Editor
EditorController
+EditorController(editor : Editor, editorview : EditorView)
+getEditor() : Editor
+modifyText(e : ModifyEvent) : void
+verifyKey(e : KeyEvent) : void
+focusGained(e : FocusEvent) : void
Editor
-source : String
+Editor()
+setSource(source : String) : void
+setView(view : EditorView) : void
+getSource() : String
+undo() : void
+redo() : void
-createMemento() : EditorMemento
-findKeywords(source : String) : void
-addKeyWordColor(position : int, keyword : String) : Keyword
-addBreakpoint(line : int) : void
EditorFrame
-parentdisplay : Display
-textfield : StyledText
-linenumbers : StyledText
-MAX_LINES : int = 500
+EditorView(parent : Composite, def : int, editor : Editor)
+updateView() : void
+getText() : String
+getTextField() : StyledText
+getLineNumbers() : StyledText
Keyword
-start : int
-length : int
-color : RGB
+Keyword(start : int, length : int, color : RGB)
+getStart() : int
+getLength() : int
+getColor() : RGB
-convertStringToRGB(colorString : String) : RGB
EditorMemento
-source : String
+EditorMemento(source : String)
+getSource() : String
Parameter & RandomTest
ParameterController
+ParameterController(exHandler : ExecutionHandler, msgsystem : MessageSystem)
+addParameterFrame(parameterframe : ParameterFrame) : void
+addRandomTestFrame(randomtestframe : RandomTestFrame, source : String) : void
+widgetSelected(e : SelectionEvent) : void
+createRandomTestValues(parameters : FunctionParameter [*]) : String [*]
-createRandomIntegerValue(begin : int, end : int) : int
-createRandomBooleanValue() : boolean
-createRandomIntegerArray(begin : int, end : int, size : int) : int [*]
-createRandomBooleanArray(size : int) : boolean [*]
+getParameterFrame() : ParameterFrame
+getRandomTestFrame() : RandomTestFrame
+widgetDefaultSelected(e : SelectionEvent) : void
RandomTestFrame
-shell : Shell
-count : Text
-intervals : Text[*]
-startButton : Button
+RandomTestFrame(parentShell : Shell)
+createFrame(parameters : FunctionParameter [*]) : void
+createEmptyFrame(s : String) : void
+getShell() : Shell
+getCount() : Text
+getIntervals() : Text [*]
+getStartButton() : Button
+getSaveButton() : Button
+close() : void
ParameterFrame
-shell : Shell
-values : Text[*]
-okButton : Button
+ParameterFrame(parentShell : Shell)
+createFrame(parameters : FunctionParameter [*]) : void
+getShell() : Shell
+getOkButton() : Button
+close() : void
Breakpoints
Breakpoint
-active : boolean
#Breakpoint()
+isActive() : boolean
+setActive(active : boolean) : void
GlobalBreakpoint
-expression : Expression
+GlobalBreakpoint(expression : Expression)
+getExpression() : Expression
StatementBreakpoint
-line : int
+StatementBreakpoint(line : int)
+getLine() : int
FileFrame
-shell : Shell
-chosenFile : File
+FileFrame(shell : Shell, def : i...
+close() : void
+getChosenFile() : File
AboutFrame
-shell : Shell
+AboutFrame(parentShell : Sh...
+close() : void

1.7 While-Sprache

- Zur Vereinfachung der Behandlung von Funktionsaufrufen im SMTLibTranslator wird ein Return-Statement nur noch am Ende von Funktionen (die main-Funktion ausgeschlossen) erlaubt. An diesen Stellen muss dieses Statement sogar stehen, sonst ist das Programm nicht syntaktisch korrekt. Dies wird schon im Parser erkannt.

- Funktionsaufrufe in Assumptions, Axioms, Ensures, Invariants und QuantifiedExpressions werden verboten, nur noch die eingebaute ¨length¨-Funktion ist erlaubt. Neben einer Vereinfachung des Beweisers dient dies auch der Benutzerfreundlichkeit. Durch die Funktionen msste nmlich intern im Single-Step-Modus durchgegangen werden, um jederzeit eine Unterprechung des Programmablaufs zu erlauben. Da die QuantifiedExpression aber eventuell ber einen sehr groen Bereich geprft wird, ist dies fr den Benutzer ein zu groer Aufwand. Die restlichen Statement-Arten werden auerhalb eines Statements gemeinsam geprft. Ein Funktionsaufruf im Singlestep wre gegen die Logik dieser Statements.

- Die Syntax der Arraydeklaration wurde gendert, da die Syntax im Entwurf unintuitiv war.

2 Umsetzung der Entwurfsentscheidungen

2.1 Visitor-Pattern

Das Visitor-Pattern bildet die Basis der Behandlungen des abstrakten Syntaxbaumes. So bilden der Interpreter, der TypeChecker und der SMTLibTranslator die Besucherklassen, die die Knoten des AST besuchen können.

Dazu besitzt die abstrakte Klasse ASTRoot, die Oberklasse jedes Element des AST ist, eine abstrakte Methode accept(ASTVisitor visitor), die einen Besucher entgegen nimmt. Das Interface ASTVisitor besitzt fr jede konkrete Unterklasse von ASTRoot eine eigene visit-Methode, die diese Klasse besucht. Als Beispiel seien hier visit(LogicalExpression logicalExpression) und visit(Loop loop) genannt. Diese Methode accept wird in allen konkreten Unterklassen des AST implementiert. Diese ruft mit visitor.visit(this) die entsprechende visit-Methode des Besuchers auf. Der Besucher seinerseits kann ein AST-Element element durch element.accept(this) besuchen, ohne die konkrete Klasse dieses Elements zu kennen.

Dieses Entwurfsmuster frdert die Erweiterbarkeit, da neue Besucher ohne nderung der AST-Klassen eingefhrt werden knnen. Auerdem ist es mglich, neue Klassen zum AST hinzuzufgen. Dann muss jeder Besucher nur die zustzliche visit-Methode implementieren.

2.2 MVC-Pattern

Es wurde bei der Implentierung der GUI stets auf das MVC-Pattern geachtet und es besteht nun eine klare Trennung zwischen den Modellen, Controllern und Views. Konkret bilden zum Beispiel die folgenden Klassen jeweils ein vollständiges MVC-System:

- Editor, EditorView und EditorController

- ExecutionHandler, BreakpointView/VariableView und TableViewController

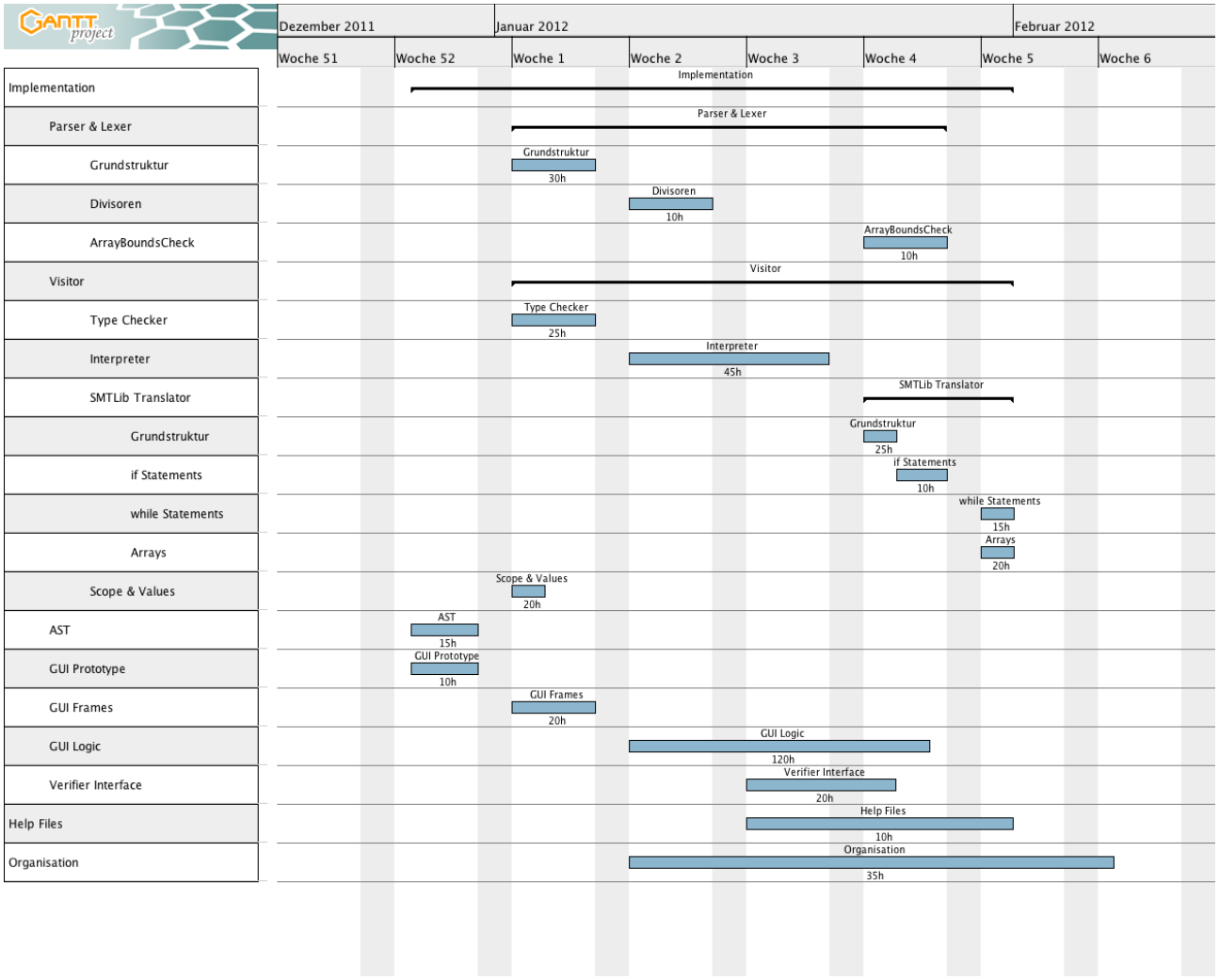
- Settings, SettingsFrame und SettingsController

2.3 Erweiterbarkeit der Beweiseranbindung

Auf die Möglichkeit einer einfachen Anbindung anderer Beweiser als Z3 wurde geachtet. Es ist nun sowohl möglich Z3 mit einer anderen Übersetzungsstrategie zu verwenden, indem der Klasse Z3 ein anderer ASTVisitor gegeben wird, als auch die Wiederverwendung des SMTLibTranslators für einen anderen Beweiser durch eine neue Klasse, die von Verifier erbt, und die Klasse SMTLibTranslator benutzt. Auerdem ist ein anderes Beweiserformat durch nderung des Translator-Visitors mglich.

3 Zeitablauf

3.1 Gantt-Diagramm



3.2 Parser und Lexer

Die Implementierung von Parser und Lexer dauerte - insbesondere aufgrund von Sonderfällen wie der Division durch Null und dem Prüfen von Arraygrenzen - länger als ursprünglich geplant.

3.3 Visitor-Klassen

Der Type Checker nahm weniger Zeit in Anspruch, als dafür veranschlagt war. So konnte parallel dazu schon mit der Implementierung der Scope- und Value-Klassen begonnen werden.
Die Fertigstellung des Interpreters verzögerte sich um etwa eine Woche, da hier Abhängigkeiten vom Interpreter bestanden.
Da die Designentscheidungen für die Beweiserschnittstelle überarbeitet wurden, fing dessen tatsächliche Implementierung später an als geplant. Durch die Verschiebungen bei Interpreter und Parser kam es ebenfalls zu einem Aufschub bei der Anbindung an den Beweiser mittles des SMTLib Translators.

3.4 GUI

Die Zeit, die für die Erstellung der GUI benötigt wurde, ist aufgrund von Änderungen des Klassenentwurfs höher ausgefallen, als ursprünglich geschätzt wurde. Durch die Änderungen des Entwurfs der GUI, die in der Implementierungsphase stattfanden, wurde diese weiter von anderen Komponenten entkoppelt, wodurch die GUI und die AST- bzw. Visitor-Komponenten größtenteils unabhängig voneinander entwickelt werden konnten. Hierdurch entstand im Zeitplan eine größere Flexibilität.

3.5 Buffer und Integration

Die Zeit für Buffer und Integration wurde hauptsächlich zur weiteren Implementierung verwendet. Die Integration fand zu großen Teilen während der Zeit der Implementierung der GUI Logic statt.