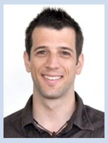


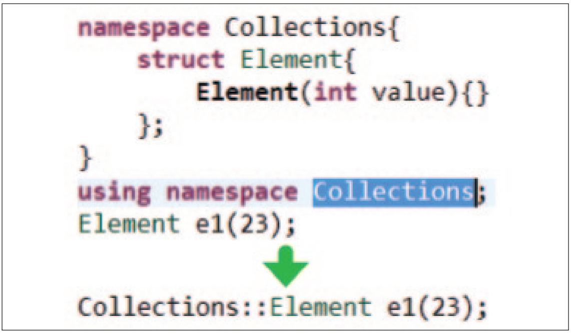
Ueli
Kunz

Julius
Weder

Graduate Candidates	Ueli Kunz, Julius Weder
Examiner	Prof. Dr. Luc Bläser
Co-Examiner	Martin Botzler, Siemens AG Zug
Subject Area	Software

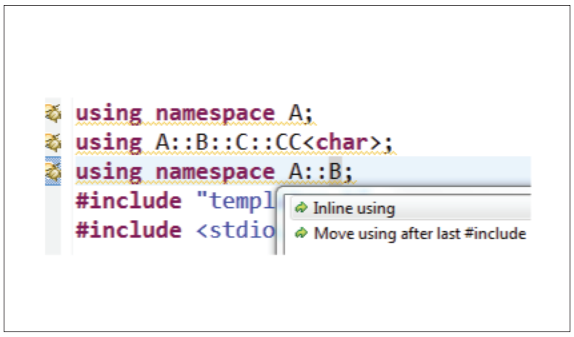
Namespactor – Eclipse CDT nNamespace Refactoring Plug-in

A tool to simplify and accelerate working with names in C++



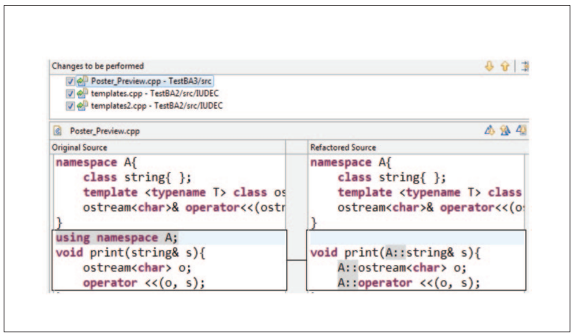
```
namespace Collections{
    struct Element{
        Element(int value){}
    };
}
using namespace Collections;
Element e1(23);

Collections::Element e1(23);
```

Inline Using Directive refactoring: qualifies all affected names in the scope of the selected using directive with its name and removes it.



```
using namespace A;
using A::B::C::CC<char>;
using namespace A::B;
#include "templ
#include <stdio
```

Problem markers and suggested resolutions



Changes to be performed
Poster_Preview.cpp - TestBA3/src
templates.cpp - TestBA2/src/IUDEC
templates2.cpp - TestBA2/src/IUDEC
Poster_Preview.cpp
Original Source
Refactored Source

```
namespace A{
    class string{ };
    template <typename T> class os
    ostream<char>& operator<<(ostr
}
using namespace A;
void print(string& s){
    ostream<char> o;
    operator <<(o, s);
```

```
namespace A{
    class string{ };
    template <typename T> class
    ostream<char>& operator<<(o
}

void print(A::string& s){
    A::ostream<char> o;
    A::operator <<(o, s);
```

Refactoring wizard previewing the changes applied by the Inline Using Directive refactoring

Introduction: C++ allows the introduction of names into program scopes by way of «using directives» and «using declarations», such that symbols can be referred to without their qualified name. However, the manual changing of names and the switching between their qualified and unqualified representation is error prone and time consuming. Therefore, we have developed Namespactor, a new automated C++ refactoring tool for names and namespaces for the Eclipse C/C++ Development Tooling platform (CDT).

Approach/Technologies: Namespactor is a new Eclipse plug-in which offers a set of common and effective namespace refactoring functions, such as switching between qualified and unqualified naming, introducing and moving «using declarations» or «using directives» and more. Namespactor is realized with the Eclipse Language Toolkit (LTK), the API for integrating automated refactorings in Eclipse IDE. There is currently no other similar namespace refactoring tool publicly available for the C++ programming language.

Result: The following refactorings are implemented in Namespactor:

- Inline Using Directive: Removes a using directive and qualifies the affected names
- Inline Using Declaration: Removes a using declaration and qualifies the occurrences of the affected name
- Qualify an Unqualified ID: Fully qualifies an unqualified ID with all required names
- Extract Using Directive: Introduces a using directive for a qualified name and removes the name qualifier(s) of the affected qualified names
- Extract Using Declaration: Introduces a using declaration for a qualified name and removes the name qualifier(s) on the occurrences of the affected qualified name

Namespactor provides static code analysis checkers in accordance with some namespace rules of good practice. These checkers mark problematic source code sections and provide the following quick fixes to solve the problems:

- Inline Using: Initiates the Inline Using Directive or the Inline Using Declaration refactoring
- Move Using after Include Directive: Moves a using directive or a using declaration after the include directives
- Qualify Using Directive: Initiates the Qualify an Unqualified ID refactoring