Identifying Data Flows Across Multiple Boundaries and Programming Languages

Dave Kell (dave.kell@unb.ca) | Andrew McAllister (andrewm@unb.ca) | Steve O'Hara (steve@eaglegacy.com)

UNB EST. 1785 UNIVERSITY OF NEW BRUNSWICK
Computer Science
Fredericton

The Importance of Data Flow

Communication is what this world is all about. Productive and understandable communication builds empires, while poor communication breaks down relationships. Software applications are no different. The scale of software holdings for most successful companies today are massive and incredibly complex. They traverse boundaries from manual to automated, legacy technology to advanced technology, one programming language to another (to name a few). These components are woven together over time like an intricate tapestry, which eventually become the application. Many software developers will have a hand in creation and maintenance of applications. New ideas and components mean modifying or modernizing other components. Business processes or technology changes can impact small or huge changes. The complexity with which data flows through such an application can be incomprehensible and formidable for an individual or even a team to shoulder and understand. Now imagine having to make pivotal business-defining decisions based on that understanding and information. How confident would you be?

Understanding Data Flows

Understanding this complex flow of data requires either the original developer, time looking at source code, compiled code, log files, and/or running specific localized tests. A new administrator of the site would have to spend significant amounts of time and resources before being able to make any modifications or updates. Now imagine this is a multi-billion-dollar company and the complexity is hundreds or thousands of times what is involved with this example. A more efficient method of extracting data flow information from a complex application can provide immense benefits to the company and enhance the value of the software.

The Challenges with Data Flows

There are some challenges with identifying data-flows that make it a complex and difficult problem.

To start, what is a data flow?

- Is it a method call within a program?
- Is it input/output of a program?
- Is the flow of data in a control-flow graph?
- Is it the flow of data across a network?
- Is it how data flows at runtime?
- Is it the capability of a application to flow data or the instances of actual flow?
- Could it be any of the above?

Assuming the concept of a data flow has been precisely defined, what information represents the flow?

- Is it a single line of code?
- Is it a single piece of data or multiple pieces?
- Is it an entire method or class?
- Is it a data structure?
- Would the answer depend on the definition?

Equally as important, how can one determine when the data flows occur during the execution of the application? Is it possible capture all the instances of it, even manual ones?

- When are data stores modified or updated manually?
- When do batch jobs run and how often?
- When are files manually created, edited, copied?
- When are databases modified manually?
- Has the data flow been triggered manually or automatically?
- People talking to applications?
- Applications talking to other applications?

Assuming all of the above is possible:

- How would the source of a data flow be represented?
- How would the destination of a data flow be represented?
- How would the semantics (such as validated data, or timing) of a data flow be represented?

A Simple Example of Complexity

The application described in "Figure 1 – Photography Website", is a personal website developed and designed by one administrator. While the website is aimed at family photos, videos, and information, it is heavily used by many visitors. Since its creation, the photos have been viewed over 6 million times. Part of the website described in the data-flow diagram, Fig.1, revolves around the photos and the people (faces) contained within each photo. A good example of the complex nature of data-flow can be illustrated by the way that the faces information is updated.

This example contains three different types of data stores (INI file, CSV file, and a database), six different applications (Google Picasa, Batch shell program, Java, PHP, SQL, and HTML), and contains manual processes (e.g. administrator updating Picasa) and automated processes (via website generating photo page).

The Impact of One Change

An administrator utilizes the Google Picasa application which updates a picasa.ini file within the Photos directory. A separate batch file reads this picasa.ini file and executes a Java program (PicasaReader.java) that updates a CSV file. A PHP program uses this file (faces.csv) and updates (via a SQL query) the FACES table within a database. Another PHP program uses this table and generates a html page that is displayed to a visitor of the website.

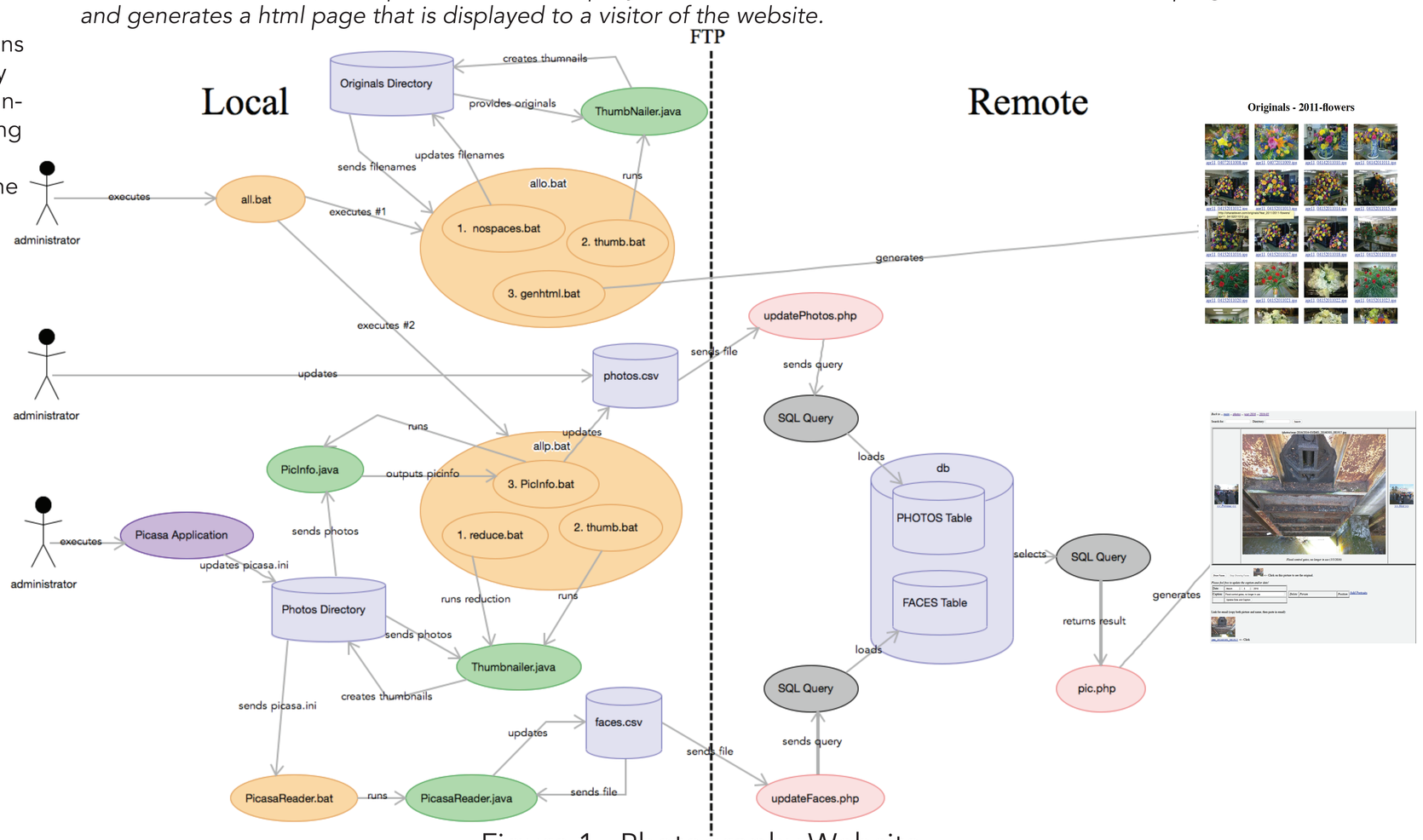

Figure 1 - Photography Website

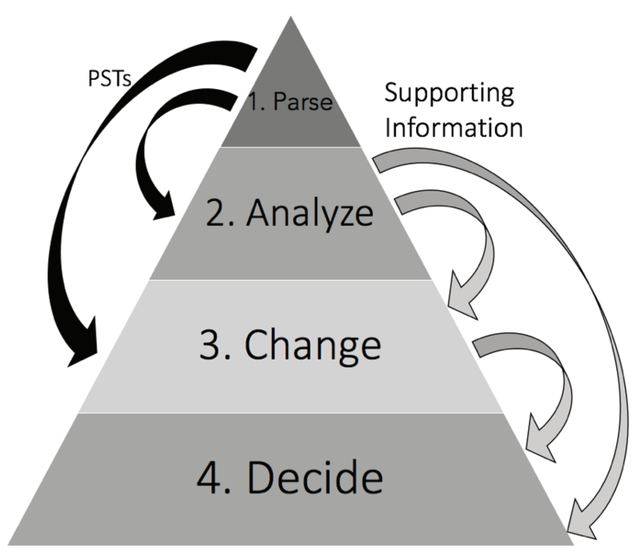

Figure 2 - A PPSM Framework

[A. McAllister, S. O'Hara, "Toward Effective Management of Large-Scale Software", 2016]

The Path Forward

As shown in the example, the challenges above pose difficulties even to a small amount of code. Using manual examination (viewing source code) and ad-hoc processing (running localized tests), some of these answers can be discovered, but manual identification is a tedious and costly endeavor. LIkewise, with large-scale applications, this approach becomes improbable for an individual or even a team to accomplish efficiently. Our research is to discover if it is possible to automatically extract this information, even from large-scale software applications. Given the following framework (Figure 2) for post-production software maintenance (PPSM), our goal is to improve or expose deeper understanding of the ability to analyze software applications (section 2 on the pyramid). If we are able to increase the efficiency, reduce resources, costs, or improve results, we will have greatly aided the efforts of making changes or decisions. For businesses, the ability to make rapid, informed, and accurate decisions is a tremendous advantage.