Candoia: A Platform for Building and Sharing Mining Software Repositories Tools as Apps

Nitin Mukesh Tiwari

Department of Computer Science
Iowa State University
nmtiwari@iastate.edu

POSC Committee
Major Advisor: Dr. Hridesh Rajan
Dr. G. Prabhu
Dr. S. Kautz

- Problem
  - Building easily customizable, adoptable and applicable mining software repository tools
- Solution
  - An ecosystem which offers suitable abstractions and computational means to realize the process for building and sharing MSR tools as apps.
- Evaluation
- Related works, Conclusion, & Future Work
  - Existing open source tools and frameworks
  - Open source datasets

Goal

- Reduce the efforts required to build MSR tools
- Ease the process of adopting, customizing and sharing MSR tools
- Allow users to run third-party tools more securely

Scenario 1: MSR Tool Building and Sharing

User wants to build a tool for Association Mining

- Source code
  - Java source code
- Version control systsm(VCS)
  - GIT
- Bug Information
  - Github-Issues Bug Tracker

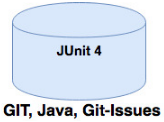


Figure: Software Repository Data

Scenario 1: Tool Building and Sharing

Build a tool for association mining

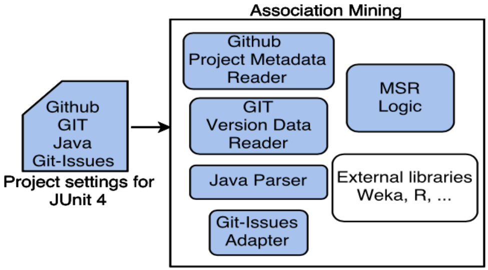

Association Mining
Github Project Metadata Reader
GIT Version Data Reader
Java Parser
Git-Issues Adapter
MSR Logic
External libraries Weka, R, ...
Github GIT Java Git-Issues
Project settings for JUnit 4

Figure: Repository Mining Tool Building

Scenario 1: Tool Building and Sharing

Share the built tool with other researchers and practitioners

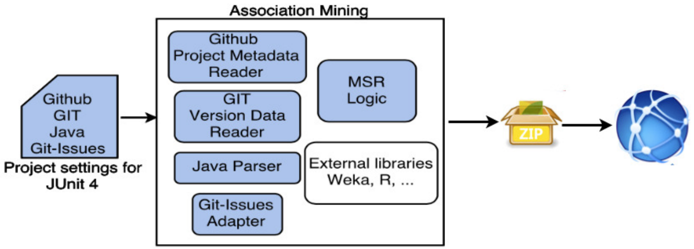


Figure: Complete process of building and sharing tool

Challenges: Scenario 1 MSR Tool Building and Sharing

- User is required to build necessary data preperation tools

Challenges of building and sharing MSR tool

- User is required to build necessary data preperation tools
- Tool are tightly coupled to other tools and particular SCM systems

Scenario 2: Adopting a shared tool

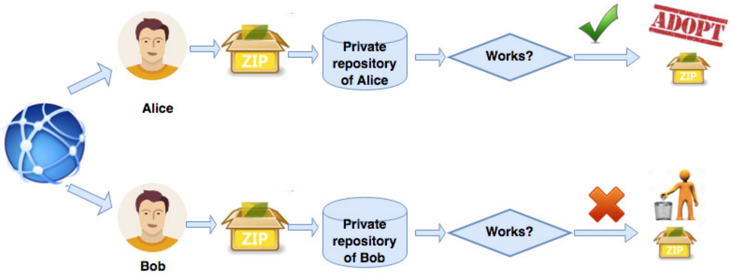


Figure: Repository Mining Tool Building

Scenario 2: Adopting a shared MSR tool

Why Alice was able to adopt but not Bob?
&
What are the possible points of failure?

How MSR tools are build?

- MSR tools are build for specific project setting.
- A project setting defines types and sources of various MSR artifacts

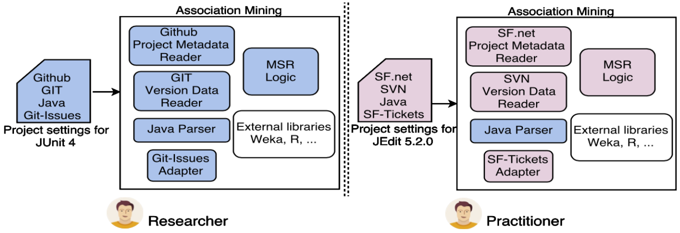
How MSR tools are build?

- MSR tools are build for specific project setting.
- A project setting defines types and sources of various MSR artifacts
- MSR Artifacts: Any kind of information realted to your software
    - Revision history from version control system (VCS)
    - Source code of programming language(s)
    - Bug data from bug trackers
    - Project metadata
    - users and teams data from forges

Potential Points of failure

Failure Cause

An MSR tool build for one project setting may not work for different project settings.


Association Mining
Github Project Metadata Reader
MSR Logic
GIT Version Data Reader
Java Parser
External libraries Weka, R, ...
Git-Issues Adapter
Github GIT Java Git-Issues
Project settings for JUnit 4
Researcher
Association Mining
SF.net Project Metadata Reader
MSR Logic
SVN Version Data Reader
Java Parser
External libraries Weka, R, ...
SF-Tickets Adapter
SF.net SVN Java SF-Tickets
Project settings for JEdit 5.2.0
Practitioner

How to make adopting MSR tools possible?

Our Solution

- Software As Apps
  - Make tool size smaller by pushing generic functionality in the platform
- Programs as Script
  - Make tool components more like scripts, than programs, easier to customize
- Platform and appstore
  - Provide a platform and ecosystem to distribute these tools

Software as Apps

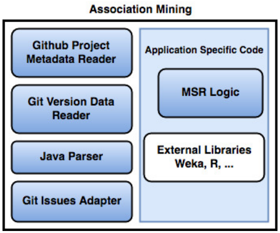

Association Mining
Github Project Metadata Reader
Git Version Data Reader
Java Parser
Git Issues Adapter
Application Specific Code
MSR Logic
External Libraries Weka, R, ...

Figure: Repository Mining Tool Building

Pull supporting tools out and make them available as part of platform

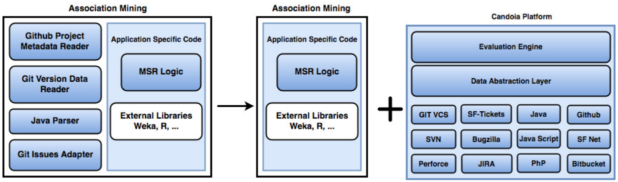


Figure: Repository Mining Tool Building

Programs as Script

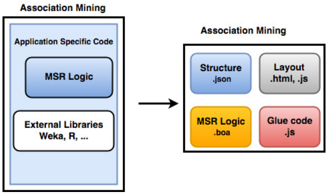


Figure: Repository Mining Tool Building

Association mining as Candoia app

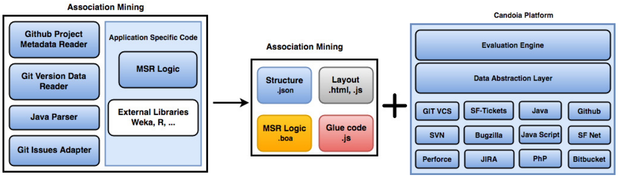


Figure: Repository Mining Tool Building

Building and Sharing Candoia app

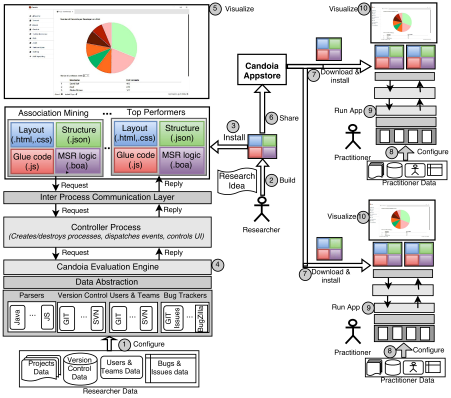
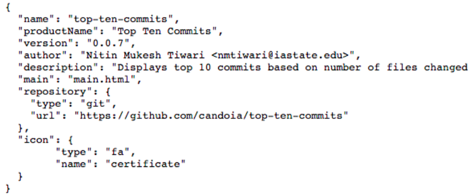


Figure: Repository Mining Tool Building

Candoia App Structure

- Mining Logic
  - Extension of Boa DSL
- Visualization Layout
  - Html
  - CSS
- Glue Code
  - Java script
- Structure Decription
  - Json (file package.json)



```
{
  "name": "top-ten-commits",
  "productName": "Top Ten Commits",
  "version": "0.0.7",
  "author": "Nitin Mukesh Tiwari <nmtiwari@iastate.edu>",
  "description": "Displays top 10 commits based on number of files changed
  "main": "main.html",
  "repository": {
    "type": "git",
    "url": "https://github.com/candoia/top-ten-commits"
  },
  "icon": {
        "type": "fa",
        "name": "certificate"
  }
}
```

Figure: Repository Mining Tool Building

Data Abstraction

- Queries are written over data abstractions
- Data abstractions provide capability of running an app over data collected from diverse sources
- Candoia's data abstraction is extension of Boa DSL.

Candoia Data Schema

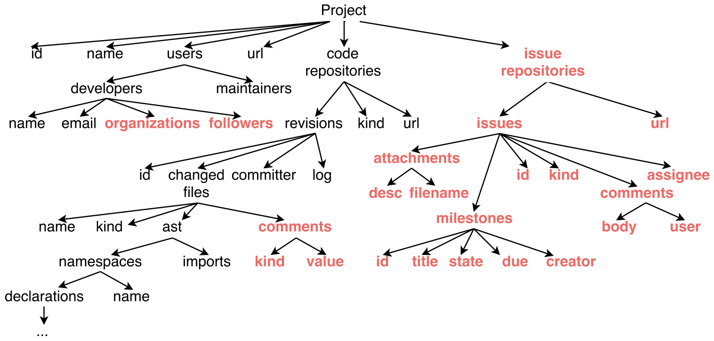


Figure: Candoia's data schema

Customizations

Two levels of customizations

- Data source customizations
- App Customization

Customizations

Two levels of customizations

- Data source customizations
    - Concerned with changing the source of the data
    - No Change in app required
    - Just rerun the application with new datasource
- Customization in Apps

Customizations

Two levels of customizations

- Data source customizations
- Customization in Apps
  - Concerned with customizing different part of the apps
  - app customizations in Candoia are more focused in terms of findings the right component(s) for customizations
  - minimizes the changes required

Customizations

Two levels of customizations

- Data source customizations
- Customization in Apps
  - Concerned with customizing different part of the apps
  - app customizations in Candoia are more focused in terms of findings the right component(s) for customizations
  - minimizes the changes required
    - Changing from Apriori algorithm to EClat in Association mining
    - Changing from File association to package level associattion

Candoia Evaluation Engine

- Interpreter based Query evaluator
- Reads data from local and remote software artifacts
  - Forges: Github, Source Forge
  - VCS: GIT, SVN
  - Bug Tracker: Bugzilla, JIRA, Github-Issue, SF-Ticket
  - Programming Language: Java, Javascript
- Process level parallelization for running multiple apps
- Thread level prarallelization for dataset creation
- Provides fine grained control to Candoia frontend

Candoia Security Architecture

- Prevent apps from corrupting each other
- Allow access to user data on a need to know basis

Chromium based User Interface

- Candoia builds on the process architecture of Chromium
- Process communicates with controller process via Inter Process Communication
- Controller process mediates interaction between file system, window data etc. by exposing APIs.
- An app can only access resources using exposed APIs.

var data = api.boa.run('myprog.boa')

Few APIs available to Candoia app

- Running MSR queries (api.boa)
- Reading (not writing) files within app (api.fs)
- Saving arbitrary data between instances (api.store)
- Getting its own package info such as version (api.meta)
- Inter-Process-Communication handle (api.ipc)

var data = api.boa.run('myprog.boa')

Candoia Exchange

- A web platform for sharing MSR apps
- Candoia frameworks can connect to exchange for gathering information and installing apps