2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion

Candoia: A Platform and Ecosystem for Mining Software Repositories Tools

Nitin M Tiwari Ganesha Upadhyaya Hridesh Rajan
Iowa State University
226 Atanasoff Hall, Ames, IA, 50010, USA
{nmtiwari,ganeshau,hridesh}@iastate.edu

We introduce Candoia, a platform and ecosystem for building Mining Software Repositories (MSR) tools. The platform is designed to support building of MSR tools by providing necessary tools and abstractions that hide the complex details of version control, bug databases, source code programming languages and forges. The ecosystem allows easy sharing and accessing of MSR apps for researchers and practitioners. We have some initial evidence about Candoia's applicability in building robust MSR tools (over two dozen prebuilt apps in the first public release of Candoia), adoptability and interoperability (apps run on widely used projects such as Apache Tomcat, Apache Hadoop etc) and easy customizability (an user study). Candoia is available for download from *http://candoia.org*.

1. INTRODUCTION

Mining Software Repositories (MSR) research has aided in solving many software engineering (SE) problems— defect prediction, source code analysis and pattern discovery, mining specifications, social network analysis for software development to name a few. In this work we describe Candoia, a platform and an ecosystem to ease building, sharing, adopting, and customizing MSR tools. In Candoia, MSR tools are built as "*apps*" akin to mobile apps for Android and iOS platforms. Candoia apps have the property "*build once, run everywhere*", where an app once built can run on a large variety of project settings[1].

Main technical benefits of Candoia come from its ability to provide necessary tools[2] and abstractions[3] to ease building of MSR tools. Candoia users are not required to build tools for making their app applicable in practice and they need not worry about making the app compatible with different project settings. Candoia platform handles the aspects of accessibility, interoperability and customizability of the Candoia apps. Here, by accessibility we mean availability of data and infrastructure used in building the MSR tool [14], by compatibility we mean ability of the MSR tool to work across different project settings without requiring rebuild, and by customizability we mean ability of the MSR tool to allow easy customization to fit usage scenarios in practice.

[1]By software project setting we mean a combination of version control systems (VCS) such as `CVS`, `SVN`, `GIT`, etc., bug databases such as `Bugzilla`, `Issues`, `Jira`, etc., forges such as `SF.net`, `GitHub`, `Bitbucket`, etc., programming languages such as `Java`, `Javascript`, `PHP`, etc.

[2]By necessary tools we mean tools such as language parsers, version control data reader, bug data adapters, etc.

[3]Abstractions unifies the differences in MSR data, such as differences in `GIT` and `SVN` version control data.

ICSE '16 Companion, May 14 - 22, 2016, Austin, TX,USA
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4205-6/16/05.
DOI: http://dx.doi.org/10.1145/2889160.2892662

2. RELATED WORK

There has been efforts along two directions to help MSR researchers and practitioners. First set of approaches provide i) platforms for reusing of tools and allow low cost addition of new tools [4], ii) frameworks that define database schemas for MSR data (such as revision history, source code, etc.) and provide access to this data via SQL [2, 1, 9, 6, 7, 3, 11] and iii) infrastructures for downloading projects from open-source repositories, analyzing the source code, revision histories and other MSR data, and building the dataset for testing the hypothesis [12, 10, 12]. Second set of approaches provide a repository of datasets from open-source repositories so that researchers do not have to collect and curate datasets [5, 13, 8]. When compared to first set of approaches that are mainly focused on enabling faster MSR prototyping, Candoia addresses easier building and customizing of MSR tools and achieves interoperability of the built tools. When compared to second set of approaches that are focused on providing standard datasets, Candoia allows MSR analysis of the user specific datasets.

3. THE PLATFORM AND ECOSYSTEM

A typical workflow of a Candoia user is shown in Figure 1. Candoia users are either researchers, who are evaluating their MSR research prototype or practitioners, such as developers, testers, designers, program managers, support engineer, etc, who are adopting and customizing MSR tools based on their needs. Candoia users use Candoia platform and ecosystem for: i) configuring their project repositories (local or remote), ii) browse and install the existing apps (apps that are prebuilt or shared by other users) and iii) run the apps and visualize the output. Candoia provides various capabilities to its users. Table 1 lists these capabilities. We now describe how Candoia platform and ecosystem is able to provide these capabilities.

Candoia apps are built using common web technologies such as `HTML5`, `CSS`, and `Javascript` which makes building

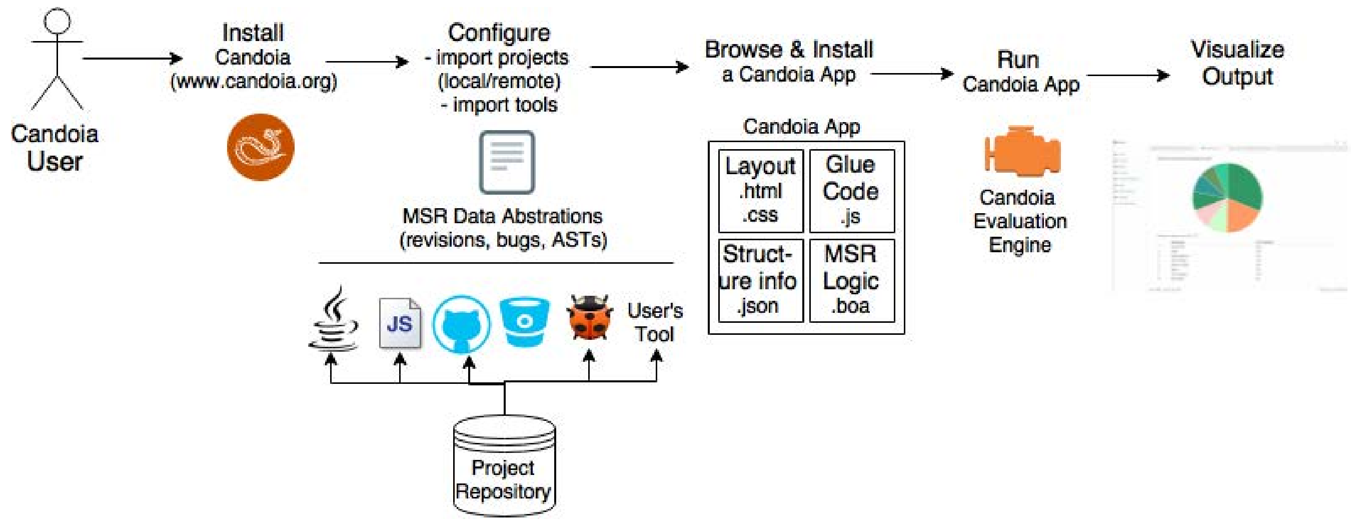

Figure 1: An Overview of a Typical Workflow for a Candoia User

apps accessible to most developers. A Candoia app has a predefined structure for developing various aspects, such as MSR logic, visualization, app structure, etc. The use of a domain-specific language for developing each aspect of an app makes it easier to build them and keeps the lines of code small. Candoia allows integration of external libraries, such as `Weka` or `R` using well defined extension points while building the app.

Table 1: Capabilities of Candoia

Applicability	Enable robust tool development by offering well-defined interfaces to MSR technologies and by providing extension points to add new technologies
Adoptability	Adopting new app is simply by "*Install & Run*"
Compatibility	An app built for one project setting works for a different setting without requiring any changes,
Customizability	Only app-specific customizations are required
Accessibility	sharing apps by publishing them on store
Scalability	Process level parallelism; each app runs as a process

Candoia platform provides tools and abstractions for facilitating the process of building, adopting and customizing Candoia apps. The platform tools read MSR data from user projects which eases adopting MSR tools. The MSR data is represented using well defined abstractions and schemas that serve two purposes: i) hides complex details of version control, bug databases, source code programming languages and forges from app developers, and ii) provides uniform access to MSR data of different sources, such as version control data from GIT and SVN. Candoia apps are built using the abstractions becomes easily compatible across different project settings. Customizing a Candoia app requires customizing only app specifics. An example app specific customization is visualizing an app output using advanced R charts (bar-charts and pie-charts) instead of tabular form.

4. PRELIMINARY RESULTS

We have built over two dozen apps that were of interest to MSR researchers— bug detection apps, software evolution apps, project management apps, static code analysis apps to name a few. Popular source code metrics app computes and displays various metrics such as Chidamber & Kemerer metrics, and object oriented metrics (NOPM, FanOut, etc).

Table 2: Evaluation projects. VCS: Version Control System, PL: Programming Language, D: Developers

Projects	Size (KB)	LOC	AST	Revs	Bugs	D	VCS	PL	Bug data
Tomcat	4900	366459	1209368	15493	3023	29	SVN	Java	Bugzilla
Hadoop	3300	1758594	7490096	11529	10333	49	GIT	Java	Jira
JUnit 4	68	29809	164754	2066	148	121	GIT	Java	Issues
SLF4j	18200	151676	67867	1330	332	53	GIT	Java	Jira
Bootstrap	2500	61513	197348	11519	213	700	GIT	JS	Issues
Node.js	542	1533926	1759485	10762	955	30	GIT	JS	Issues
Grunt	680	3509	22172	1312	155	20	GIT	JS	Issues
JQuery	20100	40626	160602	5882	165	86	GIT	JS	Issues
PMD	2700	150435	1330824	7947	1394	77	GIT	Java	Tickets
JEdit	3100	214126	580194	24025	3926	7	SVN	Java	Tickets

Wait-notify police app is a bug detection app that checks and warns against the improper usage of wait-notify features. The details about other apps can be found in our technical report [15] and these apps are available for download in our public release of Candoia.

For testing adoptability, compatibility, and scalability we have prepared a list of candidate projects, as shown in Table 2. These projects are widely used by the MSR community for testing and evaluation. The table shows project sizes in terms of `Size`, and `LOC` and it can be seen that our evaluation provides sufficiently large candidates to evaluate scalability of Candoia apps. The columns `AST`, `Revs`, `Bugs`, and `D` describes MSR data of the projects that Candoia apps can query. For instance, our source code analysis apps require source code ASTs, software evolution apps require revision data, etc. The columns `VCS`, `PL`, and `Bug data` describes different project settings for adoptability evaluation. In our evaluation we find that apps built using Candoia runs on various project settings without requiring any changes. We performed an user study for understanding the customizability aspect of Candoia. We found that participants were able to perform the customizations fairly quickly [15].

Acknowledgments. This work was supported in part by the US National Science Foundation under grants CCF-15-18897, CNS-15-13263, and CCF-14-23370. Dalton Mills and Trey Erenberger helped with the Candoia frontend code, Eric Lin helped with coding Candoia apps, and Hoan Nguyen provided us helpful feedback on the draft.

5. REFERENCES

[1] S. Bajracharya, J. Ossher, and C. Lopes. Sourcerer: An infrastructure for large-scale collection and analysis of open-source code. *Sci. Comput. Program.*, 79:241–259, Jan. 2014.

[2] J. Bevan, J. E. James Whitehead, S. Kim, and M. Godfrey. Facilitating software evolution research with kenyon. In *ESEC/FSE-13: Proceedings of the 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 177–186. ACM Press, 2005.

[3] V. Dallmeier and T. Zimmermann. Extraction of bug localization benchmarks from history. In *Proceedings of the 22nd IEEE/ACM international conference on Automated software engineering*, pages 433–436, 2007.

[4] S. Ducasse, T. Gîrba, and O. Nierstrasz. Moose: An Agile Reengineering Environment. In *Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ESEC/FSE-13, pages 99–102. ACM, 2005.

[5] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen. Boa: A Language and Infrastructure for Analyzing Ultra-Large-Scale Software Repositories. In *Proceedings of the 35th International Conference on Software Engineering*, ICSE '13, pages 422–431. IEEE Press, 2013.

[6] G. Gousios and D. Spinellis. Alitheia core: An extensible software quality monitoring platform. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 579–582. IEEE Computer Society, 2009.

[7] G. Gousios and D. Spinellis. GHTorrent: GitHub's data from a firehose. In *MSR '12: Proceedings of the 9th Working Conference on Mining Software Repositories*, MSR '12, pages 12–21. IEEE, 2012.

[8] G. Gousios, B. Vasilescu, A. Serebrenik, and A. Zaidman. Lean GHTorrent: GitHub Data on Demand. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR'14, pages 384–387. ACM, 2014.

[9] M. Grechanik, C. McMillan, L. DeFerrari, M. Comi, S. Crespi, D. Poshyvanyk, C. Fu, Q. Xie, and C. Ghezzi. An empirical investigation into a large-scale java open source code repository. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, page 11. ACM, 2010.

[10] J. Howison, M. Conklin, and K. Crowston. Flossmole: A collaborative repository for floss research data and analyses. *IJITWE '06*, 2006.

[11] R. Just, D. Jalali, and M. D. Ernst. Defects4J: A database of existing faults to enable controlled testing studies for Java programs. In *Proceedings of the 2014 International Symposium on Software Testing and Analysis*, pages 437–440, 2014.

[12] G. Pinto, W. Torres, B. Fernandes, F. Castor, and R. S. Barros. A Large-Scale Study on the Usage of Java's Concurrent Programming Constructs. *Journal of Systems and Software*, 106:59–81, 2015.

[13] Promise 2009. http://promisedata.org/2009/datasets.html.

[14] G. Robles. Replicating MSR: A study of the potential replicability of papers published in the Mining Software Repositories proceedings. In *7th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 171–180, 2010.

[15] N. M. Tiwari, D. D. Mills, G. Upadhyaya, E. Lin, and H. Rajan. Candoia: A Platform and an Ecosystem for Building and Deploying Versatile Mining Software Repositories Tools. Technical Report TR15-13, Iowa State University, Nov. 2015. In submission.