CodeSkelGen
A Program Skeleton Generator

Ricardo Queirós

June 20, 2013

SLATE’13
Faculdade de Ciências da Universidade do Porto
1. Motivation
2. Approach
3. CodeSkelGen
   1. Architecture
   2. Annotation type
   3. Annotation processor
   4. Program annotation
4. Integration into an educational setting
5. Future work
Motivation

Computer programming

- Difficult to teach
  - high number of students
  - extensive course syllabus
  - number of TA insufficient
  - lack of exercises

- Difficult to learn
  - subject complexity
  - teaching methods
  - learning styles
  - students motivation
Computer programming

- Difficult to teach
  - high number of students
  - extensive course syllabus
  - number of TA insufficient
  - lack of exercises

- Difficult to learn
  - subject complexity
  - teaching methods
  - learning styles
  - students motivation
Motivation

Learning through practice

- Best way to learn and to engage novice students
- Practice boils down to solve exercises
- Several studies reveal that novice students have a poor performance when solving programming exercises from scratch
- Main reasons:
  - have poor problem solving skills
  - lack knowledge of fundamental programming constructs
Learning through practice

- Best way to learn and to engage novice students
- Practice boils down to solve exercises
- Several studies reveal that novice students have a poor performance when solving programming exercises from scratch
- Main reasons:
  - have poor problem solving skills
  - lack knowledge of fundamental programming constructs
- Consequences:
  - discouraged at the outset
  - high failure and dropout rates
Motivation

Learning through practice

- Best way to learn and to engage novice students
- Practice boils down to solve exercises
- Several studies reveal that novice students have a poor performance when solving programming exercises from scratch
- Main reasons:
  - have poor problem solving skills
  - lack knowledge of fundamental programming constructs
- Consequences:
  - discouraged at the outset
  - high failure and dropout rates
Approach

Connecting-the-dots learning

- Use of scaffolding based tools

- Delivery of skeleton or buggy programs
  - the "problem-solving" issue is softened
  - students’ working memory is free to focus on the core of the problem and abstract their foundations
Approach

Connecting-the-dots learning

- Use of scaffolding based tools

- Delivery of skeleton or buggy programs
  - the "problem-solving" issue is softened
  - students’ working memory is free to focus on the core of the problem and abstract their foundations
CodeSkelGen

- Scaffolding tool to generate Java programs
- Based on annotated programs
- Annotations
  - described within an annotation type
  - processed by an annotation processor
- Type of programs
  1. Skeleton programs
  2. Buggy programs
Types of programs

1. **Skeleton programs**
   - include: high level structures
   - goal: accelerate the beginning of exercises resolution

2. **Buggy programs**
   - include: logic and/or execution errors
   - goal: stimulate students to debug and test their programs
Annotation type

- Syntactic metadata added to Java source code
- CodeSkelGen annotation type

```java
package CodeSkelGen;
@Retention ( RetentionPolicy . SOURCE )
public @interface CSG {
String changeOperator () default "";
String changeValue () default "";
String changeVariable () default "";
String comment () default "";
String removeRefVariable () default "";
boolean removeBody () default false;
... }
```
Annotation processor

- A compiler plug-in usually called processor
- A processor will "process" one or more annotation types.
- Excerpt of the annotation processor

```java
SupportedAnnotationTypes( "CodeSkelGen.CSG" )
@SupportedSourceVersion( SourceVersion.RELEASE_6 )
public class CSGAnnotationProc extends AbstractProcessor {
    public CSGAnnotationProc() {
        super();
    }
    @Override
    public boolean process(..., RoundEnvironment r) {
        // For each element annotated with the CSG annotation
        for (Element e : r.getElementsAnnotatedWith(CSG.class)) {
            // Process code here
        }
        return true;
    }
}
```
public class Program {
    @CSG(comment = "Calculate the factorial of the sum of two numbers")
    public static void main(String[] args) {
        long num1 = Long.parseLong(args[0]);
        long num2 = Long.parseLong(args[1]);
        long total = sum(num1, num2);
        System.out.println("Factorial of " + total + " is " + fact(total));
    }
    public static long fact(long num) {
        @CSG(changeValue=">")
        if (num <=1 ) return 1;
        else
            @CSG(changeOperator)
            return num * factorial(num - 1);
    }
    @CSG(comment="Complete the method!", removeBody=true)
    public static long sum(long num1, long num2) {
        return num1+num2;    }
}

public class Program {
    public static void main(String[] args) {
        long num1 = Long.parseLong(args[0]);
        long num2 = Long.parseLong(args[1]);
        long total = sum(num1, num2);
        System.out.println("Factorial of \(\) + total + "is\(\) +
                            fact\(\) (total));
    }

    public static long factorial(long num) {
        if (num > 1)
            return 1;
        else
            return num * factorial(num + 1);
    }

    // Complete the method!
    public static long sum(long num1, long num2) {
        return 1;
    }
}
Integration of CodeSkelGen in an Ensemble instance

- Ensemble instance
  - Relies on the practice of programming exercises to improve programming skills
  - Includes a set of components for the creation, storage, visualisation and evaluation of programming exercises
  - Orchestrated by a central component (teaching assistant) that mediates the communication among all components

- Integration steps:
  1. Include annotation processor in the exercise generator
  2. Extend PExIL schema
Integration of CodeSkelGen in an Ensemble instance

- **Ensemble instance**
  - Relies on the practice of programming exercises to improve programming skills
  - Includes a set of components for the creation, storage, visualisation and evaluation of programming exercises
  - Orchestrated by a central component (teaching assistant) that mediates the communication among all components

- **Integration steps:**
  1. Include annotation processor in the exercise generator
  2. Extend PExIL schema
Integration into an Educational Setting

1. Include annotation processor in the exercise generator
   • Programming exercises authoring process
Integration into an Educational Setting

2. Extend PExIL schema

- PExIL data model
Conclusions and Future work

Main contribution
- approach used to generate partial programs

Advantages:
- the processor is external to the source code
- the same annotated solution program can be the base for several different versions

Disadvantages:
- language dependent (Java)
- teacher must learn the elements of the annotation type

Future work
1. enrich the CSG interface with more pertinent constructs
2. implement CodeSkelGen
3. find a language-independent approach

Ricardo Queirós
CodeSkelGen
Conclusions and Future work

Main contribution

- approach used to generate partial programs

Advantages:

- the processor is external to the source code
- the same annotated solution program can be the base for several different versions

Disadvantages:

- language dependent (Java)
- teacher must learn the elements of the annotation type

Future work

1. enrich the CSG interface with more pertinent constructs
2. implement CodeSkelGen
3. find a language-independent approach
Conclusions and Future work

Main contribution

- approach used to generate partial programs

Advantages:

- the processor is external to the source code
- the same annotated solution program can be the base for several different versions

Disadvantages:

- language dependent (Java)
- teacher must learn the elements of the annotation type

Future work

1. enrich the CSG interface with more pertinent constructs
2. implement CodeSkelGen
3. find a language-independent approach
Conclusions and Future work

Main contribution
  - approach used to generate partial programs

Advantages:
  - the processor is external to the source code
  - the same annotated solution program can be the base for several different versions

Disadvantages:
  - language dependent (Java)
  - teacher must learn the elements of the annotation type

Future work
  1. enrich the CSG interface with more pertinent constructs
  2. implement CodeSkelGen
  3. find a language-independent approach
it was an elephant... :-}