Research in Engineering Education Symposium &
Australasian Association for Engineering Education Conference
5 – 8 December, 2021 - Perth, WA

Image-to-Code: Assisting Engineering Students in Relating to OOP

Matthew Eden, Maxwell Benson, Partha Roop, and Nasser Giacaman
University of Auckland, New Zealand
Corresponding Author Email: n.giacaman@auckland.ac.nz

CONTEXT

Object-oriented programming (OOP) concerns itself with modelling real-world entities. Not only is OOP widely used in the software industry, it is compulsory for undergraduate engineering students in non-software majors such as computer systems, electrical and electronics, and mechatronics engineering. The computing education literature has shown that OOP is an important threshold concept for novice programmers, and that students often face a myriad of difficulties and misconceptions in their learning of the underlying OOP concepts. Past efforts to alleviate these challenges include the use of visualisation tools designed to capitalise on visual and kinaesthetic learning. Despite such efforts, it remains a burden for instructors to create meaningful and concrete examples to help students relate the concepts to real-world entities.

PURPOSE AND GOALS

The purpose of this study is to reduce the burden on instructors when creating OOP code examples for students. In turn, by making it easier to generate such meaningful code snippets, it is hoped that students will be able to better-relate the OOP concepts to their world. The goals of this study are twofold: (i) explore the feasibility of developing a tool that automatically generates code snippets of skeleton classes, purely from a single input image, and (ii) understand the pedagogical value that such a tool provides to students as they are being introduced to OOP concepts.

APPROACH OR METHODOLOGY/METHODS

The approach included the development of a tool (dubbed Image-to-Code) employing machine learning technologies to automatically generate code from images. This includes the ability to classify images, obtain a description of that classified object, and parse that description to extract attributes of the object for use in a code template. In order to evaluate the pedagogical value of such a tool, an online learning activity was completed by 294 students in a second-year programming course for engineering students. The study included comparisons of student agreements with Image-to-Code, impact on learner confidence regarding OOP concepts, time-to-completion, and reported student satisfaction. The analysis is both quantitative (using statistical techniques) and qualitative (using thematic analysis).

ACTUAL OUTCOMES

There are a few key takeaways from this study. The most important is that the online learning activity improved self-reported confidence in students, and their understanding of how to model key OOP aspects of real-world objects. This is evidenced by the reported student confidence before and after completing the online learning activity, as well as the dominant theme from the open-ended responses that students found the activity effective. In terms of the performance of the Image-to-Code tool, the results highlight that more work is required to improve the quality of the automatically-generated words. In particular, the generation of class names and parent class names were done well, but the quality of member fields and methods need to be improved.

CONCLUSIONS

The experiences of this preliminary work opens vast opportunities for the computing education community to build on, particularly in the development of tools to help engineering students appreciate the relevance and application of fundamental OOP concepts. The Image-to-Code tool, along with the associated online activity, were highly valued by students. To the best of our knowledge, we have not seen such an approach in the literature – and we attribute this to the novelty of the underlying machine learning technologies we are employing. We recommend expanding this study to investigate further opportunities to improve the tool's quality and its impact on learning for engineering students.

KEYWORDS

Object-oriented programming, image classification, natural language processing.

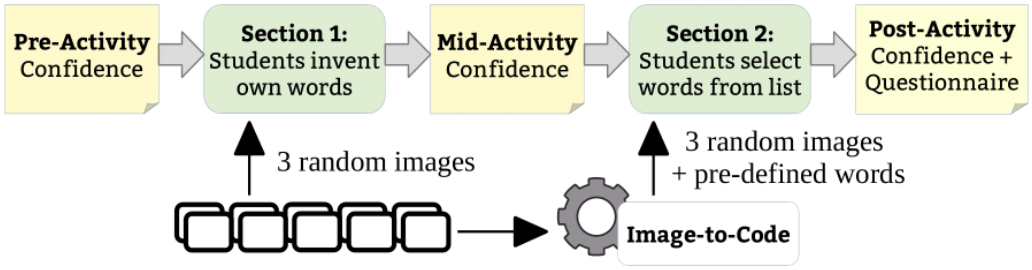


Figure 1: Design of the Image-to-Code study was comprised of two OOP activity sections, with 5-point Likert scale confidence measures in between. Each student was presented with three randomly selected images in each section.

Introduction

For the novice computing student, there are several challenging concepts they need to come to terms with in order to progress in the field (Dale, 2006). Of these concepts, there are a few well-known threshold concepts, such as Object-Oriented Programming (OOP). (Boustedt, et al., 2007) (Rountree & Rountree, 2009) (Sanders, et al., 2012) (Eckerdal, et al., 2006). To combat this, computing education instructors have employed a variety of methodologies to teach students OOP, including the creation of various tools to abstract concepts away from implementation in any one programming language (Jimenez-Diaz, Gonzalez-Calero, & Gomez-Albarran, 2012) (Yan, 2009). Such tools share a common approach: they emphasise the connected nature of objects and ground abstract concepts through the use of real-world examples. However, to create examples that relate to the real world is often difficult and time-consuming.

Related Work

Balasundaram and Ramadoss (2006) developed a tool to help students practice developing object-oriented designs from specifications, with a particular focus on collaborative learning. They found that working together helped students to perform better on this task and learn in general. This task has similarities to the process automated by *Image-to-Code* in generating class skeletons from natural language, and some of the specific challenges involved for humans are detailed. Li and Xu (2010) provide a worked example showing a process of teaching object-oriented teaching through the eight-queens puzzle, with the main takeaway being the concept of object-oriented thinking as a distinct way of viewing a problem, as opposed to simply a collection of disconnected concepts and programming syntax.

Bagert and Calloni (1997) discuss the development of an icon-based programming tool BACCII, shown to improve learning outcomes of novice programmers, which suggests that visual analogies can help students to learn OOP. Jimenez-Diaz et al. (2012) discuss their tool ViRPlay, “a 3D role play virtual environment for teaching object-oriented design”. Each student portrays a class, is given a ‘CRC card’ to represent the classes' responsibilities and dependencies, then made to act out their role in various scenarios. An evaluation of the tool was performed, showing that it improved grades, and that students and instructors both found it a useful learning/teaching tool. Among many things, the timing of teaching OOP itself presents a dilemma for instructors (Pedroni & Meyer, 2010). CS1 has the drawbacks of students not having yet mastered dependency concepts, while CS2 has the drawback of a ‘paradigm shift’ (Adams, 1996). Our work is therefore partly inspired by the arguments made by Adams, of enabling an intermediate approach that helps students identify objects and their operations early on.

```
class Bookcase: public Furniture {
  private:
    string books;
    string shelves;
  public:
    Bookcase(string books, string shelves);
    void store();
```

Figure 2: An example image with the generated code (C++ header) from the *Image-to-Code* tool

Image-to-Code Tool

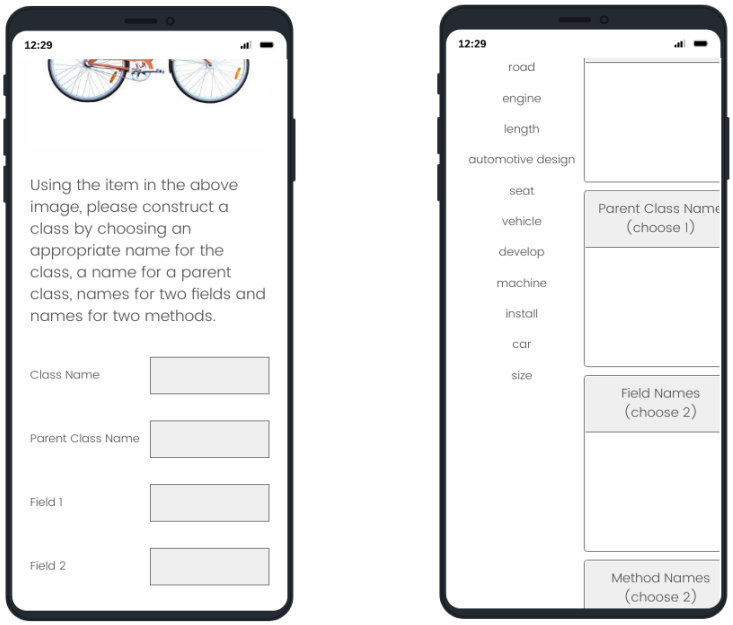
Here we introduce *Image-to*-Code, a tool designed to create simple OOP code snippets that will help students appreciate the object-oriented nature of OOP in relation to real-world objects. The tool workflow takes in an image, leverages APIs to obtain information about the image content and uses this information to construct a class skeleton in C++, Java and Python. This approach was selected because it is geared towards making it easy for instructors and students to generate compilable code, and to do so simply from images. While other approaches are possible, they would have involved more effort from instructors and therefore run counter to the intent of this research. Figure 2 illustrates the results of running this tool on a given image.

Implementation

Image-to-Code was written in Python 3, utilising the Google Vision API (Google, 2020) for the classification of input images, and a third-party Wikipedia API (Goldsmith, 2014) to scrape Wikipedia articles for content. The spaCy NLP framework (Explosion, 2020) is used to generate dependency graphs from natural language, and NLTK (Natural Language Toolkit - NLTK 3.5 documentation, 2020) is used to obtain part-of-speech statistics for a particular word in a large corpus (specifically, the Brown corpus).

Methodology

Figure 1 illustrates the evaluation design, a survey-based activity presented to students enrolled in a CS2-level course. This activity was aimed to challenge students' understanding of OOP concepts near to the time they were first introduced to them, as well as assisting in evaluation of the *Image-to-Code* tool. It was comprised of two main sections, each consisting of three randomly-selected images chosen out of a pool of ten images. Although all images had an equal chance of appearing in either section, an image would not appear more than once across both sections for any given student. In addition to these two core sections of the activity, students were also asked to rate (using a 5-point Likert scale) their level of



(b) Section 1 example (a) Section 2 example

Figure 3: Examples screenshots from Section 1 and Section 2 of the activities. The activities were web-based, and therefore accessible on either a computer or mobile device.

confidence on understanding what is a *class name*, a *parent class name*, a *field*, and a *method*. The purpose of these three confidence checks was to gauge how the activity was contributing to students' self-perceived confidence in understanding the respective OOP concepts.

Figure 3 (a) shows a screenshot example from one of the images selected to appear in Section 1 of the activity. It requires students to come up with words on their own for the given image. For each image presented in this section, students were required to provide:

- One class name,
- One parent class name,
- Two member fields,
- Two methods,
- Two 5-star ratings:
    - Level of satisfaction *with the words they selected*
    - Overall *quality of the class* representing the given image

Figure 3 (b) shows a screenshot example from Section 2. This required the students to complete the same steps of Section 1, except this time they were only allowed to select words (using drag-and-drop) from a pre-defined set of words produced by the *Image-to-Code* tool. Similar to Section 1, students were asked to rate their satisfaction of the short-listed words from the pre-defined set, and the overall quality of the class.

Table 1: Average Student Confidence

OOP Aspects	Confidence Average		
	Beginning	Middle	End
Class	4.20	4.62	4.67
Parent	4.07	4.60	4.65
Fields	3.81	4.40	4.48
Methods	3.84	4.30	4.40

Evaluation

Of the 300 students that the activity was delivered to, a total of 294 students (98%) completed the activity in full; demonstrating the simplicity of the activity that would help support such a high completion rate.

Taking into account the random allocation of images (questions) to students, there were 90 responses per image on average in each of the two sections. The number of optional feedback submissions given by students at the conclusion of the activity was 87. These responses were annotated with a tagging tool and themes identified using thematic analysis (Braun & Clarke, 2006).

NLP Vs Student Agreements

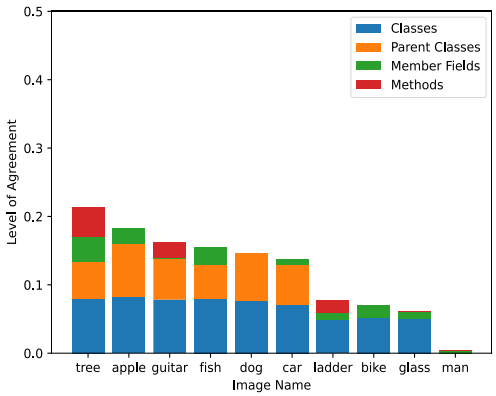


Figure 4: Average level of agreement between the Image-to-Code NLP tool and students (Section 1).

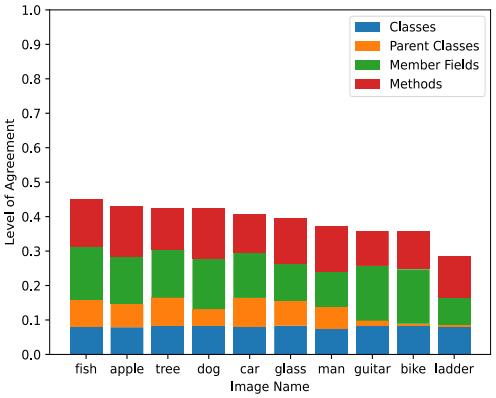


Figure 5: Average level of agreement between the Image-to-Code NLP tool and students (Section 2).

In an attempt to measure some element of quality of the *Image-to-Code* tool, a comparison is made between the word choices of students in each section of the activity, and how that compares to the word choices produced by the *Image-to-Code* tool. For Section 1, where students manually created a class description, we are interested in the agreement (or lack thereof) between the words picked by students and the words generated by the tool; the results of this comparison are represented in Figure 4. For Section 2, we were interested in seeing how students categorised the words provided to them, compared to how the tool intended them to be categorised; the results of this comparison are represented in Figure 5.

A lower level of agreement was seen in Section 1 compared to Section 2, as students were able to brainstorm their own words. With regard to Figure 4, we see that for most images *Image-to-Code* was able to at least choose a class name that students agreed with (with the exception of the ‘man’ image). For most images, *Image-to-Code* chose a parent class that

students mostly agreed with, but noticeably fewer cases where member fields and methods were chosen that matched those picked by students.

Reported Satisfaction and Quality of Words

As another measure of quality, we can compare students' satisfaction (across the two sections) with the selection of words they chose to represent the given images. The expectation is that one would be able to be ‘more satisfied’ when they are not confined to selecting from a short list, and we see this in Figure 6. Students reported an overall higher satisfaction in Section 1 ($\bar{x}$=4.30) compared to that of Section 2 ($\bar{x}$=3.67). Using a two-tailed Mann-Whitney U test, this difference is statistically significant (W=476932, p<0.0001). As a result of being less satisfied with the selection of words, this also led to students rating their overall OOP design ‘solutions’ a lower quality in Section 2 ($\bar{x}$=3.70) compared to that of Section 1 ($\bar{x}$=4.23), as demonstrated in Figure 7. Again, this difference is statistically significant (W=454565, p<0.0001).

Impact on Learner Confidence

The self-reported confidence of students was recorded at three distinct stages during the activity: at the beginning, in the middle between Sections 1 and 2, and at the end. Students were queried concerning their confidence relating to class names, parent class names, member field names and method names, and asked to rate their confidence according to a 5-point Likert scale. Averages (out of a maximum of 5) are shown in Table 1 for each of the OOP aspects. The one-tailed Wilcoxon signed-rank tests between each of these stages are shown in Table 2, showing a statistically significant increase in confidence between each stage for all categories.

Time to Completion on Images

Table 2: Comparison of Student Confidence

Stage	Category	p-value	W-value
Before Section 1 (beginning)	Class	<0.001	243.0
	Parent	<0.001	431.5
	Fields	<0.001	899.5
	Methods	<0.001	1269.0
After Section 1 (middle)	Class	<0.001	243.0
	Parent	<0.001	26430
	Fields	<0.001	799.0
	Methods	<0.001	1294.5
After Section 2 (end)	Class	0.009	183.0
	Parent	0.011	299.5
	Fields	0.002	667.0
	Methods	0.001	958.5

Figure 8 and Figure 9 show the distribution of time it took for students to select words pertaining to various OOP aspects for Section 1 and Section 2 respectively. In each chart, these are given in ascending order by median time within the respective section. This may provide us with some insight to possibly infer which images presented a bigger challenge to students. For example, Figure 8 shows that the median student needed twice the amount of

time for the ‘tree’ image (about 100 seconds) compared to the ‘dog’ image (about 50 seconds). The median time across all the images in Section 1 was 88.4 seconds.

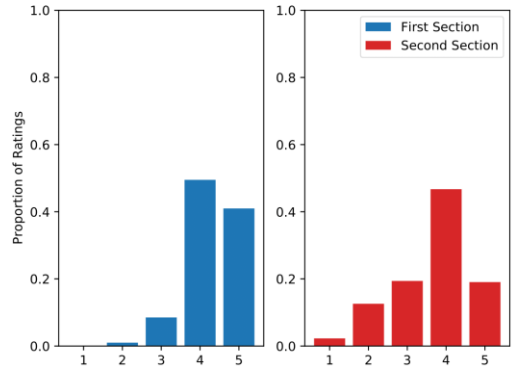


Figure 6: Students’ overall reported satisfaction with the words used in each of the respective sections.

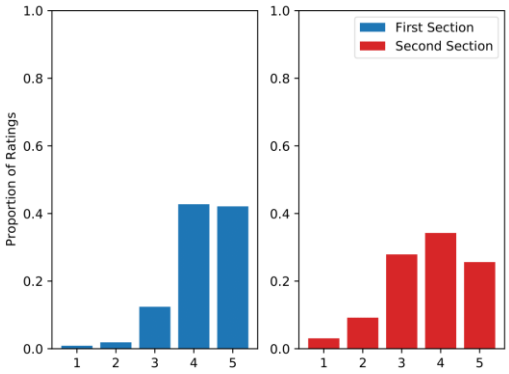


Figure 7: Students’ overall reported quality of the OOP ‘solution’ representing the images in each respective section.

Discussion

Lessons from this Experience

There are a few key takeaways from this study. The most important of which, at least in terms of what it can mean for instructors, is that the activity improved self-reported confidence in students' understanding of how to model key OOP aspects of real-world items. This is evidenced by the reported student confidence before and after completing each section of the activity, and the theme of finding the activity helpful identified in the open-ended responses. This shows that there is inherent merit to the exercise of requiring students to identify class aspects from images of real-world objects.

Even with disregard to the *Image-to-Code* tool, instructors can use the findings reported here to inspire students in CS2 courses concerned with introducing OOP. Of particular note, is that the significant increase in confidence was achieved with relatively little effort (for both instructors and students), and in itself is a worthy low-stakes assignment to consider. Considering the simplicity of this exercise, and the benefit to learners, we believe this activity will be attractive for ‘objects-first’ or ‘objects-early’ programming courses (Pedroni & Meyer, 2010).

Limitations

Although the evaluation was quite positive, there are inevitably some threats to validity. Particularly, as the activity inherently relies on the responses of students, there is the possibility that some of the data collected is not completely representative. While the timing data does show a reasonable level of dedication, there may have been students more interested in completing the activity as quickly as possible with little regard for the quality of their solution. The evaluation was conducted on students enrolled in a CS2 course for a single semester. While the number of students was reasonably large, it is difficult to infer the impact of the activity more generally. As the timing of the activity was constrained to a single point of time (when students were introduced to the basics of OOP), we may see different results if the activity was delivered at a different time in the semester. It is therefore unclear what the long-term value of this activity is. Similarly, the study did not investigate its learning impact in terms of timing, such as CS1 versus CS2, ‘objects-first’ versus ‘objects-late’, and so on.

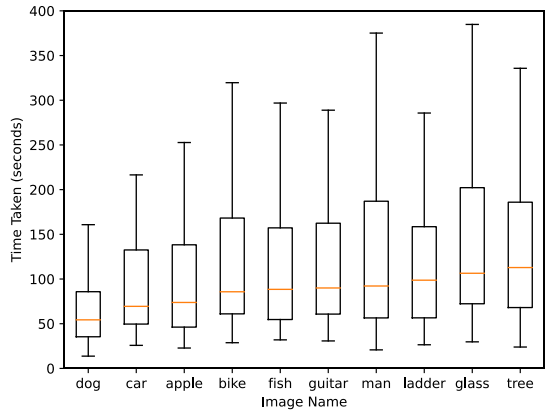


Figure 8: Time taken per image (Section 1).

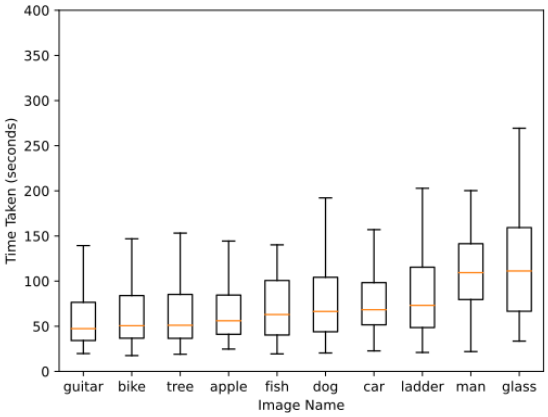


Figure 9: Time taken per image (Section 2).

The *Image-to-Code* tool itself faces some limitations which affected the quality of the output it was able to produce. The first of which being it is highly dependent on the Google Vision API proposing the initial labels for determining what the image pertains to. Although powerful, the image classification does not always label images as a human might expect, such as focusing on the clothing a person is wearing rather than the person themselves. The second of which relates to the use of Wikipedia as a knowledge source. Although the content on Wikipedia is fairly wide-ranging and comprehensive, there are several cases where the description simply lacks the key verbs or nouns that a human would associate with that object due to the academic nature of the page summaries.

Conclusions and Future Work

A tool, dubbed *Image-to-Code*, was developed as part of an attempt to address the difficulties faced by instructors in conveying OOP concepts to students. An activity was created to evaluate this tool, with discussion around the results focussing on the implication that the completion of said activity is useful for students' learning. Several much-needed improvements were identified in regard to the performance of the tool, and the quality of the output it produces. The key limitations were twofold; one being the classification of images via Google's Vision API and the other being the processing of Wikipedia's descriptions.

References

Google, 2020, *Vision AI | Derive Image Insights via ML | Cloud Vision API*

Issues Regarding Threshold Concepts in Computer Science, 2009, *Proceedings of the Eleventh Australasian Conference on Computing Education - Volume 95*139-146AUSAustralian Computer Society, Inc.

Most Difficult Topics in CS1: Results of an Online Survey of Educators, *SIGCSE Bull.38*49-53doi10.1145/1138403.1138432

Natural Language Toolkit - NLTK 3.5 documentation, 2020

Object Oriented Analysis Learning Tool using Collaborative Learning, 2006*7th International Conference on Information Technology Based Higher Education and Training,* 811-816

Object-Centered Design: A Five-Phase Introduction to Object-Oriented Programming in CS1–21996, *Proceedings of the Twenty-Seventh SIGCSE Technical Symposium on Computer Science Education,* 78–82, New York, NY, USA, Association for Computing Machinery, doi10.1145/236452.236513

Object-oriented modeling of object-oriented concepts, 2010, *International Conference on Informatics in Secondary Schools-Evolution and Perspectives,* 155–169

Putting Threshold Concepts into Context in Computer Science Education, *SIGCSE Bull.38*103-107, doi10.1145/1140123.1140154

Role-play virtual worlds for teaching object-oriented design: the ViRPlay development experience, 2012, *Software: Practice and Experience, 42*235-253, doi10.1002/spe.1071

spaCy: Industrial-Strength Natural Language Processing in Python, 2020

Teaching Object-Oriented Programming with Games, 2009, *2009 Sixth International Conference on Information Technology: New Generations,* 969-974

The teaching research on a case of object-oriented programming, 2010, *2010 5th International Conference on Computer Science Education,* 619-621

Threshold Concepts and Threshold Skills in Computing, 2012, *Proceedings of the Ninth Annual International Conference on International Computing Education Research,* 23-30New York, NY, USA, Association for Computing Machinery, doi10.1145/2361276.2361283

Threshold Concepts in Computer Science: Do They Exist and Are They Useful?, 2007, *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education,* 504-508, New York, NY, USA, Association for Computing Machinery, doi10.1145/1227310.1227482

Using an iconic design tool to teach the object-oriented paradigm, 1997, *Proceedings Frontiers in Education 1997 27th Annual Conference. Teaching and Learning in an Era of Change, 2*861 vol.2-

Using thematic analysis in psychology, *Qualitative Research in Psychology, 3*77-101, doi10.1191/1478088706qp063oa

wikpedia - PyPI, 2014