

Refactoring Techniques and Automated Approaches Through Tool Support

Seminar in Software Engineering

Peter Feichtinger
Lisa Kritzingner

Institute for Software Systems Engineering
Johannes Kepler University Linz

13. June 2017

1 Introduction

- Automated Refactoring in General

2 Automated Refactoring Approaches

- Restructuring Legacy C Code into C++
- Performance Impact of Polymorphism
- Design Differencing
- The Spartanizer: Massive Automatic Refactoring

Refactoring

- Restructuring code without change in semantics
 - Importance in software evolution is obvious
 - Tool support is important
- Topics:
 - Performance impact of refactoring
 - Modernizing code
 - Automated refactoring

Refactoring

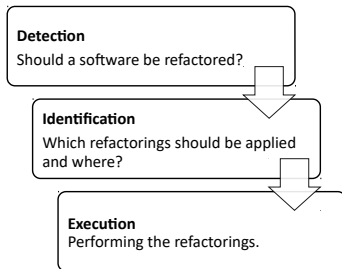
- Restructuring code without change in semantics
 - Importance in software evolution is obvious
 - Tool support is important
- Topics:
 - Performance impact of refactoring
 - Modernizing code
 - Automated refactoring

Automated Refactoring

- Goals
 - Understandability
 - Correctness
 - Ease of Maintenance and Evolution
- Automated Refactoring Steps

Automated Refactoring

- Goals
 - Understandability
 - Correctness
 - Ease of Maintenance and Evolution
- Automated Refactoring Steps



Restructuring Legacy C Code into C++

- Case study on Mosaic browser code
- Combination of refactorings to create classes
 - From C structs
 - From related variables

Restructuring Legacy C Code into C++

- Case study on Mosaic browser code
- Combination of refactorings to create classes
 - From C structs
 - From related variables

Performance Impact of Polymorphism

- Comparison of the performance of two programmes
 - one which contains large conditionals
 - one where the conditionals are implemented using polymorphism

Performance Impact of Polymorphism

```

class ConditionalWidget {
    short mType;
    int mData;
public:
    ConditionalWidget(short type, int data)
        : mType(type), mData(data) { }
    int actionIf();
    int actionSwitch();
};

int ConditionalWidget::actionIf() {
    if(mType == 0) {
        return mData + 1;
    } else if(mType == 1) {
        return mData - 3;
    }
    ...
} else {
    return -1;
}
}

int ConditionalWidget::actionSwitch() {
    switch(mType) {
        case 0: return mData + 1;
        case 1: return mData - 3;
        ...
        default: return -1;
    }
}
}

```

```

class PolymorphicWidget {
protected:
    int mData;
public:
    explicit PolymorphicWidget(int d)
        : mData(d) { }
    virtual ~PolymorphicWidget() = default;
    virtual int action() {
        return -1; // Default case
    }
};

class Widget0 : public PolymorphicWidget {
public:
    Widget0(int d) : PolymorphicWidget(d) {
    }
    int action() override {
        return mData + 1;
    }
}

class Widget1 : public PolymorphicWidget {
public:
    Widget1(int d) : PolymorphicWidget(d) {
    }
    int action() override {
        return mData - 3;
    }
}
...

```

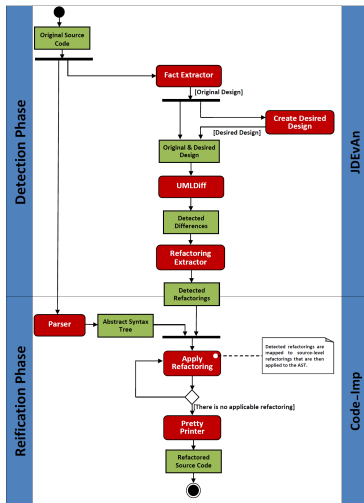
Design Differencing

- Novel refactoring approach that refactors a program based on
 - desired design
 - source code
- Using desired design as target, based on
 - current software design and
 - understanding of how it may be required to evolve

Design Differencing

- Novel refactoring approach that refactors a program based on
 - desired design
 - source code
- Using desired design as target, based on
 - current software design and
 - understanding of how it may be required to evolve

Design Differencing



- **JDEvAn**

- Eclipse plugin
- analyzes a software system's design-evolution history and provides information about the system's history
- provides information about the system's history

- **Code-Imp**

- fully automated refactoring framework
- driven by a search technique, steepest-ascent hill climbing

The Spartanizer: Massive Automatic Refactoring

- Tool demo paper
- Eclipse plugin for automatic refactoring to make code more compact
- Shows that automatic refactoring can be used effectively

The Spartanizer: Massive Automatic Refactoring

- Tool demo paper
- Eclipse plugin for automatic refactoring to make code more compact
- Shows that automatic refactoring can be used effectively

The Spartanizer: Massive Automatic Refactoring

```

public class C0<T> {
    private T inner;
    public C0(T inner) {
        super();
        this.inner = inner;
    }
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((inner==null)
            ? 0 : inner.hashCode());
        return result;
    }
    public boolean equals(Object obj) {
        if(this == obj)
            return true;
        if(obj == null)
            return false;
        if(getClass() != obj.getClass())
            return false;
        C0 other = (C0) obj;
        if(inner == null) {
            if(other.inner != null)
                return false;
        } else if(!inner.equals(other.inner))
            return false;
        return true;
    }
}

```

```

public class C1<T> {
    private final T inner;
    public C1(T inner) {
        this.inner = inner;
    }
    public int hashCode() {
        return 31 + ((inner == null) ? 0 :
            inner.hashCode());
    }
    public boolean equals(Object c) {
        return c == this ||
            c != null && getClass() ==
            c.getClass() && equals((C1) c);
    }
    private boolean equals(C1 c) {
        return inner == null ?
            c.inner == null :
            inner.equals(c.inner);
    }
}

```


Evaluation

Table: Comparison with respect to achieved goals

	Understandability	Correctness	Maintainability
C to C++	+	+	+
Polymorphism	o	+	+
Design Diff.	o	+	+
JDEvAn	o	-	+
Code-Imp	o	-	+
The Spartanizer	+	o	+

Evaluation

Table: Comparison with respect to supported steps

	Detection	Identification	Execution
C to C++	o	o	o
Polymorphism	o	o	-
Design Diff.	-	o	+
JDEvAn	+	+	+
Code-Imp	+	+	+
The Spartanizer	o	+	+

Papers



Demeyer

Maintainability versus Performance: What's the Effect of Introducing Polymorphism?

Technical Report, Lab. on Reengineering, Universiteit Antwerpe, 2002



D'Hondt, De Volder, Mens, Wuyts

Co-evolution of Object-Oriented Software Design and Implementation

Software Architectures and Component Technology, 2002



Moghadam, Ó Cinnéide

Automated Refactoring using Design Differencing

16th European Conference on Software Maintenance and Reengineering (CSMR), 2012

Papers



Fanta, Rajlich

Restructuring Legacy C Code into C++

IEEE International Conference on Software Maintenance (ICSM), 1999



Gil, Orrù

The Spartanizer: Massive Automatic Refactoring

24th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2017