

EINDHOVEN
UNIVERSITY OF TECHNOLOGY

C++ JETBRAINS MPS EXTENSION

VERSION 1.0

Unit Testing Plan

Project team:

Nicholas DONNELLY
Daan DRIJVER
Bart van HELVERT
Julia HOFs
Daan van der KALLEN
Bart MUNTER
Joris ROMBOUTS
Job SAVELSBERG
Bart SMIT
Remco SURTEL
Jelle van der STER

Customer:

Dmitrii NIKESHKIN

Supervisor:

Önder BABUR

Project Managers:

Wout de RUITER
Wesley BRANTS

July 4, 2018

Abstract

This Unit Testing Plan (UTP) contains the description of the procedures for testing of the C++ extension to the JetBrains Meta Programming System (MPS). It describes the different tests, their environment, and how to execute them. This document complies with the software requirements laid down in the Software Requirements Document¹ (SRD). This document also complies with the ESA software standard.²

Contents

1	Introduction	7
1.1	Purpose	7
1.2	Overview	7
1.3	List of Definitions	7
1.3.1	General Definitions	7
1.3.2	C++ Definitions	7
1.3.3	Abbreviations	8
1.4	List of references	8
2	Test Plan	9
2.1	Test Items	9
2.2	Features to be tested	9
2.3	Test Deliverables	9
2.4	Testing Tasks	9
2.5	Environmental Needs	10
2.6	Test Case Pass/Fail Criteria	10
3	Test Case Specifications	11
3.1	Typesystem Rules	11
3.2	C++ Code Generation	26
3.2.1	Classes	26
3.2.2	Namespaces	35
3.2.3	Auto	37
3.2.4	Casting	38
3.2.5	CharTypes	40
3.2.6	NumericTypes	41
3.2.7	New & Delete	45
3.2.8	Nullpointer	47
3.2.9	Operator Overloading	49
3.2.10	Specifiers	52
3.2.11	Templates	53
3.2.12	Virtual	56
4	Test Procedures	58
4.1	Typesystem Rules	58
4.1.1	Identifier	58
4.1.2	Purpose	59
4.1.3	Procedure Steps	60
4.2	C++ Code Generation	60
4.2.1	Identifier	60
4.2.2	Purpose	60
4.2.3	Procedure Steps	60
5	Coverage Report	61
5.1	Type System Rules	61
5.2	C++ Code Generation	61

6	Test Report	62
6.1	Type System Rules	62
6.2	C++ Code Generation	75

Document Status Sheet

General

Document title: Unit Testing Plan
Identification: UTP/ 1.0
Authors: B.I. Munter, B.A.J. van Helvert, J. van der Ster, R.J.A. Surtel,
D. van der Kallen, N.J. Donnelly, J. Savelberg
Document status: final version

Document History

Version	Date	Author(s)	Changes
0.1	25-05-2018	B.I. Munter	Initial version. Document layout
0.2.1	30-05-2018	B.A.J. van Helvert	Added type system and scoping rules
0.2.2	5-06-2018	B.A.J. van Helvert	Changed testcases to new reference format
0.2.3	7-06-2018	J. van der Ster	Added overview section, and needs, tasks and pass/fail criteria sections. Added namespace testcases
0.2.4	14-06-2018	R.J.A. Surtel	Added typecheck- ing for Global- VarDecCPP
0.3	14-06-2018	J. van der Ster	Added C++ code generation testcases
0.3.1	14-06-2018	D. van der Kallen, N.J. Donnelly	Added testcases for operator overload- ing, auto, function and class templates, and casting
0.4	14-06-2018	J. van der Ster	Added C++ code generation test procedure. Added template for test report
0.4.1	15-06-2018	J. Savelberg	Added testcases for this keyword

0.4.2	18-06-2018	J. van der Ster	Modified C++ code generation testcases, modified template for test report, added remaining type-checking testcases
0.5	19-06-2018	J. van der Ster	Revised the introduction, test plan, test case specification and test procedure sections.
0.5.1	19-06-2018	J. van der Ster, B.I. Munter	Added test cases for numeric types. Added coverage report
0.6	20-06-2018	J. van der Ster	Added result figures for type system test cases

Document Change Records

Section	Applicable Changes
0.0	Added Initial Document Layout
1.1	Added Purpose
1.3	Added General Definitions, Most Common C++ Definitions and Abbreviations List
2.1	Added Test Items
2.2	Added Features to be tested
2.3	Added Test Deliverables
3.1	Added Type System and Scoping rules
3.1	Changed Testcase format
2.4	Added Testing Tasks
2.5	Added Environmental Needs
2.6	Added Test Case Pass/Fail Criteria
3.1	Added Namespace Testcases
1.2	Added Overview
3.1	Added Class Testcases
4.1	Added Type System Test Procedure
3.2	Added C++ Code Generation Testcases
3.1	Added Operator Overloading and Casting Testcases
3.1	Added Templates and Auto Testcases
4.2	Added C++ Code Generation Test Procedure
6	Added Test Report Template
3.1	Added This Testcases
3.2	Modified C++ Code Generation Testcases
6	Modified Test Report Template
3.1	Added ArrayAttributeInit, ConstructorInitializable, StaticClassMethodCall, TemplateTypeDef and TryCatchStatement Testcases
1	Revised Introduction
2	Revised Test Plan
3.1	Revised Type System Rules
4.1.1	Updated Identifier Mapping
3.2	Added NumericTypes Testcases
5	Added Coverage Report
6.1	Added Test Report Result Figures

1 Introduction

1.1 Purpose

The Unit Testing Plan document can be seen as an overview of the developers' tests for the separate parts of the C++ extension to JetBrains MPS, based on the already existent SRD.¹ This is in contrast with the Acceptance Test Plan³ (ATP), which contains the tests designed to check conformance to the URD.⁴

1.2 Overview

Section 2 describes the items that need to be tested. Section 3 specifies the expected input and output of each uniquely identified unit test. Section 4 describes the testing procedures. Finally, sections 5 and 6 report the total coverage and test results, respectively.

1.3 List of Definitions

1.3.1 General Definitions

MPS	A meta programming system created by JetBrains used to create and utilize domain specific languages.
mbeddr ⁵	An extension to MPS created by Itemis, which allows users to develop, among many other things, C applications.
TextGen	A language component that handles the conversion from elements declared within the MPS editor, to basic text or usable code.
Generator	An mbeddr component that creates code files, header files and executables, containing the edited code.

1.3.2 C++ Definitions

Expression	A single statement containing a combination of values, constants, variables, operators and functions that produces a value.
Namespace	A scoping mechanism in the C++ environment to deal with conflicting names in large C++ projects. Every class included in a namespace is accessible from outside the namespace.
Operator Overloading	Redefining a previously defined function or operator.
Statement	A block of code that expresses some action to be carried out.
Template	A type of C++ declaration that allows types to be inferred at compile time, so identical code can be reused by many object types.

1.3.3 Abbreviations

ADD	Architectural Design Document
ATP	Acceptance Test Plan
MPS	JetBrains Metaprogramming System
SRD	Software Requirements Document
URD	User Requirements Document
UTP	Unit Testing Plan

1.4 List of references

¹ Eded, (2018), *Software Requirements Document*, version 0.3

² ESA, 1991, *software engineering standards*, ESA PSS-05-0 issue 2

³ Eded, (2018), *Acceptance Test Plan*, version 0.3

⁴ Eded, (2018), *User Requirements Document*, version 1.0.1

⁵ mbeddr, 2018, *The embeddr platform*, Web. <http://mbeddr.com/platform.html>. Accessed 01 May 2018.

2 Test Plan

2.1 Test Items

The software to be tested is the C++ Language Extension for the MPS environment.

2.2 Features to be tested

An MPS language consists of many components. For unit testing, the only components we consider are the Generator, TextGen, and Typesystem components. Besides checking types, the Typesystem component also covers scope checking and shows error/warning messages whenever necessary. All other components (Concepts, Editor, Behavior) are described in the ATP³ and are tested during the Acceptance Test.

MPS contains built-in test concepts for language testing. We use *NodesTestCases* for writing the tests for the Typesystem. MPS doesn't provide a testing solution for the Generator and TextGen. To solve this, we utilize the mbeddr Buildconfig concept to create and run test cases in the generated code. This allows us to receive compile errors in the editor when either the TextGen or the Generator fails to generate valid C++ code.

2.3 Test Deliverables

The deliverable will be in the same format as all the other tests in the mbeddr platform.⁵ Two separate packages are delivered: one containing all the executable test cases, used to test the Generator and TextGen components, and another one containing the Typesystem test cases.

After the test plan has been successfully executed, the following documents need to be delivered:

1. The results of the tests described in the UTP.
2. Problem reports (if any).

2.4 Testing Tasks

Before the testing starts, the following tasks must be fulfilled:

1. Design and specify all unit tests.
2. For each unit test, create input data within MPS.
3. Ensure all environmental needs for each unit test are satisfied.

When the above tasks are complete, unit testing can be performed according to the procedures described in section 4.

2.5 Environmental Needs

The following resources are necessary for unit testing:

- JetBrains MPS v2018.1
- Java 1.8
- Mbeddr CPP fork
- GCC 4.8
- GDB 7.6
- CBMC 5.6
- Graphviz 2.30
- Ant 1.9
- Make 3.81

2.6 Test Case Pass/Fail Criteria

For each test case, the described criteria should be met to pass that specific test case. An individual test case is passed when it produces the specified output as described in section 3.

3 Test Case Specifications

The following chapter will describe all the test cases in the C++ MPS extension. The test cases will follow the structure below:

Test component: Which items will be tested, the items will be in the form of `<model>/<concept>`.

Input specification: The input which will be tested.

Output specification: specification: The expected outcome.

Environmental needs: Other requirements for the test to have the specified output on the given input.

3.1 Typesystem Rules

UTS-1

Test component: typesystem/[check_AttributeDeclaration](#)

Input specification:

1. Attribute is prefixed by both `constexpr` and `inline`.
2. Attribute is prefixed by both `static` and `mutable`.
3. Attribute is prefixed by `constexpr` without a body.
4. Attribute is prefixed by `const` without a body.

Output specification:

1. Show [ConstexprAlreadyImplicitely](#) warning.
2. Show [StaticDataMemberCantBe](#) error.
3. Show [ConstantDataMemberMust](#) error.
4. Show [ConstantDataMemberMust](#) error.

Environmental needs: The attribute is declared inside a valid class.

UTS-2

Test component: typesystem/[check_MethodDeclaration](#)

Input specification:

1. Method is not virtual and has no body.
2. Method is prefixed by both `constexpr` and `virtual`.
3. Method is prefixed by both `constexpr` and `inline`.
4. Method is prefixed by both `static` and `virtual`.
5. Method is prefixed by both `static` and `volatile`.
6. Method is prefixed by both `static` and `const`.

Output specification:

1. Show [NonpureVirtualMethodMust](#) error.
2. Show [ConstantExpressionMember](#) error.
3. Show [ConstexprAlreadyImplicitely](#) warning.
4. Show [StaticMemberCantBeVirtual](#) error.
5. Show [StaticMemberCantBeVolatile](#) error.
6. Show [StaticMemberCantBeConst](#) error.

Environmental needs: The method is declared inside a valid class.

UTS-3

Test component: typesystem/[check_IPureVirtualFlag](#)

Input specification:

1. Method is prefixed by pure, but not virtual.

Output specification:

1. Show [NonvirtualMethodCanNot](#) error.

Environmental needs: The method is declared inside a valid class.

UTS-4

Test component: typesystem/[check_ClassType](#)

Input specification:

1. Creating a class instance of a class within that class.
2. Creating a class instance of a nested class without using the outer class prefix.

Output specification:

1. Show [ClassMayNotContainInstance](#) error.
2. Show [CantReferenceInnerClass](#) error.

Environmental needs: The nested class is declared inside a valid class.

UTS-5

Test component: typesystem/[check_InheritanceInstance](#)

Input specification:

1. Having a class inherit from itself.
2. Having a class inherit from an unexported class that exists within an imported module.

3. Having a class inherit from an exported class that exists within a module that is not imported.
4. Having a class inherit from a template class without specifying template types.

Output specification:

1. Show [ClassCannotExtendItself](#) error.
2. Show [YouCantExtendAnUnexported](#) error.
3. Show [YouCantExtendClassFrom](#) error.
4. Show [TemplateClassTypeWithout](#) error.

Environmental needs: The current module contains three valid classes and one valid template class, imports an existing module containing a valid unexported class, and does not import an existing module containing a valid exported class.

UTS-6

Test component: typesystem/[check_InheritanceNameCollision](#)

Input specification:

1. Class inherits from two classes that contain an accessible attribute with the same name.
2. Class inherits from two classes that contain an accessible method with the same name.

Output specification:

1. Show [NameCollisionAttribute](#) error.
2. Show [NameCollisionMethod](#) error.

Environmental needs: The classes involved are correctly declared in isolation.

UTS-7

Test component: typesystem/[check_ClassConstructor](#)

Input specification:

1. Destructor contains parameters.
2. Destructor is prefixed by constexpr.
3. Constructor is prefixed by both constexpr and virtual.
4. Constructor name is different from direct parent class name.
5. Destructor name is different from direct parent class name.
6. Wrong number of arguments is given to a constructor.

Output specification:

1. Show [DestructorsMayNotHave](#) error.
2. Show [DestructorCantBeConstant](#) error.
3. Show [ConstantExpressionMember](#) error.
4. Show [CantHaveConstructorFor](#) error.
5. Show [CantHaveDestructorFor](#) error.
6. Show [WrongNumberOfArguments](#) error.

Environmental needs: The constructor is declared in a valid class.

UTS-8

Test component: typesystem/[check_ClassDeclaration](#)

Input specification:

1. Class contains multiple destructors.

Output specification:

1. Show [ClassesMayOnlyHaveOne](#) error.

Environmental needs: The destructors are declared in a valid class.

UTS-9

Test component: typesystem/[check_LocalClassVariableDeclaration](#)

Input specification:

1. Class instance constructor contains no parameters.

Output specification:

1. Show [YouShouldSelectConstructor](#) error.

Environmental needs: A correctly declared class is instanced and has a valid constructor.

UTS-10

Test component: typesystem/[check_IAMConstructorInitializable](#)

Input specification:

1. Calling a base constructor from a deconstructor.

Output specification:

1. Show [InitializersAreNotAllowed](#) error.

Environmental needs: A correctly declared class contains a constructor and a deconstructor.

UTS-11

Test component: typesystem/[typeof_ConstructorInitializedAttribute](#)

Input specification:

1. A constructor contains a constructor initialized attribute for `foo` with the value 1.

Output specification:

1. Type system error.

Environmental needs: A correctly declared template class `Foo` with template type param `T` contains a public field `foo` of type `T`, and a public method `bar` with no arguments and return type `T`.

UTS-12

Test component: typesystem/[check_StaticClassMethodCallRule](#)

Input specification:

1. Referring to a static method that exists within a class.

Output specification:

1. Show [MethodOnClassIsStatic](#) warning.

Environmental needs: A correctly declared class contains a static method.

UTS-13

Test component: typesystem/[check_NamespaceDeclaration](#)

Input specification:

1. Namespace name is the same as another declared namespace.
2. Namespace name is the same as a top namespace.

Output specification:

1. Show [DuplicateNameCanMakeVariable](#) warning.
2. Show [DuplicateName](#) error.

Environmental needs: Another namespace is already correctly declared with the given name.

UTS-14

Test component: constraints/[NamespaceAttributeRef_Constraints](#)

Input specification:

1. Attribute does not exist within given namespace.

Output specification:

1. Show [ReferenceIsOutOfScope](#) error.

Environmental needs: The namespace reference is incorrectly adjusted after selecting a valid attribute.

UTS-15

Test component: constraints/[NamespaceMethodCall_Constraints](#)

Input specification:

1. Method does not exist within given namespace.

Output specification:

1. Show [ReferenceIsOutOfScope](#) error.

Environmental needs: The namespace reference is incorrectly adjusted after selecting a valid method.

UTS-16

Test component: typesystem/[check_NamespaceAttributeRef](#)

Input specification:

1. Namespace attribute reference has the same name as a local variable when it is being used.
2. Namespace attribute reference has the same name as a local variable when its namespace is being used.

Output specification:

1. Show [AttributeReferencesTo](#) error.
2. Show [AttributeReferencesTo](#) error.

Environmental needs: Either the attribute or its namespace is called within a valid Using declaration prior to referencing.

UTS-17

Test component: typesystem/[check_GlobalUsingNamespaceAttributeDeclaration](#)

Input specification:

1. Attribute does not exist within given namespace.

Output specification:

1. Show [AttributeDoesNotExist](#) error.

Environmental needs: The namespace is adjusted after selecting an attribute as reference.

UTS-18

Test component: typesystem/[check_GlobalUsingNamespaceMethodDeclaration](#)

Input specification:

1. Method does not exist within given namespace.

Output specification:

1. Show [MethodDoesNotExistWithin](#) error.

Environmental needs: The namespace is adjusted after selecting an method as reference.

UTS-19

Test component: typesystem/[check_GlobalVarDecCPP](#)

Input specification:

1. A global variable declaration is `thread_local` and stored in register.
2. The type of a new global variable is `char16_t`, and the initialization contains a negative number.
3. The type of a new global variable is `char16_t`, and the size of the initialization exceeds 16 bits.
4. The type of a new global variable is `char32_t`, and the initialization contains a negative number.
5. The type of a new global variable is `char32_t`, and the initialization exceeds 1114111.
6. The type of a new global variable is `wchar_t`, and the initialization contains a negative number.
7. The type of a new global variable is `wchar_t`, and the size of the initialization exceeds 8 bits.

Output specification:

1. Show [VariableDeclarationMay](#) error.
2. Show [ChartMustNotBeAssigned](#) error.
3. Show [ChartMustBeBetweenAnd](#) error.
4. Show [ChartMustNotBeAssigned](#) error.
5. Show [ChartMustBeBetweenAnd](#) error.
6. Show [ChartMustNotBeAssigned](#) error.

7. Show [AssignedValueMoreThan](#) warning.

Environmental needs: None.

UTS-20

Test component: typesystem/[check_IdentifierNamedConceptKeywords](#)

Input specification:

1. Declaring a class, namespace or variable with a name reserved for keywords.
(new, this, private, etc.)

Output specification:

1. Show [IsReservedKeyword](#) error.

Environmental needs: None.

UTS-21

Test component: typesystem/[supertype_Polymorphism](#)

Input specification:

1. New declaration contains a type or constructor that isn't a subtype of the original declaration.
2. Right side of pointer assignment contains a pointer type that isn't a subtype of the pointer declaration.

Output specification:

1. Show [IsNotSubtypeOf](#) error.
2. Show [IsNotSubtypeOf](#) error.

Environmental needs: Two classes are correctly declared.

UTS-22

Test component: typesystem/[typeof_NewDeclaration](#)

Input specification:

1. New declaration contains a type or constructor that is not a subtype of the original declaration.

Output specification:

1. Show [IsNotSubtypeOf](#) error.

Environmental needs: A variable of any type or a constructor is correctly declared.

UTS-23

Test component: typesystem/[check_DeleteDeclaration](#)

Input specification:

1. Delete declaration contains a variable which is not a pointer type.

Output specification:

1. Show [IsNotPointerOnlyVariables](#) error.

Environmental needs: A variable of any type or a constructor is correctly declared.

UTS-24

Test component: typesystem/[typeof_BinaryExpressionOverload](#)

Input specification:

1. Binary operator is used on a class object that has not defined it for the given types.

Output specification:

1. Show [OperatorIsNotDefined](#) error.

Environmental needs: The class objects are correctly declared.

UTS-25

Test component: typesystem/[typeof_ArrayAccessOverload](#)

Input specification:

1. Indices are accessed on a class object in which array access operator is not defined.

Output specification:

1. Show [OperatorIsNotDefined](#) error.

Environmental needs: The class object is correctly declared.

UTS-26

Test component: typesystem/[typeof_BinaryOrderedComparisonExpressionOverload](#)

Input specification:

1. Comparison operators are used on a class object that has not defined them.

Output specification:

1. Show [OperatorIsNotDefined](#) error.

Environmental needs: The class objects are correctly declared.

UTS-27

Test component: typesystem/[typeof_EqualsExpressionOverload](#)

Input specification:

1. Equality operator is used on a class object that has not defined it.

Output specification:

1. Show `OperatorIsNotDefined` error.

Environmental needs: The class object is correctly declared.

UTS-28

Test component: typesystem/`check_TemplateStub`

Input specification:

1. Template stub is created without a function or class underneath.

Output specification:

1. Show `LeftoverTemplateStub` error.

Environmental needs: None.

UTS-29

Test component: behavior/`behavior_TemplateFunctionCall`

Input specification:

1. Template method call with type implementation `int32` with value `true` filled in.
2. Template method call with type implementation `boolean` with value `true` filled in that then gets assigned to a variable of type `int32`.

Output specification:

1. Type system error.
2. Type system error.

Environmental needs: A correctly declared template function has a class type that has one parameter of that type and a return value of that type.

UTS-30

Test component: typesystem/`typeof_ITemplateImpl`

Input specification:

1. Template method call to `f` with no type implementations filled in.
2. Template method call to `f` with two type implementations filled in.
3. Template method call to `f` with value `1` filled in as type implementation.
4. Template method call to `g` with value `true` filled in as value implementation.

Output specification:

1. Show `TooFewTemplateArguments` error.
2. Show `TooManyTemplateArguments` error.

3. Show [ArgumentMustBeType](#) error.
4. Show [ArgumentMustBeValue](#) error.

Environmental needs: A correctly declared template function `f` has a class type that has one parameter of that type and a return value of that type. A correctly declared template function `g` has a value template parameter of type `int32` that has no parameters and a return value of `int32`.

UTS-31

Test component: `typesystem/typeof_TemplateValueRef`

Input specification:

1. The value of the value parameter is assigned to a variable of type `boolean`.

Output specification:

1. Type system error.

Environmental needs: A correctly declared template function has a value template parameter of type `int32` that has no parameters and a return value of `int32`.

UTS-32

Test component: `typesystem/typeof_ITemplate`

Input specification:

1. A new template function declaration first specifies a type template parameter `T` with default `int32` and then a type template parameter `S` without a default.

Output specification:

1. Shows [ParamWithoutDefaultAppearing](#) error.

Environmental needs: None.

UTS-33

Test component: `typesystem/check_TemplateTypeDef`

Input specification:

1. Template class contains two templates with the exact same type and name.

Output specification:

1. Show [DuplicateTypeName](#) error.

Environmental needs: A correctly declared template class contains with two templates.

UTS-34

Test component: `typesystem/typeof_InternalAttributeRef`

Input specification:

1. A variable declaration with type `boolean` which is initialized with the value of `foo`.

Output specification:

1. Type system error.

Environmental needs: Inside a method of a template class `Foo` with template type param `T`, public field `foo` of type `T`, and public method `bar` with no arguments and return type `T`.

UTS-35

Test component: `typesystem/typeof_InternalMethodCall`

Input specification:

1. A variable declaration with type `boolean` which is initialized with the value of `bar()`.

Output specification:

1. Type system error.

Environmental needs: Inside a method of a template class `Foo` with template type param `T`, public field `foo` of type `T`, and public method `bar` with no arguments and return type `T`.

UTS-36

Test component: `typesystem/typeof_AttributeRef`

Input specification:

1. A variable declaration with type `boolean` is initialized with the value of `fooInstance.foo`.

Output specification:

1. Type system error.

Environmental needs: A correctly declared instance of template class `Foo` called `fooInstance` has template type param `T` and contains public field `foo` of type `T`, and public method `bar` with no arguments and return type `T`.

UTS-37

Test component: `typesystem/typeof_QualifiedMethodCall`

Input specification:

1. A variable declaration with type `boolean` is initialized with the value of `fooInstance.bar()`.

Output specification:

1. Type system error.

Environmental needs: A correctly declared instance of template class `Foo` called `fooInstance` has template type param `T` and contains public field `foo` of type `T`, and public method `bar` with no arguments and return type `T`.

UTS-38

Test component: typesystem/[typeof_AutoType](#)

Input specification:

1. Assigning the value of `val` to a value with type `boolean`.
2. Declaring a variable with type `auto` and without initializer.

Output specification:

1. Type system error.
2. Show [VariableDeclarationWith](#) error.

Environmental needs: A correctly declared variable `val` has type `auto` and initializer 1.

UTS-39

Test component: typesystem/[check_DynamicCast](#)

Input specification:

1. Attempt to use [dynamic_cast](#) using non-pointer types.

Output specification:

1. Show [DynamiccastCanOnlyCast](#) error.

Environmental needs: Pointers are declared correctly.

UTS-40

Test component: typesystem/[check_ReinterpretCast](#)

Input specification:

1. Attempt to use [reinterpret_cast](#) using non-pointer types.

Output specification:

1. Show [ReinterpretcastCanOnly](#) error.

Environmental needs: Pointers are declared correctly.

UTS-41

Test component: typesystem/[check_StaticCast](#)

Input specification:

1. Attempt to use `static_cast` using inconvertible class types

Output specification:

1. Show `StaticcastCanOnlyCast` error

Environmental needs: Pointers are declared correctly.

UTS-42

Test component: typesystem/`typeof_ICPPCast`

Input specification:

1. Attempt to use any of `static_cast`, `reinterpret_cast` and `dynamic_cast` using valid pointers.

Output specification:

1. Assign the type inside of the cast expression.

Environmental needs: Pointers are declared correctly.

UTS-43

Test component: typesystem/`typeof_ConstCast`

Input specification:

1. Attempt to use any of `const_cast` using valid pointers.

Output specification:

1. Assign the type inside of the cast expression with `const` removed.

Environmental needs: Pointers are declared correctly.

UTS-44

Test component: typesystem/`typeof_This`

Input specification:

1. Referencing the enclosing class using the `this` keyword.

Output specification:

1. Assign a pointer type pointing to the enclosing class to `this`.

Environmental needs: An enclosing class is declared correctly.

UTS-45

Test component: typesystem/[check_TryCatchStatement](#)

Input specification:

1. Try-catch statement contains a catch block that catches all, followed by another catch block.

Output specification:

1. Show [CatchBlockAfterCatchall](#) error.

Environmental needs: A function contains a try-catch statement with two catch blocks.

3.2 C++ Code Generation

In this section, the text generation of the C++ code of each added element is tested.

3.2.1 Classes

UTS-46

Test component: test.ex.com.mbeddr.cpp/[Class](#)

Input specification:

```
1 class Counter {
2
3     public int8 value = 0;
4     public int8 [5] value2 = {1, 2, 3, 4, 5};
5
6     public void inc1() { value++; }
7
8     public void inc2() {
9         int [5] x = {1, 2, 3, 4, 5};
10        inc1();
11        inc1();
12    }
13
14    public void incN(int8 n) { value += n; }
15
16    public int8 getVal() {
17        return value;
18    }
19 }
20
21 void instantiationTC1() {
22     Counter counter;
23     counter.inc1();
24     counter.inc2();
25     counter.incN(42);
26 }
27
28 void instantiationTC2() {
29     Counter counter;
30     counter.inc1();
31 }
32
33 void assignmentTC1() {
34     Counter counter;
35     counter.value = 59;
36     counter.inc1();
37 }
```

Output specification:

In code file:

```
1 class Counter {
2
3     int8_t value = 0;
4     int8 [5] value2 = {
5         1, 2, 3, 4, 5 };
6 }
```

```

6
7  void inc1(void) {
8      value++;
9  }
10
11 void inc2(void) {
12     int x[5] = {
13         1, 2, 3, 4, 5 };
14     inc1();
15     inc1();
16 }
17
18 void incN(int8_t n) {
19     value += n;
20 }
21
22 int8_t getVal(void) {
23     return value;
24 }
25
26 };
27
28 static void UTP_instantiationTC1(void);
29
30 static void UTP_instantiationTC2(void);
31
32 static void UTP_assignmentTC1(void);
33
34 static void UTP_instantiationTC1(void)
35 {
36     Counter counter;
37     counter.inc1();
38     counter.inc2();
39     counter.incN(42);
40 }
41
42
43 static void UTP_instantiationTC2(void)
44 {
45     Counter counter;
46     counter.inc1();
47 }
48
49
50 static void UTP_assignmentTC1(void)
51 {
52     Counter counter;
53     counter.value = 59;
54     counter.inc1();
55 }

```

Environmental needs: The class and its members are correctly declared.

UTS-47

Test component: test.ex.com.mbeddr.cpp/[Constructor](#)

Input specification:

```
1 exported class ParentClass {
2     public ParentClass(<< ... >>){ }
3     protected int32 parentY = 0;
4 }
5
6 exported class ClassName : private ParentClass {
7     public int32 x = 0;
8     public ClassName(int32 inputX, int32 inputY): x(inputX), parentY(3){ }
9     public ClassName(<< ... >>): ParentClass(<< ... >>){ x = 235; }
10    public ~ClassName(<< ... >>){ }
11 }
12
13 void constructorTC1() {
14     ClassName twoIntsConstructor(50, 100);
15     ClassName noArgsConstructor(<no constructor>);
16 }
```

Output specification:

In header file:

```
1 class ParentClass {
2     ParentClass() {
3     }
4
5     int32_t parentY = 0;
6 };
7
8 class ClassName : private ParentClass {
9     int32_t x = 0;
10    ClassName() {
11    }
12
13    ClassName(int32_t inputX, int32_t inputY) : x(inputX), parentY(3) {
14    }
15
16    ClassName() : ParentClass() {
17    }
18
19    ~ClassName() {
20    }
21
22 };
```

In code file:

```
1 static void UTP_constructorTC1(void);
2
3 static void UTP_constructorTC1(void)
4 {
5     ClassName twoIntsConstructor(50,100);
6     ClassName noArgsConstructor;
7 }
```

Environmental needs: The classes and their members are correctly declared.

Test component: test.ex.com.mbeddr.cpp/[Extension](#)

Input specification:

```

1  exported class Adder {
2      public int32 pubfield = 5;
3      public int32 increment(int32 value) {
4          return ++value;
5      }
6  }
7
8  exported class Calculator : public Adder, private Subtractor {
9      public int32 other() {
10         return 5;
11     }
12     public int32 increment20(int32 value) {
13         for (int32 i = 0; i < 35; i++ ) {
14             value = increment(value);
15         }
16         for (int32 i = 0; i < 15; i++ ) {
17             value = decrement(value);
18         }
19         return value;
20     }
21 }
22
23 exported class MegaCalc : public Calculator {
24     public int32 someMethod() {
25         pubfield;
26         // From grandparent
27         increment(4);
28         // From parent
29         increment20(5);
30         return increment20(35);
31     }
32 }
33
34 void extensionTC1() {
35     Calculator calculator;
36     MegaCalc other_calculator(<no constructor>);
37     int32 someInt = 0;
38     calculator.increment( someInt );
39     // Grandparentfield
40     other_calculator.pubfield;
41     // Grandparentmethod
42     other_calculator.increment( 4 );
43     other_calculator.someMethod( );
44
45     calculator.increment( 24 );
46 }

```

Output specification:

In header file:

```

1  #include "ImportedModule.h"
2

```

```

3  class Adder {
4      int32_t pubfield = 5;
5      int32_t increment(int32_t value) {
6          return ++value;
7      }
8  };
9
10 class Calculator : public Adder, private Subtractor {
11     int32_t other(void) {
12         return 5;
13     }
14     int32_t increment20(int32_t value) {
15         for ( int32_t i = 0 ; i < 35; i++ )
16         {
17             value = increment(value);
18         }
19         for ( int32_t i = 0 ; i < 15; i++ )
20         {
21             value = decrement(value);
22         }
23         return value;
24     }
25 };
26
27 class OmegaCalc : public Calculator {
28     int32_t someMethod(void) {
29         pubfield;
30         /*
31          * From grandparent
32          */
33
34         increment(4);
35         /*
36          * From parent
37          */
38
39         increment20(5);
40         return increment20(35);
41     }
42 };

```

In code file:

```

1  #include "ImportedModule.h"
2
3  static void UTP_extensionTC1(void);
4
5  static void UTP_extensionTC1(void)
6  {
7      Calculator calculator;
8      MegaCalc other_calculator;
9      int32_t someInt = 0;
10     calculator.increment(someInt);
11     /*
12      * Grandparent field
13      */
14
15     other_calculator.pubfield;
16     /*

```

```

17  * Grandparent method
18  */
19
20  other_calculator.increment(4);
21  other_calculator.someMethod();
22
23  calculator.increment(24);
24 }

```

Environmental needs: The classes and their members are correctly declared.

UTS-49

Test component: test.ex.com.mbeddr.cpp/[ImportedModule](#)

Input specification:

```

1  exported class Subtractor {
2      public int32 decrement(int32 value) {
3          return —value;
4      }
5  }

```

Output specification:

In header file:

```

1  class Subtractor {
2      int32_t decrement(int32_t value) {
3          return —value;
4      }
5  };

```

Environmental needs: The class and its members are correctly declared.

UTS-50

Test component: test.ex.com.mbeddr.cpp/[Nesting](#)

Input specification:

```

1  exported class Outer {
2      public class Inner {
3          public int iVal = 1;
4      }
5
6      // Class should be able to be referenced directly if it's in the scope,
7      // or via the outer class
8      private Outer::Inner outerInner;
9      private Inner inner;
10     public int getInnerValue() {
11         return inner.iVal;
12     }
13
14     public class Other {
15         public int oVal = 2;
16     }

```



```

16 }
17
18 void nestingTC1() {
19     Outer outer;
20     Outer::Other other;
21 }

```

Output specification:

In header file:

```

1 class Outer {
2     class Inner {
3         int iVal = 1;
4     };
5
6     /*
7      * Class should be able to be referenced directly if it's in the scope,
8      * or via the outer class
9      */
10    Outer::Inner outerInner;
11    Inner inner;
12    int getInnerValue(void) {
13        return inner.iVal;
14    }
15
16    class Other {
17        int oVal = 2;
18    };
19 };

```

In code file:

```

1 static void UTP_nestingTC1(void);
2
3 static void UTP_nestingTC1(void)
4 {
5     Outer outer;
6     Outer::Other other;
7 }

```

Environmental needs: The class and its members are correctly declared.

UTS-51

Test component: test.ex.com.mbeddr.cpp/[Polymorphism](#)

Input specification:

```

1 exported class A {
2     public int8 xValue;
3     public A(int8 x): xValue(x){ }
4 }
5
6 exported class B : public A {
7     public B(int8 x): A(x){ }
8 }
9

```

```

10 exported class C : public B {
11     public C(<< ... >>): B(22){ }
12 }
13
14 void poly_external() {
15     A* actualA = new A(5) ;
16     // Child
17     A* a = new B(5) ;
18     // Grandchild
19     A* a2 = new C(<< ... >>) ;
20 }

```

Output specification:

In header file:

```

1 class A {
2     int8_t xValue;
3     A(int8_t x) : xValue(x) {
4     }
5
6 };
7
8 class B : public A {
9     B(int8_t x) : A(x) {
10    }
11
12 };
13
14 class C : public B {
15     C() : B(22) {
16     }
17
18 };

```

In code file:

```

1 static void UTP_poly_external(void);
2
3 static void UTP_poly_external(void)
4 {
5     A *actualA = new A(5);
6     /*
7      * Child
8      */
9
10    A *a = new B(5);
11    /*
12     * Grandchild
13     */
14
15    A *a2 = new C();
16 }

```

Environmental needs: The classes and its constructors are correctly declared.

UTS-52

Test component: test.ex.com.mbeddr.cpp/[This](#)

Input specification:

```
1 exported class Person {
2     private int32 age;
3
4     public Person(int32 age){ this->age = age; }
5
6     public int32 getAge() {
7         return this->age;
8     }
9 }
10
11 template<class T>
12 exported class List {
13     public T head;
14     public List <T>* tail;
15
16     public List(T h, List <T>* t): head(h), tail(t){ }
17
18     public T lastItem() {
19         List <T>* node = this;
20         while (node->tail != NULL) {
21             node = node->tail;
22         }
23         return node->head;
24     }
25 }
26
27 void thisTC1() {
28     Person henk(20);
29     List <int32> ages;
30     ages.head = henk.getAge();
31 }
```

Output specification:

In header file:

```
1 class Person {
2     int32_t age;
3
4     Person(int32_t age) {
5         this->age = age;
6     }
7
8
9     int32_t getAge(void) {
10         return this->age;
11     }
12 };
13
14 template<class T>
15 class List {
16     T head;
17     List<T> * tail;
18 }
```

```

19 List(T h, List<T> *t) : head(h), tail(t) {
20 }
21
22
23 T lastItem(void) {
24     List<T> *node = this;
25     while (node->tail != NULL)
26     {
27         node = node->tail;
28     }
29     return node->head;
30 }
31 };

```

In code file:

```

1 static void UTP_thisTC1(void);
2
3 static void UTP_thisTC1(void)
4 {
5     Person henk(20);
6     List<int32_t> ages;
7     ages.head = henk.getAge();
8 }

```

Environmental needs: The classes and their members are correctly declared.

3.2.2 Namespaces

UTS-53

Test component: test.ex.com.mbeddr.cpp/[Namespace](#)

Input specification:

```

1 namespace A {
2     int32 aInt = 25;
3     boolean aBool() {
4         return true;
5     }
6     namespace B {
7         int32 bInt = 78;
8         boolean bBool(boolean x) {
9             return x;
10        }
11    }
12    namespace C {
13        using namespace A;
14    }
15 }
16
17 namespace D {
18     int32 dInt = 96;
19     boolean dBool() {
20         return true;
21     }
22     namespace E {

```

```

23     int32 eInt = 69;
24     boolean eBool(boolean y) {
25         return y;
26     }
27 }
28 }
29
30 void namespaceLocalUsing() {
31     int32 lInt1 = A::aInt;
32     int32 lInt2 = A::B::bInt;
33     boolean lBool1 = A::aBool();
34     boolean lBool2 = A::B::bBool(false);
35
36     using A::B::bInt;
37     lInt1 = bInt;
38     using A::B::bBool;
39     lBool1 = bBool(false);
40     using namespace A;
41     lInt2 = aInt;
42     lBool2 = aBool();
43
44     lInt1 = C::aInt;
45     lBool1 = C::aBool();
46 }
47
48 using D::dInt;
49 using D::dBool;
50 using namespace D::E;
51
52 void namespaceGlobalUsing() {
53     int32 gInt1 = dInt;
54     int32 gInt2 = eInt;
55     boolean gBool1 = dBool();
56     boolean gBool2 = eBool(true);
57 }

```

Output specification:

In code file:

```

1 namespace A {
2     int32_t aInt = 25;
3     bool aBool(void) {
4         return true;
5     }
6     namespace B {
7         int32_t bInt = 78;
8         bool bBool(bool x) {
9             return x;
10        }
11    }
12    namespace C {
13        using namespace A;
14    }
15 }
16
17 namespace D {
18     int32_t dInt = 96;

```

```

19  bool dBool(void) {
20      return true;
21  }
22  namespace E {
23      int32_t eInt = 69;
24      bool eBool(bool y) {
25          return y;
26      }
27  }
28 }
29
30 using D::dInt;
31 using D::dBool;
32 using namespace D::E;
33
34 static void UTP_namespaceLocalUsing(void);
35
36 static void UTP_namespaceGlobalUsing(void);
37
38 static void UTP_namespaceLocalUsing(void)
39 {
40     int32_t lInt1 = A::aInt;
41     int32_t lInt2 = A::B::bInt;
42     bool lBool1 = A::aBool();
43     bool lBool2 = A::B::bBool(false);
44
45     using A::B::bInt;
46     lInt1 = bInt;
47     using A::B::bBool;
48     lBool1 = bBool(false);
49     using namespace A;
50     lInt2 = aInt;
51     lBool2 = aBool();
52
53     lInt1 = C::aInt;
54     lBool1 = C::aBool();
55 }
56
57 static void UTP_namespaceGlobalUsing(void)
58 {
59     int32_t gInt1 = dInt;
60     int32_t gInt2 = eInt;
61     bool gBool1 = dBool();
62     bool gBool2 = eBool(true);
63 }

```

Environmental needs: The namespaces and their elements are correctly declared.

3.2.3 Auto

UTS-54

Test component: test.ex.com.mbeddr.cpp/[Auto](#)

Input specification:

```

1 auto globalShouldBeBool = true;
2 auto globalShouldBeInt = 1;
3
4 exported class AutoContainer {
5     public static auto const shouldBeInt = 1;
6     public int32 hey = 1;
7 }
8
9 void test_auto() {
10     auto integer = 1;
11     int32 assignsToAuto = integer;
12     integer = globalShouldBeInt;
13     AutoContainer autoContainer(<no constructor>);
14     assignsToAuto = autoContainer.shouldBeInt;
15 }

```

Output specification:

In header file:

```

1 class AutoContainer {
2     static auto const shouldBeInt = 1;
3     int32_t hey = 1;
4 };

```

In code file:

```

1 static void UTP_test_auto(void);
2
3 static auto UTP_globalShouldBeBool = true;
4
5 static auto UTP_globalShouldBeInt = 1;
6
7 static void UTP_test_auto(void)
8 {
9     auto integer = 1;
10    int32_t assignsToAuto = integer;
11    integer = UTP_globalShouldBeInt;
12    AutoContainer autoContainer;
13    assignsToAuto = autoContainer.shouldBeInt;
14 }

```

Environmental needs: The class and the statements using the auto keyword are declared correctly.

3.2.4 Casting

UTS-55

Test component: test.ex.com.mbeddr.cpp/[Casting](#)

Input specification:

```

1 exported class SomeClass {
2     << ... >>
3 }

```

```

4 |
5 | exported class OtherClass : public SomeClass {
6 |     << ... >>
7 | }
8 |
9 | exported class ThirdClass : private SomeClass {
10 |     << ... >>
11 | }
12 |
13 | void test_static_casting() {
14 |     OtherClass* other_class = new OtherClass ;
15 |     ThirdClass* third_class = new ThirdClass ;
16 |     SomeClass* some_class = static_cast<SomeClass*>(other_class);
17 |
18 |     static_cast<int32>(235.3);
19 | }
20 |
21 | void test_dynamic_casting() {
22 |     OtherClass* other_class = new OtherClass ;
23 |     ThirdClass* third_class = new ThirdClass ;
24 |     dynamic_cast<ThirdClass*>(other_class);
25 | }
26 |
27 | void test_const_casting() {
28 |     ThirdClass* const third_class = new ThirdClass ;
29 |     const_cast<ThirdClass*>(third_class);
30 | }
31 |
32 | void test_reinterpret_casting() {
33 |     OtherClass* other_class = new OtherClass ;
34 |     ThirdClass* third_class = new ThirdClass ;
35 |     other_class = reinterpret_cast<OtherClass*>(third_class);
36 | }

```

Output specification:

In header file:

```

1 | class SomeClass {
2 | };
3 |
4 | class OtherClass : public SomeClass {
5 | };
6 |
7 | class ThirdClass : private SomeClass {
8 | };

```

In code file:

```

1 | static void UTP_test_static_casting(void);
2 |
3 | static void UTP_test_dynamic_casting(void);
4 |
5 | static void UTP_test_const_casting(void);
6 |
7 | static void UTP_test_reinterpret_casting(void);
8 |
9 | static void UTP_test_static_casting(void)
10 | {

```



```

11 OtherClass *other_class = new OtherClass;
12 ThirdClass *third_class = new ThirdClass;
13 SomeClass *some_class = static_cast<SomeClass*>(other_class);
14
15 static_cast<int32>(235.3);
16 }
17
18
19 static void UTP_test_dynamic_casting(void)
20 {
21     OtherClass *other_class = new OtherClass;
22     ThirdClass *third_class = new ThirdClass;
23     dynamic_cast<ThirdClass*>(other_class);
24 }
25
26
27 static void UTP_test_const_casting(void)
28 {
29     ThirdClass * const third_class = new ThirdClass;
30     const_cast<ThirdClass*>(third_class);
31 }
32
33
34 static void UTP_test_reinterpret_casting(void)
35 {
36     OtherClass *other_class = new OtherClass;
37     ThirdClass *third_class = new ThirdClass;
38     other_class = reinterpret_cast<OtherClass*>(third_class);
39 }

```

Environmental needs: All classes and casting statements are declared correctly.

3.2.5 CharTypes

UTS-56

Test component: test.ex.com.mbeddr.cpp/[Chars](#)

Input specification:

```

1 char16_t char16 = '1';
2 char32_t char32 = '1';
3 wchar_t wchar = '1';
4 char testChar = '1';
5
6 void char16Test {
7     boolean char16Test = char16 == testChar;
8 }
9
10 void char32Test {
11     boolean char32Test = char32 == testChar;
12 }
13
14 void wcharTest {
15     boolean wcharTest = wchar == testChar;
16 }

```

Output specification:

In code file:

```
1 static void UTP_char16Test(void);
2
3 static void UTP_char32Test(void);
4
5 static void UTP_wcharTest(void);
6
7 char16_t UTP_char16 = '1';
8 char32_t UTP_char32 = '1';
9 wchar_t UTP_wchar = '1';
10 char UTP_testChar = '1';
11 static void UTP_char16Test(void)
12 {
13     bool char16Test = UTP_char16 == UTP_testChar;
14 }
15
16
17 static void UTP_char32Test(void)
18 {
19     bool char32Test = UTP_char32 == UTP_testChar;
20 }
21
22
23 static void UTP_wcharTest(void)
24 {
25     bool wcharTest = UTP_wchar == UTP_testChar;
26 }
```

Environmental needs: All statements using these char types are declared correctly.

3.2.6 NumericTypes

UTS-57

Test component: test.ex.com.mbeddr.cpp/[Ints](#)

Input specification:

```
1 int testInt = 1;
2
3 int8 int8Var = 1;
4 int16 int16Var = 1;
5 int32 int32Var = 1;
6 int64 int64Var = 1;
7
8 unsigned int testUInt = 1;
9
10 uint8 uInt8Var = 1;
11 uint16 uInt16Var = 1;
12 uint32 uInt32Var = 1;
13 uint64 uInt64Var = 1;
14
15 float testFloat = 1;
```

```

16
17 double longFloatVar = 1;
18 long longIntVar = 1;
19 long long longLongIntVar = 1;
20 long double longLongFloatVar = 1;
21
22 short shortIntVar = 1;
23 unsigned short uShortIntVar = 1;
24
25
26 void int8Test {
27     boolean int8Test = int8Var == testInt;
28 }
29
30 void int16Test {
31     boolean int16Test = int16Var == testInt;
32 }
33
34 void int32Test {
35     boolean int32Test = int32Var == testInt;
36 }
37
38 void int64Test {
39     boolean int64Test = int64Var == testInt;
40 }
41
42
43 void uInt8Test {
44     boolean uInt8Test = uInt8Var == testUInt;
45 }
46
47 void uInt16Test {
48     boolean uInt16Test = uInt16Var == testUInt;
49 }
50
51 void uInt32Test {
52     boolean uInt32Test = uInt32Var == testUInt;
53 }
54
55 void uInt64Test {
56     boolean uInt64Test = uInt64Var == testUInt;
57 }
58
59
60 void longFloatTest {
61     boolean longFloatTest = longFloatVar == testFloat;
62 }
63
64 void longIntTest {
65     boolean longIntTest = longIntVar == testInt;
66 }
67
68 void longLongIntTest {
69     boolean longLongIntTest = longLongIntVar == testInt;
70 }
71
72 void longDoubleTest {
73     boolean longDoubleTest = longLongFloatVar == testFloat;

```

```

74 }
75
76
77 void shortIntTest {
78     boolean shortIntTest = shortIntVar == testInt;
79 }
80
81 void uShortIntTest {
82     boolean uShortIntTest = uShortIntVar == testUInt;
83 }

```

Output specification:

In code file:

```

1  static void UTP_int8Test(void);
2
3  static void UTP_int16Test(void);
4
5  static void UTP_int32Test(void);
6
7  static void UTP_int64Test(void);
8
9  static void UTP_uInt8Test(void);
10
11 static void UTP_uInt16Test(void);
12
13 static void UTP_uInt32Test(void);
14
15 static void UTP_uInt64Test(void);
16
17 static void UTP_longFloatTest(void);
18
19 static void UTP_longIntTest(void);
20
21 static void UTP_longLongIntTest(void);
22
23 static void UTP_longDoubleTest(void);
24
25 static void UTP_shortIntTest(void);
26
27 static void UTP_uShortIntTest(void);
28
29
30 int UTP_testInt = 1;
31
32 int8 UTP_int8Var = 1;
33 int16 UTP_int16Var = 1;
34 int32 UTP_int32Var = 1;
35 int64 UTP_int64Var = 1;
36
37 unsigned int UTP_testUInt = 1;
38
39 uint8 UTP_uInt8Var = 1;
40 uint16 UTP_uInt16Var = 1;
41 uint32 UTP_uInt32Var = 1;
42 uint64 UTP_uInt64Var = 1;
43

```

```

44 float UTP_testFloat = 1;
45
46 double UTP_longFloatVar = 1;
47 long UTP_longIntVar = 1;
48 long long UTP_longLongIntVar = 1;
49 long double UTP_longLongFloatVar = 1;
50
51 short UTP_shortIntVar = 1;
52 unsigned short UTP_uShortIntVar = 1;
53
54
55 static void UTP_int8Test(void){
56     boolean int8Test = UTP_int8Var == UTP_testInt;
57 }
58
59 static void UTP_int16Test(void) {
60     boolean int16Test = UTP_int16Var == UTP_testInt;
61 }
62
63 static void UTP_int32Test(void) {
64     boolean int32Test = UTP_int32Var == UTP_testInt;
65 }
66
67 static void UTP_int64Test(void) {
68     boolean int64Test = UTP_int64Var == UTP_testInt;
69 }
70
71
72 static void UTP_uInt8Test(void) {
73     boolean uInt8Test = UTP_uInt8Var == UTP_testUInt;
74 }
75
76 static void UTP_uInt16Test(void) {
77     boolean uInt16Test = UTP_uInt16Var == UTP_testUInt;
78 }
79
80 static void UTP_uInt32Test(void) {
81     boolean uInt32Test = UTP_uInt32Var == UTP_testUInt;
82 }
83
84 static void UTP_uInt64Test(void) {
85     boolean uInt64Test = UTP_uInt64Var == UTP_testUInt;
86 }
87
88
89 static void UTP_longFloatTest(void) {
90     boolean longFloatTest = UTP_longFloatVar == UTP_testFloat;
91 }
92
93 static void UTP_longIntTest(void) {
94     boolean longIntTest = UTP_longIntVar == UTP_testInt;
95 }
96
97 static void UTP_longLongIntTest(void) {
98     boolean longLongIntTest = UTP_longLongIntVar == UTP_testInt;
99 }
100
101 static void UTP_longDoubleTest(void) {

```

```

102     boolean longDoubleTest = UTP_longLongFloatVar == UTP_testFloat;
103 }
104
105
106 static void UTP_shortIntTest(void) {
107     boolean shortIntTest = UTP_shortIntVar == UTP_testInt;
108 }
109
110 static void UTP_uShortIntTest(void) {
111     boolean uShortIntTest = UTP_uShortIntVar == UTP_testUInt;
112 }

```

Environmental needs: All statements using these char types are declared correctly.

3.2.7 New & Delete

UTS-58

Test component: test.ex.com.mbeddr.cpp/[NewDelete](#)

Input specification:

```

1  exported class NewDeleteClass1 {
2      public NewDeleteClass1(<< ... >>){ }
3      public NewDeleteClass1(int8 x){ }
4
5      public int16* newDeleteInt1;
6      public int64* newDeleteIntToBeExtended;
7  }
8
9  exported class NewDeleteClass2 : public NewDeleteClass1 {
10     public int64* extendedInt;
11 }
12
13 void newDeleteTC1 {
14     int64* newDeleteInt2 = new (std::nothrow) int64[23];
15     delete [] newDeleteInt2;
16
17     NewDeleteClass1 ndc1;
18     int16* ndc1Int = ndc1.newDeleteInt1;
19     ndc1Int = new int16;
20     *ndc1Int = 500;
21     delete ndc1Int;
22
23     NewDeleteClass2 ndc2;
24     int64* ndc2Int = ndc2.extendedInt;
25     *ndc2Int = 700;
26     delete ndc2Int;
27
28     NewDeleteClass2 ndc2_2;
29     int64* ndc2_2Int = ndc2_2.newDeleteIntToBeExtended;
30     ndc2_2Int = new int64;
31     delete ndc2_2Int;
32 }
33
34 void newDeleteTC2 {

```

```

35 NewDeleteClass1* ndc1constructor(<< ... >>);
36 ndc1constructor = new NewDeleteClass1;
37 delete ndc1constructor;
38
39 NewDeleteClass1* ndc1constructorInt = new NewDeleteClass1(50);
40 ndc1constructorInt = new NewDeleteClass1;
41 delete ndc1constructorInt;
42 }

```

Output specification:

In header file:

```

1 #include <new>
2
3 class NewDeleteClass1 {
4     NewDeleteClass1() {
5     }
6
7     NewDeleteClass1(int8_t x) {
8     }
9
10
11     int16_t * newDeleteInt1;
12     int64_t * newDeleteIntToBeExtended;
13 };
14
15 class NewDeleteClass2 : public NewDeleteClass1 {
16     int64_t * extendedInt;
17 };

```

In code file:

```

1 static void UTP_newDeleteTC1(void);
2
3 static void UTP_newDeleteTC2(void);
4
5 static void UTP_newDeleteTC1(void)
6 {
7     int64_t *newDeleteInt2 = new (std::nothrow) int64_t [23];
8     delete [] newDeleteInt2;
9
10    NewDeleteClass1 ndc1;
11    int16_t *ndc1Int = ndc1.newDeleteInt1;
12    ndc1Int = new int16_t;
13    *ndc1Int = 500;
14    delete ndc1Int;
15
16    NewDeleteClass2 ndc2;
17    int64_t *ndc2Int = ndc2.extendedInt;
18    *ndc2Int = 700;
19    delete ndc2Int;
20
21    NewDeleteClass2 ndc2_2;
22    int64_t *ndc2_2Int = ndc2_2.newDeleteIntToBeExtended;
23    ndc2_2Int = new int64_t;
24    delete ndc2_2Int;
25 }
26

```

```

27
28 static void UTP_newDeleteTC2(void)
29 {
30     NewDeleteClass1 *ndclconstructor;
31     ndclconstructor = new NewDeleteClass1;
32     delete ndclconstructor;
33
34     NewDeleteClass1 *ndclconstructorInt = new NewDeleteClass1(50);
35     ndclconstructorInt = new NewDeleteClass1;
36     delete ndclconstructorInt;
37 }

```

Environmental needs: The classes and the statements using the new and delete keywords were declared correctly.

3.2.8 Nullpointer

UTS-59

Test component: test.ex.com.mbeddr.cpp/[NullPointer](#)

Input specification:

```

1  exported class TestClass {
2      public TestClass(<< ... >>){ }
3      public TestClass(std::nullptr_t null_ptr_type){ }
4      public TestClass(int16* int_ptr_type){ }
5
6      public std::nullptr_t null_ptr = nullptr;
7
8      public void testMethodNullPtr(std::nullptr_t null_ptr) { }
9      public void testMethodIntPtr(int16* null_ptr) { }
10 }
11
12 void nullPointerTC1() {
13     // Test for field type std::nullptr_t such that NULL can be assigned
14     TestClass testClass(<< ... >>);
15     testClass.null_ptr = NULL;
16
17 }
18
19 void nullPointerTC2() {
20     // Test for constructor with integer pointer (TestClass(int16*
21         int_ptr_type)) and constructor with nullpointer (TestClass(std::
22         nullptr_t null_ptr_type))
23     TestClass testClass(nullptr);
24     TestClass testClass2(((int16*) (NULL)));
25
26     TestClass testClass3(nullptr);
27     TestClass testClass4(((std::nullptr_t) (NULL)));
28 }
29
30 void nullPointerTC3() {
31     // Test for nullpointer and integer pointer methods, they both accept
32     NULL and nullptr as arguments

```



```

31     TestClass testClass(<< ... >>);
32
33     testClass.testMethodNullPtr(NULL);
34     testClass.testMethodNullPtr(nullptr);
35
36     testClass.testMethodIntPtr(NULL);
37     testClass.testMethodIntPtr(nullptr);
38 }

```

Output specification:

In header file:

```

1  class TestClass {
2      TestClass() {
3      }
4
5      TestClass(std::nullptr_t null_ptr_type) {
6      }
7
8      TestClass(int16_t *int_ptr_type) {
9      }
10
11
12     std::nullptr_t null_ptr = nullptr;
13
14     void testMethodNullPtr(std::nullptr_t null_ptr) {
15     }
16     void testMethodIntPtr(int16_t *null_ptr) {
17     }
18 };

```

In code file:

```

1  static void UTP_nullPointerTC1(void);
2
3  static void UTP_nullPointerTC2(void);
4
5  static void UTP_nullPointerTC3(void);
6
7  static void UTP_nullPointerTC1(void)
8  {
9      /*
10     * Test for field type std::nullptr_t such that
11     * NULL can be assigned
12     */
13
14     TestClass testClass;
15     testClass.null_ptr = NULL;
16
17 }
18
19
20 static void UTP_nullPointerTC2(void)
21 {
22     /*
23     * Test for constructor with integer pointer
24     * (TestClass(int16* int_ptr_type))
25     * and constructor with null pointer

```

```

26     * (TestClass(std::nullptr_t nullptr_type))
27     */
28
29     TestClass testClass(nullptr);
30     TestClass testClass2(((int16_t *) (NULL)));
31
32     TestClass testClass3(nullptr);
33     TestClass testClass4(((std::nullptr_t) (NULL)));
34
35 }
36
37
38 static void UTP_nullPointerTC3(void)
39 {
40     /*
41      * Test for null pointer and integer pointer methods,
42      * they both accept NULL and nullptr as arguments
43      */
44
45     TestClass testClass;
46
47     testClass.testMethodNullPtr(NULL);
48     testClass.testMethodNullPtr(nullptr);
49
50     testClass.testMethodIntPtr(NULL);
51     testClass.testMethodIntPtr(nullptr);
52 }

```

Environmental needs: The class and its elements, and the statements using the null-pointer keywords, are all declared correctly.

3.2.9 Operator Overloading

UTS-60

Test component: test.ex.com.mbeddr.cpp/[OperatorOverloading](#)

Input specification:

```

1  exported class SomeClass {
2      public boolean setFalse(boolean boo) {
3          boo = false;
4          return boo;
5      }
6
7      public int8 _x = 5;
8
9      public void takesSomeClass(SomeClass param) { }
10
11     public SomeClass operator+(SomeClass other){
12         return other;
13     }
14
15     public int8 operator+(int8 other){
16         return other;
17     }

```

```

18
19 public SomeClass operator%(SomeClass other){
20     return other;
21 }
22
23 public SomeClass operator+=(int8 b){
24     return *this;
25 }
26
27 public int8 operator [] (int8 index){
28     return index;
29 }
30
31 public SomeClass operator [] (SomeClass index){
32     return index;
33 }
34
35 public SomeClass operator++(<< ... >>){
36     return *this;
37 }
38
39 public boolean operator==(SomeClass other){
40     return true;
41 }
42
43 private void dancesWithOperators() {
44     SomeClass a(<no constructor>);
45     SomeClass b(<no constructor>);
46     int8 x = 5;
47     a.takesSomeClass( a + b );
48     a[ ((int8) (5)) ];
49
50     a++;
51     ++a;
52     a == b;
53     a.takesSomeClass( ++a );
54     a + b;
55     a + x;
56     a;
57 }
58 }
59
60 void test_operators() {
61     SomeClass a(<no constructor>);
62     SomeClass b(<no constructor>);
63     int8 x = 5;
64     a.takesSomeClass(a + b);
65     a[ ((int8) (5)) ];
66     a += ((int8) (5));
67     a++;
68     a == b;
69     a.takesSomeClass(++a);
70 }

```

Output specification:

In header file:

```

1  class SomeClass {
2      bool setFalse(bool boo) {
3          boo = false;
4          return boo;
5      }
6
7      int8_t _x = 5;
8
9      void takesSomeClass(SomeClass param) {
10     }
11
12     SomeClass operator+(SomeClass other){
13         return other;
14     }
15
16     int8_t operator+(int8_t other){
17         return other;
18     }
19
20     SomeClass operator%(SomeClass other){
21         return other;
22     }
23
24     SomeClass operator+=(int8_t b){
25         return *this;
26     }
27
28     int8_t operator[](int8_t index){
29         return index;
30     }
31
32     SomeClass operator[](SomeClass index){
33         return index;
34     }
35
36     SomeClass operator++(){
37         return *this;
38     }
39
40     bool operator==(SomeClass other){
41         return true;
42     }
43
44     void dancesWithOperators(void) {
45         SomeClass a;
46         SomeClass b;
47         int8_t x = 5;
48         a.takesSomeClass(a + b);
49         a[((int8_t)((5)))];
50
51         a++;
52         ++a;
53         a == b;
54         a.takesSomeClass(++a);
55         a + b;
56         a + x;

```

```

57     a;
58 }
59 };

```

In code file:

```

1 static void UTP_test_operators(void);
2
3 static void UTP_test_operators(void)
4 {
5     SomeClass a;
6     SomeClass b;
7     int8_t x = 5;
8     a.takesSomeClass(a + b);
9     a[((int8_t)((5)))];
10    a += ((int8_t)((5)));
11    a++;
12    a == b;
13    a.takesSomeClass(++a);
14 }

```

Environmental needs: The classes and its statements using operator overloading are all declared correctly.

3.2.10 Specifiers

UTS-61

Test component: test.ex.com.mbeddr.cpp/[Specifiers](#)

Input specification:

```

1 class SomeClass {
2
3     public constexpr static int32 staticField = 111;
4
5     public constexpr int32 const constField = 3;
6
7     public int32 volatile volatileField = 5;
8
9     public int32 const volatile constVolatileField = 21;
10
11    public inline int32 inlinedMethod() {
12        return 32;
13    }
14
15    public static int32 staticMethod() {
16        int32 test = 1;
17        return 46;
18    }
19
20    public static inline int32 staticInlinedMethod() {
21        return 74;
22    }
23 }

```

Output specification:

In code file:

```
1 class SomeClass {
2     constexpr static int32_t staticField = 111;
3
4     constexpr int32_t const    constField = 3;
5
6     volatile int32_t volatileField = 5;
7
8     volatile int32_t const    constVolatileField = 21;
9
10    int32_t inlinedMethod(void) {
11        return 32;
12    }
13
14    int32_t staticMethod(void) {
15        int32_t test = 1;
16        return 46;
17    }
18
19    int32_t staticInlinedMethod(void) {
20        return 74;
21    }
22 };
23
24 static void UTP_test_specifiers(void);
25
26 static void UTP_test_specifiers(void)
27 {
28     SomeClass anInstance;
29     int32_t a = anInstance.staticField;
30     int32_t b = anInstance.constField;
31     int32_t c = anInstance.volatileField;
32     int32_t d = anInstance.constVolatileField;
33     int32_t e = anInstance.inlinedMethod();
34     int32_t f = anInstance.staticMethod();
35     int32_t g = anInstance.staticInlinedMethod();
36 }
```

Environmental needs: The class and its elements using the specifiers are declared correctly.

3.2.11 Templates

UTS-62

Test component: test.ex.com.mbeddr.cpp/[Templates](#)

Input specification:

```
1 template<class T>
2 exported T identity(T value) {
3     return value;
4 }
```

```

5
6 template<class T>
7 int8 compare(T a, T b) {
8     return (a < b)?(-1):((a > b)?(1):(0));
9 }
10
11 template<int32 n = 2>
12 int32 multiplyBy(int32 x) {
13     return x * n;
14 }
15
16 template<class T, T def>
17 T deref(T* ref) {
18     if (ref == NULL) {
19         return def;
20     } else {
21         return *ref;
22     } if
23 }
24
25 template<class T>
26 exported class List {
27     public T head;
28     public List <T>* tail;
29
30     public List(T h, List <T>* t): head(h), tail(t){ }
31
32     public T lastItem() {
33         foobar <int32>( head, 1 );
34         List <T>* node = this;
35         while (node->tail != NULL) {
36             node = node->tail;
37         }
38         return node->head;
39     }
40
41     template<class S>
42     public S foobar(T first, S second) {
43         return second;
44     }
45 }
46
47 exported class IntList : public List <int32> {
48     public IntList(int32 h): head(h){ }
49     public IntList(int32 h, List <int32>* t): List(h, t){ }
50 }
51
52 int32 sum(List <int32>* list) {
53     int32 res = 0;
54     while (list != NULL) {
55         res += list->head;
56         list = list->tail;
57     } while
58     return res;
59 }

```

Output specification:

In header file:

```
1  template<class T>
2  T UTP_identity(T value)
3  {
4      return value;
5  }
6
7
8  template<class T>
9  class List {
10     T head;
11     List<T> * tail;
12
13     List(T h, List<T> *t) : head(h), tail(t) {
14     }
15
16
17     T lastItem(void) {
18         foobar<int32_t>(head, 1);
19         List<T> *node = this;
20         while (node->tail != NULL)
21         {
22             node = node->tail;
23         }
24         return node->head;
25     }
26
27     template<class S>
28     S foobar(T first, S second) {
29         return second;
30     }
31 };
32
33 class IntList :public List<int32_t> {
34     IntList(int32_t h) : head(h) {
35     }
36
37     IntList(int32_t h, List<int32_t> *t) : List(h, t) {
38     }
39
40 };
41
42 UTP_identity( value);
```

In code file:

```
1  static int8_t UTP_compare(T a, T b);
2
3  static int32_t UTP_multiplyBy(int32_t x);
4
5  static T UTP_deref(T *ref);
6
7  static int32_t UTP_sum(List<int32_t> *list);
8
9  template<class T>
10 T UTP_identity(T value)
11 {
12     return value;
```



```

13 }
14
15
16 template<class T>
17 static int8_t UTP_compare(T a, T b)
18 {
19     return (a < b) ? (-1) : ((a > b) ? (1) : (0));
20 }
21
22
23 template<int32_t n = 2>
24 static int32_t UTP_multiplyBy(int32_t x)
25 {
26     return x * n;
27 }
28
29
30 template<class T, T def>
31 static T UTP_deref(T *ref)
32 {
33     if ( ref == NULL )
34     {
35         return def;
36     }
37     else
38     {
39         return *ref;
40     }
41 }
42
43
44 static int32_t UTP_sum(List<int32_t> *list)
45 {
46     int32_t res = 0;
47     while (list != NULL)
48     {
49         res += list->head;
50         list = list->tail;
51     }
52     return res;
53 }

```

Environmental needs: The classes, their template declarations and their elements are all declared correctly.

3.2.12 Virtual

UTS-63

Test component: test.ex.com.mbeddr.cpp/[Virtual](#)

Input specification:

```

1 exported class Foo {
2     public int32 foo() {
3         return 0;

```

```

4     }
5     public virtual int32 bar() {
6         return 0;
7     }
8 }
9 exported class Bar : public Foo {
10     public int32 foo() {
11         return 1;
12     }
13     public int32 bar() {
14         return 1;
15     }
16 }
17
18 void virtualTest1() {
19     Foo* foo = new Foo;
20     Bar* bar = new Bar;
21     Foo* baz = new Bar;
22 }

```

Output specification:

In header file:

```

1 class Foo {
2     int32_t foo(void) {
3         return 0;
4     }
5     int32_t bar(void) {
6         return 0;
7     }
8 };
9
10 class Bar : public Foo {
11     int32_t foo(void) {
12         return 1;
13     }
14     int32_t bar(void) {
15         return 1;
16     }
17 };

```

In code file:

```

1 static void UTP_virtualTest1(void);
2
3 static void UTP_virtualTest1(void)
4 {
5     Foo *foo = new Foo;
6     Bar *bar = new Bar;
7     Foo *baz = new Bar;
8 }

```

Environmental needs: The classes and their elements are all declared correctly.

4 Test Procedures

4.1 Typesystem Rules

4.1.1 Identifier

Each unit test is located within a specific file. Given below is the mapping of unit tests to node test files.

UTS-1	AttrClassMemberSpecifier
UTS-2	MethClassMemberSpecifier
UTS-3	MethClassMemberSpecifier
UTS-4	ClassNesting
UTS-5	InheritanceInstance
UTS-6	NameCollisions
UTS-7	ClassConstructor
UTS-8	ClassConstructor
UTS-9	ClassConstructor
UTS-10	ConstructorInitializable
UTS-11	ClassTemplates
UTS-12	StaticClassMethodCallRule
UTS-13	NamespaceScope
UTS-14	NamespaceScope
UTS-15	NamespaceScope
UTS-16	LocalUsingKeyword
UTS-17	GlobalUsingKeyword
UTS-18	GlobalUsingKeyword
UTS-19	GlobalVarDecCPP
UTS-20	IdentifierNamedConceptKeywords
UTS-21	Polymorphism
UTS-22	NewDelete
UTS-23	NewDelete

UTS-24	OperatorOverloading
UTS-25	OperatorOverloading
UTS-26	OperatorOverloading
UTS-27	OperatorOverloading
UTS-28	FunctionTemplates
UTS-29	FunctionTemplates
UTS-30	FunctionTemplates
UTS-31	FunctionTemplates
UTS-32	FunctionTemplates
UTS-33	TemplateTypeDef
UTS-34	ClassTemplates
UTS-35	ClassTemplates
UTS-36	ClassTemplates
UTS-37	ClassTemplates
UTS-38	Refactor
UTS-39	Casting
UTS-40	Casting
UTS-41	Casting
UTS-42	Casting
UTS-43	Casting
UTS-44	ThisPointer
UTS-45	TryCatchStatement

4.1.2 Purpose

This test procedure executes all test cases associated with a test file in section 4.1.1. This consists of all test cases defined in section 3.1.

4.1.3 Procedure Steps

The subsequent procedure should be followed to run each unit test.

1. Open the project within MPS.
2. Within the ‘test.ts.com.mbeddr.cpp’ solution, right-click the node test file associated with the unit test in section 4.1.1.
3. Select the option ”Run” followed by the test file name.
4. The results of the tests are shown in the ’Run’ window.

4.2 C++ Code Generation

4.2.1 Identifier

Each unit test is located within a specific file, denoted as their test component, and each file hosts one unit test.

4.2.2 Purpose

This test procedure executes all code generation test cases. This consists of all test cases defined in section 3.2.

4.2.3 Procedure Steps

The subsequent procedure should be followed to run each unit test.

1. Open the project within MPS.
2. Within the ‘test.ex.com.mbeddr.cpp’ Solution, open the node test file associated with the unit test in section 3.2.
3. Copy all code within the file to another solution, containing both a build configuration file, and a C++ Module with the name ”UTP” and a main function.
4. Change all test-cases to void functions, by creating void functions with equal names and moving all statements within each test-case to their associated void function.
5. Remove all ’assert’ statements.
6. Ensure that 8 or less tabs are opened within the MPS editor.
7. Right-click within the file and select the “Preview Generated Text” option.
8. The results of the tests are shown in the generated header and code files as tabs within the MPS editor.

5 Coverage Report

This section describes the total coverage of all test cases.

5.1 Type System Rules

Each test case in section 3 tests a specific file within the Typesystem component. As such, every existing typesystem file that contains error and warning messages to test, is covered by the test cases in section 3.

5.2 C++ Code Generation

Each test case in section 3.2 tests part of the TextGen component. As such, every existing text generation file associated with the C++ elements, along with the support from the compiler and module text generator within mbeddr for these elements, is covered by the test cases in section 3.2.

6 Test Report

This section contains the results of all test cases.

6.1 Type System Rules

The type system test cases are described in section 3.1. The procedure to obtain the result figures shown below is described in section 4.1.3. The figures also show that every test case passed.

TSF-1

Test File: AttrClassMemberSpecifier

Test Cases: UTS-1

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.028 s	20180 Kb	1322738 Kb	1342918 ...	P:5
test.ts.com.mbeddr.cpp.attributes.AttrClassMemberSpecifier_Test...	1.028 s	20180 Kb	1322738 Kb	1342918 ...	P:5
test_NodeUseOfInlineNotNecessaryCheck74411442836336572	0.048 s	14632 Kb	1322738 Kb	1337370 Kb	Passed
test_NodeStaticDataMemberMustNotCheck74411442836337521	0.078 s	15628 Kb	1307109 Kb	1322738 Kb	Passed
test_NodeConstantDataMemberMustCheck74411442836337671	0.662 s	270283 Kb	957528 Kb	1227811 Kb	Passed
test_NodeConstantDataMemberMustCheck74411442836338052	0.206 s	79297 Kb	1227811 Kb	1307109 Kb	Passed
test_ErrorMessagesCheck3508428030145628485	0.034 s	5548 Kb	1337370 Kb	1342918 Kb	Passed

- ✓ AttrClassMemberSpecifier_Test (test.ts.com.mbeddr.cpp.attributes)
 - ✓ test_NodeUseOfInlineNotNecessaryCheck74411442836336572
 - ✓ test_NodeStaticDataMemberMustNotCheck74411442836337521
 - ✓ test_NodeConstantDataMemberMustCheck74411442836337671
 - ✓ test_NodeConstantDataMemberMustCheck74411442836338052
 - ✓ test_ErrorMessagesCheck3508428030145628485

Figure 1: AttrClassMemberSpecifier Test Results

TSF-2

Test File: MethClassMemberSpecifier

Test Cases: UTS-2, UTS-3

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	2.683 s	-150482 Kb	1025903 Kb	875420 Kb	P:7
test.ts.com.mbeddr.cpp.method.MethClassMemberSpecifier_Test (t...	2.683 s	-150482 Kb	1025903 Kb	875420 Kb	P:7
test_NodeNonpureVirtualMethodMustCheck74411442836409379	1.714 s	273234 Kb	1025903 Kb	1299137 Kb	Passed
test_NodeConstantExpressionMemberCheck74411442833389993	0.050 s	14607 Kb	875420 Kb	890027 Kb	Passed
test_NodeConstexprAlreadyImplicitelyCheck74411442833391262	0.038 s	6976 Kb	890027 Kb	897004 Kb	Passed
test_NodeStaticMemberMayNotBeVirtualCheck74411442833391469	0.045 s	5874 Kb	897372 Kb	903247 Kb	Passed
test_NodeStaticMemberMayNotBeVolatileCheck74411442835027061	0.047 s	13304 Kb	904059 Kb	917364 Kb	Passed
test_NodeStaticMemberMayNotBeConstCheck8479367613881128499	0.707 s	-444843 Kb	1299137 Kb	854294 Kb	Passed
test_NodeMonvirtualMethodMustNotCheck74411442836409097	0.082 s	21126 Kb	854294 Kb	875420 Kb	Passed

- ✓ MethClassMemberSpecifier_Test (test.ts.com.mbeddr.cpp.method)
 - ✓ test_NodeNonpureVirtualMethodMustCheck74411442836409379
 - ✓ test_NodeConstantExpressionMemberCheck74411442833389993
 - ✓ test_NodeConstexprAlreadyImplicitelyCheck74411442833391262
 - ✓ test_NodeStaticMemberMayNotBeVirtualCheck74411442833391469
 - ✓ test_NodeStaticMemberMayNotBeVolatileCheck74411442835027061
 - ✓ test_NodeStaticMemberMayNotBeConstCheck8479367613881128499
 - ✓ test_NodeMonvirtualMethodMustNotCheck74411442836409097

Figure 2: MethClassMemberSpecifier Test Results

TSF-3

Test File: ClassNesting

Test Cases: UTS-4

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.176 s	-172696 Kb	1372766 Kb	1200069 Kb	P:5
test.ts.com.mbeddr.cpp.classnesting.ClassNesting_Test (test.ts.co...	1.176 s	-172696 Kb	1372766 Kb	1200069 Kb	P:5
test_NodeClassMayNotContainInstanceCheck2532724293696919801	0.739 s	270852 Kb	1372766 Kb	1643619 Kb	Passed
test_NodeClassMayNotContainInstanceCheck2532724293696926042	0.227 s	-473399 Kb	1643621 Kb	1170221 Kb	Passed
test_NodeClassMayNotContainInstanceCheck2532724293696932229	0.067 s	6986 Kb	1170226 Kb	1177213 Kb	Passed
test_NodeCantReferenceInnerClassCheck2532724293696932598	0.100 s	10980 Kb	1177215 Kb	1188196 Kb	Passed
test_ErrorMessagesCheck3508428030145634703	0.043 s	11870 Kb	1188199 Kb	1200069 Kb	Passed

- ✓ ClassNesting_Test (test.ts.com.mbeddr.cpp.classnesting)
 - ✓ test_NodeClassMayNotContainInstanceCheck2532724293696919801
 - ✓ test_NodeClassMayNotContainInstanceCheck2532724293696926042
 - ✓ test_NodeClassMayNotContainInstanceCheck2532724293696932229
 - ✓ test_NodeCantReferenceInnerClassCheck2532724293696932598
 - ✓ test_ErrorMessagesCheck3508428030145634703

Figure 3: ClassNesting Test Results

TSF-4

Test File: InheritanceInstance

Test Cases: UTS-5

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.164 s	16620 Kb	936597 Kb	953217 Kb	P:5
test.ts.com.mbeddr.cpp.inheritance.InheritanceInstance_Test (test.t...	1.164 s	16620 Kb	936597 Kb	953217 Kb	P:5
test_NodeClassCannotExtendItselfCheck2532724293690159694	0.037 s	5213 Kb	936597 Kb	941810 Kb	Passed
test_NodeYouCantExtendAnUnexportedCheck2532724293690161417	0.121 s	21849 Kb	909512 Kb	931361 Kb	Passed
test_NodeYouCantExtendClassFromCheck2532724293690163166	0.038 s	4715 Kb	931882 Kb	936597 Kb	Passed
test_NodeTemplateClassTypeWithoutCheck2532724293690165144	0.933 s	-197752 Kb	1107264 Kb	909512 Kb	Passed
test_ErrorMessagesCheck3508428030145626209	0.035 s	11406 Kb	941810 Kb	953217 Kb	Passed

- ✓ InheritanceInstance_Test (test.ts.com.mbeddr.cpp.inheritance)
 - ✓ test_NodeClassCannotExtendItselfCheck2532724293690159694
 - ✓ test_NodeYouCantExtendAnUnexportedCheck2532724293690161417
 - ✓ test_NodeYouCantExtendClassFromCheck2532724293690163166
 - ✓ test_NodeTemplateClassTypeWithoutCheck2532724293690165144
 - ✓ test_ErrorMessagesCheck3508428030145626209

Figure 4: InheritanceInstance Test Results

TSF-5

Test File: NameCollisions

Test Cases: UTS-6

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.147 s	20028 Kb	743401 Kb	763430 Kb	P:5
test.ts.com.mbeddr.cpp.scope.NameCollisions_Test (test.ts.co...	1.147 s	20028 Kb	743401 Kb	763430 Kb	P:5
test_NodeNameCollisionAttributeCheck3910253520674809875	0.090 s	14495 Kb	743401 Kb	757897 Kb	Passed
test_NodeNameCollisionAttributeCheck3910253520674809584	0.048 s	12457 Kb	730944 Kb	743401 Kb	Passed
test_NodeNameCollisionMethodCheck3910253520674986883	0.868 s	-181785 Kb	893807 Kb	712021 Kb	Passed
test_NodeNameCollisionMethodCheck3910253520674987177	0.083 s	18610 Kb	712021 Kb	730631 Kb	Passed
test_ErrorMessagesCheck3508428030145634703	0.058 s	5532 Kb	757897 Kb	763430 Kb	Passed

- ✓ NameCollisions_Test (test.ts.com.mbeddr.cpp.scope)
 - ✓ test_NodeNameCollisionAttributeCheck3910253520674809875
 - ✓ test_NodeNameCollisionAttributeCheck3910253520674809584
 - ✓ test_NodeNameCollisionMethodCheck3910253520674986883
 - ✓ test_NodeNameCollisionMethodCheck3910253520674987177
 - ✓ test_ErrorMessagesCheck3508428030145634703

Figure 5: NameCollisions Test Results

TSF-6

Test File: ClassConstructor

Test Cases: UTS-7, UTS-8, UTS-9

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.173 s	28046 Kb	1056147 Kb	1084194 Kb	P:8
test.ts.com.mbeddr.cpp.constructor.ClassConstructor_Test (test.ts.co...	1.173 s	28046 Kb	1056147 Kb	1084194 Kb	P:8
test_NodeClassesMayOnlyHaveOneCheck1892164344769520647	0.062 s	17297 Kb	1056147 Kb	1073444 Kb	Passed
test_NodeDestructorsMayNotBeConstantCheck1892164344769517235	0.041 s	12722 Kb	1104015 Kb	1116738 Kb	Passed
test_NodeConstantExpressionMemberCheck4368610415798303781	0.054 s	12859 Kb	1131004 Kb	1143864 Kb	Passed
test_NodeMayNotHaveConstructorCheck4368610415798303935	0.845 s	355252 Kb	700895 Kb	1056147 Kb	Passed
test_NodeDestructorsMayNotHaveCheck4368610415798304811	0.055 s	14266 Kb	1116738 Kb	1131004 Kb	Passed
test_NodeMayNotHaveDestructorForCheck4368610415800007521	0.041 s	8589 Kb	1084194 Kb	1092783 Kb	Passed
test_NodeYouShouldSelectConstructorCheck4368610415798305189	0.034 s	10735 Kb	1092783 Kb	1103519 Kb	Passed
test_NodeWrongNumberOfArgumentsCheck5674958623601135996	0.041 s	10749 Kb	1073444 Kb	1084194 Kb	Passed

- ✓ ClassConstructor_Test (test.ts.com.mbeddr.cpp.constructor)
 - ✓ test_NodeClassesMayOnlyHaveOneCheck1892164344769520647
 - ✓ test_NodeDestructorsMayNotBeConstantCheck1892164344769517235
 - ✓ test_NodeConstantExpressionMemberCheck4368610415798303781
 - ✓ test_NodeMayNotHaveConstructorCheck4368610415798303935
 - ✓ test_NodeDestructorsMayNotHaveCheck4368610415798304811
 - ✓ test_NodeMayNotHaveDestructorForCheck4368610415800007521
 - ✓ test_NodeYouShouldSelectConstructorCheck4368610415798305189
 - ✓ test_NodeWrongNumberOfArgumentsCheck5674958623601135996

Figure 6: ClassConstructor Test Results

TSF-7

Test File: ConstructorInitializable

Test Cases: UTS-10

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	0.877 s	-195744 Kb	1069339 Kb	873594 Kb	P:2
test.ts.com.mbeddr.cpp.constructorinitializable.ConstructorInitia	0.877 s	-195744 Kb	1069339 Kb	873594 Kb	P:2
test_NodeInitializersAreNotAllowedCheck1963037008638239145	0.835 s	-208303 Kb	1069339 Kb	861036 Kb	Passed
test_ErrorMessagesCheck3508428030145626209	0.042 s	12558 Kb	861036 Kb	873594 Kb	Passed

- ✓ ConstructorInitializable_Test (test.ts.com.mbeddr.cpp.constructorinitializable)
 - ✓ test_NodeInitializersAreNotAllowedCheck1963037008638239145
 - ✓ test_ErrorMessagesCheck3508428030145626209

Figure 7: ConstructorInitializable Test Results

TSF-8

Test File: ClassTemplates

Test Cases: UTS-11, UTS-34, UTS-35, UTS-36, UTS-37

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.374 s	0 Kb	868152 Kb	868152 Kb	P:8
test.ts.com.mbeddr.cpp.templates.ClassTemplates_Test...	1.374 s	0 Kb	868152 Kb	868152 Kb	P:8
test_ErrorMessagesCheck8029970258445532190	0.063 s	14974 Kb	868152 Kb	883126 Kb	Passed
test_NodeTypeSystemCheck8029970258445539357	0.894 s	-189460 Kb	1015108 Kb	825647 Kb	Passed
test_NodeTypeSystemCheck8323098634528583755	0.055 s	13085 Kb	897075 Kb	910161 Kb	Passed
test_NodeTypeSystemCheck8323098634528580149	0.059 s	13949 Kb	883126 Kb	897075 Kb	Passed
test_NodeTypeSystemCheck8323098634528603724	0.056 s	16510 Kb	910680 Kb	927191 Kb	Passed
test_NodeWarningCheck2532724293689606323	0.112 s	14998 Kb	825647 Kb	840646 Kb	Passed
test_NodeTypeSystemCheck8323098634528608131	0.069 s	14467 Kb	927191 Kb	941658 Kb	Passed
test_NodeWarningCheck2532724293689606991	0.066 s	26985 Kb	841166 Kb	868152 Kb	Passed

```

✓ ClassTemplates_Test (test.ts.com.mbeddr.cpp.templates)
  ✓ test_ErrorMessagesCheck8029970258445532190
  ✓ test_NodeTypeSystemCheck8029970258445539357
  ✓ test_NodeTypeSystemCheck8323098634528583755
  ✓ test_NodeTypeSystemCheck8323098634528580149
  ✓ test_NodeTypeSystemCheck8323098634528603724
  ✓ test_NodeWarningCheck2532724293689606323
  ✓ test_NodeTypeSystemCheck8323098634528608131
  ✓ test_NodeWarningCheck2532724293689606991

```

Figure 8: ClassTemplates Test Results

TSF-9

Test File: StaticClassMethodCallRule

Test Cases: UTS-12

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	0.966 s	-174168 Kb	972073 Kb	797904 Kb	P:2
test.ts.com.mbeddr.cpp.staticclassmethodcall.StaticClassM...	0.966 s	-174168 Kb	972073 Kb	797904 Kb	P:2
test_NodeUnnamedWarningCheck1963037008638412851	0.915 s	-182714 Kb	972073 Kb	789358 Kb	Passed
test_ErrorMessagesCheck3508428030145626209	0.051 s	8546 Kb	789358 Kb	797904 Kb	Passed

```

✓ StaticClassMethodCallRule_Test (test.ts.com.mbeddr.cpp.static)
  ✓ test_NodeUnnamedWarningCheck1963037008638412851
  ✓ test_ErrorMessagesCheck3508428030145626209

```

Figure 9: StaticClassMethodCallRule Test Results

TSF-10

Test File: NamespaceScope

Test Cases: UTS-13, UTS-14, UTS-15

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.419 s	503147 Kb	693508 Kb	1196656 Kb	P:10
test.ts.com.mbeddr.cpp.namespace.NamespaceScope_Test (test...	1.419 s	503147 Kb	693508 Kb	1196656 Kb	P:10
test_NodeDuplicateNamesMayMakeCheck7557806416399944275	0.041 s	15345 Kb	693508 Kb	708854 Kb	Passed
test_NodeDuplicateNamesMayMakeCheck7557806416399944709	0.064 s	12273 Kb	708854 Kb	721127 Kb	Passed
test_NodeErrorCheck405507513514921377	0.052 s	13398 Kb	1196656 Kb	1210055 Kb	Passed
test_NodeDuplicateNamesMayMakeCheck7557806416399941681	0.091 s	-516546 Kb	1210055 Kb	693508 Kb	Passed
test_NodeErrorCheck8863643635247286085	0.906 s	352436 Kb	776483 Kb	1128920 Kb	Passed
test_NodeErrorCheck8863643635247286208	0.042 s	16030 Kb	1128920 Kb	1144951 Kb	Passed
test_NodeErrorCheck8863643635243063637	0.044 s	14091 Kb	1145633 Kb	1159725 Kb	Passed
test_NodeErrorCheck4973973365658388120	0.053 s	15457 Kb	1159725 Kb	1175183 Kb	Passed
test_NodeYouShouldSelectConstructorCheck405507513512651862	0.064 s	19690 Kb	721127 Kb	740818 Kb	Passed
test_ErrorMessagesCheck405507513512503374	0.062 s	21473 Kb	1175183 Kb	1196656 Kb	Passed

- ✓ NamespaceScope_Test (test.ts.com.mbeddr.cpp.namespace)
 - ✓ test_NodeDuplicateNamesMayMakeCheck7557806416399944275
 - ✓ test_NodeDuplicateNamesMayMakeCheck7557806416399944709
 - ✓ test_NodeErrorCheck405507513514921377
 - ✓ test_NodeDuplicateNamesMayMakeCheck7557806416399941681
 - ✓ test_NodeErrorCheck8863643635247286085
 - ✓ test_NodeErrorCheck8863643635247286208
 - ✓ test_NodeErrorCheck8863643635243063637
 - ✓ test_NodeErrorCheck4973973365658388120
 - ✓ test_NodeYouShouldSelectConstructorCheck405507513512651862
 - ✓ test_ErrorMessagesCheck405507513512503374

Figure 10: NamespaceScope Test Results

TSF-11

Test File: LocalUsingKeyword

Test Cases: UTS-16

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.038 s	0 Kb	1005092 Kb	1005092 Kb	P:5
test.ts.com.mbeddr.cpp.namespace.LocalUsingKeyword_Test (test...	1.038 s	0 Kb	1005092 Kb	1005092 Kb	P:5
test_NodeWarningCheck405507513514950604	0.056 s	11975 Kb	1005092 Kb	1017068 Kb	Passed
test_NodeAttributeReferencesToCheck405507513514908200	0.837 s	-191724 Kb	1162964 Kb	971239 Kb	Passed
test_NodeErrorCheck4482119505957031049	0.042 s	8649 Kb	971239 Kb	979888 Kb	Passed
test_NodeErrorCheck4482119505957031592	0.045 s	12872 Kb	979888 Kb	992760 Kb	Passed
test_ErrorMessagesCheck405507513512661318	0.058 s	12331 Kb	992760 Kb	1005092 Kb	Passed

- ✓ LocalUsingKeyword_Test (test.ts.com.mbeddr.cpp.namespace)
 - ✓ test_NodeWarningCheck405507513514950604
 - ✓ test_NodeAttributeReferencesToCheck405507513514908200
 - ✓ test_NodeErrorCheck4482119505957031049
 - ✓ test_NodeErrorCheck4482119505957031592
 - ✓ test_ErrorMessagesCheck405507513512661318

Figure 11: LocalUsingKeyword Test Results

TSF-12

Test File: GlobalUsingKeyword

Test Cases: UTS-17, UTS-18

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.041 s	-169886 Kb	921089 Kb	751203 Kb	P:3
test.ts.com.mbeddr.cpp.namespace.GlobalUsingKeyword_Test (te...	1.041 s	-169886 Kb	921089 Kb	751203 Kb	P:3
test_NodeAttributeDoesNotExistCheck8863643635247277263	0.735 s	266349 Kb	921089 Kb	1187439 Kb	Passed
test_NodeMethodDoesNotExistWithinCheck8863643635247277001	0.283 s	-446641 Kb	1187439 Kb	740797 Kb	Passed
test_ErrorMessagesCheck405507513514988374	0.023 s	10311 Kb	740891 Kb	751203 Kb	Passed

- ✓ GlobalUsingKeyword_Test (test.ts.com.mbeddr.cpp.namespace)
 - ✓ test_NodeAttributeDoesNotExistCheck8863643635247277263
 - ✓ test_NodeMethodDoesNotExistWithinCheck8863643635247277001
 - ✓ test_ErrorMessagesCheck405507513514988374

Figure 12: GlobalUsingKeyword Test Results

TSF-13

Test File: GlobalVarDecCPP

Test Cases: UTS-19

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.210 s	0 Kb	1162863 Kb	1162863 Kb	P:8
test.ts.com.mbeddr.cpp.globalvars.GlobalVarDecCPP_Test (test.ts...	1.210 s	0 Kb	1162863 Kb	1162863 Kb	P:8
test_NodeVariableDeclarationMayCheck2532724293691724724	0.050 s	13711 Kb	1162863 Kb	1176574 Kb	Passed
test_NodeChartMustNotBeAssignedCheck2532724293691724918	0.047 s	11004 Kb	1120176 Kb	1131181 Kb	Passed
test_NodeErrorCheck2532724293691725162	0.834 s	344740 Kb	750650 Kb	1095390 Kb	Passed
test_NodeChartMustNotBeAssignedCheck2532724293691725380	0.043 s	7968 Kb	1131181 Kb	1139150 Kb	Passed
test_NodeErrorCheck2532724293691725582	0.084 s	19452 Kb	1095390 Kb	1114843 Kb	Passed
test_NodeWchartMustNotBeAssignedCheck2532724293691725800	0.044 s	17368 Kb	1139150 Kb	1156519 Kb	Passed
test_NodeWarningCheck2532724293691771967	0.037 s	5333 Kb	1114843 Kb	1120176 Kb	Passed
test_ErrorMessagesCheck3508428030145626209	0.071 s	6344 Kb	1156519 Kb	1162863 Kb	Passed

- ✓ GlobalVarDecCPP_Test (test.ts.com.mbeddr.cpp.globalvars)
 - ✓ test_NodeVariableDeclarationMayCheck2532724293691724724
 - ✓ test_NodeChartMustNotBeAssignedCheck2532724293691724918
 - ✓ test_NodeErrorCheck2532724293691725162
 - ✓ test_NodeChartMustNotBeAssignedCheck2532724293691725380
 - ✓ test_NodeErrorCheck2532724293691725582
 - ✓ test_NodeWchartMustNotBeAssignedCheck2532724293691725800
 - ✓ test_NodeWarningCheck2532724293691771967
 - ✓ test_ErrorMessagesCheck3508428030145626209

Figure 13: GlobalVarDecCPP Test Results

TSF-14

Test File: IdentifierNamedConceptKeywords

Test Cases: UTS-20

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.654 s	16528 Kb	1081273 Kb	1097802 Kb	P:21
test.ts.com.mbeddr.cpp.identifiernames.IIdentifierNamedCo	1.654 s	16528 Kb	1081273 Kb	1097802 Kb	P:21
test_NodelsReservedKeywordCheck2532724293691473801	0.856 s	-204497 Kb	1081273 Kb	876775 Kb	Passed
test_NodelsReservedKeywordCheck2532724293691473885	0.080 s	18882 Kb	876775 Kb	895657 Kb	Passed
test_NodelsReservedKeywordCheck2532724293691473932	0.037 s	12655 Kb	895657 Kb	908313 Kb	Passed
test_NodelsReservedKeywordCheck2532724293691473999	0.043 s	16050 Kb	908313 Kb	924364 Kb	Passed
test_NodelsReservedKeywordCheck2532724293691474086	0.032 s	11076 Kb	924364 Kb	935440 Kb	Passed
test_NodelsReservedKeywordCheck2532724293691474193	0.048 s	11503 Kb	935440 Kb	946944 Kb	Passed
test_NodelsReservedKeywordCheck2532724293691474320	0.045 s	12652 Kb	946944 Kb	959597 Kb	Passed
test_NodelsReservedKeywordCheck2532724293691474467	0.072 s	15517 Kb	959597 Kb	975115 Kb	Passed
test_NodelsReservedKeywordCheck2532724293691474821	0.060 s	13703 Kb	975115 Kb	988818 Kb	Passed
test_NodelsReservedKeywordCheck2532724293691475008	0.032 s	6256 Kb	988818 Kb	995075 Kb	Passed

- ✓ IIdentifierNamedConceptKeywords_Test (test.ts.com.mbeddr.cpp.identifiernames)
 - ✓ test_NodelsReservedKeywordCheck2532724293691473801
 - ✓ test_NodelsReservedKeywordCheck2532724293691473885
 - ✓ test_NodelsReservedKeywordCheck2532724293691473932
 - ✓ test_NodelsReservedKeywordCheck2532724293691473999
 - ✓ test_NodelsReservedKeywordCheck2532724293691474086
 - ✓ test_NodelsReservedKeywordCheck2532724293691474193
 - ✓ test_NodelsReservedKeywordCheck2532724293691474320
 - ✓ test_NodelsReservedKeywordCheck2532724293691474467
 - ✓ test_NodelsReservedKeywordCheck2532724293691474821
 - ✓ test_NodelsReservedKeywordCheck2532724293691475008
 - ✓ test_NodelsReservedKeywordCheck2532724293691475215
 - ✓ test_NodelsReservedKeywordCheck2532724293691475442

Figure 14: IIdentifierNamedConceptKeywords Test Results

TSF-15

Test File: Polymorphism

Test Cases: UTS-21

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.542 s	179276 Kb	846735 Kb	1026012 Kb	P:9
test.ts.com.mbeddr.cpp.polymorphism.Polymorphism_Test (...)	1.542 s	179276 Kb	846735 Kb	1026012 Kb	P:9
test_NodeTypeCheck3508428030145339855	0.061 s	22113 Kb	846735 Kb	868848 Kb	Passed
test_NodeTypeCheck3508428030145339415	0.940 s	-181060 Kb	1027795 Kb	846735 Kb	Passed
test_NodeTypeCheck3508428030145340495	0.074 s	15679 Kb	868848 Kb	884527 Kb	Passed
test_NodeTypeCheck3508428030145344310	0.060 s	22603 Kb	885047 Kb	907651 Kb	Passed
test_NodeErrorCheck9083970262003174642	0.099 s	23527 Kb	908170 Kb	931698 Kb	Passed
test_NodeWarningCheck2532724293689524442	0.067 s	23984 Kb	952788 Kb	976773 Kb	Passed
test_NodeErrorCheck9083970262003183965	0.065 s	21090 Kb	931698 Kb	952788 Kb	Passed
test_NodeWarningCheck2532724293689526160	0.096 s	26778 Kb	976773 Kb	1003551 Kb	Passed
test_ErrorMessagesCheck3508428030145362576	0.080 s	21941 Kb	1004071 Kb	1026012 Kb	Passed

- ✓ Polymorphism_Test (test.ts.com.mbeddr.cpp.polymorphism)
 - ✓ test_NodeTypeCheck3508428030145339855
 - ✓ test_NodeTypeCheck3508428030145339415
 - ✓ test_NodeTypeCheck3508428030145340495
 - ✓ test_NodeTypeCheck3508428030145344310
 - ✓ test_NodeErrorCheck9083970262003174642
 - ✓ test_NodeWarningCheck2532724293689524442
 - ✓ test_NodeErrorCheck9083970262003183965
 - ✓ test_NodeWarningCheck2532724293689526160
 - ✓ test_ErrorMessagesCheck3508428030145362576

Figure 15: Polymorphism Test Results

TSF-16

Test File: NewDelete

Test Cases: UTS-22, UTS-23

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.250 s	-14932 Kb	770910 Kb	755977 Kb	P:7
test.ts.com.mbeddr.cpp.dynamicmemory.NewDelete_Test (test.t...	1.250 s	-14932 Kb	770910 Kb	755977 Kb	P:7
test_NodeTypeSystemCheck8123081327724939154	0.071 s	17466 Kb	770910 Kb	788376 Kb	Passed
test_NodeTypeSystemCheck8123081327724942966	0.069 s	12806 Kb	788396 Kb	801202 Kb	Passed
test_NodeAssignedVariablesNotCheck3508428030145611594	0.095 s	15492 Kb	817679 Kb	833172 Kb	Passed
test_NodeTypeSystemCheck8123081327724944128	0.058 s	15438 Kb	801721 Kb	817160 Kb	Passed
test_NodeAssignedVariablesNotCheck3508428030145613378	0.094 s	16599 Kb	833172 Kb	849771 Kb	Passed
test_NodeAssignedVariablesNotCheck8123081327725135343	0.060 s	14856 Kb	755977 Kb	770834 Kb	Passed
test_NodelsNotPointerOnlyVariablesCheck8123081327758339700	0.803 s	-199837 Kb	955814 Kb	755977 Kb	Passed

- ✓ NewDelete_Test (test.ts.com.mbeddr.cpp.dynamicmemory)
 - ✓ test_NodeTypeSystemCheck8123081327724939154
 - ✓ test_NodeTypeSystemCheck8123081327724942966
 - ✓ test_NodeAssignedVariablesNotCheck3508428030145611594
 - ✓ test_NodeTypeSystemCheck8123081327724944128
 - ✓ test_NodeAssignedVariablesNotCheck3508428030145613378
 - ✓ test_NodeAssignedVariablesNotCheck8123081327725135343
 - ✓ test_NodelsNotPointerOnlyVariablesCheck8123081327758339700

Figure 16: NewDelete Test Results

TSF-17

Test File: OperatorOverloading
Test Cases: UTS-24, UTS-25, UTS-26, UTS-27
Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.289 s	-485582 Kb	1224646 Kb	739063 Kb	P:6
test.ts.com.mbeddr.cpp.operatorOverloading.OperatorOverloading_Test ...	1.289 s	-485582 Kb	1224646 Kb	739063 Kb	P:6
test_NodeErrorCheck9083970262012151328	0.093 s	-505586 Kb	1224646 Kb	719059 Kb	Passed
test_NodeErrorCheck2532724293689398670	0.693 s	282217 Kb	844102 Kb	1126319 Kb	Passed
test_NodeNotAnArrayCheck2532724293689399627	0.065 s	27469 Kb	739063 Kb	766533 Kb	Passed
test_NodeErrorCheck2532724293689414707	0.277 s	98326 Kb	1126319 Kb	1224646 Kb	Passed
test_NodeNotAnArrayCheck2532724293689413653	0.075 s	25689 Kb	766533 Kb	792223 Kb	Passed
test_NodeErrorCheck9083970262012202444	0.086 s	20003 Kb	719059 Kb	739063 Kb	Passed

- ✓ OperatorOverloading_Test (test.ts.com.mbeddr.cpp.operatorOverloading)
 - ✓ test_NodeErrorCheck9083970262012151328
 - ✓ test_NodeErrorCheck2532724293689398670
 - ✓ test_NodeNotAnArrayCheck2532724293689399627
 - ✓ test_NodeErrorCheck2532724293689414707
 - ✓ test_NodeNotAnArrayCheck2532724293689413653
 - ✓ test_NodeErrorCheck9083970262012202444

Figure 17: OperatorOverloading Test Results

TSF-18

Test File: FunctionTemplates
Test Cases: UTS-28, UTS-29, UTS-30, UTS-31, UTS-32
Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	2.317 s	421835 Kb	719629 Kb	1141465 Kb	P:20
test.ts.com.mbeddr.cpp.templates.FunctionTemplates_Test (...)	2.317 s	421835 Kb	719629 Kb	1141465 Kb	P:20
test_NodeLeftoverTemplateStubCheck8323098634512839298	0.095 s	24206 Kb	719629 Kb	743835 Kb	Passed
test_NodeWarningCheck2532724293689633491	0.057 s	21612 Kb	800015 Kb	821627 Kb	Passed
test_NodeTypeSystemCheck8029970258445498966	0.101 s	-508426 Kb	1193837 Kb	685410 Kb	Passed
test_NodeUnreachableNodeErrorCheck8323098634512844317	0.077 s	30737 Kb	769277 Kb	800015 Kb	Passed
test_NodeTypeSystemCheck8323098634512844319	0.060 s	23128 Kb	1063843 Kb	1086971 Kb	Passed
test_NodeWarningCheck2532724293689634981	0.056 s	17581 Kb	821627 Kb	839209 Kb	Passed
test_NodeTypeSystemCheck8323098634512893156	0.065 s	25327 Kb	1086971 Kb	1112298 Kb	Passed
test_NodeWarningCheck2532724293689635926	0.065 s	17604 Kb	839209 Kb	856813 Kb	Passed
test_NodeParamWithoutDefaultAppearingCheck8323098634512908311	0.282 s	112561 Kb	951281 Kb	1063843 Kb	Passed
test_NodeWarningCheck2532724293689636881	0.078 s	23662 Kb	856813 Kb	880476 Kb	Passed

- ✓ FunctionTemplates_Test (test.ts.com.mbeddr.cpp.templates)
 - ✓ test_NodeLeftoverTemplateStubCheck8323098634512839298
 - ✓ test_NodeWarningCheck2532724293689633491
 - ✓ test_NodeTypeSystemCheck8029970258445498966
 - ✓ test_NodeUnreachableNodeErrorCheck8323098634512844317
 - ✓ test_NodeTypeSystemCheck8323098634512844319
 - ✓ test_NodeWarningCheck2532724293689634981
 - ✓ test_NodeTypeSystemCheck8323098634512893156
 - ✓ test_NodeWarningCheck2532724293689635926
 - ✓ test_NodeParamWithoutDefaultAppearingCheck8323098634512908311
 - ✓ test_NodeWarningCheck2532724293689636881
 - ✓ test_NodeTypeSystemCheck8029970258445495907
 - ✓ test_NodeWarningCheck2532724293689637796

Figure 18: FunctionTemplates Test Results

TSF-19

Test File: TemplateTypeDef

Test Cases: UTS-33

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	0.923 s	364929 Kb	762231 Kb	1127161 Kb	P:3
test.ts.com.mbeddr.cpp.templateypes.TemplateTypeDef_Test (t...	0.923 s	364929 Kb	762231 Kb	1127161 Kb	P:3
test_NodeDuplicateTypeNameCheck1963037008640599117	0.858 s	357244 Kb	762231 Kb	1119476 Kb	Passed
test_NodeDuplicateTypeNameCheck1963037008640599128	0.032 s	3108 Kb	1119480 Kb	1122588 Kb	Passed
test_ErrorMessagesCheck3508428030145626209	0.033 s	4570 Kb	1122590 Kb	1127161 Kb	Passed

- ✓ TemplateTypeDef_Test (test.ts.com.mbeddr.cpp.templateypes)
 - ✓ test_NodeDuplicateTypeNameCheck1963037008640599117
 - ✓ test_NodeDuplicateTypeNameCheck1963037008640599128
 - ✓ test_ErrorMessagesCheck3508428030145626209

Figure 19: TemplateTypeDef Test Results

TSF-20

Test File: Refactor

Test Cases: UTS-38

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	0.977 s	-10045 Kb	866386 Kb	856340 Kb	P:4
test.ts.com.mbeddr.cpp.typeInference.Refactor_Test (test.ts.co...	0.977 s	-10045 Kb	866386 Kb	856340 Kb	P:4
test_NodeTypeSystemCheck8323098634532843986	0.042 s	6838 Kb	866386 Kb	873224 Kb	Passed
test_NodeWarningCheck2532724293690135501	0.830 s	-208892 Kb	1050495 Kb	841603 Kb	Passed
test_NodeVariableDeclarationWithCheck8323098634532844948	0.054 s	9004 Kb	857381 Kb	866386 Kb	Passed
test_ErrorMessagesCheck4996299911451114940	0.051 s	14737 Kb	841603 Kb	856340 Kb	Passed

- ✓ Refactor_Test (test.ts.com.mbeddr.cpp.typeInference)
 - ✓ test_NodeTypeSystemCheck8323098634532843986
 - ✓ test_NodeWarningCheck2532724293690135501
 - ✓ test_NodeVariableDeclarationWithCheck8323098634532844948
 - ✓ test_ErrorMessagesCheck4996299911451114940

Figure 20: Refactor Test Results

TSF-21

Test File: Casting

Test Cases: UTS-39, UTS-40, UTS-41, UTS-42, UTS-43

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	1.134 s	61354 Kb	845723 Kb	907078 Kb	P:5
test.ts.com.mbeddr.cpp.casting.Casting_Test (test.ts.com.mbeddr.c...	1.134 s	61354 Kb	845723 Kb	907078 Kb	P:5
test_NodeStaticcastCanOnlyCastCheck2882715192743049420	0.098 s	16771 Kb	845723 Kb	862494 Kb	Passed
test_NodeReinterpretcastCanOnlyCheck4996299911456233341	0.849 s	-184730 Kb	1030454 Kb	845723 Kb	Passed
test_NodeDynamiccastCanOnlyCastCheck4996299911457691056	0.064 s	15185 Kb	862523 Kb	877709 Kb	Passed
test_NodeConstTypeWithoutInitValueCheck2532724293689338547	0.063 s	23530 Kb	877713 Kb	901244 Kb	Passed
test_NodeTypeCheck4996299911459184108	0.060 s	5830 Kb	901248 Kb	907078 Kb	Passed

- ✓ Casting_Test (test.ts.com.mbeddr.cpp.casting)
 - ✓ test_NodeStaticcastCanOnlyCastCheck2882715192743049420
 - ✓ test_NodeReinterpretcastCanOnlyCheck4996299911456233341
 - ✓ test_NodeDynamiccastCanOnlyCastCheck4996299911457691056
 - ✓ test_NodeConstTypeWithoutInitValueCheck2532724293689338547
 - ✓ test_NodeTypeCheck4996299911459184108

Figure 21: Casting Test Results

TSF-22

Test File: ThisPointer

Test Cases: UTS-44

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	0.847 s	371983 Kb	719872 Kb	1091856 Kb	P:2
test.ts.com.mbeddr.cpp.thisPointer.ThisPointer_Test...	0.847 s	371983 Kb	719872 Kb	1091856 Kb	P:2
test_NodeTypeSystemCheck2992941503319134866	0.801 s	355903 Kb	719872 Kb	1075776 Kb	Passed
test_ErrorMessagesCheck2992941503319092462	0.046 s	16079 Kb	1075776 Kb	1091856 Kb	Passed




 ThisPointer_Test (test.ts.com.mbeddr.cpp.thisPointer)
 test_NodeTypeSystemCheck2992941503319134866
 test_ErrorMessagesCheck2992941503319092462

Figure 22: ThisPointer Test Results

TSF-23

Test File: TryCatchStatement

Test Cases: UTS-45

Result Figure:

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	0.944 s	378096 Kb	836204 Kb	1214301 Kb	P:3
test.ts.com.mbeddr.cpp.trycatch.TryCatchStatement_Test (test...	0.944 s	378096 Kb	836204 Kb	1214301 Kb	P:3
test_NodeWarningCheck2532724293690129946	0.850 s	358090 Kb	836204 Kb	1194295 Kb	Passed
test_NodeCatchBlockAfterCatchallCheck1963037008640623672	0.042 s	6519 Kb	1194295 Kb	1200815 Kb	Passed
test_ErrorMessagesCheck3508428030145626209	0.052 s	13482 Kb	1200818 Kb	1214301 Kb	Passed





 TryCatchStatement_Test (test.ts.com.mbeddr.cpp.trycatch)
 test_NodeWarningCheck2532724293690129946
 test_NodeCatchBlockAfterCatchallCheck1963037008640623672
 test_ErrorMessagesCheck3508428030145626209

Figure 23: TryCatchStatement Test Results

6.2 C++ Code Generation

The code generation test cases are described in section 3.2. To obtain the results for the C++ code generation test cases, the steps described in section 4.2.3 should be followed. For each test case, the header and code file tabs within the MPS editor show that it passed.