Daisy: A Block-Based Environment for Learning Data Modeling

Jessica Belicia Cahyono
jessica@prg.is.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

Youyou Cong
cong@c.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

Hidehiko Masuhara
masuhara@acm.org
Tokyo Institute of Technology
Tokyo, Japan

ABSTRACT

When solving a problem through programming, we start with data modeling, i.e., defining data structures in a programming language that represent information in the problem description. Despite its crucial role in program design, data modeling is difficult for novice programmers. The main reasons include the lack of clear instructions, the need for syntax knowledge, and unavailability of feedback when solving problems on their own.

We aim to support learners in mastering data modeling skills. To achieve this goal, we elaborate on the steps of the data modeling process and implement a block-based environment Daisy for solving data modeling exercises. We also report the results from a preliminary experiment, which shows the potential usefulness of Daisy.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Social and professional topics** → **Computing education**.

KEYWORDS

data modeling, algebraic data type, block-based environment, auto-generated feedback, programming education

ACM Reference Format:
Jessica Belicia Cahyono, Youyou Cong, and Hidehiko Masuhara. 2024. Daisy: A Block-Based Environment for Learning Data Modeling. In *Proceedings of Proceedings of the 36th Symposium on Implementation and Application of Functional Languages (IFL '24).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Problem solving through programming is better taught by taking multiple steps. These steps include thinking about the shapes of data, giving concrete examples of inputs and outputs, and writing and verifying a function [3, 7].

The conversion from information to data, which we call *data modeling*, is essential to be mastered by programmers. This is because the representation of data generally affects the planning of the whole program. In fact, Felleisen et al. [2] describe the process of designing programs in such a way that data representation drives later design steps.

IFL '24, August 26–28, 2024, Nijmigen, The Netherlands

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

However, learning the data modeling process is challenging, especially for novice programmers. For instance, in an undergraduate course taught at the authors' institution, we observed that students often have difficulty completing the process for the following reasons.

- Lack of instructions on the data modeling process
- Need for knowledge of programming language syntax
- Unavailability of feedback on the outcomes in the absence of instructors

To the best of our knowledge, there do not seem to exist clear instructions on data modeling, nor are there any tools that support data modeling. This could reduce the learner's motivation to start solving the exercise, let alone finishing it.

We aim to assist learners in acquiring data modeling skills. To achieve this goal, we make the following contributions.

- **Give explicit instructions on how to do data modeling.** In particular, we decompose the data modeling process into two steps: information mining and data representation.
- **Develop Daisy, a block-based environment for solving data modeling exercises.** To make data modeling accessible to beginners, Daisy provides block components and automatic feedback generation features.

In the rest of this paper, we discuss related work (Section 2), define the data modeling process (Section 3), and explain the features of Daisy (Section 4). We then report on the user experiment (Section 5) and discuss the results (Section 6). Lastly, we describe future work (Section 7) and conclude the paper (Section 8).

2 RELATED WORK

2.1 Design Recipe

The program design recipe [2] is a sequence of steps that guide learners through the process of developing a program in a systematic manner. It breaks down the program design process into the following six steps:

(1) Define data types and create data examples
(2) Write a signature, purpose statement, and function header
(3) Create input/output examples
(4) Develop a function template (i.e., an incomplete function definition)
(5) Complete the function definition
(6) Run the tests

Among these steps, Step 1 is equivalent to the data modeling process. The outcome of Step 1 is used extensively in later steps. For example, in Step 3, we create an input/output example for every data example created in Step 1. As another example, in Step 4, we introduce a case analysis covering all possible patterns and extract

the components of data according to the data definition from Step 1.

2.2 Block-Based Environments and Tools

Block-based programming environments are widely used in introductory computer science courses. These environments ensure syntax error freedom, allowing novice programmers to focus on the essential task of building programs [1]. Blocks are also effective: they make it possible for a learner to understand a wide variety of concepts within a short amount of time [4].

Besides coding, blocks have been used to assist program design. Closest to ours is Mio [5], a programming environment in which one uses blocks to complete tasks in the design recipe. Another study in this line of work is Rivera et al. [6], who develop a tool for planning programs using higher-order function blocks.

3 DEFINITION OF DATA MODELING PROCESS

To make data modeling easier to learn, we define data modeling as a two-step process.

(1) **Information Mining**
In this step, the learner identifies relevant information in the problem description, then extracts and refines keywords necessary for representing the information and solving the problem.
(2) **Data Representation**
In this step, the learner develops a representation of the information as data, using the keywords obtained in the previous step.

In Figure 1, we provide an example showing how we solve the Shape exercise below following the above steps.

Exercise: Shape

Define the function `area` to calculate the area of a shape. The shape is either a square or a triangle.

In the information mining step, the learner extracts the words *shape*, *square*, and *triangle* from the problem description. Then, the learner refines the set of keywords by adding new keywords *side*, *base*, and *height*, which are needed to calculate the area of the shape, and *Int*, which is the type of side, base, and height. Subsequently, in the data representation step, the learner defines data representing shapes using those keywords. In our example, shapes are represented as an algebraic data type (ADT) in Scala, but there are many other possible representations, such as structures and classes.

4 DESIGN OF DAISY

For smooth acquisition of data modeling skills, we develop ***Daisy*** *(**Da**ta modeling **i**s ea**s**y to learn)*, an exercise environment for practicing data modeling. The kind of exercise supported by Daisy is one that asks creation of an ADT that is needed to solve a given problem. To make such exercises feasible for novice programmers, we implement the following features in Daisy.

(1) **Block Components**
Daisy provides components for building ADTs as blocks. This allows the learner to focus on the essential task of data modeling without spending time struggling with syntax errors.
(2) **Automatic Feedback Generation**
Daisy generates feedback based on the ADTs created. The feedback is composed according to the correct solution prepared by the instructor and helps the learner identify their mistakes and stay motivated.

Our ultimate goal is to support both steps of data modeling in Daisy. So far, we have implemented the easier step, namely data representation, assuming that all necessary keywords are given by the instructor. In what follows, we demonstrate how data representation is done in the current implementation of Daisy.

4.1 Overview

Daisy is implemented on top of *Snap!*[1]. As shown in Figure 2, Daisy has a block palette on the left, a workspace area in the center, a feedback area at the top right, and a set of exercises at the lower right. The blocks provided in the block palette include ones for defining a data type (green), data constructors (yellow), and constructor arguments (orange), as well as keywords (purple) from the information mining step (which are currently provided by the instructor).

4.2 Demonstration

Let us guide the reader through the steps of data representation using Daisy. We use the Shape exercise from Section 3 for the demonstration.

Constructing ADT using Block Components. Based on the information given in the problem description, the learner constructs an ADT by dragging and dropping the blocks from the block palette. In our example, the exercise requires an ADT representing shapes, which are either a square or a triangle. Thus, the learner defines a data type called `Shape`, which has two constructors `Square` and `Triangle`. The exercise also requires measuring the area of a given shape, and to do that, we need the length of a square's side and the length of a triangle's base and height. Hence, the learner gives the `Square` constructor an argument `length`, the `Triangle` constructor two arguments `base` and `height`, all having type `Int`. Figure 3 shows the outcome of the Shape exercise, which is a complete definition of the `Shape` data type.

Requesting Feedback. While constructing an ADT, the learner can request feedback regarding their outcome by clicking the feedback generator button. If the outcome is correct, the learner receives a positive message as shown in Figure 4. If the outcome is incorrect, the learner sees a warning as shown in Figure 5. The feedback can be requested at any time during the construction of an ADT, even when the ADT is incomplete.

4.3 Comparison with Previous Work

The design of Daisy is inspired by Mio [5], a block-based environment for learning program design. Like Daisy, Mio allows construction of ADTs using blocks and generates feedback on constructed ADTs. However, there are two key differences. First, while blocks in Daisy are based on natural language, blocks in Mio reflect the

[1] Snap! Build Your Own Blocks. https://snap.berkeley.edu/

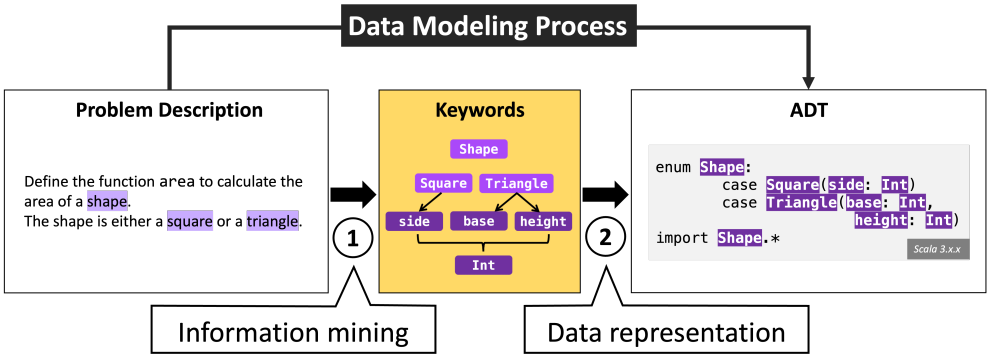

Figure 1: Decomposition of Data Modeling Process

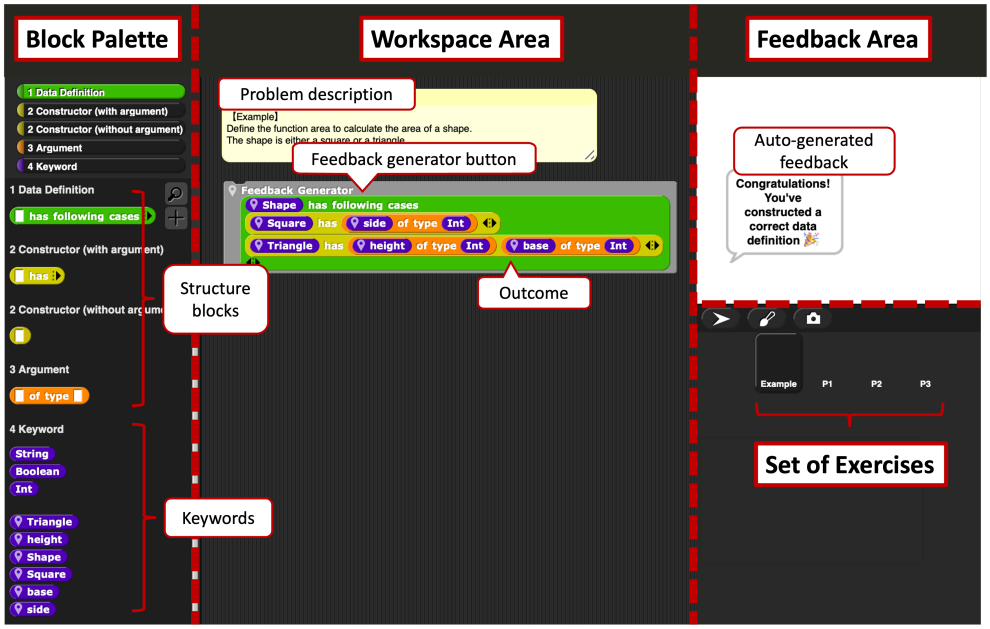

Figure 2: Overview of Daisy

Figure 3: Shape Data Type Constructed Using Daisy

Figure 4: Feedback on Correct ADT

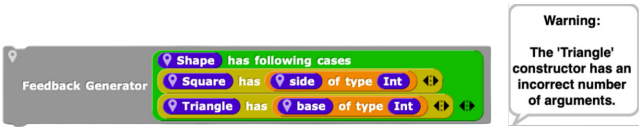

Figure 5: Feedback on Incorrect ADT

syntax of Scala. The use of natural language makes the environment more beginner-friendly and applicable to a wider range of programming languages. Second, while Daisy provides keywords as building components, Mio asks the learner to fill in the holes of blocks by text. The use of keywords makes it possible to generate problem-specific feedback, rather than general warnings such as the presence of unfilled holes.

5 EXPERIMENT

To see whether Daisy would potentially be helpful for learning data modeling, we conducted a preliminary experiment. The experiment

was done in a course called *Introduction to Computer Science*, an undergraduate course taught by the second author. The course was about basic functional programming in Scala, and it introduced the students to design recipe-based program development. Among the enrolled students, 27 of them participated in the experiment.

In the experiment, we asked the participants to solve three data modeling exercises using Daisy. Since the course was taught in Japanese, we used a version of Daisy in which blocks had Japanese text, while leaving the feedback in English to make it close to conventional error messages.

Here are the three data modeling problems we provided to the participants.

> Problem 1: Money
>
> Define the data type `Money` that represents cash used in Japan. Cash is either a banknote or a coin. A banknote has the information of its amount and the name of the person whose portrait is on it. A coin has the information of its amount, color, and whether it has a hole or not.

> Problem 2: Device
>
> Define the data type `Device` that represents devices.
> A device is either a laptop or a television.

> Problem 3: Path
>
> Define the data type `Path` that represents the file location in a file system. For example, /Home/Download/ex1.scala is a path to the ex1.scala file.
> The path is either a file or a directory. A file has information of its name and extension. A directory has the information of its name and the file or directory inside it. We assume that each directory can only have one directory or file.

In Problems 1 and 2, we asked the participants to define a non-recursive data type. For Problem 2, we did not specify the arguments of the constructors in the problem statement, but instead provided necessary argument names as keywords. In Problem 3, we asked them to define a recursive data type.

After the exercises, we asked the participants to share their thoughts about how the environment helps in learning data modeling. This was done through a questionnaire survey consisting of the following questions.

1. **Helpfulness of the Environment**
   How helpful was the environment with respect to the following aspects?
   1a. Concentrating on constructing ADTs without worrying about the syntax.
   (a. Very helpful, b. Somewhat helpful, c. Not very helpful, d. Not helpful at all)
   1b. Checking the correctness of the ADTs you constructed.
   (a. Very helpful, b. Somewhat helpful, c. Not very helpful, d. Not helpful at all)
2. **Block Components**
   2a. Did you think the process of constructing ADTs becomes easier when constructors and argument names are given by the environment?
   (a. Agree, b. Disagree)
   2b. Which mode do you think is easier to use when constructing ADTs, blocks or text?
   (a. Block, b. Text, c. Other)
   2c. Please write your thoughts about the block components feature.
3. **Automatic Feedback**
   3a. Did you get any feedback from the environment during the exercises?
   (a. Yes, b. No)
   3b. For which kind of mistake(s) did you receive feedback?
   3c. Was the feedback appropriate?
   (a. Yes, b. No, c. I did not get any feedback while solving the exercises)
   3d. Did the feedback help you correct your mistake?
   (a. Yes, b. No, c. I did not get any feedback while solving the exercises)
   3e. Please write your thoughts about the automatic feedback feature.

6 RESULTS AND DISCUSSION

Overall, we received positive reactions regarding the helpfulness of Daisy in learning data modeling. Below we provide the details of the results.

6.1 Helpfulness of the Environment

Figure 6 shows the participants' responses regarding their thoughts about the benefits of block components (Question 1a) and automatic feedback (Question 1b). More than 80% of the participants agreed that the environment helps them concentrate on the construction of ADTs and check the correctness of their outcomes.

6.2 Usefulness of the Block Components

As we can see in Figure 7, most participants agreed that having block components in the environment makes the process of constructing ADTs easier. In particular, the availability of constructor and argument names helps them figure out what to do.

Regarding the ease of use, two-thirds of the participants thought that a block-based environment is easier to use compared to a text-based one. They said that the usage of blocks makes the process of constructing ADTs more intuitive and easier to understand.

On the other hand, several participants indicated that they prefer a text-based environment. Their arguments include "It takes more time to drag-and-drop the block components" and "Drag-and-drop is troublesome; I prefer typing the code." One participant also argued that a block-based environment is not really suitable for teaching data modeling because ADTs are an advanced programming concept typically taught to students who have prior programming experience.

6.3 Automatic Feedback

Of the 27 participants, 20 of them received feedback while solving the exercises (Figure 8, Question 3a). We observed that a majority of them received feedback for Problem 3, which involves recursive data

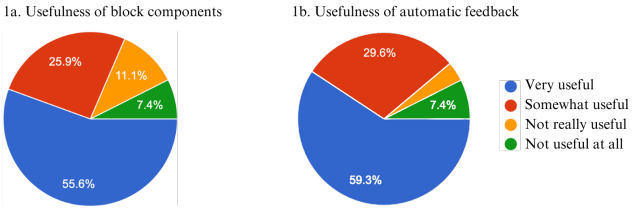

Figure 6: Responses of Questionnaire - Q1

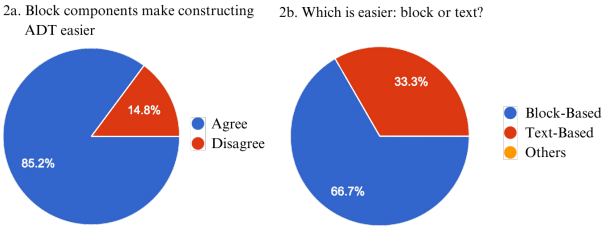

Figure 7: Responses of Questionnaire - Q2 (multiple choice)

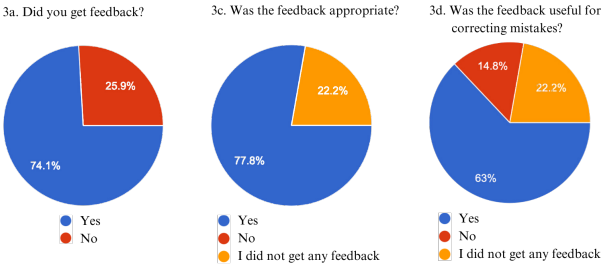

Figure 8: Responses of Questionnaire - Q3 (multiple choice)

type. Most of the generated feedback addressed incorrect argument names, argument numbers, and type mismatch (Question 3b).

Among those who received feedback, all of them agreed that the content of the feedback was appropriate, although a few of them did not feel that the feedback was helpful in correcting their mistakes (Questions 3c and 3d). This might be because the feedback was given in English, which is not their mother tongue.

As additional comments (Question 3e), we received suggestions such as keeping previous feedback messages when generating a new message and requests for more detailed messages including the exact location of errors.

6.4 Summary

The results indicate the potential usefulness of Daisy in learning data modeling. In particular, the positive reactions on Daisy's blocks and feedback imply that the overall design of the environment is on the right track.

7 FUTURE WORK

7.1 Support for Information Mining Step

An obvious next step is to support the information mining step in Daisy. We do so by letting the learner select words from the problem description and add more necessary keywords by themselves, as shown in Figure 9. Thus, the learner will be able to practice the whole process of data modeling in Daisy.

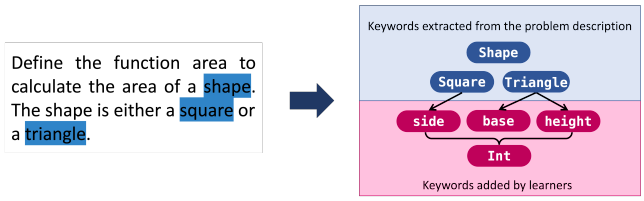

Figure 9: Implementation Idea of Information Mining Step

7.2 Quantitative Assessment of the Environment

Another thing we plan to do is to conduct quantitative assessments on the effectiveness of Daisy in learning data modeling. Specifically, we are interested in investigating how long-term use of Daisy improves learning outcomes. We are currently brainstorming possible ways of doing this assessment, and we look forward to discussing this with the IFL participants.

8 CONCLUSION

Data modeling is an important step in program design, yet the lack of instructions and tool support makes data modeling hard for novices to master. To solve this problem, we defined the steps of data modeling and implemented one of them into Daisy. We also conducted a preliminary experiment in the classroom and obtained results suggesting the potential helpfulness of Daisy. We believe that the full version of Daisy would be an effective tool for learning data modeling.

REFERENCES

[1] David Bau, Jeff Gray, Caitlin Kelleher, Josh Sheldon, and Franklyn Turbak. 2017. Learnable programming: blocks and beyond. *Commun. ACM* 60, 6 (May 2017), 72–80. https://doi.org/10.1145/3015455

[2] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. 2018. *How to Design Programs: An Introduction to Programming and Computing*. The MIT Press.

[3] Rajib Mall. 2014. *Fundamentals of Software Engineering, Fourth Edition*. PHI Learning Private Limited, Delhi.

[4] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. *ACM Trans. Comput. Educ.* 10, 4, Article 16 (nov 2010), 15 pages. https://doi.org/10.1145/1868358.1868363

[5] Junya Nose, Youyou Cong, and Hidehiko Masuhara. 2022. Mio: A Block-Based Environment for Program Design. In *Proceedings of the 2022 ACM SIGPLAN International Symposium on SPLASH-E* (Auckland, New Zealand) *(SPLASH-E 2022)*. Association for Computing Machinery, New York, NY, USA, 62–69. https://doi.org/10.1145/3563767.3568127

[6] Elijah Rivera, Shriram Krishnamurthi, and Robert Goldstone. 2022. Plan Composition Using Higher-Order Functions. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1* (Lugano and Virtual Event, Switzerland) *(ICER '22)*. Association for Computing Machinery, New York, NY, USA, 84–104. https://doi.org/10.1145/3501385.3543965

[7] Lorrie Willey, Barbara White, and Cynthia Deale. 2023. Teaching AI in the college course: Introducing the AI Prompt Development Life Cycle (PDLC). 24 (10 2023), 123–138. https://doi.org/10.48009/2_iis_2023_111

A ADDITIONAL INFORMATION ABOUT EXPERIMENT EXERCISES

In this section, we provide some additional information about the exercises given in the preliminary experiment. Specifically, we present the keywords provided for each exercise, as well as the expected answers.

Problem 1: Money

Define the data type `Money` that represents cash used in Japan. Cash is either a banknote or a coin. A banknote has the information of its amount and the name of the person whose portrait is on it. A coin has the information of its amount, color, and whether it has a hole or not.

Keywords: String, Boolean, Int, Coin, portrait, hasHole, Money, color, Bill, value

Answer:

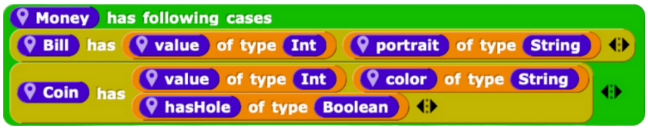

Figure 10: Expected Answer for Problem 1

Problem 2: Device

Define the data type `Device` that represents devices.
A device is either a laptop or a television.

Keywords: String, Boolean, Int, brand, Device, size, Laptop, memory, Television

Answer:

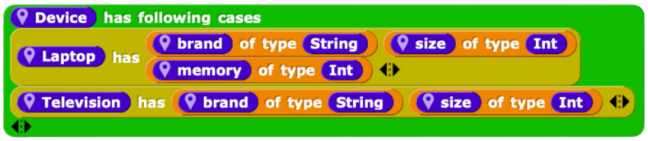

Figure 11: Expected Answer for Problem 2

Problem 3: Path

Define the data type `Path` that represents the file location in a file system. For example, /Home/Download/ex1.scala is a path to the ex1.scala file.
The path is either a file or a directory. A file has information of its name and extension. A directory has the information of its name and the file or directory inside it. We assume that each directory can only have one directory or file.

Keywords: String, Boolean, Int, name, Directory, content, File, Path, extension

Answer:

Figure 12: Expected Answer for Problem 3