ASE TEAM 4

SYSTEM DOCUMENTATION

HUSACCT – Analyse C#

June 24, 2013

BARUCH BERGER - GERARD BOSMA - BART GOES
CHRISTIAAN DE JONG - NIELS HUBREGTSE

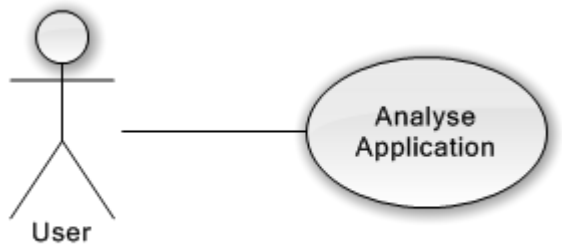
CONTENTS

INTRODUCTION

The following document describes the functionality and internal workings of the component analyse C#. The component is designed to work independently according to component-based architecture. The C# analyse component is a part of the analyse component. The C# analyse component has the task to find the dependencies in C# source files and store them in a Famix model. This model is a language independent model.
To do this, a 3th party lexer and parser are used to convert the c# source code to an abstract syntax. An abstract syntax tree (AST) is a tree which contains all the logical information about the source code. To extract the information that is needed, the convert controller, which is the main entry point of the AST, walks through the tree to extract any data useful for this case and sends it to the generators. There are multiple generators and they have the responsibility to extract information from a specific part, for instance when we find an import, we delegate it to a generator which contains the logic to handle a tree with an import, and store it in the Famix model.

FUNCTIONALITY

USE CASE

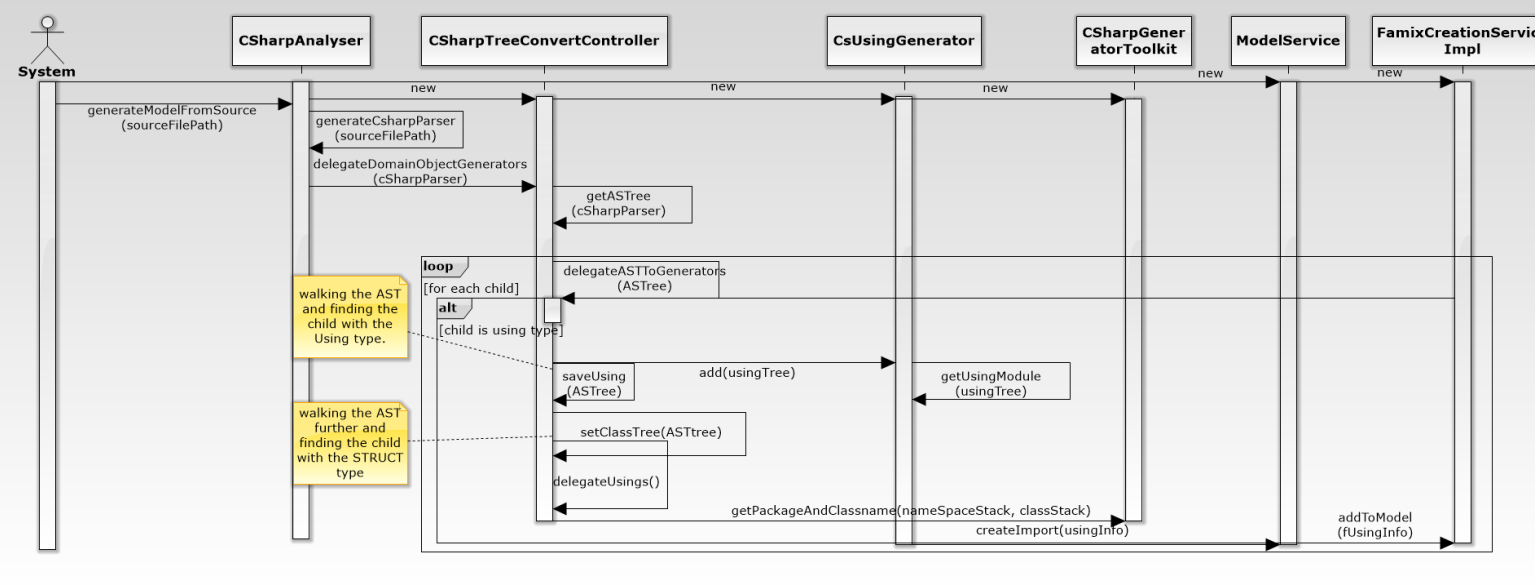


1 ANALYSE C# USE CASE

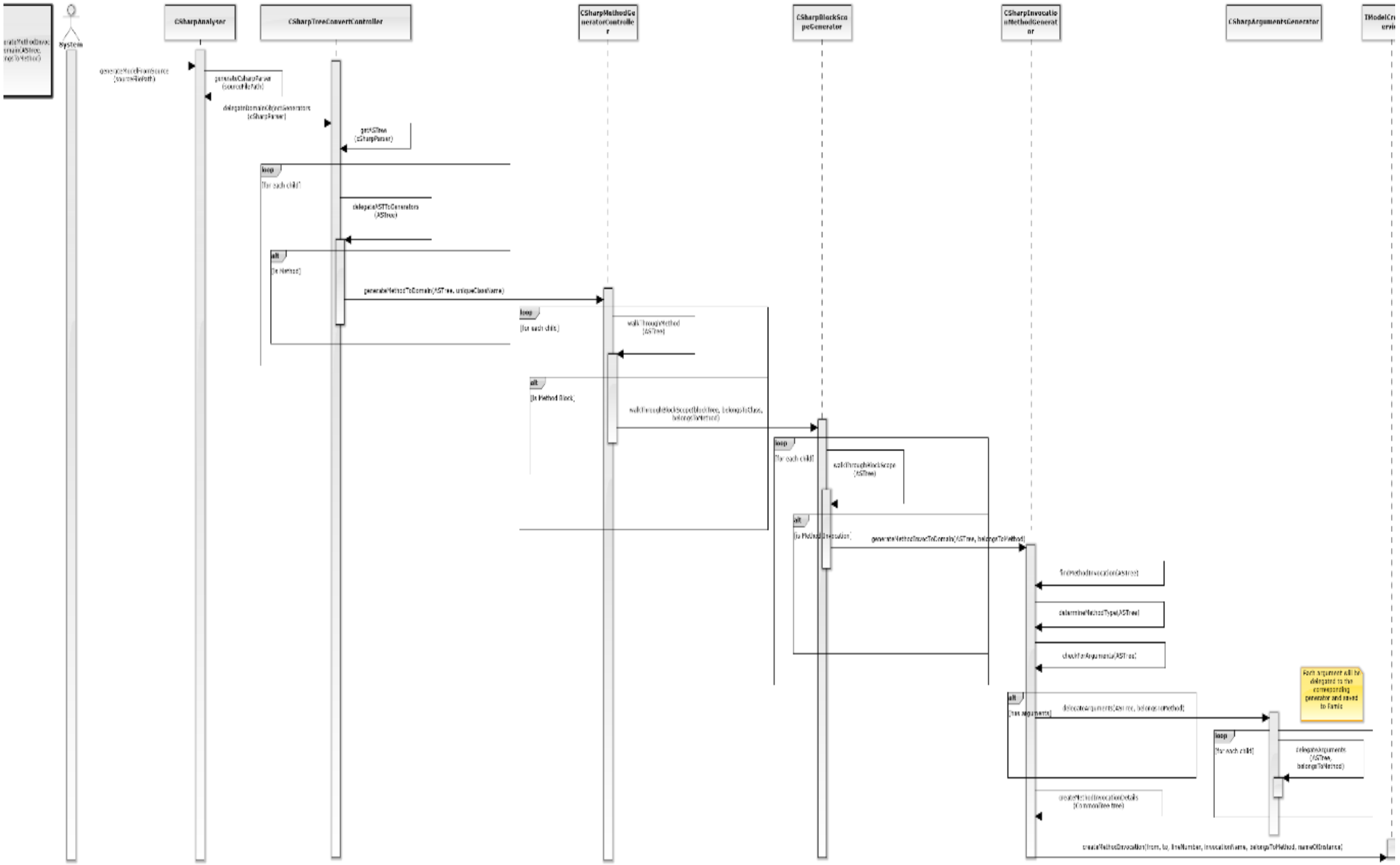
ID	**C# Use case 1**
Name	Analyse Application
Actors	End-User
Precondition	Actor submitted the root path or paths of an application and language.
Postcondition	The application is scanned and a FAMIX model is generated.
Description	Analyses the application and stores the information, making it possible for other components to query the service.
Constraints	- The duration of the analyse of a project must be less than 15 minutes. (NF6 architecture Notebook vs 2012-03-08). - The information found should be stored in a language independent model.

SEQUENCE DIAGRAM

To show the general idea of the project we have elaborated two sequence diagrams. The first one shows how a *'using'* is handled once it is found and the second one shows when and how invocations are found and stored.



2 USING SEQUENCE DIAGRAM



3 INVOCATION SEQUENCE DIAGRAM

REQUIREMENTS

This part will be split into functional requirements, non-functional requirements and constrains.

FUNCTIONAL REQUIREMENTS

The following functional requirements concern the c# analyse part.
Different types of dependencies can be established in object oriented program code. All types of dependencies, analyzable with static analysis, should be recognized by HUSACCT.

Direct dependency: A class uses another class with explicit reference to the class. An import command is specified.

Table 1. Direct dependencies

CFR#	Category	Dependency Types	Example Code from Class1
5.1	Method call	Method Class method Constructor	linkToC2 = new Class2();
5.2	Variable access	Instance variable Instance variable, constant Class variable Class variable , constant Enumeration	private String variable; variable = linkToC2.variable;
5.3 5.4	Inheritance	Extends class Extends abstract class Implements interface	public class Class1 extends Class2 { }
5.5	Type declaration	Instance variable Parameter Return type Exception Type cast	private Class2 linkToC2;
5.6	Annotation	Class annotation	@Class2
5.7	Import	Class import	Import ModuleB.ModuleB1.Class2;

CFR#: Identification of the Core Functional Requirement as specified in 2012
Indirect dependency: A class uses another class without an explicit reference to the class. No import command is specified.

Table 2. Indirect dependencies

CFR#	Category	Dependency Types	Example Code from Class1
5.1	Method call	Method Class method Constructor	private String variable; variable = linkToC2.linkToC3.method();
5.2	Variable access	Instance variable Class variable	private String variable; variable = linkToC2.linkToC3.variable;
5.3	Inheritance	Extends - extends	public class Class1 extends Class2 { }

5.4		Extends - implements Implements - extends	Example Code from Class2: public class Class2 extends Class3 { }

Double indirect: variable = linkToC2.linkToC3.linkToC4.variable;

NON-FUNCTIONAL REQUIREMENTS

The following non-functional requirements concern the C# analyse part.

NFR#	ISO 9126 attr.	Requirement	Reference
1.	**Functionality**		
1.1.	Accuracy	All true violations must be reported. False negatives must not occur.	Rutar 2004;
1.2.	Accuracy	All reported violations must be true. False positives must not occur.	Rutar 2004;
1.3.	Suitability	Configuration of the user language must be possible.	Rutar 2004;
1.5.	Suitability	The tools must be useful on existing code, but also for projects from the start.	Rutar 2004;
1.6.	Suitability	The architecture definition, analysis and validation data of a compliance check must be reusable (for other versions of the project (evolution); for other projects (arch. definition)).	Knodel 2007;
2.	**Reliability**		
2.1.	Maturity	The tool must not go down in case of a failure, but generate a meaningful error message.	
2.2.	Fault tolerance	There must be no restrictions in the size of the project regarding number of classes, lines of code, components...	Knodel 2007;
3.	**Usability**		
3.7.	Understandability	A violation must be reported only once.	Rutar 2004;
4.	**Efficiency**		
4.1.	Time behavior	Tools must not take long (<= 15 min; 1.000.000 LOC) to analyse/validate the code, and to generate an error report.	Rutar 2004;
5.	**Maintainability**		
5.1.	Analyseability, Testability	Taking over the development of the tool by other development teams must be unproblematic.	

NFR#: Identification of the Non Functional Requirement as specified in 2012

CONSTRAINTS

There are a few constraints for this project which are relevant for the analysis of C#. The biggest one is the Famix model, which is said to be language independent but it isn't. The biggest issue with Famix is that there is no support for Lambdas. The Famix model also doesn't save the file name because in java this is the same as the class, but in C# this is not necessarily the case.

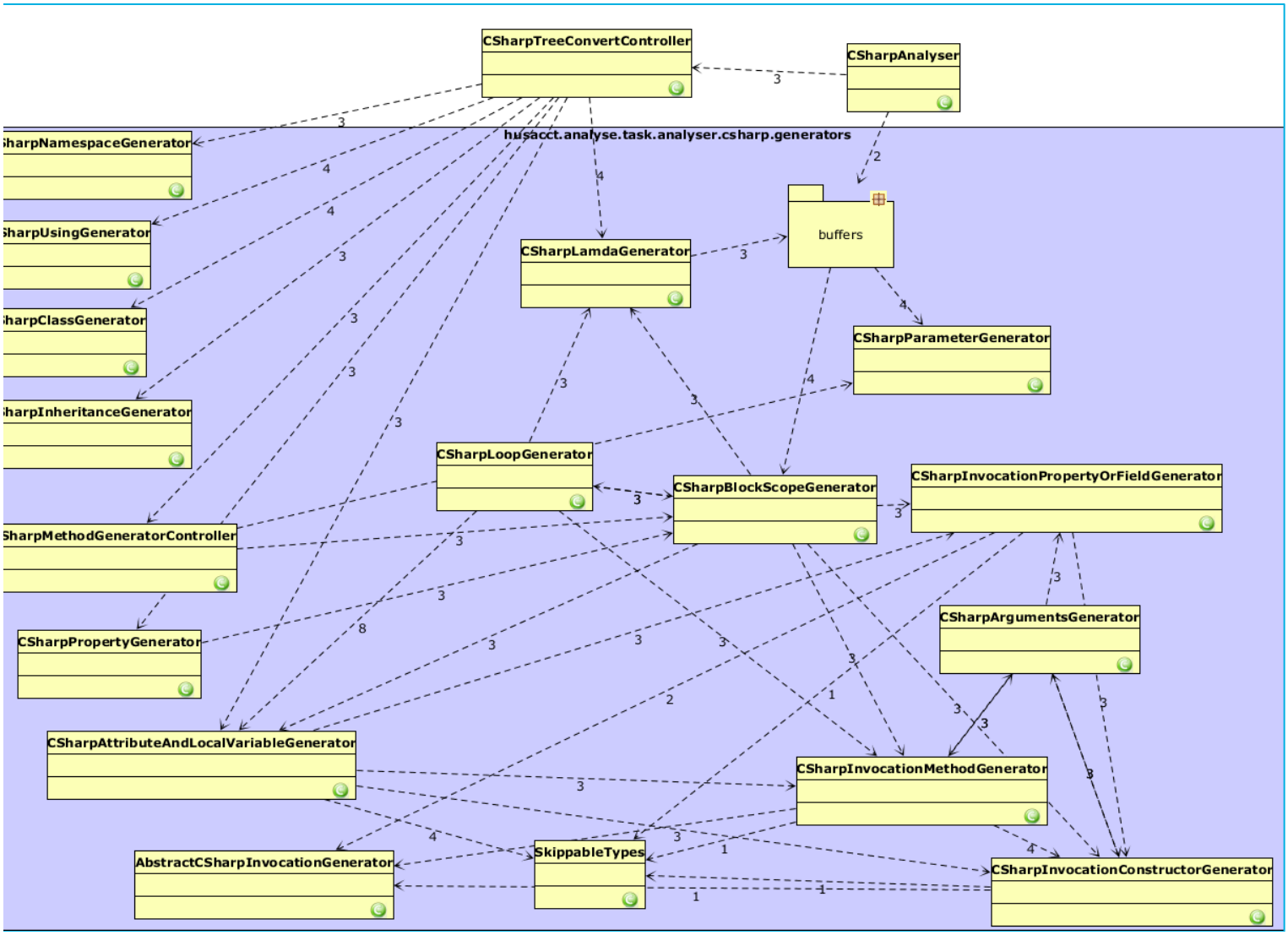
The version of ANTLR is also a constraint, this couldn't be updated to the correct version because the Java analyse component uses the same ANTLR Library as C#. This did probably invoke the problem that a lambda expression wasn't found as a lambda but as attribute or local variable.

SOFTWARE PARTITIONING MODEL

The model that is shown here is only an overview of the C# analyse component and does not contains the global analyse structure.

For the view ability the following classes have been removed:

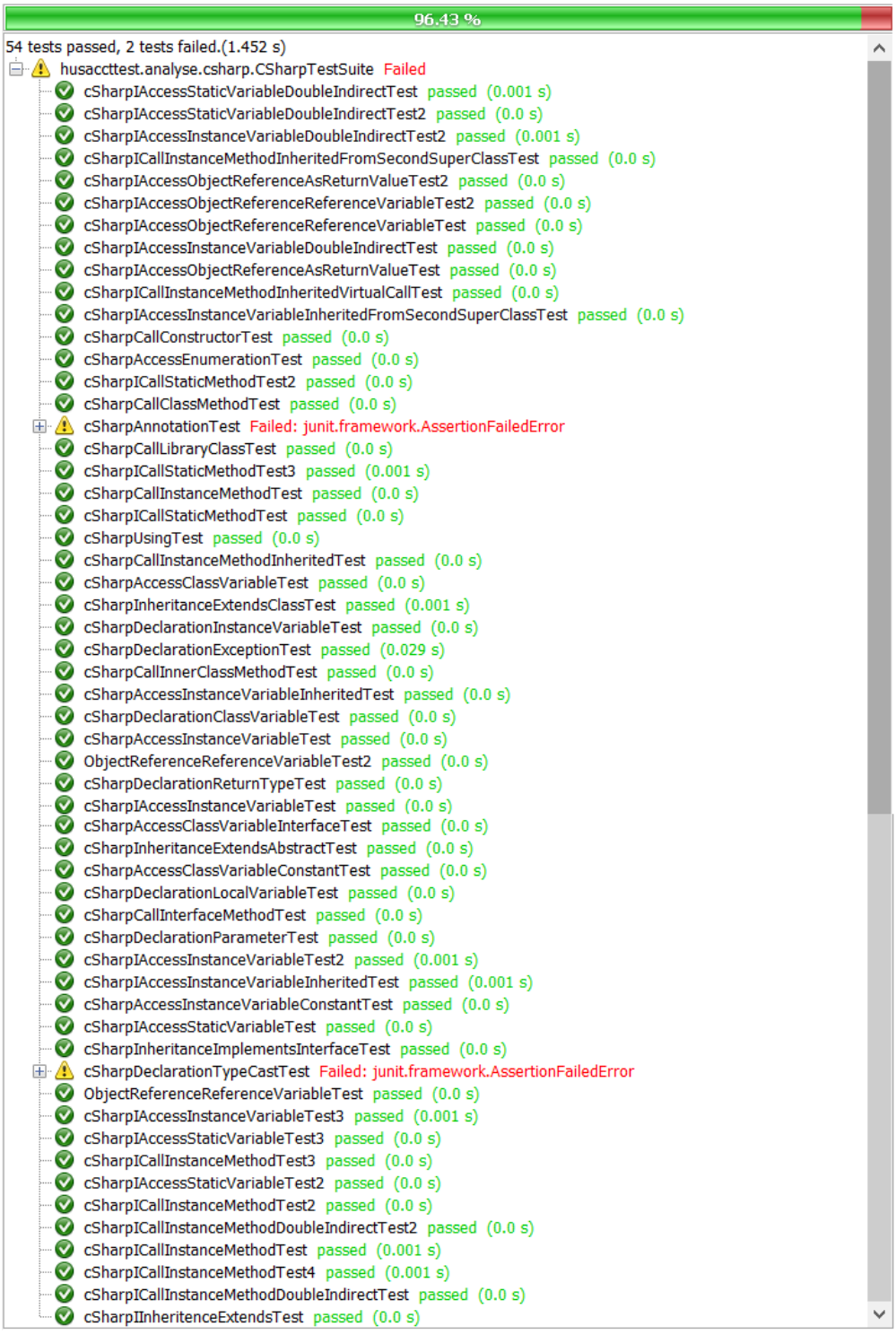
- CSharpToolkitGenerator, this class holds functions that multiple generators use.
- CSharpGenerator, this class is extended by all specific generators and merely provides a handle to the Famix façade.
- CSharpException, can be used to catch and print exceptions.



ANALYSE C# MODEL

TESTS

To ensure the quality and maintainability of this project, unit tests have been written. Those unit tests test the same things as the java tests. The project on which the tests are running is implemented in the *testproject* folder.



5 ANALYSE C# TEST RESULTS (21-06-2013)

FUTURE WORK

The C# analyse is not fully finished yet. The key functionalities are supported, but there is still room for improvements or extensions.

KNOWN BUGS

As of right now there are no known bugs.

FUTURE IMPROVEMENTS

Currently lambdas are only partially supported and the way they are found and stored isn't clean this definitely needs an upgrade. To support this correctly the following has to be added to the Famix:
The ModelService is a service which adds a FamixObject to FamixModel. We need to create a FamixLambda for this purpose first. In this class we extend FamixStructuralEntity, and then we add Lambda-specific logic. Next, we add a method to the IModelService, this is an interface that is necessary for the ModelService.

The time an analysis of C# source code takes could also be improved, however it is within the non-functional requirement.

FUTURE EXTENSIONS

A start has been made on multiple-projects support, however this wasn't implemented. In the future this might be a useful extension and to implement it you would have to give every object in the Famix an id.
Annotations are not supported right now, this could be useful to have. The reason why we didn't added it was because in C# you can give more information that just a class this makes it a lot more work and there wasn't a lot of time to do this.