Vrije Universiteit Brussel

Faculty of Science
Department of Computer Science
and Applied Computer Science

TIAMAT: A Multi-touch Tablet IDE for AmbientTalk

Ayrton Vercruysse

Promotor: Prof. Dr. Wolfgang De Meuter
Advisors: Dries Harnie
Lode Hoste

2012-2013

Contents

Chapter 1

Introduction

1.1 Goal

The goal of this document is to be a guide with the organisation, implementation and evaluation of the TIAMAT project. It will explain made choises concerning the implementaion and will evaluate the relevances of the chosen solution. To illustrate this all the specifications of this project will be listed and we will explain how these are implemented. Next to a reflection about how we implemented the project we will alse provide some context. This being why the project is made, who will be using it and as last we will have a look at some related work. Next to the implementation we will have a social experiment with the application. We will distribute the app to several people and let them test the app. We will evaluate if the app is an improvement to the old apps which provide the possibility to create some AmbientTalk programs. Thanks to the implementationdetails the users of the application will be able to use this document as a reference to find what the possibilities of the app are. They can guide a user to use the app to it's fullest potential.

1.2 Scope

The scope of this project is a third bachelor thesis at the Vrije Universiteit Brussel. The 3th bachelor thesis is given at third bachelor students computersciences. The goal of this bachelorthesis is to let the bachelorstudents participate in some actual research on the university. This has been chosen over giving the students a predifined task.

1.3 Definitions and Acronyms

VUB	Vrije Universiteit Brussel
IDE	Integreted Develepment Enviorment
AST	Abstract Syntax Tree
MANET	Mobile Ad-Hoc Network
OHA	Open Handset Alliance
App	Application. Programs ons smartphones and tablets
MT4j	Multi-Touch for Java

1.4 Product Placement

This product is an IDE for AmbientTalk on Android device. AmbientTalk is a programminglanguage developed to mainly program within Mobile Ad-Hoc networks. MANETs are mostly used to program for mobile devices like smarphones and tablets. AmbientTalk is developed to be able to handle with possible disconnects within the MANET.

As AmbientTalk programs are made for Android devices one should be able to program on the Andeoid device itselve. Currently on pc's AmbientTalk is programmed in Eclipse. There are currently several apps which port Eclipse to Android. The only drawback is that these are not developed to be really useable on these devices. Most of them even assume that a keyboard is connected to the tablet. Other drawbacks are that the screen of the pc version of Eclipse would not fit a much smaller tablet or smartphonescreen.

The goal of this product is to be a usefull alternative to program AmbientTalk code on an Android device. We won't make use of the exsisting technologies of Eclipse but we will develop an entire new technique basic on the usage the internal AST. We will manipulate the AST directly in stead of create a sequence of statements which will be mapped to an AST afterwards.

By manipulating the AST we can make use of predefined templates. This will help with the fact to create sourcecode without defining of sequence of statements. For every possible part of an AST a template is created which can be added to the current AST making it possible to create an entire program.

1.5 Users

The goal of the Application is to be able to create programs on the Android devices. As AmbientTalk is programming language one should have basic knowledge of programming within the AmbientTalk enviorment. Without a knowledge

of the AmbientTalk language it is not easy to create a sound program. As AmbientTalk is developed at the university and momentarely is used as an example on different conferences the app can be helpfull to create a small example of an AmbientTalk program instead of always needing to use a computer.

1.6 Used Software

1.6.1 AmbientTalk

AmbientTalk is a programming language develeped at the VUB in 2005. The goal of the language was to focus on making programs within Ad-Hoc networks. This means that AbmientTalk is developed mainly to create programs on mobile devices. AmbientTalk combines elements from other programming languages such like Scheme (closures), Smalltalk (pure OO), Self (prototypes and delegation) and some other languages.

AmbientTalk was originally a part of a doctorate study made by Jessie Dedecker. His goals was to create an extension to the already excisting programminglanguage Pic% (pic-oh-oh). Pic% itselve was an expension of Pico. Pico was developed by professor Theo D'Hondt in 1996. Pico was developed merely to be used with educational purposes. Pico is being used as a programminglanguage to illustrate how programming languages are being made.

Pico got it's design principes and concepts from Scheme. The goal of Pico was to be a programming language based on simple rules, easy to extend. Thanks to the simplicity and extandability of Pico this language is often used to do some research to possible extensions and features for programming languages. This is the reason why a lot of offsprings of Pico all created, all with their own special attention to certain problems or extensions. When AmbientTalk was being developed special attention was given create distributed programs within an ad-hoc network. Momentarely AmbientTalk/2 is being used. It's the successor of the original AmbientTalk. Even though AmbientTalk/1 was already succesfull concerning the programmingfeatures for mobile Ad-Hoc networks it lacked some important features to create bigger software applications, such like exception handling,...

In 2006 Tom Van Cutsem and Stijn Mostinckx started developing AmbientTalk/2, de current version of AmbientTalk. The changes between AbmientTalk/1 and AbmientTalk/2 are quite big. There has been made a new total new design for most aspects of the language, including the syntax. The reason to change the syntax was to make AmbientTalk more accesible for people who don't have any exerperiance with Pico.

AmbtientTalk is still mainly used by students at the VUB to do some research.

But, slowly but surely, AmbientTalk became a usefull, handy programming language which was usable to create some rather big programs. Because the focus of AmbientTalk lies within the distributed networks and this is still an area in which a lot of research being done there aren't a lot of good alternatives. This made AmbientTalk a good alternative to create some real-life software.

The symbiosis with the underlaying Java Virtual Machine enables the possibility to use some parts of the Java programming languages within AmbientTalk, which makes it possible to combine within one project.

The renewing element of AmbientTalk is so big that it became the main subject of the Distributed and Mobile Programming Paradigms course teached at the VUB.

1.6.2 Google Android

1.6.3 MT4j

Chapter 2

Implementation

2.1 ASTs

2.1.1 Making ASTs

Description: Every piece of code used in the IDE will be directly linked to an AST. For this matter there has to be a possibility to create new ASTs from scratch. To create new AST trees we will need to implement the different parts of which an AST can exist. These different parts of an AST will be called a Node. We will start with a list of possible nodes with which basic programs can be made. Later extra kinds of nodes can be added to the AST. A superclass Node will be implemented which will organise the tree structure by providing operations like getParent(), which gets the parent of a Node, setChild(oldChild, newChild) which will set a child of a node and getChilderen() to get the childeren of a Node. By using these operations new trees can be created. The types of nodes that will be implemented are: Placeholder, Begin, Block, Definition, Value, Argument List, Function, Function Call, Function Definition, Operation, Table, Table Call Table Definition.
Priority: High
Status:

2.1.2 Deleting ASTs

Description: When we have an excisting AST and parts of this AST became irreleveant there should be a possibility to remove this AST. The removing of an AST, in total, or only a part of an AST will happen by replacing this AST by the Placeholder node. This possibility will be, as said in 1.1 implemented in the Node class, by means of the setChild(oldChild, newChild) function.

Priority: High
Status:

2.1.3 Replacing parts of ASTs

Description: When new Nodes will be created it's childeren, if there are any, will be make by Placeholders. When making a function for example, we'll implement the operation, but the two operands will be initialized with a Placeholder node. To be able to change these Placeholders to more relevant Nodes we should be able to replace existing Nodes by other Nodes, in this example maybe a Value, or an other Operation. This can be done on a similar way as the deletion of ASTs, by making use of the replaceChild(oldChild, newChild) method.
Priority: High
Status:

2.1.4 Rendering ASTs to text

Description: To be able to pass the constructed code to the AmbientTalk interpreter we have to convert the AST to an understandable format for the interpreter. The interpreter works with a textfile containing AmbientTalk code. This means that the ASTs have to be converted to text. Converting an AST to a text should be a function within each Node, calling the same function recursively on all of it's childeren.
Priority: High
Status:

2.2 Templates

2.2.1 Templates creating ASTs

Description: For convenience we will make use of templates. Templates will be frequently used parts of code that will be saved. The way templates will be stored in the memory will be by an AST. Every time a template is used withing the program a copy of this AST will be made and inserted into our current program. These templates can be frequently used functions, but mainly these will be the ASTs representing basic elements of the language like the if-then-else sturcture, when-discovered,...

Priority: High
Status:

2.2.2 Creating templates from XML

Description: To keep a clear view on what template functions we have, and make it easy for users to add templates, without having to know the implementation of the entire program we will keep the Templates in a separate XML file. When having an XML file containing the structure to create new Templates a function will be implemented that creates, from the XML file, a new AST, which can be used as a Template within the program.
Priority: High
Status:

2.2.3 Create templates from functions

Description: When a newly created function is often used by the user and he wants to save this function, the possibility to write this function, in correct XML, to the existing XML file will be provided. This means that whenever the program is terminated functions can be stored inside the XML file.
Priority: High
Status:

2.3 Interface

2.3.1 Rendering of nodes

Description: Each type of node has to have a screen representation. For each node we need a function that displays the Node on the screen, and that recursively calls the display function of all children of this node. We won't implement these displays within the separate types of nodes, discribed in 1.1, but for each node we will implement a separate display function. This with as goal that we can re-use our AST implementation without having to dead with the program specific display possibilities.
Priority: High
Status:

2.3.2 Lay-out engine

Description: We will make a render manager, this manager will render all nodes and make sure they are organised on a ordered way. The render mananger will use indetion to organise nodes across the screen. **Priority:** High
Status:

2.3.3 Creating functions on the spot

Description: Whenever a new function (or variable) is created within the program a link to call this function will be added to the menu's. This means that we can easily re-use a made variable or have an easy way to call an earlier created function.
Priority: High
Status:

2.3.4 Buttons

Menu's

Description: To keep track of all templates or calls that can be made we will need a menu that gives us the possibility to insert templates in to the code. We will use one a menu with some submenu's, devided for each kind of templates. These can be provided functions self implemented function, used variables,...
Priority: High
Status:

Run button

Description: Thanks to requirements 4.2 we can evaluate made code with an external AmbientTalk app. To do this we'll add a button to the main screen which will first make sure requirement 1.4 is done, the creation of a textfile with the current code, and later on run the AmbientTalk app, as described in requirement 4.2.
Priority: High
Status:

Delete button

Description: To be able to delete some code, as described in 1.2 we need a way to select what code will be deleted. Herefore we will implement a delete button on the screen, which deletes the current selected piece of code. This can be done by pressing this button or by dragging the selected code to this place. **Priority:** High
Status:

2.3.5 Views

Colorcoding keywords

Description: As in many languages the use of colorcoding should be done here aswell. We will try to achieve the same colorcoding as being used in the official AmbientTalk IDE in Eclipse.
Priority: Low
Status:

Colorcoding types

Description: Next to the colorcoding of keywords we will implement colorcoding on the types of words. This means that we will give Placeholders different colors, just like VariableNames,...
Priority: Low
Status:

Codefolding

Description: The use of AST should gives us the possibility to easily fold in certain parts (read piece of the AST) of our code. This can be done by just not rendering certain nodes, and it's childeren, of the AST.
Priority: Low
Status:

Move code

Description: By using the possibility of deleting parts of ASTs and saving parts of ASTs a copy-pasty system, or even moving of code to places that make sense belongs to the possibilities.
Priority: Low
Status:

2.3.6 Gestures

Selection of code

Description: To be able to replace, delete, move,... code we need a possibility to select certain parts of the code. This will be done by a tap or doubletap gesture. The selected code will be marked and it will be possible to makes changes to this code.
Priority: High
Status:

Pinch-to-zoom

Description: When starting on a piece of code by using the pinch-to-zoom function the selected part of code should be extended. This can be done by enlarging the selected area from the current AST to the parent of this AST.
Priority: High
Status:

90 degrees turning for comments

Description: When a piece of code is selected, and afterwards this piece of code gets turned around for 90 degrees, this piece of code should be commented out. Commented code should not be written to the textfile when this is called on this AST.
Priority: Low
Status:

Fast scrolldown

Description: When we want to scroll down a longer piece of code we can use the scroll with two fingers getures to get us scroll faster.
Priority: Low
Status:

2.4 Evaluating code

2.4.1 Writing code to textfile

Description: To make use of the AmbientTalk app we need a textfile with the AmbientTalk code. We can make use of the toText() function of each node to translate our current AST to plain text, and afterwards we will need to write this text, into a textfile.
Priority: High
Status:

2.4.2 Call external AmbientTalk app

Description: The external AmbientTalk app, which makes use of the textfile from the previous - insert number- should be called. To get this app called we will implement a button on the screen, which first gets the textfile created and afterwards calls the AmbientTalk app.
Priority: High
Status:

2.4.3 Return to TIAMAt after evaluating code

Description: The AmbientTalk app will evaluate the program. After the evaluation of this program and returning the result we should get back the our application to create the possibility to edit our current program
Priority: High
Status:

2.5 Features

2.5.1 Extra interface for comments

Description: An extra interface can be created in which we can store extra comments. This interface should only be called on object which could get any use of extra comments.
Priority: Low
Status:

2.5.2 Speaking comments

Description: A nice feature is to add spoken comments to our program. The possibility to add spoken comments will be integrated in the extra interface for spoken comments, announced in requirement 5.1. There will be added extra buttons to record comments and play these comments.
Priority: Low
Status:

2.5.3 Selector for Java classes

Description: Within the AmbientTalk language it's possible to make use of a Java class selector. Adding this feature to this implementation will make sure that our project becomes a better alternative to the pc one.
Priority: Low
Status:

2.5.4 Multiple tabs

Description: If we want to make bigger programs it would be easy to be able to have multiple tabs. Certainly when big programs exsist of multiple files. To make this possible we would need to save multiple AST trees and implement a possibility to switch between the different tabs.
Priority: Low
Status:

2.5.5 Saving files

Description: As we will implement a write to XML to save new Template into an XML file we could use this procedure to save an entire program in XML, and later rebuild our AST and on this way reconstruct our program.
Priority: Low
Status:

Chapter 3

Related work

3.1 TouchDevelop

TouchDevelop is a programming enviorment made for Windows phones. TouchDevelop allows you to create some simple applications for your phone and add these later to the Windows Phone Marketplace. Because TouchDevelop is made to create applications for the device on which they are developed they can make use of really powerfull templates. TouchDevelop offers you the possibility to make use of most of the built in sensors and files upon your device, such like media files, web cloud, network, gps,... This makes it possible to develop apps which only react when one shakes the phone or only when the phone is faced up or down.

TouchDevelop is ment for students and people who like to program in their free time, and not for professional developers. It makes the making of own applications so easy that everyone with a basic knowledge of programming can start making his own applications. Different than with TIAMAT these applications are being made within an entire new programming language. This language is made from scratch to be work with Windows Phones, while AmbientTalk was developed to work on plurar platforms.

The goal of TouchDevelop was not to be and IDE for an exsisting programming language, like TIAMAT, but to develop an entire way to create some application on the device for which the application is ment to run.

3.2 Touch Scheme

Touch Scheme is developed by a programmer who wanted to program during his vacation on his iPad. To be able to do so he developed an app. His main idea was to edit the parse tree within the app.

3.3 Editting Haskell as AST rather than text

Christopher Done has created 'A concept for editing Haskell as an AST rather than text'. This works in a similar way as Tiamat. The idea here is to change certain parts of the AST to the piece of code we want. In the video made by Chistopher Done you can see how he edits his Haskell program by clicking on the places where new parts of the code should come. He fills in the parts of the AST by clicking on them and choosing from a list (of templates) what should be entered on this spot. This idea is very similar to TIAMAT where we work with Placeholders which can be replaced with actual nodes within the AST of the program.

Chapter 4

Evaluation

Chapter 5

Conclusion