Western Norway University of Applied Sciences

Technology for Programming Course Assignments

Erlend Berntsen

28.02.2022

Høgskulen på Vestlandet

Contents

Introduction

Most courses at informatic departments in universities revolve around programming. Students will generally have programming assignments throughout the courses to assess their knowledge and skills, which are often graded by lecturers and/or teaching assistants with some feedback added to the grade. Producing such assignments can be a long and tedious task and is prone to many human errors. Programming assignments is split into many closely related but independent parts. Often there is some starting code produced to help the students focus on the specific purpose of the assignment without doing everything from scratch. The creator of the assignments generally writes the “solution code” before publishing the assignment, often accompanied by tests to make sure the solution is sound. The solution code is then removed, and the students receive some starting code and tests. The assignment text is usually written completely independent from the assignment code. Lastly, the assignment is corrected when the students hand in their solutions. This step is usually a combination of automated tests and manual inspection of the source code. Some courses focus more on code quality, which is usually done by manually inspecting the solution. In recent years, there has been a rapidly growing development of tools and frameworks that can automatically evaluate code quality.

Most of these tasks in creating programming assignments involves manual work by a human. A change in one part may need to be reflected in the other parts. For example, a change in the starting code may produce necessary changes to certain tests and the assignment text. Since these parts are mostly independent of each other, manual refactoring is often needed. Manual refactoring is prone to errors since developers have a tendency to be lazy, forgetful or just inadvertently producing errors.

Background

Code generation and transformation has been around for a long time, but nothing is directly tied to the case of generating different artefacts for programming assignments. The term “low code” has had a steady increase in popularity the last decades, which focuses on quickly producing applications with minimal effort. Low-Code programming is based upon model-driven software development, rapid application development, automatic code generation and visual programming [1]. This project is also inspired by these paradigms, with less emphasis on visual programming.

There is a plethora of tools for generating and transforming code available today. Meta3 is a conceptual framework for creating code generators for domain specific languages (DSLs) [2]. This framework identifies several layers of a system for generating code from DSLs: input languages and metamodels, conceptual metamodels, and output artifacts and related metamodels. Changes between these layers are done by model-to-model (M2M) and model-to-text (M2T) transformations. Other transformation tools, such as the C/C++ refactoring tool Nobrainer, are built and used in the same language, and argues that avoiding using DSLs increases the ease of use [6].

Xevolver is a framework for code translation of source code by letting developers specify their own code translation rules [3]. It is mainly focused on transforming code by optimizing it for better performance of applications. This is done by turning the source code into an abstract syntax tree (AST) and converting that into an XML document. User-defined code transformations can then be written with XML data translation tools. Xevtgen builds on similar ideas but generates the transformation rules automatically based on input and output code patterns [4]. Other transformation frameworks make use of machine learning [5].

There is a lot of related work done on programming assignments in an educational context when it comes to correcting and giving feedback to solutions automatically or semi-automatically. In [7] a tool was developed to give more specific feedback to students when solutions were incorrect as a way of having more positive effect on student engagement, while also giving tutors an overview of the different problems students encounter. An inquiry-based tool with conceptual feedback is also shown to help students produce better code compared to normal code coverage feedback [8].

Real-time feedback on assignments can enhance the learning experience of students, and a scoreboard based on the grading results can motivate students to work harder on the assignments before submitting them [9]. Penalties can be introduced to stop students from relying too much on autograders [10]. This results in students testing their own solutions more before checking it with an autograder.

Research questions

The current issues with programming assignments lead to an interesting research question:

> **How can we create a better and more automated workflow for creating and correcting programming assignments?**

This master project will focus on developing a better approach and technical solution for creating and correcting programming assignments. By automating steps wherever possible we hope that this will reduce both inconsistencies and time spent creating programming assignments. The solution is aimed at connecting the currently independent parts so that there is always consistency between them. Continuous integration technology and model-driven software engineering will be the foundation of the project implementation. Code generation and transformation will also play a central role in the implementation. Testing and autograding of student solutions is the last part of the implementation. This will be mostly concerned about integrating with preexisting tools since there is already much work done on this topic.

Ideally, the solution will be programming language agnostic to be as accessible and useful as possible. However, most programming courses today focus either on Java or Python (occasionally C, Haskell, or JavaScript). To keep the project focused, and to identify a feasible path, I will work only with Java at the start, and will reevaluate later to see if a more general solution is feasible. This is in turn leads to a second research question:

- **To what degree is making a general language agnostic solution possible?**

How the solution will be provided still needs to be investigated. To keep the development environment for the author of the programming assignments as familiar as possible, the solution should ideally be seamlessly integrated into an IDE without being tied to a specific IDE. Maven and Gradle are popular build tools that may be suited to keep the solution IDE independent. A Java specific solution using annotations could be developed as a library, perhaps in combination with Maven or Gradle.

In recent years, cloud technology has been used to hand out/in assignments to students. From personal experience, I have used cloud technologies such as GitLab and GitHub for retrieving and handing in assignments, and CodeGrade (as a TA) to grade solutions from students. Ideally, this project will be rather successful and actually be used in programming courses. To make that more likely, the solution should take such cloud technologies into consideration and be integrated with them in some way.

An implementation is hopefully ready by fall so that we can evaluate it in a real educational environment. The plan is to evaluate it on DAT100 and DAT110 programming assignments. DAT100 is an introductory course for programming with Java, while DAT110 focuses on distributed systems and networking, also with Java as the programming language. This will provide a good opportunity to do qualitative research by getting feedback and evaluation of the system from both students and teachers.

Research methods and evaluation

The main focus of this project is the development of a software tool/platform. The result of the project will most likely be a language specific (Java) solution. Hopefully, the solution works well enough so that it can be used as a new and better procedure for creating and correcting programming assignments in courses at universities. We plan to use preexisting programming assignments to test and further improve the tool. If a general, language agnostic solution is not implemented, then a descriptive model of the architectural style can be useful for evaluating and illustrating the possibility of such a general solution. Research into similar projects and literature about code generation, meta-programming, compiler technology, and more, is needed to accurately create and analyze such a model.

Figuring out if this new way of creating and correcting programming assignments is better than before, and if so, the question of how much better is difficult to answer. Experiences from a real scenario would be important to determine its usefulness. By taking use of the project in courses we can do qualitative research by getting feedback from both students and teachers in the form of questionnaires. Although this is insightful in determining the success of the project, it will be mostly defined by personal opinions and preferences.

Quantitative research will be detrimental to analyze the usefulness of the project in a more objective manner. One of the goals of the project is to save time producing all the different artefacts of a programming assignment. The effectiveness of the project can be determined by comparing how much resources are used writing assignments with and without the tool.

“Resources” here is intentionally left vague since we have yet to determine how to measure it. There are many metrics that can be suited in this regard, but it is difficult to know which one is the best one to measure effectiveness. Some examples are

- Time spent writing code
- Lines of code written
- A custom measuring system

Reducing errors and always keeping the different artefacts consistent is also one of the main goals of the project. One way to compare how consistent or how easily the artefacts are kept consistent, is to inject changes into to the assignments and check how much work is needed to maintain consistency. Again, “work” is vague for the same reason.

In the case that only a language specific solution is implemented, a descriptive model of a language agnostic solution is likely created. Analysis and evaluation of such a model is used to determine its feasibility. Persuasion would be used to hypothesize a working solution and to discuss the next steps to take from the language specific solution.

Engineering method

One of the challenges is injecting metadata in the source code that describes what should be generated from it. Source code that doesn’t conform to the Java syntax will result in compilation errors. To circumvent this, a custom grammar that handles both the metadata information and source code can be created. This would essentially be a domain specific language (DSL) that needs a compilation of the DSL file, and then an additional compilation of the Java source code. A simpler solution that doesn’t involve creating a new DSL would be to take advantage of the Java annotations API. This makes it easy to write metadata directly in the source code without introducing new grammar and syntax. An additional advantage of using annotations is that metadata about the source code, such as method and file names for example, can easily be retrieved with reflection. The annotations need to be handled by a *processor* that gathers information about the annotations, source code, and other metadata. After processing the data, a *builder* will handle what will be generated, including its location and content.

An example of a programming assignment in an introductory programming course is given in listing 1. The example includes only the first subtask of the first exercise of the assignment due to brevity, but there is already clear structure of the assignment to be seen. It consists of several exercises, which all have a number, title, description, and a possibility of including several subtask. The subtasks can also include a number/letter, title, and description. There is also a lot of metadata that is referred to in the description of the exercise, e.g. the class and file name, the package name, constructor and variables. Listing 2 shows the source code of the solution for the exercise and listing 3 shows the start code that the students receive.

Oppgave 1: Klasse for GPS datapunkter

I denne oppgaven skal der implementeres en klasse `GPSPoint.java` for å kunne representere GPS datapunkter.

Ideen er at et GPS punkt skal representeres som et objekt av `GPSPoint` -klassen og programmet skal ha et slikt objekt for hvert GPS punkt. For hvert GPS punkt får vi behov for å lagre tidspunkt, breddegrad, lengdegrad og høyde.

For denne oppgaven er spesielt Kap. 6.1 - 6.3 i Java-boken samt Undervisning 13 relevant.

a) Objektvariable og konstruktør

Se på start-koden for klassen `GPSPoint.java` i pakken `no.hvl.dat100ptc.oppgave1`. Utvid startkoden for klassen slik klassen får følgende objektvariable:

- `time` (heltall) som angir tiden i sekunder
- `latitude` (desimaltall) som angir breddegrad
- `longitude` (desimaltall) som angir lengdegrad
- `elevation` (desimaltall) som angir høyde i meter

som alle skal være `private` dvs. kun synlige innenfor klassen.

Videre skal klassen ha en *konstruktør*

```
public GPSPoint(int time, double latitude, double longitude, double elevation)
```

som kan gi verdi til alle objektvariable.

Test implementasjonen ved å kjøre testene i test-klassen `GPSPointeTester.java`

Listing 1. A programming assignment description.

```
public class GPSPoint {

        // TODO - objektvariable

        private int time;
        private double latitude;
        private double longitude;
        private double elevation;

        public GPSPoint(int time, double latitude, double longitude, double elevation) {

                // TODO - konstruktur

                // throw new UnsupportedOperationException(TODO.construtor("GPSPoint"));
                this.time = time;
                this.latitude = latitude;
                this.longitude = longitude;
                this.elevation = elevation;
        }
```

Listing 2. The solution of the exercise.

```java
public class GPSPoint {

        // TODO - objektvariable

        public GPSPoint(int time, double latitude, double longitude, double elevation) {

                // TODO - konstruktur

                throw new UnsupportedOperationException(TODO.construtor("GPSPoint"));

        }
```

Listing 3. Student start code.

Obviously there is a lot of repeated information and text, spread among different files. This information stems mostly from the source code of the solution. The files for the start code and exercise description can be generated by expanding the exercise solution with additional metainformation. An example of how the source code of the solution might look like with annotated metainformation can be seen in listing 4. The annotations would be predefined with elements (fields) that describe the structure and information about an exercise. Other metadata can be referenced in descriptions by specifically marking it as metainformation, for example by using @ signs.

```java
public class GPSPoint {
    @Task(nr = 1,
            title = "Klasse for GPS datapunkter",
            description = """
                    I denne oppgaven skal der implementeres en klasse @GPSPoint.javafor å kunne representere GPS datapunkter.

                    Ideen er at et GPS punkt skal representeres som et objekt av @GPSPoint-klassen og programmet skal ha et slikt objekt for hvert GPS punkt.\s
                    For hvert GPS punkt får vi behov for å lagre tidspunkt, breddegrad, lengdegrad og høyde.

                    For denne oppgaven er spesielt Kap. 6.1 - 6.3 i Java-boken samt Undervisning 13 relevant.""")

    @Subtask(nr = "a",
                supertask = 1,
                title = "Objektvariable og konstruktør",
                description = """
                            Se på start-koden for klassen @GPSPoint.java i pakken @no.hvl.dat100ptc.oppgave1.
                            Utvid startkoden for klassen slik klassen får følgende objektvariable:

                            @time (heltall) som angir tiden i sekunder
                            @latitude (desimaltall) som angir breddegrad
                            @longitude (desimaltall) som angir lengdegrad
                            @elevation (desimaltall) som angir høyde i meter
                            som alle skal være private dvs. kun synlige innenfor klassen.

                            Videre skal klassen ha en konstruktør
                            @constructor
                            som kan gi verdi til alle objektvariable.

                            Test implementasjonen ved å kjøre testene i test-klassen @GPSPointeTester.java""")
    private int time;
    private double latitude;
    private double longitude;
    private double elevation;

    public GPSPoint(int time, double latitude, double longitude, double elevation) {
        this.time = time;
        this.latitude = latitude;
        this.longitude = longitude;
        this.elevation = elevation;
    }
```

Listing 4. An example of a solution based on java annotations.

Results

This project is still in its infancy and is currently focused on finding background information from related literature and technologies that may be useful. The next step is to get a deeper understanding of Java's annotation and processor API before developing a small prototype. There are no actual results from the project at this point in time, since the technological solution has not been developed yet. Currently, only speculative guesses can be done about the results of this project. However, hypothesizing if a programming language agnostic solution is feasible, or how it would look like at this stage of the project seems meaningless due to the lack of knowledge and experience.

Centralizing the development of programming assignments into a single project will likely improve the workflow for creating assignments. Manually creating different files and projects can be cumbersome and opens many possibilities for inconsistencies between them. Although some extra time will be spent writing metainformation, it is more than likely that time will be saved by being able to generate different artefacts from this additional information. Inconsistency between these artefacts should be nonexistent, since they will be generated from the same source. The creator of an assignment only needs to worry about making changes in one location, which should prove to be useful when refactoring.

There is already much research into automating correcting and giving good feedback to student solutions of programming assignments, so there is no need to reinvent the wheel from scratch. This project is an opportunity to develop a tool that combines different technologies, approaches, and methods from earlier work that has shown to have a positive effect on the experience of students. There is no reason to believe an autograding system based on a combination of previous successful work shouldn't be just as good. There is also something invaluable by having solutions inspected and reviewed by a real human, allowing for personalized feedback directed at each student. No matter how successful an autograding system becomes, it is difficult seeing it completely replacing lecturers and teacher assistants when it comes to grading assignments. The goal of this project isn't to do that either, but rather be a tool to help them become more effective when grading solutions.

Conclusion

A new approach and tool for creating and correcting programming assignments in programming courses was presented and discussed in this paper. As a first step, the project is focused on developing a solution exclusively for Java. An example solution was presented that added extra metadata through annotations into the source code that would determine the structure and content of the generated output. Additional artefacts, such as exercise description and start code for students, can then be generated from the source code of the solution to the assignment. Inconsistency between artefacts, and the amount of work spent writing assignments, will likely be reduced by this new approach. In addition, an automated grading system of student solutions based on previous research will be integrated into the

project. This should act as a tool for teachers and tutors for effectively correcting and giving feedback to students.

References

[1] R. Waszkowski, “Low-code platform for automating business processes in manufacturing,” *IFAC-PapersOnLine*, vol. 52, no. 10, pp. 376–381, Jan. 2019, doi: 10.1016/j.ifacol.2019.10.060.

[2] G. Kövesdán and L. Lengyel, “Meta3: a code generator framework for domain-specific languages,” *Softw Syst Model*, vol. 18, no. 4, pp. 2421–2439, Aug. 2019, doi: 10.1007/s10270-018-0673-6.

[3] H. Takizawa, S. Hirasawa, Y. Hayashi, R. Egawa, and H. Kobayashi, “Xevolver: An XML-based code translation framework for supporting HPC application migration,” in *2014 21st International Conference on High Performance Computing (HiPC)*, Dec. 2014, pp. 1–11. doi: 10.1109/HiPC.2014.7116902.

[4] H. Takizawa, T. Yamada, S. Hirasawa, and R. Suda, “A Use Case of a Code Transformation Rule Generator for Data Layout Optimization,” in *Sustained Simulation Performance 2016*, Cham, 2016, pp. 21–30. doi: 10.1007/978-3-319-46735-1_3.

[5] A. Laukaitis, “Code Transformation Pattern Alignments and Induction for ERP Legacy Systems Migration,” in *Perspectives in Business Informatics Research*, Cham, 2015, pp. 228–240. doi: 10.1007/978-3-319-21915-8_15

[6] V. V. Savchenko *et al.*, “NOBRAINER: A Tool for Example-Based Transformation of C/C++ Code,” *Program Comput Soft*, vol. 46, no. 5, pp. 362–372, Sep. 2020, doi: 10.1134/S0361768820040052.

[7] F. Miranda, “Using Failing Test Cases to Semi-Automate Feedback for Beginner Programmers,” in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, New York, NY, USA: Association for Computing Machinery, 2021, p. 1384. [Online]. Available: https://doi.org/10.1145/3408877.3439696

[8] L. Cordova, J. Carver, N. Gershmel, and G. Walia, “A Comparison of Inquiry-Based Conceptual Feedback vs. Traditional Detailed Feedback Mechanisms in Software Testing Education: An Empirical Investigation,” in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 87–93. [Online]. Available: https://doi.org/10.1145/3408877.3432417

[9] H. H. Lee, “Effectiveness of Real-Time Feedback and Instructive Hints in Graduate CS Courses via Automated Grading System,” in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 101–107. [Online]. Available: https://doi.org/10.1145/3408877.3432463

[10] E. Baniassad, L. Zamprogno, B. Hall, and R. Holmes, “STOP THE (AUTOGRADER) INSANITY: Regression Penalties to Deter Autograder Overreliance,” in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 1062–1068. [Online]. Available: https://doi.org/10.1145/3408877.3432430