

Killer Lab: Flow Simulation and Lead Poisoning Study

James Heliotis, Computer Science

Carl Lutzer, Mathematics

Rochester Institute of Technology

The Rochester Institute of Technology offers several computing-related curricula that begin with a 4-quarter sequence of courses in computer science. Through these introductory courses we help students learn object-oriented programming (in two languages), basic software engineering skills, abstraction, and important computer science concepts mainly revolving around data structures.

During the 2004-05 school year some faculty had the opportunity to experience teaching students who stayed together as a group through their math, English, and computing courses for the entire first year. This facilitated coordination and joint planning by the three involved faculty members.

In this document we present a joint project assigned to our students in both Calculus and Computer Science in the spring of their first year. The problem was to observe the movement of lead through various organs in the human body utilizing observed rates of transfer between those organs.

Calculus Project

In the spring term calculus Professor Lutzer found an assignment from a textbook on applications of calculus [1]. The assignment asked students to investigate whether it was possible that people could recover from prolonged exposure to toxic elements such as lead. If the body's organs absorbed the toxin more quickly than they released it and than the body excreted it, the answer would be negative.

The assignment went on to explain that the rate of transfer of lead between blood, bone, and tissue is at a rate proportional to the amount present. The students were expected to write differential equations and eventually determine that they needed to use Euler's Method (basically discretizing the process) to get an answer.

Professor Heliotis then adapted the assignment to a class exercise in graph implementation and an out-of-class project that would answer the same question using a graph data structure and discrete-time simulation.

The Problem Domain

For the computer science project the students were told that the problem could be modeled as an instance of a system where *material* flows into and out of *reservoirs*

through *conduits*. In order to determine the amount of material in a reservoir at any given point in time, an algorithm must be used that asks each reservoir to calculate its new amount based on the previous state of the system. Once all reservoirs know their new amounts, they “expose” them as their official amounts. This two-phase approach is typical of discrete-time simulations. In addition, it allows the update algorithm to be written abstractly as a template method and therefore to apply to many different problems. These ideas will be discussed in the Design section.

The Analysis Pattern

Patterns exist at all stages of software development. Martin Fowler [2] was one of the earliest to notice that patterns can even be discovered when discussing requirements in the problem space. The problem in this lab fit a pattern seen quite often in systems that involve flow of material, as described above. Figure 1 shows a sample instance of this “Reservoir-and-Conduit” pattern.

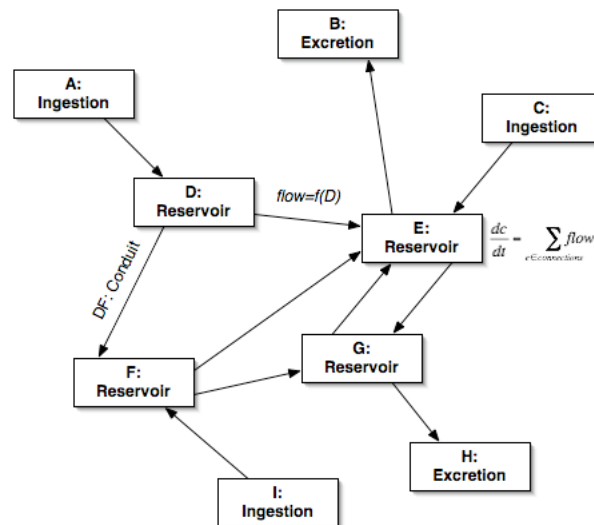


Figure 1: The Reservoir-and-Conduit Analysis Pattern

Design

This type of problem can be described using a network of nodes in a graph. Each node represents a reservoir of material, while directed edges denote the presence of a transfer connection (conduit) between two reservoirs. The “state” of each reservoir could be considered to be the amount of material it currently holds, while each conduit’s state would actually be the current transfer rate between the conduit’s source and destination reservoirs.

In a computer simulation of such a process, a discrete-time simulation is desirable. At each unit of time the amount of material in the destination reservoir is augmented by the conduit’s weight multiplied by the amount in the source reservoir, and the amount in the source reservoir is decremented by the same amount.

The two-phase discrete-time algorithmic pattern works as follows:

```
For each unit of time do
  For each component do
    Compute the new value of the component based on neighbors' values.
    Store the new value in a temporary location.
  For each component do
    Make the value stored in the temporary location the component's true value.
```

As we shall see, this algorithm was used to construct an instance of the template method pattern.

Because the real-world problem included ingesting and excreting of the toxin into and out of the body, three variations of network nodes were suggested:

- Ingestion
- Reservoir
- Excretion

Reservoir nodes behave as described above. Ingestion nodes provide a constant supply of the material, whereas excretion nodes absorb material the same way as reservoir nodes, but don't hold onto it. Clearly there are many opportunities for polymorphism in the way a node's (reservoir's) new amount is computed. Figure 2 shows a sample instance of the problem.

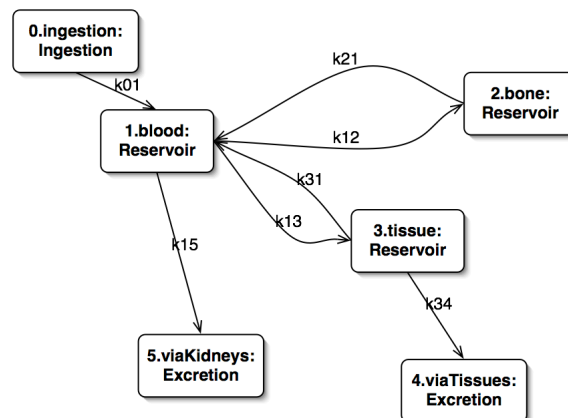


Figure 2: Modeling the Lead Absorption Problem

The body of the simulation main loop becomes an application of the template method pattern. It is a traversal of the graph's edges:

```
For each edge in the graph do
  Compute the contribution to the flow on this graph from the source node.
  Subtract the flow from the source node.
  Add the flow contribution to this edge.
For each edge in the graph do
  Add the edge's flow to the destination node.
```

Note that the temporary storage stated in the earlier algorithm is now handled by the *flow* in the edge. The statements marked in italics represent polymorphic calls that depend on the exact type of node being addressed: ingestion, reservoir, or excretion.

What It Teaches

In the computer science class this lab project has many teaching points going for it. First, graph theory is reinforced through its practical application. One especially strong point is choosing the proper traversal algorithm for cycling through the elements of the graph in the simulation. Second, from an object-oriented standpoint, the students learn to design the graph class according to their needs: what operations are needed in its interface, and which operations must be efficient. (Graph classes are notorious for not being one-size-fits-all.) Third, they must design the template method for the two-phase discrete time simulation algorithm. Although none of this is unusual for a spring first-year project, the fact that it is related to a current real-world problem can help as a motivating factor, especially for those students who are not captivated by abstract puzzles or computer-self-serving projects like computer simulators and language interpreters.

Advantages of a Collaborative Project

Although there were many “neat” aspects to this assignment from a computer science point of view, the most interesting aspect was probably the fact that it involved collaboration between faculty members from different departments and that the project was assigned to the students twice in different ways. One obvious advantage of doing this is that the students get more exposure to the problem, which results in better comprehension and more in-depth discussions. In our particular case they also learned about the connection between mathematics and computer science, and the differences in the approaches to solve the same problem. They can even compare answers they get doing the problem two different ways!

Conclusions

Providing the students with a project in an application area and bringing them all the way through to fundamental principles in mathematics and computer science is highly desirable. Were we to repeat the idea, we might assign additional problems that map to similar structures and algorithms so that the students could see the value of the abstract designs, techniques, and formulae they came up with for the first assignment. Some possibilities might be pumping water out of flooding lowlands, digital circuit simulation, or a neural network problem.

Bibliography

- [1] Borrelli, Robert, Coleman, Courtney, Differential Equations: A Modeling Perspective 2nd edition, Wiley, 2004
- [2] Fowler, Martin, Analysis Patterns – Reusable Object Models, Addison-Wesley, 1997

The writeup for the project currently exists at the following URL:

<http://www.cs.rit.edu/~vcss233/Projects/newproj02/writeup.html>

