Fall 2023 SSW-345 Homework

by

Benjamin Knobloch

Stevens.edu

October 13, 2023

Fall 2023 SSW-345 Homework

Benjamin Knobloch
Stevens.edu

This document provides the requirements and design details of Homework. The following table (Table 1) should be updated by authors whenever changes are made to the architecture design or new components are added.

Table 1: Document Update History

Date	**Updates**
09/07/2023	BCK: • Created document from template. • Added the team chapter (Chapter 1).
09/12/2023	BCK: • Created new chapter for initial homework assignment (Chapter 2).
09/17/2023	BCK: • Created new chapter for second homework assignment (Chapter 3). • Completed the four sections of the second homework assignment.
10/05/2023	BCK: • Created new chapter for third homework assignment (Chapter 4). • Completed the third homework assignment.
10/12/2023	BCK: • Created new chapter for fourth homework assignment (Chapter 5). • Completed the fourth homework assignment.

Table of Contents

List of Tables

List of Figures

Chapter 1

Team

– Benjamin Knobloch

1.1 About Ben

Ben Knobloch is a student of Software Engineering in his Junior year at Stevens Institute of Technology. He is a student tutor on campus, as well as the News Editor for the campus paper and secretary for the Stevens Honor Board and Engineers Without Borders SIT. In his free time, he enjoys reading and playing board games with his friends.

Chapter 2

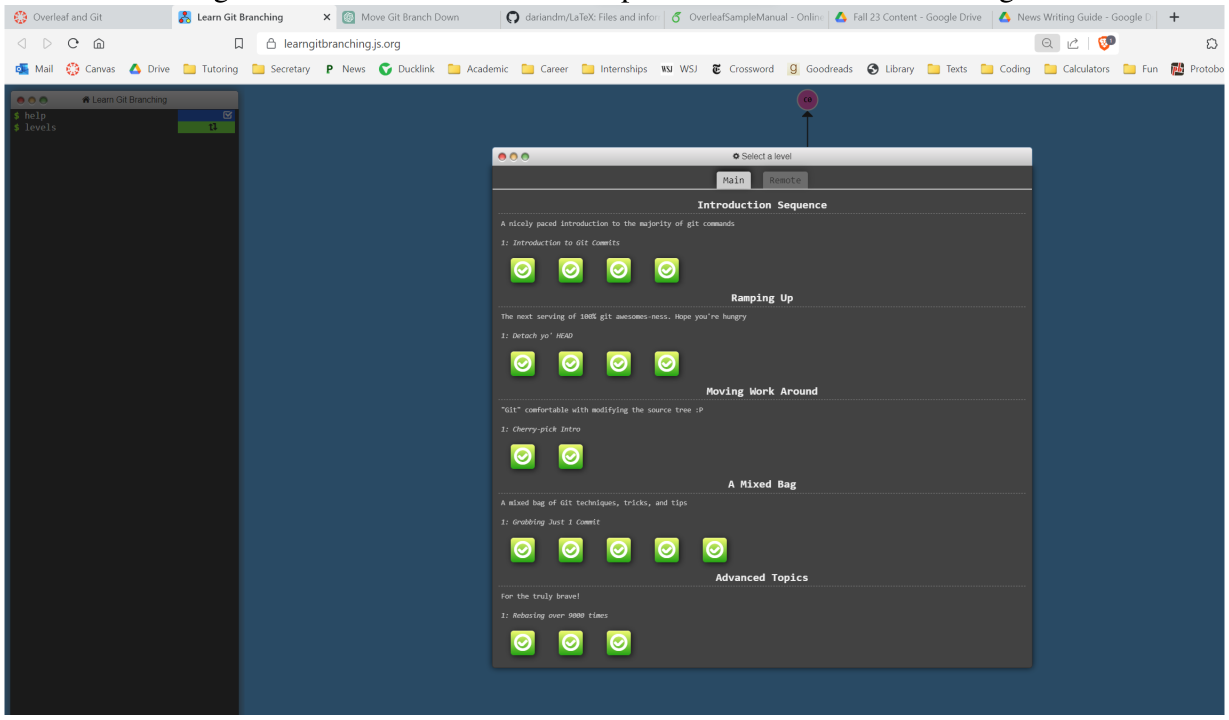
Learning Git

2.1 Assignment Description

I completed an interactive online course communicating the basics of using Git on the command line. Then, I set up my GitHub repository ("repo") for this course and created an issue that acts as my initial to-do list.

2.2 Completed Course Screenshot

Figure 2.1: Screenshot of the completed set of basic Git challenges



```
$ help
$ levels
```

Introduction Sequence

A nicely paced introduction to the majority of git commands

1: Introduction to Git Commits

Ramping Up

The next serving of 100% git awesomes-ness. Hope you're hungry

1: Detach yo' HEAD

Moving Work Around

"Git" comfortable with modifying the source tree :P

1: Cherry-pick Intro

A Mixed Bag

A mixed bag of Git techniques, tricks, and tips

1: Grabbing Just 1 Commit

Advanced Topics

For the truly brave!

1: Rebasing over 9000 times

2.3 Creating a GitHub Issue

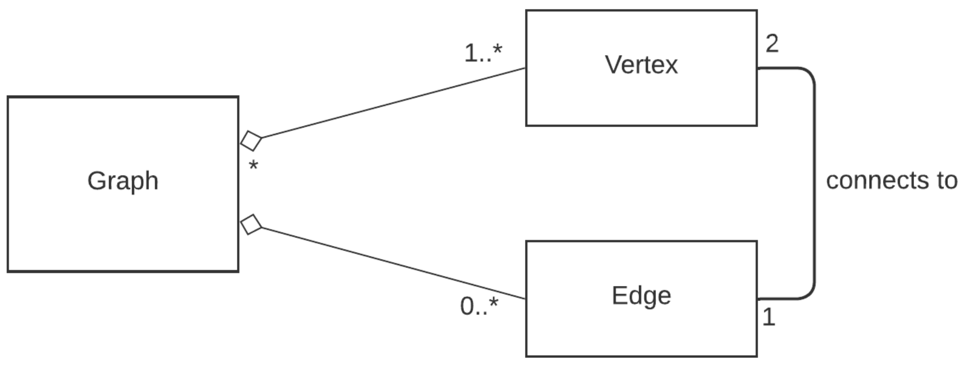
Here is the link to my repo, which contains both a copy of this document as a PDF and the screenshot of the Learning Git page.

Chapter 3

UML Class Modeling

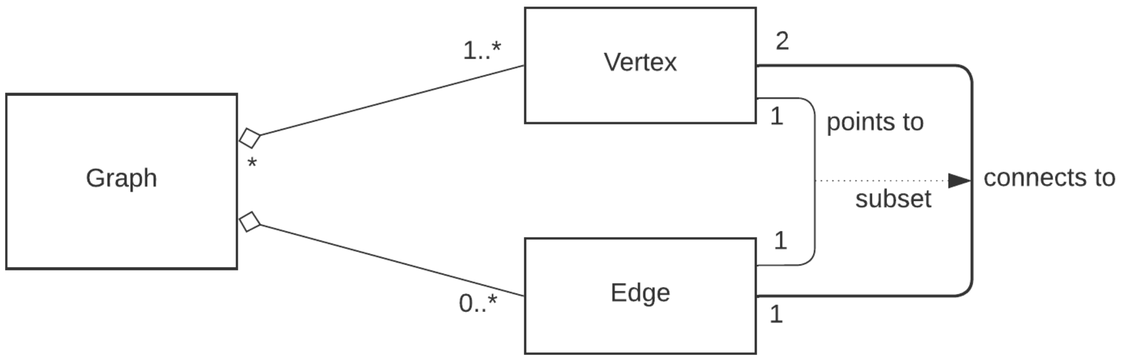
3.1 Exercise 1

Figure 3.1: Undirected Graph - UML Class Diagram



3.2 Exercise 2

Figure 3.2: Directed Graph - UML Class Diagram



3.3 Exercise 3

A window can come in the form of a scrolling window, a canvas, or a panel. Scrolling windows can also be text windows or scrolling canvases, the latter of which are also canvases. A single canvas can be associated with any number of shape elements. Shapes can be a line or a closed shape, and closed shapes can be either a polygon or an ellipse. A polygon contains any number of ordered points as vertices. A panel is associated with zero or one panel item, any number of which can be tied to an event as a notify event. Panel items can come in the form of buttons, choice items, or text items (any number of which can be tied to an event as a keyboard event). A choice item can be associated with any number of choice entries, and a subset of this relationship is that zero or one choice items can be associated with the current choice.

3.4 Exercise 4

Any number of customers can be the account holder for any number of mailing addresses. Each mailing address can be tied to any number of credit card accounts. Meanwhile, an institution has an account number that is associated with a credit card account, if it exists. Each credit card account, qualified by a statement date, is tied to a statement if it exists, which is tied by a transaction number to a transaction if it exists. Transactions can come in the form of cash advances, interest, purchases, fees, or adjustments. Any number of purchases is associated with a merchant.

Chapter 4

Advanced Class Modeling

4.1 Premise

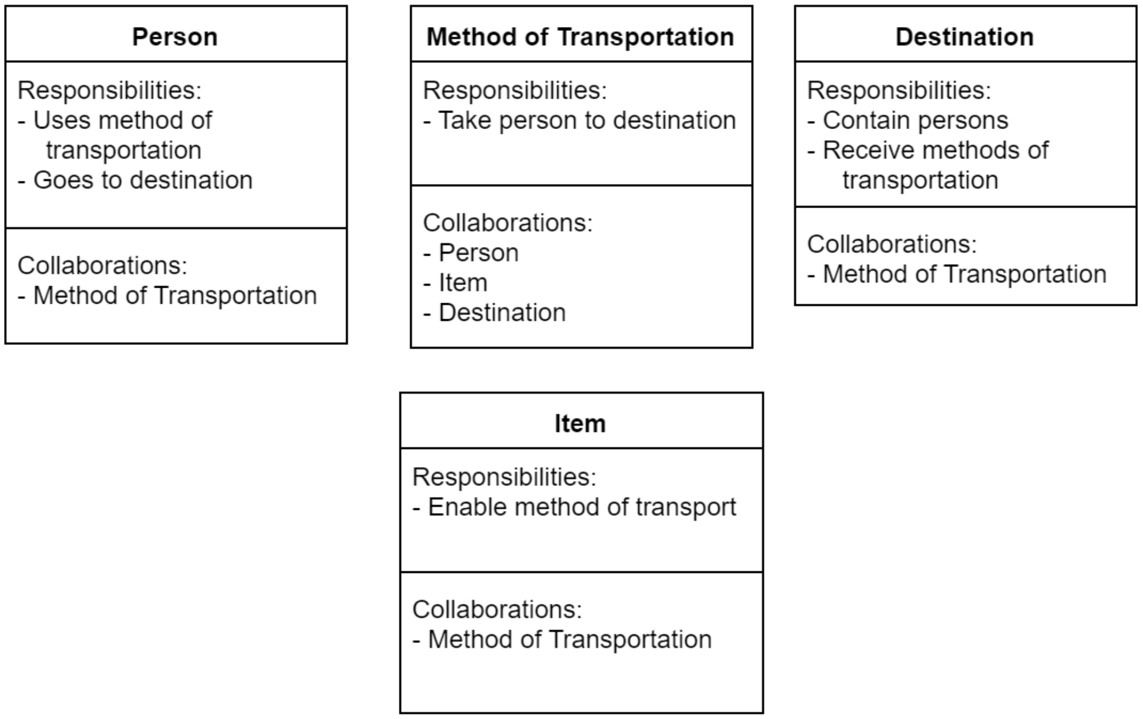
Given the following story:

> John and Maria are two students at Stevens Institute of Technology, a local university. John gets to his classes by riding his bike and Maria gets to her class by putting on her shoes and walking to class.

I created a CRC Cards model, Use-Case Diagram, Object Diagram, and Class Diagram representing the described situation.

4.2 CRC Cards

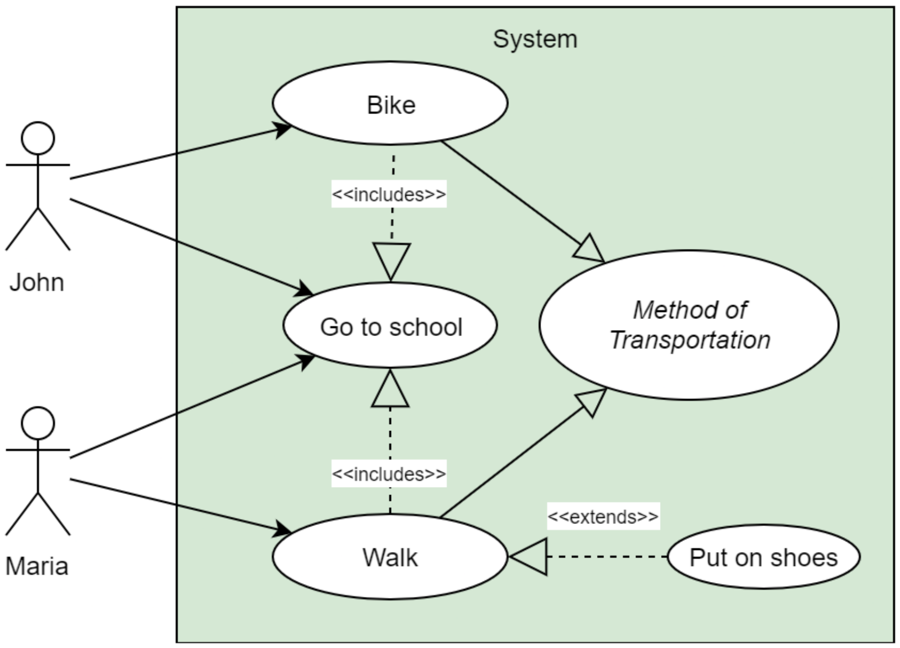
Figure 4.1: CRC cards for the John-Maria story



First, I created a CRC card for each of the principal kinds of nouns in the story: The people, their methods of transportation, any peripheral items they used, and their destination. I then determined responsibility by thinking about the role each of the nouns plays in the story, and derived collaborations from those responsibilities appropriately.

4.3 Use-Case Diagram

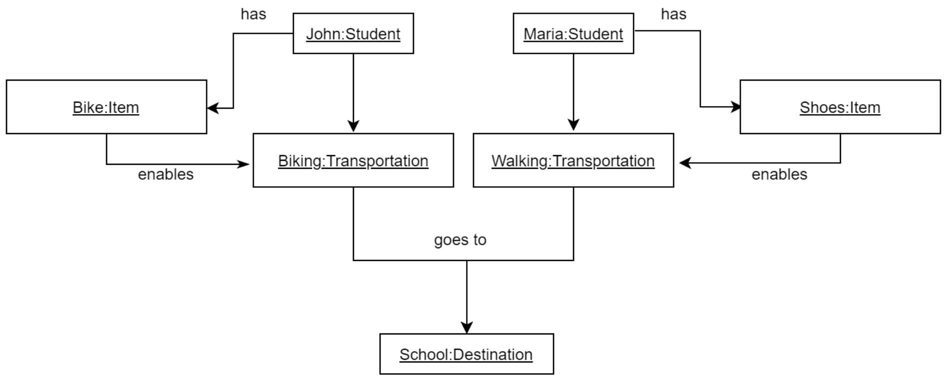
Figure 4.2: Use-Case Diagram for the John-Maria story



I began drawing this diagram from the actor side, considering what John and Maria are described to have done in this story. They bike or walk, respectively, in order to go to school. I then added additional use cases and relationships that expound on these primary use cases. For instance, ”go to school” including both biking and walking, each of which are children of generic methods of transportation. Walking, additionally, extends the action of putting on shoes.

4.4 Object Diagram

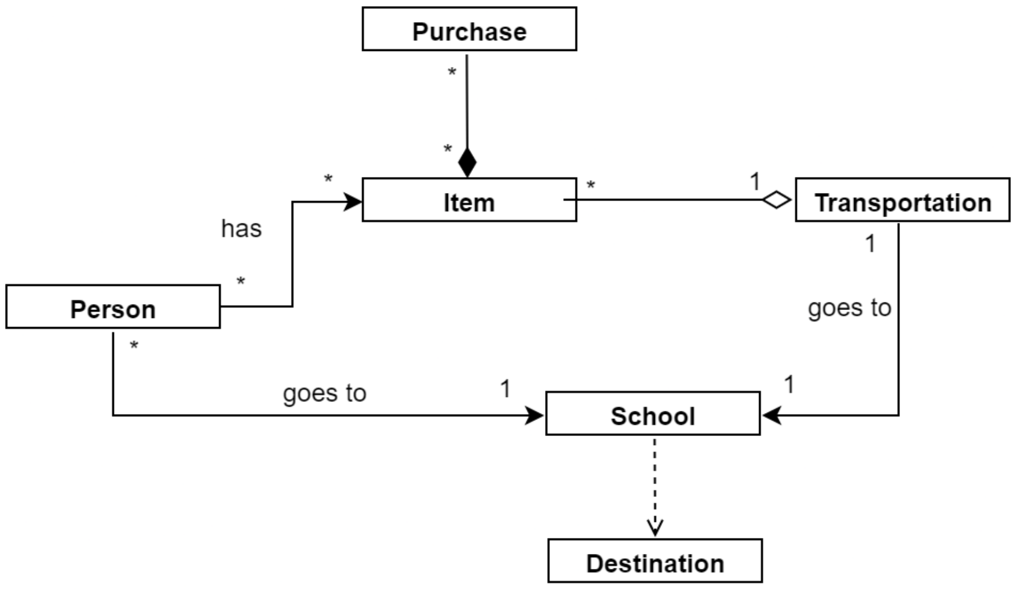
Figure 4.3: Object Diagram for the John-Maria story



This object diagram shows the relationship between each of the main instances of the classes, and so I derived it mainly from the CRC cards for the story. Each student has their respective item which enables their method of transportation, which links to the destination.

4.5 Class Diagram

Figure 4.4: Class Diagram for the John-Maria story



Lastly, I generalized the object diagram to compose the class diagram. Any number of persons can have any number of items, which are composed of purchases. These constitute a single form of transportation, which ”goes to” a single school, which is a child of a destination.

Chapter 5

Software Execution Model

5.1 Component Process Calculations

Given a provided software execution model, the best, worse, and average demand must be calculated.

P_0 is an extended node consisting of six processing steps.

- The process makes 775 calls to the CPU in total. A call results in 0.0001 seconds of service time. $775 \times 0.0001 = 0.0775$ seconds of total CPU time for P_0.
- The process makes 14 calls to disk in total. Each disk call takes 0.02 seconds. $14 \times 0.02 = 0.28$ seconds of total disk time for P_0.
- Lastly, the process makes 244 calls to the network in total. Each network call takes 0.01 seconds. $224 \times 0.01 = 2.44$ seconds of total network time for P_0.

In total, P_0 will result in $0.0775 + 0.28 + 2.44 = 2.7975$ seconds of demand time.

For all other processes,

- The ”Work” operation results in 400 calls to the CPU, so each Work operation takes $400 \times 3 \times 0.0001 = 0.12$ seconds.
- The ”DB” operation results in 1500 calls to the CPU, 150 calls to disk, and 2 calls to the network, so each DB operation takes $1500 \times 3 \times 0.0001 + 150 \times 2 \times 0.02 + 2 \times 0.01 = 0.45 + 6 + 0.02 = 6.47$ seconds.
- Lastly, the ”Msg” operation results in 10 calls to the CPU and 1 call to the network, so each Msg operation takes $10 \times 3 \times 0.0001 + 1 \times 0.01 = 0.003 + 0.01 = 0.013$ seconds.

- P_1 executes Work four times and Msg one time. Its demand time is $4 \times 0.12 + 1 \times 0.013 = 0.493$ seconds.

- P_3 executes Msg one time. Its demand time is 0.013 seconds.
- P_5 executes Work one time, DB two times, and Msg three times. Its demand time is $1 \times 0.12 + 2 \times 6.47 + 3 \times 0.013 = 13.099$ seconds.
- P_6 is identical to P_3. Its demand time is 0.013 seconds.
- P_7 executes DB one time. Its demand time is 6.47 seconds.
- P_{10} executes work one time, DB one time, and Msg one time. Its demand time is $10 \times 0.12 + 1 \times 6.47 + 1 \times 0.013 = 6.603$ seconds.
- P_{11} is identical to P_3. Its demand time is 0.013 seconds.
- P_{12} executes work 3 times, DB 2 times, and Msg 4 times. Its demand time is $3 \times 0.12 + 2 \times 6.47 + 4 \times 0.013 = 13.352$ seconds.
- P_{13} executes work 8 times, DB 3 times, and Msg 1 time. Its demand time is $8 \times 0.12 + 3 \times 6.47 + 1 \times 0.013 = 20.383$ seconds.

Partway through the execution model, there is a single case node that divides the execution into four branches. To calculate the best, average, and worst case demand times, the time of each branch must be calculated and compared to the others.

- The first branch proceeds to P_5 and then a pardo node. For a pardo node, the longest-running node is relevant since all derivative threads must complete before the execution of the program can continue. In this case, P_7 runs longer than P_6, so this branch's demand time is $13.099 + 6.47 = 19.569$ seconds.
- The second branch just executes P_0. This branch takes 2.7975 seconds to execute.
- The third branch proceeds to P_1 and then a split node. For a split node, not all threads need to complete, so the shortest-running node is relevant. Here, P_{11} is the fastest node, and so this branch's demand time is $0.493 + 0.013 = 0.506$ seconds.
- The fourth branch just executes P_{13}. This branch takes 20.383 seconds to execute.

5.2 Overall Calculations

With those preliminary calculations, we can now find the best, worst, and average case demand times for the described program.

- The program at its beginning is deterministic, with a demand time of $2 \times 2.7975 + 0.493 = 6.088$ seconds. We then enter the looping segment of the program. For the best case, which will always enter the third branch, each loop has $0.013 + 0.506 + 2.7975 = 3.3165$ seconds of demand time, for a total of $4 \times 3.3165 = 13.266$ seconds of time in this loop. **Thus, the best case demand time for the execution path is 19.354 seconds.**

- In the worst case, the program will always enter the fourth branch within the loop. This means that each loop will have $0.013 + 20.383 + 2.7975 = 23.1935$ seconds of demand time for a total of $4 \times 23.1935 = 92.774$ of demand time looping. **Thus, the worst case demand time for the execution path is 98.862 seconds.**

- On average, each loop will have a demand time of $0.013+2.7975+((0.1)19.569+(0.2)2.7975+(0.3)0.506 + (0.4)20.383) = 13.6319$ seconds. For four loops, that makes a total of $4 \times 13.6319 = 54.5276$ seconds. **This means that the average demand time for the entire execution path is 60.6156 seconds.**

Bibliography

Index