Empir Software Eng (2017) 22:1028–1062
DOI 10.1007/s10664-016-9476-y

Automated training-set creation for software architecture traceability problem

Waleed Zogaan[1] · **Ibrahim Mujhid**[1] · **Joanna C. S. Santos**[1] · **Danielle Gonzalez**[1] · **Mehdi Mirakhorli**[1]

Published online: 10 November 2016

Abstract Automated trace retrieval methods based on machine-learning algorithms can significantly reduce the cost and effort needed to create and maintain traceability links between requirements, architecture and source code. However, there is always an upfront cost to train such algorithms to detect relevant architectural information for each quality attribute in the code. In practice, training supervised or semi-supervised algorithms requires the expert to collect several files of architectural tactics that implement a quality requirement and train a learning method. Establishing such a training set can take weeks to months to complete. Furthermore, the effectiveness of this approach is largely dependent upon the knowledge of the expert. In this paper, we present three baseline approaches for the creation of training data. These approaches are (i) Manual Expert-Based, (ii) Automated Web-Mining, which generates training sets by automatically mining tactic's APIs from technical programming websites, and lastly (iii) Automated Big-Data Analysis, which mines ultra-large scale code repositories to generate training sets. We compare the trace-link creation accuracy achieved using each of these three baseline approaches and discuss the costs and benefits associated with them. Additionally, in a separate study, we investigate the impact

Communicated by: Patrick Mäder, Rocco Oliveto and Andrian Marcus

✉ Mehdi Mirakhorli
mxmvse@rit.edu

Waleed Zogaan
waz7355@rit.edu

Ibrahim Mujhid
ijm9654@rit.edu

Joanna C. S. Santos
jds5109@rit.edu

Danielle Gonzalez
dng2551@rit.edu

[1] Software Engineering Department, Rochester Institute of Technology, Rochester, NY, USA

of training set size on the accuracy of recovering trace links. The results indicate that automated techniques can create a reliable training set for the problem of tracing architectural tactics.

Keywords Architecture traceability · Dataset generation · Architecturally significant requirements · Automation

1 Introduction

Software architecture design is the first and the fundamental step towards addressing non-functional requirements such as security, privacy, safety, reliability, and performance (Bass et al. 2003). To satisfy such requirements an architect must consider alternate design solutions, evaluate their trade-offs, identify the risks and select the best solution (Bass et al. 2003). Such design decisions are often based on well-known architectural tactics (Bachmann et al. 2003), defined as reusable techniques for achieving specific quality concerns. Tactics come in many different shapes and sizes (Bachmann et al. 2003; Bass et al. 2003). For example, reliability tactics provide solutions for fault mitigation, detection, and recovery; performance tactics provide solutions for resource contention in order to optimize response time and throughput, and security tactics provide solutions for authorization, authentication, non-repudiation and other such factors.

Establishing round-trip traceability between architectural tactics and the source code supports several activities such as architecture-level change-impact analysis, design reasoning, and long-term system maintenance (Mirakhorli et al. 2012; Mirakhorli and Cleland-Huang 2011a). For instance, practice has shown that erosion of architecture often occurs when developers make changes to the code without fully understanding the underlying architectural decisions (Mirakhorli et al. 2012; Mirakhorli and Cleland-Huang 2011b). However if trace links are available, they can be used to keep developers informed of underlying architectural decisions in order to reduce the likelihood of undermining previous design decisions (Mirakhorli et al. 2014).

In prior work we proposed Archie (Mirakhorli et al. 2014; Mirakhorli et al. 2012; Mirakhorli 2014), an automated technique based on a data mining approach to establish trace links between architectural tactics and source code. Archie included a set of code-based classifiers constructed to detect different architectural tactics in the source code (Mirakhorli 2014; Mirakhorli et al. 2012). Archie's individual classifiers have been trained to detect audit, asynchronous method invocation, authentication, checkpoint, heartbeat, role-based access control (RBAC), resource pooling, scheduler, and secure session tactics. The classifiers were trained using files containing different architectural tactics collected from hundreds of high-performance, open source projects.

While the results were promising (Mirakhorli et al. 2012; Mirakhorli 2014), we observed that extending this approach to a larger number of tactics requires significant involvement of experts to create new training-sets. This prevents the technique from being practically applied in an industrial traceability setting. This is a significant barrier for software architecture traceability and software traceability in general.

Collecting tactical files from real systems requires a deep understanding of quality attributes and architectural tactics, as well as how these tactics can be implemented. This can be challenging if students or less experienced developers are involved in establishing the training set. Furthermore, even when experts (e.g. architects) are involved in the process of creating such datasets, *the process is very time consuming* as they need to search across various systems, understand their files, and select those which are the best representative of the tactics. As a result of such barriers, the community is conducting research using very small, student-generated datasets or limited industrial datasets which are not shareable, impacting both generalizability and reproducibility of research results.

1.1 Contribution

In this paper we present an empirical study and novel techniques that advances our previous work as well as of future software architecture traceability research in several important ways. We first develop new approaches based on (i) *Web-Mining* and (ii) *Big-Data Analysis* to automate the creation of traceability datasets. The Web-Mining technique generates training sets by automatically mining tactic's APIs from technical programming websites. In contrast, the Big-Data Analysis technique uses an ultra-large scale code repository established in this work to automatically generate quality training sets. The code repository we have established for this paper contains over 116,609 open source projects.

As the second contribution, we report a series of empirical studies conducted to compare the accuracy of a traceability technique trained using the automatically generated training-sets versus the datasets which are manually established by the experts.

As the third contribution, we provide a tool called BUDGET (available online)[1] that implements our automated approaches. BUDGET enables traceability researchers to mine a collection of software repositories containing 22 million source files to create training sets. BUDGET also implements several data sampling techniques.

Along with development of novel techniques, we report the results of empirical studies designed to investigate the following research questions:

Research Question 1: Does the training method based on automated web-mining result in higher trace-links classification accuracy compared to an expert-created training set?

Through a set of experiments reported in Section 4 we investigated this research question. Our empirical analysis indicates that the web-mining approach presented in this paper produces high quality training set. The accuracy of trace-link classifier trained using the web-mining approach is comparable to the classifier trained using expert-created dataset. The statistical analysis shows that the differences are not statistically significant. This finding can expand the current state of software architecture traceability, by facilitating the creation of training data through use of our proposed automated technique.

Research Question 2: Does the training method based on automated big-data analysis result in higher trace-links classification accuracy compared to expert-created training set?

The results of our empirical study indicate that the proposed novel Big-Data Analysis approach creates high quality training-set similar to a dataset created by the experts. Manually creating such dataset is very time consuming and our automated technique provides a significant reduction in training set creation time. Therefore, the Big-Data Analysis approach can be used to help researchers obtain their datasets of tactical files without an extensive cost.

[1] http://design.se.rit.edu/budget/

Research Question 3: What is the Impact of Training Set Size on the Accuracy of Trace Link Classification?

Through this research question we aim to perform a cost-benefit analysis for the cases where the training-set is established manually by experts. The goal is to investigate whether it is worth the effort to manually create large training set with the hope of achieving higher accuracy. In an experiment we compared the trace link classification accuracy of a classifier trained using 10, 20, 30, 40 and 50 projects. The results indicate that there is not a significant difference in the accuracy of the classifier for different training set sizes.

1.2 Originality and Extension

This work differs from our previous ICSE 2012 and TSE 2015 publications in different ways. In those papers (Mirakhorli et al. 2012; Mehdi Mirakhorli 2015) we proposed the original classification technique used to trace architectural tactics in the source code. In this paper we recognize training set creation as a problem in supervised software traceability approaches including our previous architecture traceability technique. Then we present novel Web-Mining and Big-Data analysis techniques which can help software traceability researchers in establishing training sets. In a series of experiment we evaluate the accuracy of automated dataset generation techniques.

1.3 Organization of the Paper

The reminder of this paper is organized as following: Section 2 provides the background for the architecture traceability problem and the related work in the area of automated dataset creation. In Sections 3 and 4 we present the overview of the three training set creation techniques and then present the results of our comparison study in Section 5. We also study the impact of training-set size on accuracy of creating trace links in Section 6. Section 7 briefly present our tool BUDGET. Section 8 discusses threats to validity. In Section 9 we provide additional generalization of results, qualitative insights, and potential additional applications. Finally we present conclusions and future work in Section 10.

2 Background

This section provides the background for the problem of architecture traceability. It summarizes the related work in the area of automated dataset creation and dataset augmentation.

2.1 Architectural Tactics

Tactics serve as a building block of software architecture and are used to satisfy a specific quality. A definition of tactics is provided by Bachmann et al. (2003) who define a tactic as a “means of satisfying a quality-attribute-response measure by manipulating some aspects of a quality attribute model through architectural design decisions”.

We limited the focus of this work to five tactics: *heartbeat*, *scheduling*, *resource pooling*, *authentication*, and *audit trail*. These were selected because they represent a variety of reliability, performance, and security requirements. They are defined as follows Bass et al. (2003):

- **Heartbeat**: A reliability tactic for fault detection, in which one component (sender) emits a periodic heartbeat message while another component listens for the message (receiver). The original component is assumed to have failed when the sender stops sending heartbeat messages. In this situation, a fault correction component is notified.
- **Scheduling**: Resource contentions are managed through scheduling policies such as FIFO (First in first out), fixed-priority, and dynamic priority scheduling.
- **Resource pooling**: Limited resources are shared between clients that do not need exclusive and continual access to a resource. Pooling is typically used for sharing threads, database connections, sockets, and other such resources. This tactic is used to achieve performance goals.
- **Authentication**: Ensures that a user or a remote system is who it claims to be. Authentication is often achieved through passwords, digital certificates, or biometric scans.
- **Audit trail**: A copy of each transaction and associated identifying information is maintained. This audit information can be used to recreate the actions of an attacker, and to support functions such as system recovery and nonrepudiation.

In the following subsection we present a classification-based approach for tracing architectural tactics in the source code. Through the rest of the paper, we train this classifier using datasets generated by different training set creation techniques and compare the accuracy of generated trace links.

2.2 Traceability Challenge: Identifying Tactic-Related Classes

In previous work (Mirakhorli et al. 2012; Mirakhorli 2014) we presented a novel approach for tracing architectural tactics to the source code. As a tactic can be implemented in several ways, through structuring the source code in many different forms, we cannot use structural analysis as the primary means of identification. Our approach therefore relied primarily on information-retrieval (IR) and machine learning techniques to train a classifier to recognize tactics in the source code through learning the specific terms that occur commonly across implemented tactics. A tactic-classifier was used to identify all classes related to a given tactic, thereby establishing traceability links from tactics to the code (Mirakhorli and Cleland-Huang 2011b). The classifier needs to be first trained by several sample implementations of each tactic, and it includes three phases of preparation, training, and classification which are defined as follows:

Data Preparation Phase In this phase, the source code of both the training set and the system under test are preprocessed using standard information retrieval techniques (stemming, stop terms removal, etc). This way, each source code is represented as a vector of terms.

Training Phase The training phase takes a set of preclassified code segments (i.e., training set) as input, and produces a set of *indicator terms*, that are considered to be representative of each tactic type, along with a *weight score*, which shows the level of importance of a specific indicator term with respect to the tactic. For example, a term such as *priority* is found more commonly in code related to the *scheduling* tactic than in other kinds of code and therefore receives a higher weighting with respect to that tactic. In short, the weight score is the probability that a given term is related to a specific tactic in the training set provided as input.

The training formula has three parts. The first part normalizes the frequency with which term t occurs in the training document with respect to its length. The second part computes the percentage of training documents of type q containing term t. Lastly, the third part decreases the weight of terms that are project-specific.

More formally, let q be a specific tactic such as *heartbeat*. Indicator terms of type q are mined by considering the set S_q of all classes that are related to tactic q. The cardinality of S_q is defined as N_q. In (1) $N_q(t)$ refers to the total number of tactic related training files containing the term t, while $N(t)$ refers to the total number of documents containing term t. $NP_q(t)$ refers to the total number of tactical projects containing term t and NP_q refers to the total number of projects in the training data. Through the training process, each term t is assigned a weight score $Pr_q(t)$ that corresponds to the probability that a particular term t identifies a class associated with tactic q. The frequency $freq(c_q, t)$ of term t in a class description c related with tactic q is computed for each tactic description in S_q. $Pr_q(t)$ is then computed as:

$$Pr_q(t) = \frac{1}{N_q} \sum_{c_q \in S_q} \frac{freq(c_q, t)}{|c_q|} * \frac{N_q(t)}{N(t)} * \frac{NP_q(t)}{NP_q} \tag{1}$$

After calculating the weight score ($Pr_q(t)$) for each term t, this phase consider that a term is an *indicator term* if it is higher than a fixed value (*term threshold*).

Classification Phase During the classification phase, the indicator terms computed in (1) are used to evaluate the likelihood ($Pr_q(c)$) that a given class c is associated with the tactic q. Let I_q be the set of indicator terms for tactic q identified during the training phase. The classification score that class c is associated with tactic q is then defined as follows:

$$Pr_q(c) = \frac{\sum_{t \in c \cap I_q} Pr_q(t)}{\sum_{t \in I_q} Pr_q(t)} \tag{2}$$

where the numerator is computed as the sum of the term weights of all type q indicator terms that are contained in c, and the denominator is the sum of the term weights for all type q indicator terms. The probabilistic classifier for a given type q will assign a higher score $Pr_q(c)$ to class c that contains several strong indicator terms for q. This score $Pr_q(c)$ indicates how similar the code is to an implementation of the tactic. Hence, if this probability is higher than a fixed value (*classification threshold*), the code is considered as tactical file.

2.3 Related Work on Dataset Creation/Augmentation

There have been many works in the area of data mining and information retrieval to facilitate training set *selection* in text classification problems. However, the fundamental assumptions in this line of research is that a large number of labeled data points exists and these approaches try to incorporate various sampling (Skalak 1994; Wilson and Martinez 2000), instance selection (Cano et al. 2003; Gates 1972; Brodley 1993) and data reduction techniques (Passini et al. 2013; Wilson and Martinez 2000) to obtain a small representative sample. Unlike these approaches, we do not make such assumption and the main problem in the area of software traceability is the lack of any labeled data.

Web-mining has been previously used in the area of requirements traceability, for example, Jane Cleland-Huang et al. (2010) have used web-mining to replace a hard-to-retrieve query with an augmented query obtained from the web. This work does not focus on creating datasets; it is about expanding the terms in a requirements with more domain specific terms, so it can be better detected by traceability techniques. In contrast, our approach is designed to create code based training set for architectural tactics. Similarly, Anas Mahmoud (2015) has used query augmentation techniques based on text clustering to classify non-functional requirements. In this approach the terms in the specification of non-functional requirements are augmented with the semantically similar terms extracted from the content of Wikipedia.

Recently, there have been a number of projects in the area of mining ultra-large-scale open source software repositories (Dyer et al. 2014; Zhu et al. 2014), these works primarily focus on studying source code and coding issues. There is a limited experimental research on using such resource to generate scientific datasets, particularly for requirements and architecture research. To the best of our knowledge there is no concrete automated technique to help scientists generate their datasets for architecture traceability.

Several independent software engineering communities are providing mechanisms for publishing and sharing datasets. The Mining Software Repositories (MSR) conference holds a Data Track every year where researchers can publish and share their dataset. The Center of Excellence for Software Traceability holds traceability challenges where researchers can share their datasets related to a software traceability challenge. Most works published on these repositories are based on manually created datasets (Liebchen and Shepperd 2008). In our work, we utilize massive amount of public data on the web and large scale software repositories and provide required automation to create high quality datasets.

3 Overview of the Three Baseline Techniques

As previously stated, this paper presents novel automated techniques to create training sets for the problem of tracing architectural tactics. These automated techniques are designed to create software traceability datasets with little or no upfront cost while achieving similar (or better) quality than datasets established by experts.

The proposed automated training set creation techniques as well as the traditional expert-created approach are illustrated in Fig. 1. In the case of the manual expert-created approach,

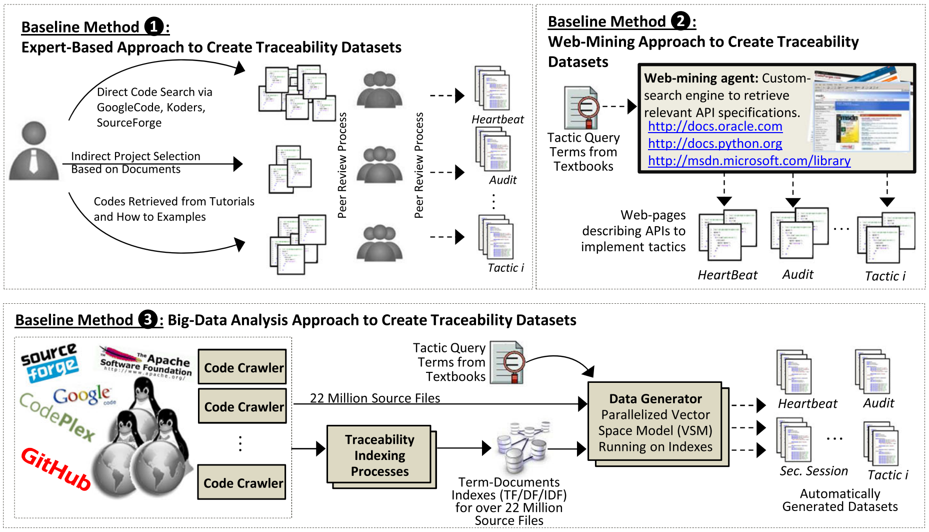

Baseline Method 1:
Expert-Based Approach to Create Traceability Datasets
Direct Code Search via GoogleCode, Koders, SourceForge
Indirect Project Selection Based on Documents
Codes Retrieved from Tutorials and How to Examples
Peer Review Process
Peer Review Process
Heartbeat
Audit
Tactic i
Baseline Method 2:
Web-Mining Approach to Create Traceability Datasets
Web-mining agent: Custom-search engine to retrieve relevant API specifications.
http://docs.oracle.com
http://docs.python.org
http://msdn.microsoft.com/library
Tactic Query Terms from Textbooks
Web-pages describing APIs to implement tactics
HeartBeat
Audit
Tactic i
Baseline Method 3: Big-Data Analysis Approach to Create Traceability Datasets
sourceforge
The Apache Software Foundation
Google code
CodePlex
GitHub
Code Crawler
Code Crawler
Code Crawler
Tactic Query Terms from Textbooks
22 Million Source Files
Traceability Indexing Processes
Term-Documents Indexes (TF/DF/IDF) for over 22 Million Source Files
Data Generator Parallelized Vector Space Model (VSM) Running on Indexes
Heartbeat
Audit
Sec. Session
Tactic i
Automatically Generated Datasets

Fig. 1 Overview of Automated Approaches to Create Tactic Traceability Training-sets

architects collect, review and refine the training set. In the case of the automated techniques, a description of the tactic from textbooks (or a set of tactic related terms) can be used as a tactic-query. Then, in each approach, advanced searching and filtering techniques are used to identify API descriptions or actual implementation of the tactic from technical libraries or open source software repositories.

In the following subsections we describe each of these baseline techniques. Then in section 4 we report empirical studies conducted to compare them.

3.1 Baseline Method 1: Expert-Created Approach

In previous work (Mirakhorli et al. 2012; Mirakhorli et al. 2012) we used a manual approach to collect datasets to train our tactic classifier. The training set shown in Table 1 was established by experts in the area of software architecture and requirements engineering. Then this dataset was peer reviewed and evaluated by two additional independent evaluators. The subject matter experts involved in the project had two to eight years of experience as software architects. The dataset of files implementing architectural tactics were discovered through the following process:

– *Direct Code Search:* The source code search engine (Koders 2014) was used to search for the tactic. The tactic-query for each tactic was composed of keywords used in descriptions of the tactic found in textbooks, articles, and white papers or the libraries that architects have previously used to implement the tactics. All the returned files were

Table 1 Manual Dataset Generated by Experts

Tactic	Projects
Audit	1-Jfolder(Programming), 2-Gnats(Bugs Tracking), 3-Java ObjectBase Manager(Database), 4-Enhydra Shark(Business, workflow engine), 5-Openfire aka Wildfire(Instant messaging), 6-Mifos(Financial), 7-Distributed 3D Secure MPI(Security), 8-OpenVA.(Security), 9-CCNetConfig(Programming), 10-OAJ (OpenAccountingJ)(ERP)
Scheduling	1-CAJO library(Programming), 2-JAVA DynEval(Programming), 3-WEKA Remote Engine(Machine Learning), 4-Realtime Transport Protocol(Programming), 5-LinuxKernel(Operating Systems), 6-Apache Hadoop(Parallel Computing), 7-ReactOS(Operating Systems), 8-Java Scheduler Event Engine(Programming), 9-XORP(Internet Protocol), 10-Mobicents(Mobile Programming)
Authentication	1-Alfresco(Content management), 2-JessieA Free Implementation of the JSSE(Security), 3-PGRADE Grid Portal(Business, workflow engine), 4-Esfinge Framework(Programming), 5-Classpath Extensions(Workflows Management), 6-Jwork(Programming), 7-GVC.SiteMaker(Programming), 8-WebMailJava(Programming), 9-Open Knowledge Initiative(OKI)(Education), 10-Aglet Software Development Kit(Programming)
Heartbeat	1- Real Time Messaging-Java(Programming), 2-Chat3(Instant messaging), 3-Amalgam(Content Management), 4-Jmmp(Programming), 5-RMI Connector Server(Web Programming), 6-SmartFrog(Parallel Computing), 7-F2(Financial), 8-Chromium NetworkManager(Web Programming), 9-Robot Walk Control Behavior(Programming), 10-Apache(Programming)
Pooling	1-ThreadPool Class(Programming), 2-Open Web Single Sign On(Web Programming), 3-ThreadStateMapping2(Programming), 4-RIFE(Web Programming), 5-Mobicents(Mobile Programming), 6-Java Thread Pooling Framework(Programming), 7-Concurrent Query(Programming), 8-RIFE(Web Programming), 9-RIFE(Web Programming), 10-EJBs(Web Programming)

reviewed by two other experts to determine whether they were relevant (i.e. related to the current architectural tactic) or not.

- *Indirect Code Search:* Project-related documents, such as design documents, online forums, etc. were searched for references and pointers to architectural tactics. This information was then used to identify and retrieve relevant code. Similarly all the retrieved files were peer-reviewed to ensure that they were implementing the targeted tactic.
- *"How to" examples:* Online materials, libraries (e.g. MSDN), technical forums (such as Stack Overflow) and tutorials were used to extract concrete examples of implemented architectural tactics.

The rigorous search and validation approach used in this manual data collection resulted in a high quality and precise traceability dataset. However, the cost associated with this approach is substantially high. For instance, it took us about 3 months to collect and peer review tactical data from 10 different projects for 5 architectural tactics. For each of the project, we identified (i) if the tactic is implemented in the project, (ii) which files are involved in the implementation of the tactic, and (iii) how and why the tactic is being used in the project (rationale for the design decision). Through this process we eliminated cases that the tactic was used outside its intended context. This dataset is released at http://coest.org/mt/27/150.

3.2 Web-Mining Approach

Web based libraries, such as *msdn*[2] or *oracle*[3], are one of the resources which contain a rich set of information about implementing architectural tactics as well as many other design and programming concerns. Our initial hypothesis was that creating training sets from these libraries will result in a high quality training set for the classifiers. Figure 2a and b illustrates sample implementation guidelines retrieved from these libraries to implement reliability requirements through *Heartbeat* and security requirements through *Audit Trail* tactics.

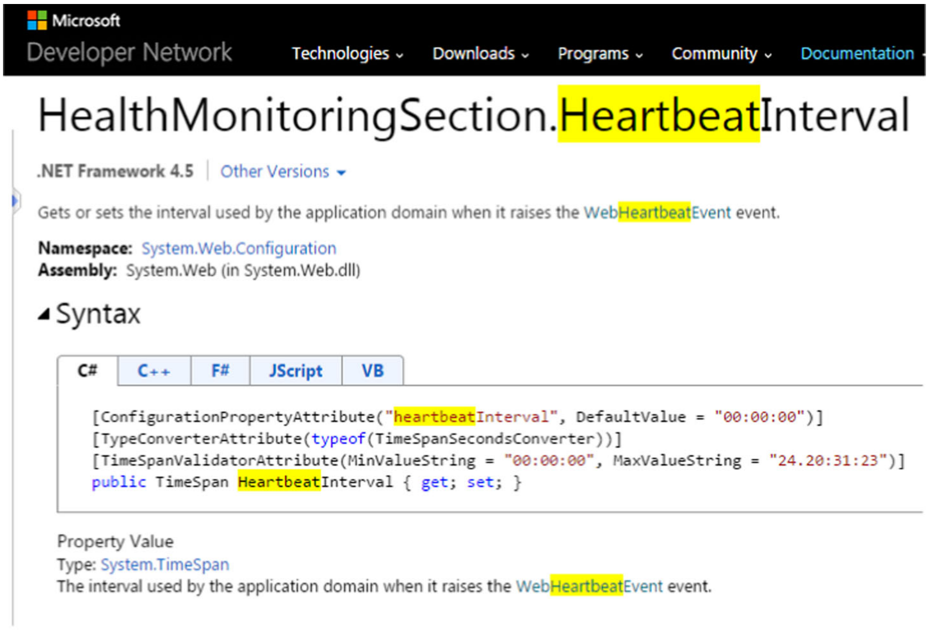
3.2.1 Data Collection Agent

We developed a custom web scraper which uses the search engine APIs of Google to query the content of predefined technical libraries (e.g. *msdn* and *oracle*).

The tactic-query used in this approach contains keywords describing the tactic (drawn from descriptions of the tactic found in textbooks). For example to find APIs related to *HeartBeat* tactic, we used the following textual description from a book (Bass et al. 2003): "***Heartbeat** is a **fault detection** mechanism that employs a periodic message exchange between a system **monitor** and a process being monitored.*" We generated the following *trace query* from this description: *Heartbeat* OR *fault* OR *detection* OR *monitoring*. Although the tactic queries can be more complex than OR joints of search terms, in this paper we only use simple tactic-queries. Table 2 shows the list of tactic-queries that were used.

[2]https://msdn.microsoft.com

[3]http://www.oracle.com



HealthMonitoringSection.HeartbeatInterval

.NET Framework 4.5 | Other Versions

Gets or sets the interval used by the application domain when it raises the WebHeartbeatEvent event.

Namespace: System.Web.Configuration
Assembly: System.Web (in System.Web.dll)

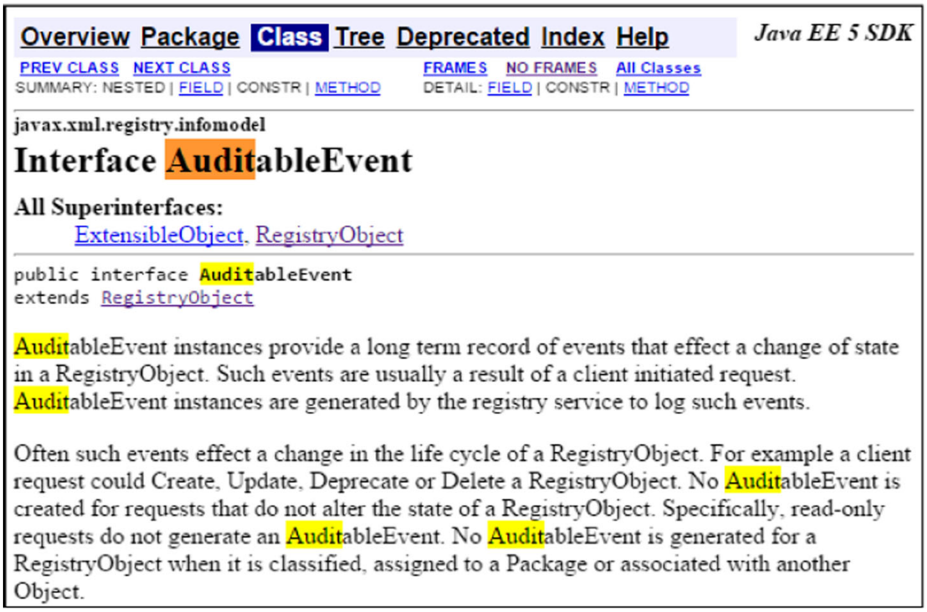
Syntax

C# C++ F# JScript VB

```
[ConfigurationPropertyAttribute("heartbeatInterval", DefaultValue = "00:00:00")]
[TypeConverterAttribute(typeof(TimeSpanSecondsConverter))]
[TimeSpanValidatorAttribute(MinValueString = "00:00:00", MaxValueString = "24.20:31:23")]
public TimeSpan HeartbeatInterval { get; set; }
```

Property Value
Type: System.TimeSpan
The interval used by the application domain when it raises the WebHeartbeatEvent event.

a implementing reliability concerns through Heartbeat tactic from msdn.com


Overview Package Class Tree Deprecated Index Help
Java EE 5 SDK
PREV CLASS NEXT CLASS
FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD

javax.xml.registry.infomodel

Interface AuditableEvent

All Superinterfaces:
ExtensibleObject, RegistryObject

```
public interface AuditableEvent
extends RegistryObject
```

AuditableEvent instances provide a long term record of events that effect a change of state in a RegistryObject. Such events are usually a result of a client initiated request. AuditableEvent instances are generated by the registry service to log such events.

Often such events effect a change in the life cycle of a RegistryObject. For example a client request could Create, Update, Deprecate or Delete a RegistryObject. No AuditableEvent is created for requests that do not alter the state of a RegistryObject. Specifically, read-only requests do not generate an AuditableEvent. No AuditableEvent is generated for a RegistryObject when it is classified, assigned to a Package or associated with another Object.

b addressing security concerns through Audit Trail tactic from Oracle.com

Fig. 2 Two sample API descriptions from technical libraries of (**a**) MSDN and (**b**) Oracle

Table 2 List of queries used in the Web-Mining approach

Tactic	Query
Audit	Audit OR trail OR logging OR log OR history OR security
Scheduling	Scheduling OR task OR worker OR execution OR time OR FIFO OR scheduler OR priority
Authentication	password OR authentication OR sign in OR login OR username OR user OR access
Heartbeat	HeartBeat OR fault OR detection OR health OR alive OR monitoring
Pooling	Thread OR pooling OR resource OR share OR performance OR suspend OR pool

For each tactic, a number of highly-relevant web pages were collected. The scraper-agent returns the ranked web-pages containing relevant API documentations and sample codes to implement the tactic. The information within each Web page is filtered, so the HTML tags are removed and only textual content is stored in a plain text file.

3.2.2 Generated Data

The generated data to train the classifier is a balanced dataset, containing the same number of positive and negative samples. In this case, a balanced samples of text files (web page contents) that are either tactic-related (positive samples) or non-tactical (negative samples). Although the Web-Mining approach is able to generate unbalanced training sets, for the sake of comparing different baseline techniques we generate balanced datasets.

The positive samples are API documentations for a tactic or sample tactical files. The negative or non-tactical samples are sets of documents which have the highest dissimilarity to the originated query. Negative samples would help to remove the terms which are dominant in the Web pages of the library (e.g. Microsoft in MSDN library).

3.3 Big-Data Analysis Approach

This approach relies on using machine learning approaches to create the code-based training sets by mining ultra large scale open source repositories. Our approach includes several different components as illustrated in Fig. 1.

3.3.1 Creating Ultra-Large Scale Repository of Open Source Projects

The first component is the source code scraper, responsible for mining source code of projects from a wide range of open source repositories.

For the purpose of this study, we have extracted over 116,609 projects from Github, Google Code, SourceForge, Apache, and other software repositories. We have developed different code crawling applications to extract projects from all these different code repositories. To extract the projects from Github, we make use of a torrent system known as GHTorrent[4] that acts as a service to extract data and events and gives it back to the

[4]http://ghtorrent.org/

community in the form of MongoDB data dumps. The dumps are composed of information about projects in the form of users, comments on commits, languages, pull requests, follower-following relations, and others.

We also utilized (University of California 2010), an automated crawling, parsing, and fingerprinting application developed by researchers at the University of California, Irvine. *Sourcerer* has been used to extract projects from publicly available open source repositories such as Apache, Java.net, Google Code and Sourceforge. The *Sourcerer* repository contains versioned source code across multiple releases, documentation (if available), project metadata, and a coarse-grained structural analysis of each project. We have downloaded the entire repository of open source systems from these code repositories.

After having extracted all these projects from Github and other repositories, we performed a data cleaning where we removed all the empty or very small projects (i.e. projects that have less than 20 source files). Table 3 shows the frequency of all the projects in different languages in our repository.

3.3.2 Indexing the Data

The second component of the Big-Data Analysis approach is a term-document indexing module, which indexes the occurrence of terms across source files of each project in our code repository. This component, which is called *Traceability Indexing*, first pre-processes each source file, removes the stop words, stems the terms to its root form and then indexes source files. The index stores statistics about each documents (source files) such as *term frequency (TF)*, *document frequency (DF)*, *TF/IDF* and *location of source file* in order to make term-based search more efficient. This is an inverted index which can list, for a term, the source files that contain it McCandless et al. (2010).

3.3.3 Data Generator Component

The third component is a paralleled version of Vector Space Model (VSM) (Salton 1989) capable running over 22 million source files in a few seconds. The VSM is a standard approach in which a query q and a source file f are both represented as a vector of weighted

Table 3 Overview of the projects in Source Code Repository of Big-Data Analysis Approach

Language	Freq.	Language	Freq.	Language	Freq.	Language	Freq.	Language	Freq.
Java	32191	Go	1614	Emacs Lisp	321	ActionScript	120	F#	74
JavaScript	22321	CoffeeScript	1187	Visual Basic	134	Elixir	82	Kotlin	43
Python	9960	Scala	729	Erlang	154	Scheme	80	Bison	39
CSS	9121	Perl	699	Processing	152	Prolog	77	Cuda	37
Ruby	8723	Arduino	321	PowerShell	151	D	72	LiveScript	32
PHP	8425	Lua	458	TypeScript	139	Common Lisp	65	AGS Script	29
C++	5271	Clojure	456	OCaml	105	Pascal	60	SQF	26
C	4592	Rust	308	XSLT	102	Haxe	60	Mathematica	25
C#	4230	Puppet	286	ASP	85	FORTRAN	45	Apex	22
Objective-C++	33	Groovy	253	Dart	84	OpenSCAD	44	PureScript	22
Objective-C	2616	SuperCollider	185	Julia	84	Racket	44	DM	21

*Total number of projects:116,609, *Total number of source files: 23M terms. Therefore, a source file f is represented as a vector $\vec{f} = (w_{1,f}, w_{2,f}, ..., w_{n,f})$ and a query q is represented as $\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{n,q})$, where $w_{i,f}$ represents the weight of the term i for source file f. We used the standard weighting scheme known as $tf - idf$ to assign weights to individual terms (Salton 1989). In this scheme, the tf represents the term frequency, and the idf corresponds to the inverse document frequency. The term frequency is computed for source file f as $tf(t_i, f) = (freq(t_i, f))/(|f|)$, where $freq(t_i, f)$ is the frequency of the term in the file, and $|f|$ is the length of the file. The inverse document frequency idf, is typically computed as:

$$idf_{ti} = log_2 \frac{n}{n_i} \tag{3}$$

where n is the total number of source files in the corpus (our repository) and n_i is the number of source files in which term t_i occurs. Thus, the individual term weight for term i in source file f is then computed as $w_{id} = tf(t_i, f) \times idf_{ti}$. Given these definitions, the similarity score $Sim(f, q)$ between a source file f and technical query q is computed as the cosine of the angle between the two vectors as:

$$Sim(f, q) = \frac{(\sum_{i=1}^{n} w_{i,f} w_{i,q})}{\left(\sqrt{\sum_{i=1}^{n} w_{i,f}} \cdot \sqrt{\sum_{i=1}^{n} w_{i,q}}\right)} \tag{4}$$

This component is used to generate a tactical dataset based on a query provided by a trace user. It calculates the cosine similarity score between provided query and all the source files, using the formula in (4), in the ultra large scale software repository. For each tactic, the most relevant source files exhibiting highest similarity to the trace query are selected. In order to avoid domain specific files, this component also retrieves n samples of non-tactical files for each tactic from the same project (n is defined by the user). Previously it has been proven that unrelated sample data has significant impact on quality of trained indicator terms for the classifier presented in this paper (Mirakhorli et al. 2012; Mirakhorli 2014; Cleland-Huang et al. 2007).

3.3.4 Generated Data

The generated data contains a balanced dataset of tactical and non-tactical files retrieved from 10 open source projects. From each project, a tactical file and one non-tactical file is retrieved.

4 Experiment Design

This section presents the experiment design to compare three baseline training-set creation techniques and to answer our research questions.

In the following we describe the justification for selection of these techniques, and the details of the methodology used to conduct the comparison and validate the results.

4.1 Justification for Selection of Approaches

The domain of automatically-generated training sets for machine learning is relatively new. Although there are previous studies on trace-query replacement and augmentation, the idea of automatically generating training-set has not been explored.

Development of such approaches relies on the existence of large, (un)structured and rich knowledge bases. Since both Web and ultra-large-scale code repositories have such characteristics, one key novelty of the proposed work in this paper is to utilize such resources and develop new techniques to help scientists in the area of software architecture traceability to obtain high quality datasets.

4.2 Expert-Created Dataset Used as Testing Set

The expert-created dataset of architectural tactics was used as the testing-set and a measurement for comparison of the three baseline techniques. This dataset was manually collected and peer reviewed by experts over the time frame of three months.

For each of the five tactics, the experts have identified 10 open-source projects in which the tactic was implemented. For each project, they performed an architectural biopsy ((random sampling of tactical files)) to retrieve a source file in which the targeted tactic was implemented and also retrieved one randomly selected non-tactical file. Using this data we built a balanced training set for each tactic which included 10 tactic-related files and 10 non-tactical ones.

4.3 Assessment Structure

Three different experiments were designed to answer research questions related to comparison of baseline techniques. In the next subsections we describe the experiments for addressing the research questions 1 and 2. Then, in Section 6 we explain the fourth experiment performed for answering the third research question.

4.3.1 Experiment Design for Using Baseline Method 1

The accuracy of classification techniques trained using the expert-created approach was evaluated using a standard 10-fold cross-validation process. In this experiment, the expert-created dataset served as both the training and testing set. This is a classic evaluation technique widely used in the area of data mining and information retrieval and automated requirements traceability (Cleland-Huang et al. 2014; Cleland-Huang et al. 2010; Mirakhorli et al. 2012; Kohavi 1995).

For each execution, we had ten runs such that in the first run nine projects, each including one related and four unrelated files, were used as the training set and one project was used for testing purposes. Following ten such runs, each of the projects was classified one time. In each executions, different pairs of term thresholds and classification thresholds were used.

4.3.2 Experiment Design for Using Baseline Method 2

In the second baseline approach we used a web-mining technique to automatically extract data from technical libraries such as *MSDN* and *ORACLE*. The tactic classifier was trained using this dataset, and then tested against the expert-created dataset of files established by experts (Table 1). The experiment was repeated using a variety of term thresholds and classification thresholds required for (1) and (2). The term thresholds were used for deciding which terms should be part indicator terms list and the classification thresholds were utilized to classify a given source file into tactical/non-tactical.

4.3.3 Experiment Design for Using Baseline Method 3

The third baseline method was trained by the training set generating using Big-Data Analysis approach. Then, the trained classifier was used against the expert-created dataset of tactical files collected by the experts. The training data was sampled from over 116,609 open source projects in our code repository.

4.4 Evaluation Metrics

The results were evaluated using four standard metrics of recall, precision, F-Measure, and specificity computed as follows where *code* is short-hand for *tactical code files*.

$$Recall = \frac{|RelevantCode \cap RetrievedCode|}{|RelevantCode|} \tag{5}$$

while precision measures the fraction of retrieved files that are relevant and is computed as:

$$Precision = \frac{|RelevantCode \cap RetrievedCode|}{|RetrievedCode|} \tag{6}$$

Because it is not feasible to achieve identical recall values across all runs of the algorithm the F-Measure computes the harmonic mean of recall and precision and can be used to compare results across experiments:

$$FMeasure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{7}$$

Finally, specificity measures the fraction of unrelated and unclassified files. It is computed as:

$$Specificity = \frac{|NonRelevantCode|}{|TrueNegatives| + |FalsePositives|} \tag{8}$$

We report all these metrics for each of the baseline techniques so that the results can be interpreted better. F-Measure will be used for the sake of comparing the performance of the baseline approaches. Statistical tests will be performed to compare the F-Measure achieved using each the baseline techniques and obtain further confidence on the answers to the research questions.

4.5 Minimizing Biases

To avoid the impact of datasets size, all the datasets that were automatically generated from our automated approaches (Big-data and Web-mining) included 10 projects (or 10 related web-pages). We trained the classifier using the files automatically extracted using our own primitive big-data analysis technique and then attempted to classify the expert-created dataset of manually established and reviewed files.

In order to avoid the bias of datasets size and primarily comparing the quality of training sets, we decided to use the dataset size equal to manual training-set. Therefore, we only included 10 sample API specifications. Furthermore, for training purposes, similar to manual case, this dataset also includes 40 descriptions of non-tactic-related IT documents collected by our web-scraper.

To minimize the biases toward selection of terms in the tactic-query, we solicited terms from text book descriptions of the tactic. More systematic approaches were conducted to address other related threats to validity, which are thoroughly discussed in Section 8.

5 Results

The experiments design described in Section 4 was followed to train the tactic classifier using three baseline approaches and compare the results. Table 4 shows the top ten indicator terms that were learned for each of the five tactics using the three training techniques. While there is significant overlap, the tactical file approaches unsurprisingly learned more code-oriented terms such as *ping*, *isonlin*, and *pwriter*.

The automated approaches work similar to trace query replacement techniques, which replace a query with a more sophisticated one. However, the data that we generate is a labeled dataset. We have observed that the original query will not identify all the tactic instances in a project. For instance, there are implementations of heartbeat that dont use the term "heartbeat", but contain terms such as "isAlive". There are also source files which use terms such as "detect" and "monitor" but do not implement the heartbeat tactic. A comparison of terms in Table 4 and those we used in our search query (Table 2) supports our claim that our automated approach augments the query with the new terms.

Figure 3 reports the F-Measure results for classifying classes by tactic using several combinations of threshold values. Overall three baseline methods obtained similar accuracy. In two cases, namely *audit* and *heartbeat* the classifier trained using expert-collected files outperformed the classifier trained using automated techniques. In case of *authentication* the

Table 4 Indicator terms learned during training

Tactic Name	Web-Mining trained indicator terms	Big-Data trained indicator terms	expert-created trained indicator terms
Heartbeat	nlb cluster balanc wlb ip unicast network subnet heartbeat host	counter, fd, hb, heartbeat, member, mbr, suspect, ping, hdr, shun	heartbeat, ping, beat, heart, hb, outbound, puls, hsr, period, isonlin
Scheduling	schedul parallel task queue partition thread unord ppl concurr unobserv	schedul, prioriti, task, feasibl, prio, norm, consid, paramet, polici, thread	schedul, task, prioriti, prcb, sched, thread, rtp, weight, tsi
Authentication	authent, password, user, account, credenti, login, membership, access, server, sql	password, login, usernam, rememb, form, authent, persist, sign, panel, succeed	authent, credenti, challeng, kerbero, auth, login, otp, cred, share, sasl
Resource Pooling	thread, wait, pool, applic, perform, server, net, object, memori, worker	pool, job, thread, connect, idl, anonym, async, context, suspend, ms	pool, thread, connect, sparrow, nbp, processor, worker, timewait, jdbc, ti
Audit Trail	audit, transact, log, sql, server, secur, net, applic, databas, manag	trail, audit, categori, observ, udit, outcom, ix, bject, acso, lesser	audit, trail, wizard, pwriter, lthread, log, string, categori, pstmt, pmr

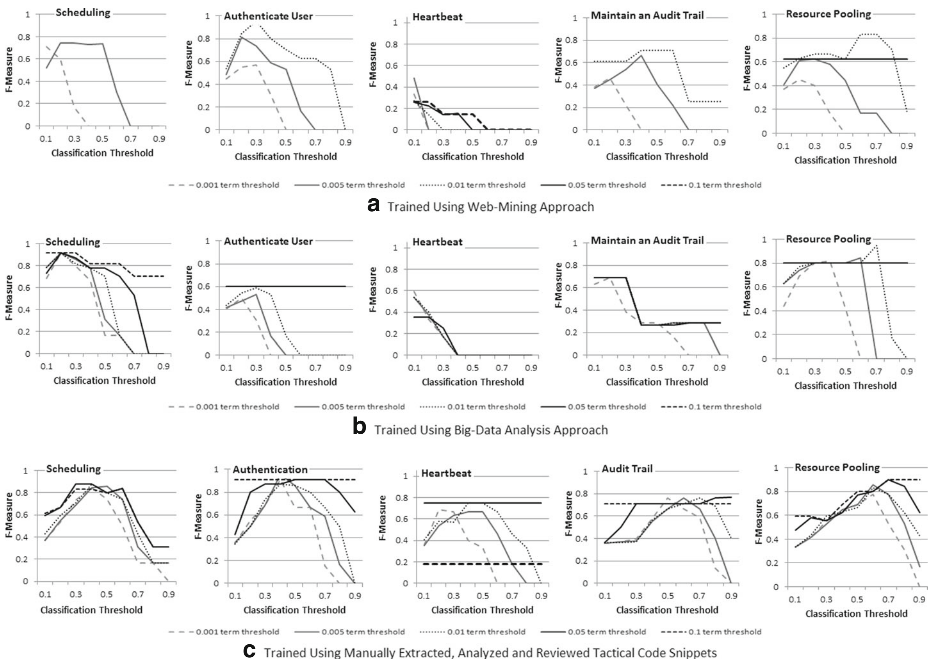


Fig. 3 Results for Detection of Tactic-related Classes at various Classification and Term Thresholds for five Different Tactics

classifier trained using the manually collected dataset achieved the same level of accuracy as the Web-Mining-trained classifier.

In the case of *pooling* and *scheduling*, the Big-data-trained classifier outperformed the other approaches at term threshold values of 0.01 and 0.001 and classification thresholds of 0.7 to 0.3. One phenomenon that needs explaining in these graphs is the horizontal lines in which there is no variation in F-Measure score across various classification values. This generally occurs when all the terms scoring over the term threshold value also score over the classification threshold.

Table 5 reports the optimal results for each of the tactics i.e. a result which achieved the high levels of recall (0.9 or higher if feasible) while also returning as high precision as possible. The results show that in four cases the classifier trained using manually collected data recalled the entire tactic related classes, while also achieving reasonable precision.

The Big-Data-trained classifier achieved recall of 0.909 in one case and recall of 1 for two of the tactics. The classifier trained using Web-based approach achieved recall of 1, in two cases and 0.909 for two other tactics.

RQ 1: Manual Expert-Created Training vs. Automated Web-Mining The above results indicate that, in four out of five cases the manual expert-created approach outperformed the automated Web-Mining technique. However, the differences were very small. Table 6 shows the differences between the F-Measure of the manual expert-created and the automated Web-Mining approaches (Subtract Expert-Created F-Measure from Web-Mining F-Measure).

Table 5 A summary of the highest scoring results

Tactic	Training Method	FMeasure	Recall	Prec.	Spec.	Term/ Classification threshold
Audit	Web-Mining	0.71	1	0.55	0.785	0.01 / 0.4
	Big-Data	0.687	1	0.523	0.762	0.001 / 0.2
	Expert-Created	0.758	1	0.611	0.833	0.001 / 0.5
Authentication	Web-Mining	0.956	1	0.916	0.9772	0.01 / 0.3
	Big-Data	0.6	0.545	0.666	0.931	0.05 /0.1
	Expert-Created	0.956	1	0.916	0.977	0.005 / 0.4
Heartbeat	Web-Mining	0.48	0.545	0.428	0.813	0.005 / 0.1
	Big-Data	0.592	0.727	0.5	0.813	0.001 / 0.1
	Expert-Created	0.689	1	0.526	0.775	0.001 / 0.2
Pooling	Web-Mining	0.833	0.909	0.769	0.931	0.01 / 0.6
	Big-Data	0.952	.909	1	1	0.01 /0.7
	Expert-Created	0.9	0.818	1	1	0.05 / 0.7
Scheduling	Web-Mining	0.740	0.909	0.625	0.863	0.005 /0.2
	Big-Data	0.916	1	0.846	0.954	0.001 / 0.2
Expert-Created	0.88	1	0.785	0.931	0.01 / 0.4	

Based on this limited observation, we can rank the manual expert-created baseline method equivalent to the automated Web-Mining approach. In order to evaluate whether the differences in obtained F-Measures were statistically significant we performed *Wilcoxon* tests as well as the *Friedman ANOVA* test which is a non-parametric test for comparing the medians of paired samples (Note: The data was not normally distributed). Both tests have been recommended for small datasets (as small as 5 per group) (De Winter 2013).

In both statistical tests, we could not reject the null hypothesis (p-value: 0.05) that there is a difference in median/mean-rank of two groups.

Result 1: There is no statistically significant difference between the trace link classification accuracy for a classifier trained using expert-created approach and Automated Web-Mining.

RQ2: Manual Expert-Created Training vs. Automated Big-Data Analysis Table 7 shows the differences between the F-Measure of two approaches (Subtract Expert-Created F-Measure from Big-Data F-Measure). As we can see in the table for the classification of one out of the five tactics (Authenticate), the Big-Data Analysis method outperformed the manual expert-created approach. In all the remaining cases, both methods returned very close results.

Similar to RQ1, *Wilcoxon* and *Friedman ANOVA* tests were conducted to compare the medians of paired samples. In both cases, the null hypothesis was retained (p-value: 0.05).

Table 6 Differences in F-measure of the expert-created and web-mining approaches

Audit	Authenticate	Heartbeat	Pooling	Scheduling
0.048	0	0.209	0.067	0.14

Table 7 Differences in F-measure of manual expert-created and automated big-data analysis approach

Audit	Authenticate	Heartbeat	Pooling	Scheduling
0.071	0.356	0.097	−0.052	−0.036

Result 2: There is no statistically significant difference between the trace link classification accuracy for a classifier trained using expert-created approach and automated Big-Data Analysis. This indicates that Big-Data Analysis approach can be used as a practical technique to help software traceability researchers generate datasets.

6 Cost-Benefit Analysis

Our empirical study of the tree baseline training set creation techniques suggests that there is no statistically significant differences between trace link accuracy for a classification technique trained using each of these techniques. However, the cost of employing experts to help in establishing the training set is significantly higher than automated approaches, while the obtained results are not different. The primary cost for automated techniques is query formation. However, this cost is significantly minimal compared to the manual search for the artifacts. For instance, in our experiments each query was formed by going through the definition of the tactic, and it took approximately less than 2 minutes to finalize the query.

Cost-Comparison In an earlier work, the estimated cost (in terms of time) for creating the training set using the expert-created approach for 5 tactics was 1080 hours. Taking into account the hourly salary of an expert or even a student in terms of dollar per hour will make this approach cost thousands of dollars.
The automated Big-Data approach generates a similar file dataset within a few seconds.

One drawback for any automated data-mining based approach is the inherent inaccuracy of these techniques. To better investigate this fact in our automated techniques, two members of our team manually evaluated the automatically generated training-sets. The accuracy of each training-set per tactic is shown in Table 8. The accuracy is represented by the percentage of correct data points in the generated training-sets that are related to the requested tactic Overall, the automated approach based on Big-Data analysis has created more correct data points (tactical files) than the web-mining approach. This might be because of the amount of noise on the technical libraries as well as inaccuracies in the underlying search technique used by the web-mining approach.

Table 8 Accuracy of automatically generated training-set

	Audit	Scheduling	Authentication	Heartbeat	Pooling
Web-mining	0.6	1	0.91	0.6	0.8
Big-data analysis	1	1	1	0.9	0.9

We also compared the data quality in two baseline methods of Big-Data analysis and Manual method. While in over 90 % of cases the Big-Data approach has successfully retrieved correct files from our large scale software repository, we observed that the data collected by experts exhibits higher internal quality. The manually collected training-set not only contains 100 % accurate data points (due to the rigorous data collection), the experts have also taken into account the representativeness, diversity, generalizability, as well as quality of these samples for training purposes. The manually collected files are richer in terms of vocabularies, APIs and comments. Based on our observation, we believe this is one of the main differences in the underlying baseline methods.

Investigating the score assigned to each indicator terms across three baseline techniques, we observed that the indicator terms generated by manually created dataset have bigger probability scores, and are ordered better with less noises (e.g. unrelated terms). In future work, we aim to augment our automated approach so that not only can they find related files, they will also take into account metrics related to data quality and sampling strategies.

6.1 To What Extent Can an Expert-Created Approach be Practical?

An ongoing debate exists on the research techniques examined/developed using student-generated datasets. The community has utilized different mitigation techniques to minimize the biases and threats related to this set of approaches (Gibiec et al. 2010; Liebchen and Shepperd 2008). At the same time, the community has praised the notion of an Expert-Created approach for obtaining datasets. Unfortunately there are several threats related to this approach as well, which some of them are similar to student-generated datasets. In this section, we will explore one of these challenges, which is related to the extent such datasets can be useful. It is commonly perceived that the larger the size of training set, the more accurate and generalizable the underlying learning method will be. This is essentially because, when the sample size is large enough, it will more accurately reflect the population it was, and therefore the sample is distributed more closely around the population mean. Based on this, and our research question RQ3, we make the following **Null Hypothesis**: *a larger training-set size will not result in a more accurate and generalizeable learning method.*

However, to the best of our knowledge no one has explored whether there is a benefit in extending the training set size generated by the experts, especially because such an extension can have a significant cost. With all the mitigation techniques used to minimize the threat to validity and create generalizable training sets, we do not have scientific confidence in this matter.

Experiment to Investigate In the next experiment we investigate the impact of different dataset sizes on the accuracy of traceability link discovery. The goal of this experiment is not to prove that the training set size matters or does not matter. Instead we aim to perform a cost-benefit analysis for the cases where the training-set is established manually by experts.

In our very first work in this area (Mirakhorli et al. 2012), we used training sets of files from 10 software systems. In extension of this work (Mirakhorli 2014) we used files sampled through a peer-reviewed process from 50 open source projects. Considering that the training sets were established using systematic manual peer review, extending them from 10 open source projects to 50 projects took almost 6 additional months of work. The experiment described in the next sub-section aims to investigate the increase in accuracy of classifiers for such additional cost.

Experiment Design For each of the five tactics included in this study, three experts identified 50 open-source projects in which the tactic was implemented. For each project an architecture biopsy (random sampling of tactical files) was performed to retrieve the source file in which each utilized tactic was implemented. In addition, for each project a randomly selected non-tactical source files was retrieved.

Impact of Training Set Size on Trace Link Accuracy (Two Case Studies) In this part of our research, we investigate RQ3: What is the Impact of Training-set Size on the Accuracy of Trace Link Classification? For this investigation we use two software systems outside of our original dataset, Apache Hadoop and OfBiz. These two projects are widely used in industry and are representative of complex software systems. We made sure, that these two projects are not part of 50 projects used in the training set. So there was no overlap between training data and the case studies used as the testing set. First, the classifiers were trained using 5 randomly-selected sub-samples of the 50 projects in the size ranges of 10, 20, 30, 40, and 50 sample files. Then, each classifier was used against the source code of the two case studies of Hadoop and OfBiz.

We compare the trace-link accuracy of classifiers trained using the different training set sizes to see if there is a return-on-investment for employing experts to establish large(er) training sets.

The accuracy metrics are reported in Fig. 4. The bars in this graph show precision, recall and specificity (Mirakhorli et al. 2012). The red line shows the F-Measure metric. Except *heartbeat* architectural tactic that exhibits major changes across different training set sizes, in all the other four tactics, the training-set size did not show any significant changes in the accuracy of trained classifier.

Result 3: This observation supports the notion that in case of manually creating a high quality training set, the size of the dataset will not have a significant impact on the accuracy of the classification technique described in equations 2 and 1. Collecting of more data-points by experts will not increase the accuracy or generalizability of the trained classifier.

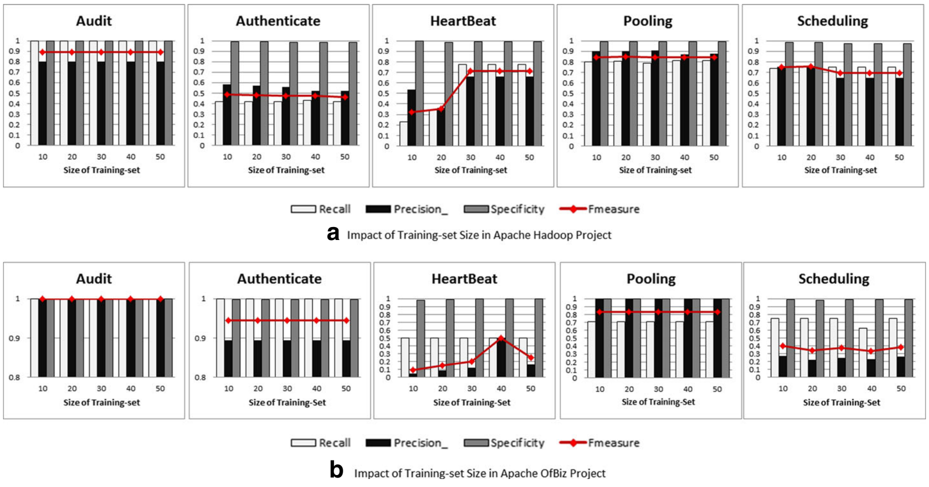
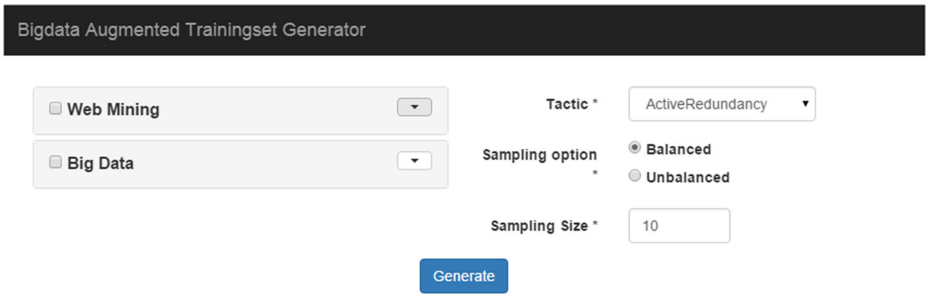


Fig. 4 The impact of training-set size in manually established dataset on accuracy of recovering trace links



Tactic * ActiveRedundancy
Sampling option * [✓ Balanced] Unbalanced
Sampling Size * 10

Fig. 5 User interface for selecting the data generation parameters of BUDGET tool

This observation is only supported by the data obtained from these two case studies. In future works, we will run more experiments, to investigate if this would be valid across different systems.

7 Tool Support

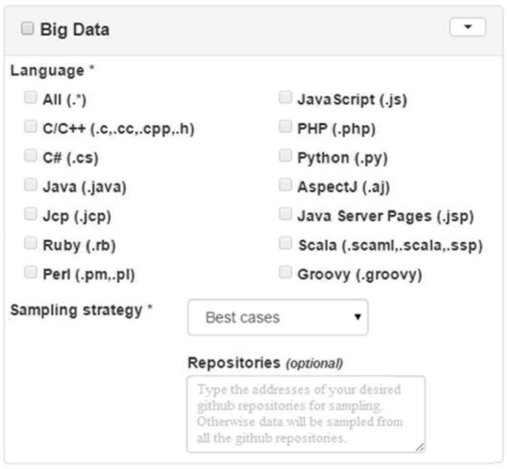
A functional prototype of the automated approaches is developed and released as a web-based tool called BUDGET (Bigdata aUgmented Dataset GEneration Tool).[5] BUDGETs inputs are the name of the tactic of interest, which approach(es) to use when collecting the data - *Web-Mining* or *Big-Data Analysis* - and the dataset size to be generated. Furthermore, there are more advanced sampling parameters that can be tuned if a particular data sampling strategy needs to be followed. This becomes especially useful for Big-Data analysis approach where the user has access to index source code of more than 100,000 applications. The sampling gives the user the flexibility to retrieve the tactical implementations from a single project, many projects, or the entire repository.

Figure 5 shows the user interface for specifying generic parameters of the BUDGET tool. As shown in this figure, the generated dataset size can be either balanced (equal amount of negative and positive samples generated) or unbalanced (different sizes of positive and negative samples) and the datasets can be automatically created using both Web-Mining and Big Data analysis techniques or only one.

When the *Web-Mining* approach is chosen, BUDGET will collect a set of web pages related to a tactic selected from technical libraries. The user can specify the list of technical libraries in a comma-separated list of URLs. By default, the tool uses MSDN as an information source if no other libraries are provided. Figure 6 shows the form field for indicating the technical libraries.

In order to retrieve tactical-related web pages, BUDGET uses Google Search Engine APIs to query technical programming libraries. Tactical terms collected from textbooks are used as a tactic-query. Then, BUDGET creates positive/tactical samples by extracting the content of web pages in top search results (i.e. HTML tags are filtered out). A similar process is followed to generate negative samples; the only difference is that the tactic-query is modified to only return web pages that do not contain any of the tactic-related terms.

[5] http://design.se.rit.edu/budget/



Big Data

Language *

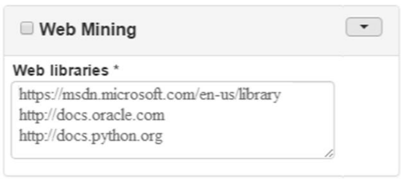
All (.*)	JavaScript (.js)
C/C++ (.c,.cc,.cpp,.h)	PHP (.php)
C# (.cs)	Python (.py)
Java (.java)	AspectJ (.aj)
Jcp (.jcp)	Java Server Pages (.jsp)
Ruby (.rb)	Scala (.scaml,.scala,.ssp)
Perl (.pm,.pl)	Groovy (.groovy)

Sampling strategy * Best cases

Repositories *(optional)*

Type the addresses of your desired github repositories for sampling. Otherwise data will be sampled from all the github repositories.

a Configuration parameters for Big Data Analysis

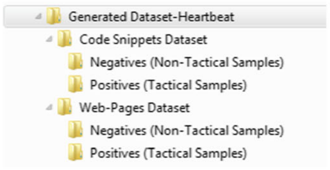


Web Mining

Web libraries *

https://msdn.microsoft.com/en-us/library
http://docs.oracle.com
http://docs.python.org

b UI for specifying the technical libraries for Web Mining



```
Generated Dataset-Heartbeat
  Code Snippets Dataset
    Negatives (Non-Tactical Samples)
    Positives (Tactical Samples)
  Web-Pages Dataset
    Negatives (Non-Tactical Samples)
    Positives (Tactical Samples)
```

a Output of the tool, downloaded as a zip file

Fig. 6 Configuration parameters for Big Data Analysis, Web-Mining Approach and Generated Output

When using the *Big Data Analysis* technique, BUDGET will retrieve source code files from public code repositories to generate the datasets. Currently BUDGET's source code repository contains over 116,609 projects, continuously more open source projects are being added to this repository.

BUDGET's parameter for generating training set of tactical files include programming languages of the source codes, and a sampling strategy (Fig. 6). The sampling strategy defines how BUDGET should sample the tactic-related files from the our ultra-large scale repositories. The three possible sampling strategies are: *Best Cases*, *Random Sampling* and *Stratified Sampling*. These strategies work as follows:

– *Best Cases*: In this strategy, the tactical files with the highest similarity score are returned. By default, the entire source code repository is used for drawing the samples, unless the user specifies a list of repositories to limit the sampling.
– *Random Sampling*: In this strategy, first the user specifies the sampling population by defining the percentage of tactical files to be included in the base population. BUDGET first separates top P % of tactical files (P defined by the user), then randomly generates a dataset size of N (where N is defined by the user).
– *Stratified Sampling*: For each project in the repository (or user defined list), only X tactical source code files are randomly selected. The value of X is also indicated by the user.

After selecting the sampling strategy, the sampled tactical files are sorted based on the similarity score to the tactic-query. Subsequently, the tool generates the N positive and M negative samples defined by the user. For that, the tool selects the N most similar tactical files and the M least related files for generating the positive and negative samples, respectively.

Besides using the tactical terms for the Big-Data Analysis and Web-Mining approaches, the tool has the flexibility of using user-defined terms to generate datasets. In this situation, instead of using our own set of tactical terms, BUDGET applies the terms specified by the user in the *Web-Mining* and *Big-Data Analysis* techniques.

After the datasets are generated, the BUDGET makes them available for download as a compressed file in ZIP format. This ZIP file will contain two folders: one has textual files obtained from Web-Mining and the other has source code files generated from Big-Data Analysis. Each folder has two subdirectories for separating positive from negative cases. Figure 6 shows the folder hierarchy of the datasets generated.

8 Threats To Validity

Threats to validity can be classified as *construct*, *internal*, *external*, and *statistical* validity. We discuss the threats which potentially impacted our work, and the ways in which we attempted to mitigate them. Since BUDGET is accessible for the public, it would enable the researchers in the community to conduct similar experiments, reproduce the results and expand this work. The expert-created dataset used in investigating the impact of dataset size is also released on-line at COEST.org.

External validity evaluates the generalizability of the results. One of the primary threats is related to the construction of the datasets for this study. The manual dataset included over 250 samples of tactic-related code. The task of locating and retrieving these files was conducted primarily by two experts in the area of requirements and software architecture and was reviewed by two additional experts. This was a very time-consuming task that was completed over the course of three months. The systematic process we followed to find tactic related classes and the careful peer-review process gave us confidence that each of the identified files was indeed representative of its relevant tactic. In addition, all of the experiments conducted in our study were based on Java, C# and C code. Some of the identified keyterms are influenced by the constructs in these programming languages such as calls to APIs that support specific tactic implementation. On the other hand, the majority of identified keyterms are non-language specific. Therefore, the experimental results reported in this paper will not be impacted by this language-specific keyterms.

Furthermore, the Hadoop and OfBiz case studies were used to evaluate the impact of dataset size on accuracy of tactic classification on a large and realistic system. We recognize that these are only two case studies and cannot be representative of all typical software engineering environments. In future work, we will to explore additional case studies in an effort to generalize our observations.

Construct validity evaluates the degree to which the claims were correctly measured. The n-fold cross-validation experiments we conducted are a standard approach for evaluating results when it is difficult to gather larger amounts of data. To avoid the impact of dataset size on training-set quality, all the comparison experiments were conducted on the training set of equal size.

Internal validity reflects the extent to which a study minimizes systematic error or bias, so that a causal conclusion can be drawn. A greater threat to validity is that the search for specific tactics was limited by the preconceived notions of the researchers, and that additional undiscovered tactics existed that used entirely different terminology. However we partially mitigated this risk through locating tactics using searching, browsing, and expert opinion. Since multiple data collection mechanisms were used by experts, the dataset is not dependent on a limited number of terms, in fact in some cases there a large diversity in terminologies. In the case of the Hadoop project, we elicited feedback from Hadoop developers on the open discussion forum. An additional threat in this category is that the accuracy of the automated techniques can be dependent on the tactic-query used in the study. To avoid this bias, we pre-selected the queries from description of tactics from text book. In future work we are planning to run different experiments to identify the impact of domain knowledge of the person who creates the query on the quality of datasets. Another threat to validity is related to the expert-created approach and the process which was followed by the experts to search for tactics implementation samples. However we mitigated this risk by asking the experts first to use their expertise and background to search for projects. We also asked them to search within open source community, identify projects with design documents, review these documents and see if there is any tactic implemented in the system. Then they were asked to collect as many of such system. After this step, they conducted code review to identify tactical files. However, we also, asked them in case they cannot find such projects on open source community, they should consider toy systems or tutorial on codeproject.com/[6] or other resources. While they might have used the web search to find some of these project, the main difference between their approach and web-mining approach is that web-mining approach collects web-content from technical libraries, while experts used web search engines to identify projects with tactic. Then they labeled tactical files in these projects.

Statistical validity concerns whether the statistical analysis has been conducted correctly. In order to address this threat appropriate statistical techniques were used. For reliability of conclusions we used two non-parametric tests. Uniformly both tests indicated that there is no statistically differences between the accuracy of training methods although manual ones ranks the best.

9 Discussions

Extrapolating the results of empirical studies beyond the context of the experiments and the data used in them can be risky. Our empirical study is not an exception. We compared three baseline data generation techniques. The results indicate that automated data generation techniques resulted in the same trace link accuracy as expert-created approach. This conclusion was drawn based on study of five architectural tactics. However our further experiments have shown that several tactics share similar characteristics at the code level, and can be detected using text analysis. Therefore, it is possible to utilize BUDGET for automating generation of training set for a large number of architectural tactics. We expect to observe different accuracy.

The impact of dataset size in case of expert-created training set was evaluated using two industrial case study. Although these are large scale, representative industrial projects, we believe further experiments would be beneficial to support/disprove our observation. Since BUDGET is accessible for the public, it would enable the researchers in the community to conduct similar experiments, reproduce the results and expand this work. The expert-created dataset used in investigating the impact of dataset size is also released on-line at COEST.org.[7]

[6]http://www.codeproject.com

[7]http://coest.org/mt/27/150

Table 9 Differences in F-measure of expert-created and big-data analysis approach in naïve bayes approach

Audit	Authenticate	Heartbeat	Pooling	Scheduling
0.038	0	0.179	0.185	-0.21

9.1 Generalization of Results to Other Classification Techniques

Throughout this paper we used our own custom-made classification technique. This was primarily done because our previous work (Mehdi Mirakhorli 2015) shows that our tactic classifiers outperform off-the-shelf classifiers in identifying tactical files. In this section we investigate how another baseline classifier will perform using the data automatically generated by our approach.

To do so, we repeated the experiments in Section 4 using a Naïve classifier.[8] Table 9 reports the difference between F-Measure of Expert-Created and Big-Data Analysis approaches. In case of Scheduling tactic, the Naïve classifier trained using Big-Data Analysis approach outperformed the Naïve classifier trained using the expert-created dataset. In case of Authentication tactic, both classifiers achieved similar F-Measure. In case of Audit the F-Measure of both approach was very close while in the remaining two tactics (Heartbeat and Pooling) the expert-created dataset resulted in better F-Measure.

These results are inline with our observation of how our custom-made classifier performed in the experiment described in the Section 5. Similarly, in order to evaluate whether differences were statistically significant we performed *Wilcoxon* tests as well as the *Friedman ANOVA*. In both statistical tests, we could not reject the null hypothesis that there is a difference in median/mean-rank of two groups ((p-value: 0.05)).

While we used Web-Mining approach to train the Naïve classifier, the F-Measure values obtained were lower than the values obtained when this classifier was trained using expert-created datasets as shown in Table 10. Similar statistical test were performed (*Wilcoxon* and *Friedman ANOVA*) and the results indicate that Naïve classifier while trained using expert-created dataset outperforms the Web-Mining technique.

Our inspection of this issue indicates that Web-Based dataset has more diverse terminologies as well as noises compared to the source files extracted using Big-Data analysis approach. An appropriate feature selection algorithms (Molina et al. 2002) can help identify key discriminating terms in the training set.

Our previous work (Mehdi Mirakhorli 2015) has shown that our custom tactic-classifier outperforms off-the-shelf classifiers. The key factor is the way our classification technique identifies indicator terms, that takes into account the impact of domain terms. The results obtain in this section, provide support that both Big-Data Analysis and Web-Mining approaches are as effective as Expert-Based approach when the data is collected for the tactic-classifier technique (Mirakhorli et al. 2012). Furthermore, Big-Data Analysis approach is as effective as Expert-Based approach when Naïve classifier is used. Considering the empirical results we obtained in an earlier work (Mehdi Mirakhorli 2015) that recognizes the tactic-classifier as best technique to identify tactical files, we can conclude that combined use of our automated dataset generation and tactic-classifier technique can result in the most cost effective way to create traceability datasets and trace tactics to the source code.

[8]Weka's NaiveBayesMultinomialText method was used.

Table 10 Differences in F-measure of expert-created and web-mining approach in naïve bayes approach

Audit	Authenticate	Heartbeat	Pooling	Scheduling
0.187	0.184	0.364	0.368	0.086

9.2 Qualitative Insights

The quantitative experiments reported in previous sections provide evidence that automated techniques can be used to help researchers obtain labeled datasets of software artifacts. In order to gain further insight into how these automated techniques work, and how datasets generated this way differentiate from expert-created datasets, we present a qualitative study in this section. We first compare the automatically generated datasets to the expert-created datasets from various perspectives. Then we analyze random samples of false positives and false negatives for each technique as well as cases reported as true positive for one technique and false positive/negative for another technique.

9.2.1 Datasets Comparison

In Section 6 we evaluated the quality of datasets generated using the Web-Mining and Big-Data Analysis approaches. This evaluation focused on the correctness of the items in the datasets. In this section, we compare the content of datasets generated using automated techniques with those obtained by the expert. Our comparison shows that the tactical source files labeled by the experts are richer in terms of terminology and they tend to have more tactic related terms than those generated automatically. The automated approach normalizes the term frequencies over the the source file length, so most labeled files were relatively small. Comparatively, we found that the experts sometimes included very big source files which also contained a diverse set of tactical terms. Overall, the data generated automatically needed more context about the source file to understand how it related to a tactic, while source files labeled by experts were easier to understand. From the perspective of training a classifier, such differences can be less of an issue, however we plan to expand our approach and include files which are not only representative of a tactic, but also expose qualities similar to those identified by experts. To do this we will develop an algorithm which samples the source files based on their centrality to the tactic's implementation, or retrieves all the files involved in the implementation of the tactic. Our tool will be expanded to enable both of these sampling strategies.

9.2.2 Comparison of Classification Features

In the next qualitative investigation we looked into the indicator terms identified from the expert-created and automatically generated datasets. The top ten indicator terms are depicted in Table 4. While there are several commonalities between top 10 terms, we observed that among the top 30 terms, the Big-Data Analysis dataset contained more diverse terminologies. TheBig-Data Analysis datasets contain other terms associated with the tactic which are not found in the expert-created dataset. As an example, for heartbeat the Big-Data Analysis data contains terms such as *Pinger, live, monitor, msg, health, timeout, ack, failure, heard, master, timer*. However these terms are not used during the classification process, because their score is smaller that indicator term threshold. This indicate a potential for augmenting the automated approach with other techniques such as Natural Language Processing (NLP) and Information Retrieval (IR) to better identify indicator terms.

This would enhance the generalizability of a trained classifier, so that it can identify tactics which are implemented using different frameworks and terminologies.

9.2.3 *Comparison of Results*

For each tactic we inspected a random misclassified case(true negatives, false positive/negative). The results show that in most of these cases a source file was misclassified as false positive because one or two of the indicator terms occurred within that source file. For instance, in case of Audit, the term Audit existed in one of non-tactic related files and that file was always classified as Audit Tactic, despite the fact that other relevant indicator terms such as (Trail, log, records) were missing. We believe that this can be improved by extending the classification equation to take into account the context in which the indicator terms appear as well as reducing the sensitivity of the classifier to a single feature. In cases of false negatives, the source file contained more of indicator terms with lower score (1), therefore the classification score was bellow the chosen classification thresholds. We could not identify specific reasons for why different results was obtained by training the classifier over automated and manually generated datasets. The results primarily depends on the indicator terms discovered in the training phase and which of these terms occur in the testing datasets.

9.3 Application to the other Areas of Requirements Engineering

The experiments reported in this paper show the feasibility and practicality of automated training set generation techniques. The results indicate that the Web-Mining and Big-Data Analysis approaches can automatically generate training set with similar quality to expert-created ones. Here we aim to conduct a feasibility study on using the proposed automated dataset creation technique to support research in different areas of requirements engineering.

In an initial study of resources in technical libraries such as MSDN as well as open source repositories, we identified artifact types which can easily be automatically generated using our approach. Therefore in the following sections we report traceability scenarios where these artifacts can be used.

9.3.1 *Usage Scenario#1: Tracing Regulatory Codes*

We now present the first potential usage scenario for applying the automated dataset generation techniques in the area of tracing regulatory codes.

Problem One of the challenges faced by community of researchers in the area of requirements traceability is the lack of datasets such as requirements, implementation or documentations related to regulatory codes within a software domain. There are a limited number of datasets commonly used such as CCHIT or HIPAA which can be found on COEST.ORG website. The proposed research techniques in this area are primarily evaluated by running experiments over sections of the same dataset, or by tracing one of these to the source code of two open source software systems of WorldVista and Itrust.

Feasibility Study Technical libraries such as MSDN have several documentations, technical guidelines, best practices and preselected technologies and APIs which can be used to address a wide range of regulatory codes, such as HIPAA and SOX (Beeler and Gardner 2006). For example, in Table 11 we list a set of regulatory-compliance acts which we found significant technical discussions about them on MSDN library.

Table 11 Sample regulations discussed on technical libraries

ACT Name	Aplies to
Sarbanes oxley act	Legislation passed by the U.S. Congress to protect shareholders and the general public from accounting errors and fraudulent practices in the enterprise, as well as improve the accuracy of corporate disclosures (Beeler and Gardner 2006). More on (http://www.sec.gov/)
HIPAA	The federal Health Insurance Portability and Accountability Act of 1996. The primary goal of the law is to make it easier for people to keep health insurance protect the confidentiality and security of health care information and help the health care industry control administrative costs. Beeler and Gardner (2006)
PCI	Payment Card Industry Data Security Standard (PCI DSS) is a proprietary information security regulation for organizations that handle branded credit cards. (Council and Payment card industry (pci) data security standard Available over the Internet July 2010)
The Gramm-LeachBliley Act (GLBA)	Also known as the Financial Services Modernization Act of 1999, is an act of the 106th United States Congress, removing barriers for confidentiality and integrity of personal financial information stored by financial institutions. Congress (1999)
SB 1386	California S.B. 1386 was a bill passed by the California legislature. The first of many U.S. and international security breach notification laws. Enactment of a requirement for notification to any resident of California whose unencrypted personal information was, or is reasonably believed to have been, acquired by an unauthorized person. California Senate Bill SB 1386 (2002)
BASEL II	Is recommendations on banking laws and regulations issued by the Basel Committee on Banking Supervision. Aplies to: Confidentiality and integrity of personal financial information stored by financial institutions. Availability of financial systems. Integrity of financial information as it is transmitted. Authentication and integrity of financial transactions (Beeler and Gardner 2006).
Health level seven (HL7)	Provides regulations for the exchange of data among health care computer applications that eliminate or substantially reduce the custom interface programming and program maintenance that may otherwise be required (Beeler and Gardner 2006).

In a preliminary study, we used our automated technique to create a dataset for the domain of “Tracing Regulatory Code”. We ran a sample experiment to create a dataset for technologies which can be used to address HIPAA regulations related to Database and Security. Three independent traceability researchers have evaluated the accuracy of the extracted data. The results are presented in Table 12 and indicated that 63 % of automatically generated data points were correct. These are the extracted files while related to the search query used by our approach. The traceability researchers through a peer-review process evaluated each individual artifact and inspected whether the artifact is about Database Security concerns in HIPA or not. Here we provide an excerpt of two sample data points extracted using our approach:

Table 12 Accuracy of automatically generated datasets in two different areas of requirements engineering

Approach	Query	Correct
Big-data	Query 1: Billing, Bill Calculation, Invoice Generation	90 %
	Query 2: Balance Management, Credit Management, Account Management, Credit Card Processing	100 %
	Query 3: Business Intelligence, SLA Management, Database Marketing	100 %
	Query 4: Product Shipment, Shopping	100 %
Web-Mining	Database Security HIPAA	63 %

- “HIPAA compliance: Healthcare customers and Independent Software Vendors (ISVs) might choose SQL Server in Azure Virtual Machines instead of Azure SQL Database because SQL Server in an Azure Virtual Machine is covered by HIPAA Business Associate Agreement (BAA). For information on compliance, see Azure Trust Center.”
- “Confidentiality: Do not rely on custom or untrusted encryption routines. Use OS platform provided cryptographic APIs, because they have been thoroughly inspected and tested rigorously. Use an asymmetric algorithm such as RSA when it is not possible to safely share a secret between the party encrypting and the party decrypting the data....”

In the area automatically generating dataset precision or accuracy of generated dataset is more important than recall (retrieving all the artifacts from web). In our baseline Big-Data Analysis and Data-Mining approaches we have not utilized any tweaking to improve the precision of artifact retrieval. Common NLP techniques can result in greater artifact accuracy. Our study reported here, provides the accuracy metric of a simplified search technique. This is adequate to support the fact that automated dataset generation techniques will work and to draw the community attention to this research area. In future, work we plan to enhance the accuracy of our artifact retrieval techniques using NLP.

9.4 Usage Scenario#2: Classifying Functional Requirements

Another area where automated techniques can be used is the generation of datasets for classifying functional requirements. Classification in this context is primarily for tracibility purposes and to distinguish functional and nonfunctional requirements.

Problem Traditionally the VSM (Gethers et al. 2011) technique has been widely used to trace functional requirements to source code. On the other hand, there are studies showing the feasibility of using supervised learning methods to trace reoccurring functional requirements (Anish et al. 2015). The biggest drawback for this approach is the difficulty of obtaining several samples of the same functional requirements or files.

Feasibility Study Using Big-Data analysis we observed that, in our ultra-large code repository, there are a large number of software systems from the same domain, therefore the files to implement functional requirements will also reoccur across these systems. Therefore it is possible to collect datasets of such implementation and use different supervised learning techniques to detect these types of requirements in the source code or utilize such dataset for other purposes. Table 12 shows the accuracy of the Big-Data analysis in establishing datasets for files implementing functional requirements of an ERP (Enterprise Resource Planning) software system. In fact, in all cases, our approach successfully created datasets of files to implement those requirements. The terms in the queries to retrieve the implementation of functional requirements were directly extracted from an on-line document of a similar system.[9] Although this is a feasibility study, it highlights the fact that our approach can be applicable to the other types of datasets beyond tactical-files. The purpose of our two usage scenario reported in this section is to show the potential for extending the automated dataset generation techniques to the other areas of software traceability. The positive results reported in the Table 12 indicate that in presence of large corpus, information retrieval techniques can assist developers in obtaining the datasets which require extensive human involvement. In future work, we aim to expand these two usage scenarios, design and execute experiments that thoroughly support the hypothesis articulated for these two scenarios.

10 Summary and Conclusion

In this paper we presented three baseline techniques for creating the training-set data necessary to train a classifier to detect architectural tactics and establish traceability links between tactics and source code. Using five architectural tactics and over 2 million source files, we performed experiments to compare the accuracy and quality of the datasets generated by these techniques. Based on these results, we found no statistically significant differences in accuracy between the expert-based, Big-Data, and Web-Mining dataset generation techniques. This indicates that automated techniques can generate useful training-sets with similar quality to expert-created datasets. The long-term goal of this project is to develop automated techniques capable of creating scientific datasets of higher quality than the expert-created datasets. The impact of dataset size in the case of expert-created training sets was evaluated using two industrial case studies: Apache Hadoop and OfBiz Results from this experiment show that manually creating a larger training will not result in a significantly more accurate classification technique.

In a comparison of our results to those of other baseline approaches, we found that our observations were inline with those of a naïve classifier. We concluded the paper with a feasibility study on potential applications of these techniques in other areas of requirements engineering. Due to the high costs associated with expert-created datasets, we find that using automated techniques to generate the datasets is a cost effective approach which will not require a loss of accuracy as a trade-off. In future work, we will investigate the practicality of the use of automated techniques in other traceability domains. We will also extend our work to provide more fine-grained sampling of the source files or web-pages and therefore reduce the potential noise in the final training sets. We also plan to run more experiments to understand the impact of the researchers' domain knowledge forming the search query on the quality of the established datasets. We also believe that further experimentation would be beneficial to support/disprove our observation regarding the impact of the dataset size on the classifiers accuracy.

[9]Please see terms in the figures: http://www.1tech.eu/clients/casestudy_ventraq

References

Anish PR, Balasubramaniam B, Cleland-Huang J, Wieringa R, Daneva M, Ghaisas S (2015) Identifying architecturally significant functional requirements. In: Proceedings of the Fifth International Workshop on Twin Peaks of Requirements and Architecture, TwinPeaks '15. IEEE Press, NJ, USA, pp 3–8

California Senate Bill SB 1386 (2002) http://www.leginfo.ca.gov/pub/13-14/bill/sen/sb_1351-1400/sb_1351_bill_20140221_introduced.pdf

Congress US (1999) Gramm-Leach-Bliley Act, Financial Privacy Rule. 15 USC:6801–6809. http://www.law.cornell.edu/uscode/usc_sup_01_15_10_94_20_I.html

Council PCI, Payment card industry (pci) data security standard Available over the Internet (July 2010). https://www.pcisecuritystandards.org

Bachmann F, Bass L, Klein M (2003) Deriving Architectural Tactics: Architectural A Step Toward Methodical Architectural Design. Technical Report, Software Engineering Institute

Bass L, Clements P, Kazman R (2003) Software Architecture in Practice. Adison Wesley

Beeler GW Jr, Gardner D (2006) A requirements primer. Queue 4(7):22–26

Brodley CE (1993) Addressing the selective superiority problem: Automatic algorithm/model class selection

Cano JR, Herrera F, Lozano M (2003) Using evolutionary algorithms as instance selection for data reduction in kdd An experimental study. Trans Evol Comp 7(6):561–575

Cleland-Huang J, Czauderna A, Gibiec M, Emenecker J (2010) A machine learning approach for tracing regulatory codes to product specific requirements. In: ICSE (1), pp 155–164

Cleland-Huang J, Gotel O, Huffman Hayes J, Mader P, Zisman A (2014) Software traceability: Trends and future directions. In: Proceedings of the 36th International Conference on Software Engineering (ICSE), India

Cleland-Huang J, Settimi R, Zou X, Solc P (2007) Automated detection and classification of non-functional requirements. Requir Eng 12(2):103–120

Dyer R, Rajan H, Nguyen HA, Nguyen TN (2014) Mining billions of ast nodes to study actual and potential usage of java language features. In: Proceedings of the 36th International Conference on Software Engineering, ICSE 2014. ACM, NY, USA, pp 779–790

Gates G (1972) The reduced nearest neighbor rule (corresp). IEEE Trans Inf Theory 18(3):431–433

Gethers M, Oliveto R, Poshyvanyk D, Lucia A (2011) On integrating orthogonal information retrieval methods to improve traceability recovery. In: 2011 27th IEEE International Conference on Software Maintenance (ICSM), pp 133–142

Gibiec M, Czauderna A, Cleland-Huang J (2010) Towards mining replacement queries for hard-to-retrieve traces. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, ASE '10. ACM, NY, USA, pp 245–254

Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. Morgan Kaufmann

Koders (2014) http://www.koders.com

Liebchen GA, Shepperd M (2008) Data sets and data quality in software engineering. In: Proceedings of the 4th International Workshop on Predictor Models in Software Engineering, PROMISE'08. ACM, NY, USA, pp 39–44

Mahmoud A (2015) An information theoretic approach for extracting and tracing non-functional requirements. In: Proceedings RE. IEEE, pp 36–45

McCandless M, Hatcher E, Gospodnetic O (2010) Lucene in Action, 2nd edn. Covers Apache Lucene 3.0. Manning Publications Co, CT, USA

Mehdi Mirakhorli J. C.-H. (2015) Detecting, tracing, and monitoring architectural tactics in code. IEEE Trans Software Eng

Mirakhorli M (2014) Preserving the quality of architectural decisions in source code. PhD Dissertation, DePaul University Library

Mirakhorli M, Cleland-Huang J (2011) Tracing Non-Functional Requirements. In: Zisman A, Cleland-Huang J, Gotel O (eds) Software and Systems Traceability. Springer-Verlag

Mirakhorli M, Cleland-Huang J (2011) Using tactic traceability information models to reduce the risk of architectural degradation during system maintenance. In: Proceedings of the 2011 27th IEEE International Conference on Software Maintenance, ICSM '11. IEEE Computer Society, DC, USA, pp 123–132

Mirakhorli M, Fakhry A, Grechko A, Wieloch M, Cleland-Huang J (2014) Archie: A tool for detecting, monitoring, and preserving architecturally significant code. In: CM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2014)

Mirakhorli M, Mäder P., Cleland-Huang J (2012) Variability points and design pattern usage in architectural tactics. In: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, FSE '12. ACM, pp 52:1–52:11

Mirakhorli M, Shin Y, Cleland-Huang J, Cinar M (2012) A tactic centric approach for automating traceability of quality concerns. In: International Conference on Software Engineering, ICSE (1)

Molina LC, Belanche L, Nebot À (2002) Feature Selection Algorithms: A Survey and Experimental Evaluation. In: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9–12 December 2002, Maebashi City, Japan, pp 306–313. doi:10.1109/ICDM.2002.1183917

Passini MLC, Estbanez K. B., Figueredo GP, Ebecken NFF (2013) A strategy for training set selection in text classification problems. (IJACSA) International Journal of Advanced Computer Science and Applications 4(6):54–60

Salton G (1989) Automatic text processing: The transformation, analysis, and retrieval of information by computer. Addison-Wesley Longman Publishing Co., Inc., MA, USA

Skalak DB (1994) Prototype and feature selection by sampling and random mutation hill climbing algorithms. In: Machine Learning: Proceedings of the Eleventh International Conference. Morgan Kaufmann, pp 293–301

University of California I (2010) The sourcerer project. sourcerer.ics.uci.edu

De Winter JCF (2013) Using the Student's t-test with extremely small sample sizes

Wilson DR, Martinez TR (2000) Reduction techniques for instance-basedlearning algorithms. Mach Learn 38(3):257–286

Zhu J, Zhou M, Mockus A (2014) Patterns of folder use and project popularity: A case study of github repositories. In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement ESEM '14, vol 4, pp 30:1–30:4

Waleed Zogaan is currently a Ph.D. student in the Software Engineering Department at Rochester Institute of Technology (RIT). He has received an MSc in Software Engineering from RIT and B.S. in Computer Science from King Abdul-Aziz University (KAU, Saudi Arabia). His main research interests are on Software Architecture and Software Requirements Traceability.

Ibrahim Mujhid is currently a lecturer in the College of Computer Science and Mathematics at University of Mosul (IRAQ). He has received an MSc in Software Engineering from RIT and B.S. in Computer Science University of Mosul (IRAQ). His main research interests are on Software Architecture and Recommender systems.

Joanna C. S. Santos is currently a Ph.D. student in the Software Engineering Department at Rochester Institute of Technology (RIT). She has an MSc in Software Engineering from RIT and a B.S. in Computer Engineering from Federal University of Sergipe (UFS, Brazil). Her main research interests are on Software Architecture and Software Security.

Danielle Gonzalez is currently a Ph.D. student in the Software Engineering Department at Rochester Institute of Technology (RIT). She has received a B.S. in Software Engineering from RIT. Her research focuses on enhancing the security, reliability, and safety of software through testing.

Mehdi Mirakhorli is an assistant professor at Rochester Institute of Technology. His research interest is in the area of software architecture design, implementation, and maintenance, and development of tools to support software engineers. Previously, he was a software architect on large data intensive software systems in the banking, meteorological, and health care domains. He currently serves as Associate Editor of IEEE Software Blog on Software Architecture.