Our Experience Replacing the Brooks2 Filter Using Various Abstractions in Java

Danny Mickens
North Carolina State University
dsmicken@ncsu.edu

Ross Eby
North Carolina State University
rneby@ncsu.edu

Michael McMillan
North Carolina State University
mmcmill@ncsu.edu

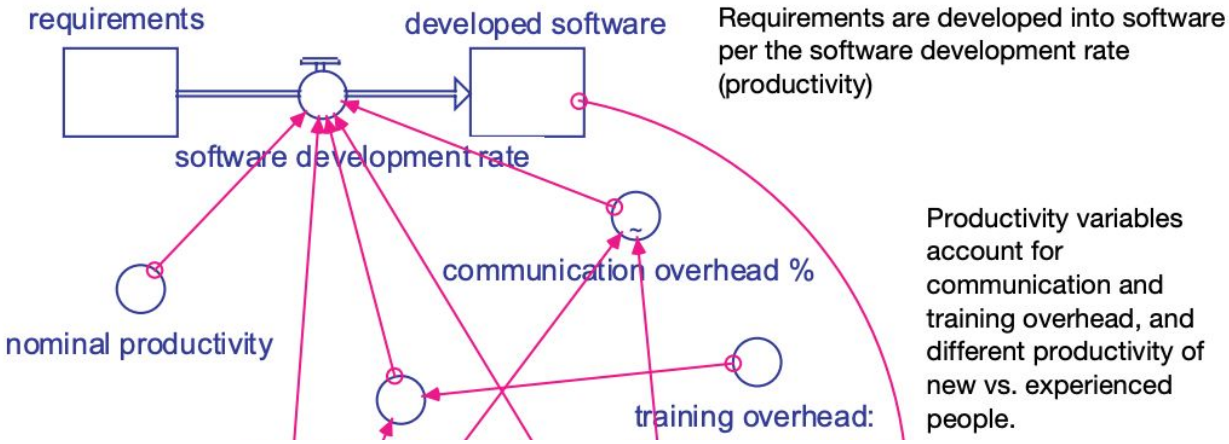

Figure 1: Part of a visual representation of a compartmental model for Brook's Law.
Source: https://github.com/txt/plm19/blob/master/doc/tuts/brooks.md

1. Introduction

In project 2A for Theory of Programming Languages we are exploring the use of various programming abstractions while replacing one filter of a pipeline. The filter our group has chosen to replace is the Brooks2 Filter. The Brooks2 filter is a model for Brook's Law which states that adding programmers to a late project may in fact make it later. Brook's law is explained by the additional training and communication overhead that gets added when new programmer's are being integrated into the existing team. Brooks Law is easily modeled by a compartmental model connecting a software development flow chain with a personnel flow chain. Our group attempted to recreate a python implemented compartmental model of Brook's Law using Java and additional abstractions.

2. Abstractions Overview

Prior to implementation, we initially made a list of ten different abstractions that could prove useful in our implementation. These abstractions include: Inheritance, Delegation, Polymorphism, Pipeline (Pipe and Filter), State Machines, Compartmental Models, Go Forth, Tantrum, Quarantine, and Map Reduction. These abstractions, or patterns, are simply different strategies to go about implementing a design that serve different purposes and provide different strengths and weaknesses. The details of these abstractions relevant to our implementation will be described in the subsections following.

2.1 Inheritance

Inheritance is a very common object-oriented pattern with a variety of uses. It creates an easily understood hierarchy structure that can be clearly

visually understood and designed through a UML class diagram. The main principle behind inheritance is that classes can be structured as super classes and subclasses where subclasses acquire the properties of their super classes.

2.2 Delegation

Delegation is another object-oriented specific design pattern which achieves similar goals to that of inheritance through a different implementation. Delegation works by having an