Journal of Visual Languages and Computing 21 (2010) 230–245

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Sharing, finding and reusing end-user code for reformatting and validating data

Christopher Scaffidi*

School of Electrical Engineering and Computer Science, Oregon State University, 1148 Kelley Engineering Center, Oregon State University, Corvallis, OR 97331-4501, USA

ARTICLE INFO

Article history:
Received 16 December 2009
Received in revised form
14 May 2010
Accepted 19 June 2010

Keywords:
End-user programming
End-user software engineering
Data
Reuse
Spreadsheets

ABSTRACT

To help users with automatically reformatting and validating spreadsheets and other datasets, prior work introduced a user-extensible data model called "topes" and a supporting visual programming language. However, no support has existed to date for users to exchange and reuse topes. This functional gap results in wasteful duplication of work as users implement topes that other people have already created.

In this paper, a design for a new repository system is presented that supports sharing and finding of topes for reuse. This repository tightly integrates traditional keyword-based search with two additional search methods whose usefulness in repositories of end-user code has gone unexplored to date. The first method is "search-by-match", where a user specifies examples of data, and the repository retrieves topes that can reformat and validate that data. The second method is collaborative filtering, which has played a vital role in repositories of non-code artifacts.

The repository's search functionality was empirically tested on a prototype repository implementation by simulating queries generated from real user spreadsheets. This experiment reveals that search-by-match and collaborative filtering greatly improve the accuracy of search over the traditional keyword-based approach, to a recall as high as 95%. These results show that search-by-match and collaborative filtering are useful approaches for helping users to publish, find, and reuse visual programs similar to topes.

1. Introduction

Everyday tasks often require reformatting and validating short human-readable strings such as phone numbers and employee ID numbers. For example, an administrative assistant at a university might gather professors' contact information from multiple web pages into a spreadsheet. Since data on different sites are often formatted differently, putting values into a consistent format can take some effort. For instance, some phone numbers might be formatted like "(541) 737-5572" while others might have a university-specific format like "7-5572", calling for tedious and error-prone reformatting into a consistent format. The user might accidentally make mistakes (e.g., dropping a digit as in "7-557"), and noticing such errors can be difficult when they are buried amid hundreds or thousands of strings.

To help users with these tasks, prior work introduced a user-extensible data model called "topes" and a supporting visual programming language to automate the reformatting and validation of strings [20–23]. Each tope is an abstraction that describes how to reformat and validate instances of one data category. For example, a user might create a tope for phone numbers. Topes have proven highly reusable across spreadsheets and even across applications (e.g., reusing a tope initially created for spreadsheets to validate databases) [21,23]. Studies have shown that topes can precisely reformat and validate many kinds of strings [23], that people can use the toolset's visual language to quickly and

* Tel.: +1 541 737 5572; fax: +1 360 935 7708.
E-mail address: cscaffid@eecs.oregonstate.edu

1045-926X/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2010.06.001

correctly create topes [21,22], and that users are extremely satisfied with the toolset and would like to see it deployed commercially [21,22].

However, no support has been provided to date in order to help people reuse one another's topes. That is, while a user could search through and reuse his or her own topes, there was no way for people to reuse each other's topes. This has been a major limitation because many people use the same kinds of data, and it is inefficient for each person to essentially repeat one another's work in creating a tope or grammar from scratch. Moreover, this limitation has been present in other similar systems, including tools for creating context-free and similar grammars [1,12,14,17].

Outside of grammars, a common approach for tackling this problem is to provide a repository where end users can publish, find, and reuse code. Repositories typically provide mechanisms so users can try out code before downloading it, and they provide minimalist support for browsing and search. Specifically, search is generally based on keywords, tags and categories (e.g., [15,24,25]). However, the effectiveness of these search approaches is limited by the well-known "vocabulary problem", in which different people use widely different labels and tags for the same things [9].

The objective in the current paper is to present and evaluate a repository approach for publishing and reusing topes, with a strong emphasis on discovering effective search algorithms that are well-suited for finding topes, context-free grammars, and similar programmatic descriptions of string data.

Two key insights drive this investigation. The first is that users probably will not want to reuse a tope until they have some data that they need to reformat or validate. For example, a user might want a tope for reformatting phone numbers because he has some phone numbers that need to be reformatted. Consequently, it would be ideal if it were possible to search the tope repository by specifying examples of the strings to match; this approach will be referred to as "search-by-match".

The second insight is that although some kinds of data (such as URLs) could be called "generic" or "universal" as they are used by virtually everyone, many kinds of data are specific to industries, organizations, regions, or other groups of people. For example, the spreadsheets of stock traders might have stock ticker symbols ("GOOG") and stock exchange symbols ("AMEX"), while the spreadsheets of college teachers might have grades ("B+") and course numbers ("COS-101"). Some kinds of data will even be specialized within a particular school or other organization. This insight drives a decentralized repository architecture, where users can "subscribe" to specific repositories that contain topes particular to their organization or industry. In addition, since some public repositories might contain topes relevant to more than one group of users, this insight calls for empirically evaluating whether a dynamic clustering mechanism such as collaborative filtering [3] can be used to improve the accuracy of search results, by matching topes to people based on what other topes have previously been used by "similar" people.

The central hypothesis is that it is possible to quickly and accurately find a tope in a repository based on three pieces of information: keywords, example strings that the tope should match, and a set of recently used topes. To evaluate this hypothesis, a prototype tope repository called "TopeDepot" has been implemented. The repository's search functionality was then empirically tested by simulating queries generated from real user spreadsheets whose data might need to be reformatted or validated.

This experiment has shown that the search-by-match approach is substantially more accurate than simply searching based on keywords and/or tags as in most existing end-user code repositories. Moreover, collaborative filtering further increases accuracy, to a recall as high as 95%. These results suggest that search-by-match and collaborative filtering will support reuse of topes and similar grammars at least as well as many existing code repositories that have already proven successful in practice.

The remainder of this paper is organized as follows. Section 2 discusses related work highlighting the need for mechanisms to facilitate sharing and reuse of grammars like topes. Section 3 presents the topes data model and summarizes prior experiments showing that users can successfully create topes through a visual programming language and use them to reformat and validate data. Section 4 discusses the TopeDepot prototype, including its subscription-based architecture and search interface. Section 5 specifies the search problem to be solved within a topes repository and describes three candidate algorithms that each combine aspects of search-by-match with collaborative filtering. Section 6 presents the empirical evaluation of the search algorithms, revealing that search-by-match and collaborative filtering improve search accuracy over the traditional keyword-based approach. Section 7 discusses implications for repository design. Section 8 summarizes the key conclusions.

2. Related work

Topes are the first abstraction to integrate support for *reformatting* and *validating* the short, human-readable strings of everyday life. Other prior research initiatives addressed either reformatting or validating alone.

In the area of reformatting, Potluck [11] and Lapis [14] support simultaneous editing, where a user edits one string, and the system automatically makes similar changes to other strings. These edits are instantaneous—not stored as rules that can be shared and later executed on other data. Nix's editing-by-example technique is similar, and it infers a macro-like program that records edits and can be replayed later [18], but no mechanism was provided so users could exchange macros.

In the area of validation, Grammex [12], Lapis [14], and Apple Data Detectors [17] each enable users to create context-free or similar grammars using textual programming languages assisted with a visual interface. In addition, SWYN presents a visual language of colored boxes and bubbles for creating regular expressions [1]. None of these systems includes a repository where users