UrbanIt: Visualizing Repositories Everywhere

Andrea Ciani, Roberto Minelli, Andrea Mocci, Michele Lanza
REVEAL @ Faculty of Informatics — Università della Svizzera italiana (USI), Switzerland

***Abstract*—Software evolution is supported by a variety of tools that help developers understand the structure of a software system, analyze its history and support specific classes of analyses. However, the increasingly distributed nature of software development requires basic repository analyses to be *always* available to developers, even when they cannot access their workstation with full-fledged applications and command-line tools.**

We present URBANIT, a gesture-based tablet application for the iPad that supports the visualization of software repositories together with useful evolutionary analyses (*e.g.,* version diff) and basic sharing features in a portable and mobile setting. URBANIT is paired with a web application that manages synchronization of multiple repositories.

Website URL: `http://urbanit.inf.usi.ch`
App Store URL: `http://appstore.com/urbanit`

I. Introduction

Tools are fundamental to support many software engineering activities. The approaches that support analysis of software repositories, and in particular the ones that support the understanding of their evolution, are no exception. For example, research has shown the benefits of 3D visualization of the structure of a software system to support development and maintenance tasks [1].

Most tools available to developers are desktop applications and command line utilities that require access to full-fledged workstations to be effective. The increasingly distributed and collaborative nature of software development [2], instead, requires that basic analyses are available to developers *on the go*, without forcing them to sit in front of a full-fledged computer.

We argue that this availability need can be supported by touch-based tablet computers, which provide a good trade-off between portability and effectiveness for visualization of complex repositories. In fact, such devices provide a dedicated gesture-based user interface, which enables novel and powerful exploration and interaction capabilities.

We present URBANIT, a dedicated iPad[1] application that supports the visualization of software repositories in a mobile and portable setting. URBANIT provides the following contributions and supports the following features: i) the synchronization with public `Git`[2] repositories through a web application; ii) a visualization, based on the *city metaphor*, to display the status of a repository, supporting both code and non-code files; iii) basic version diff visualization that can be easily shared with other developers.

[1]See http://www.apple.com/ipad/
[2]See https://git-scm.com

II. UrbanIt in a Nutshell

URBANIT is implemented as an iPad application paired with a web application whose role is to manage users and do the pre-processing of the repositories to be analyzed. At the moment, URBANIT supports public Git repositories (*e.g.,* the ones that can be cloned without authentication at Github[3]).

Figure 1 shows the main user interface of the URBANIT app, visualizing a snapshot of a project, *i.e.,* a commit.

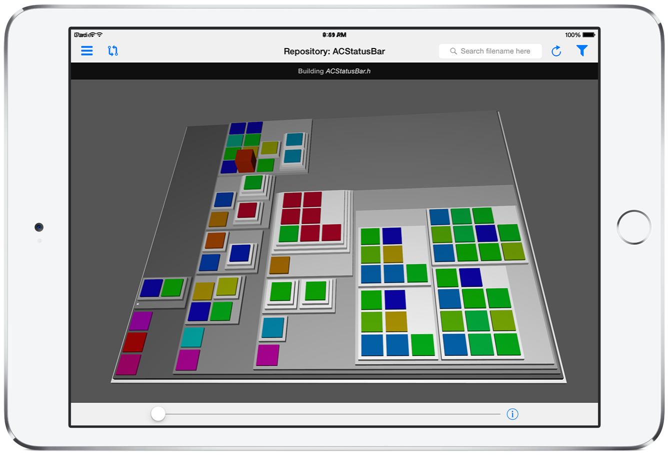


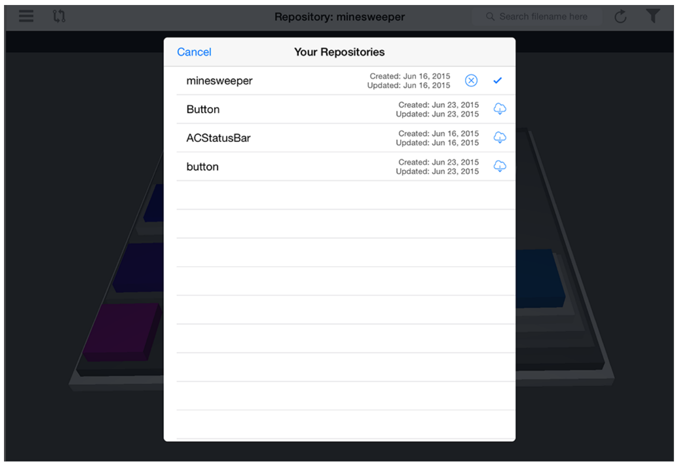
Fig. 1. The Main View of URBANIT, showing a snapshot of a project

URBANIT adopts the *city metaphor* [1], [3] to visualize the logical structure of a software repository for a particular commit. Each directory, starting from the root of the repository, is represented as a *district*. Districts can be contained in each other, as directories. Each file in a directory is represented as a *building* in the corresponding district. The view uses rectangle-packing for laying out districts and buildings. The height of the "buildings" is proportional to the physical size of the corresponding file (in kilobytes). Each building is also colored depending on the corresponding file type (derived from its extension). The user interface of URBANIT allows to explore the structure of a repository in a given commit with typical gesture-based interactions, *i.e.,* moving by panning and zooming by pinching. On the upper part of the view, a toolbar provides basic functionalities to change the selected software repository, search for specific files, and to filter the visualization to specific file types. On the bottom part of the view, a timeline enables the exploration of the repository history, *i.e.,* it can be used to visualize a specific commit, starting from the oldest to the newest.

[3]See http://github.com

III. Features and Usage Scenarios

Repository Subscription. After registering at the website[4], the user can add a set of public Git repositories. They are cloned and processed server-side, and the user receives a push notification when they are ready to be visualized in the app. When she opens URBANIT, the app displays the list of available repositories. She can choose which ones to download and save on the iPad, which ones to delete, or select one to visualize and analyze, as shown in Figure 2.



minesweeper	Created: Jun 16, 2015 Updated: Jun 16, 2015
Button	Created: Jun 23, 2015 Updated: Jun 23, 2015
ACStatusBar	Created: Jun 16, 2015 Updated: Jun 16, 2015
button	Created: Jun 23, 2015 Updated: Jun 23, 2015

Fig. 2. Repository List

Filtering. In the main view of URBANIT, the user can select the types of files she wants to focus on and filter out the others. The filtered files are displayed with translucent colors, *i.e.,* almost transparent. For example, in Figure 3, the developer chose to focus only on code artifacts that are present in the repository, selecting just the files with `.scala` extension (*i.e.,* colored in red) and `.java` extension (*i.e.,* colored in mustard).

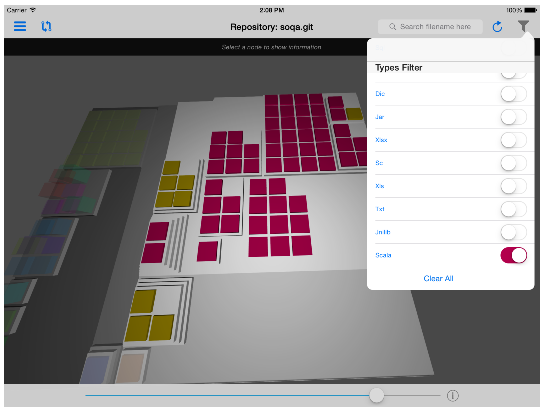
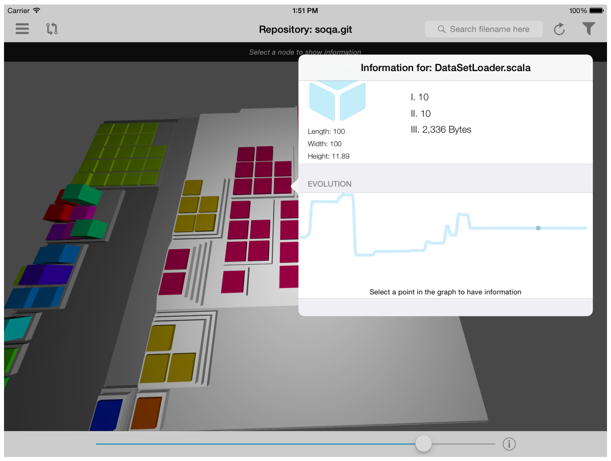


Fig. 3. File type filtering

Selection. The user can select one or more files and obtain additional details by *"long-tapping"* (*i.e.,* tap and hold) on the selection. URBANIT also visualizes a brief overview of the evolution of the total size of the selected files.

[4]See http://urbanit.inf.usi.ch

Figure 4 shows a relatively big Scala source file, called `DataSetLoader.scala`, with a long history and that went through two major refactorings/modifications that reduced its size, before becoming stable and unmodified.


Carrier
1:51 PM
100%
Repository: soqa.git
Search filename here
Select a node to show information
Information for: DataSetLoader.scala

Length: 100
Width: 100
Height: 11.89

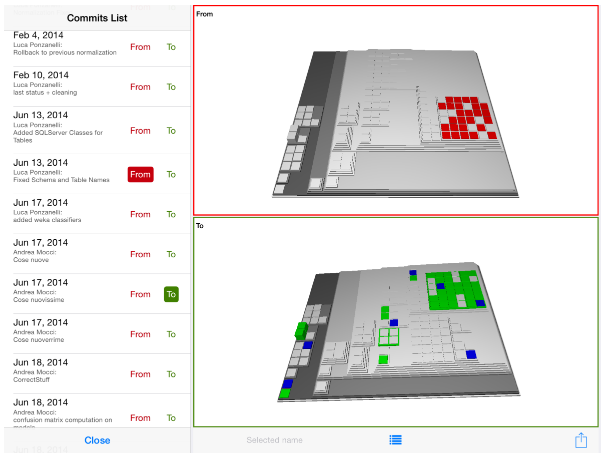
I. 10
II. 10
III. 2,336 Bytes

EVOLUTION
Select a point in the graph to have information

Fig. 4. File Selection and Details

Diff View. The user can open a *diff view* to understand how the repository changed from one version to another (Figure 5). On the left hand side, the user can select two versions (*i.e., From* and *To*), which are then portrayed on the right part of Figure 5. Files (or directories) that have been removed are colored in red on the upper view (representing the older version, *i.e., From*), while files that have been added are colored in green in the bottom view (representing the newer version, *i.e., To*). All modified files are colored in blue.

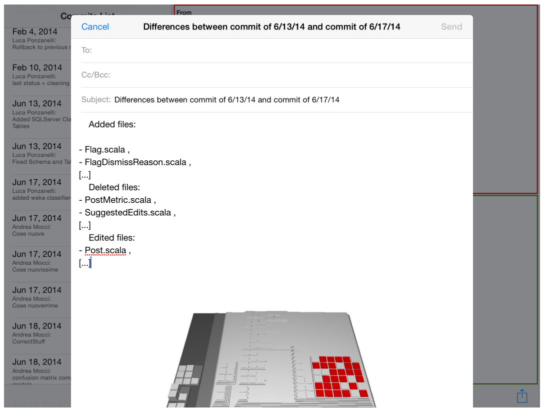
Figure 5 shows a diff view for two versions in the repository, before and after a relatively major series of commits. The developer can spot a major group of files that likely have been moved (*i.e.,* they have been removed from the location in the older version, and added in a new location in the newer version), two directories that have been created, and a group of modified files.



Feb 4, 2014 Luca Ponzanelli: Rollback to previous normalization	From	To
Feb 10, 2014 Luca Ponzanelli: last status + cleaning	From	To
Jun 13, 2014 Luca Ponzanelli: Added SQLServer Classes for Tables	From	To
Jun 13, 2014 Luca Ponzanelli: Fixed Schema and Table Names	[✓ From]	To
Jun 17, 2014 Luca Ponzanelli: added weka classifiers	From	To
Jun 17, 2014 Andrea Mocci: Cose nuove	From	To
Jun 17, 2014 Andrea Mocci: Cose nuovissime	From	[✓ To]
Jun 17, 2014 Andrea Mocci: Cose nuoverrime	From	To
Jun 18, 2014 Andrea Mocci: CorrectStuff	From	To
Jun 18, 2014 Andrea Mocci: confusion matrix computation on	From	To

Fig. 5. Diff view for a Major Refactoring

Communication. The diff view can be easily shared with other developers with a dedicated button located on the lower right side of the diff view. URBANIT is able to generate a simple mail report that contains the list of added, removed and modified files for the selected pair of commits and the corresponding views as generated by the app. Without exiting the app, the developer can add recipients to the mail, add other useful information and send it (see Figure 6).



Feb 4, 2014
Luca Ponzanelli:
Rollback to previous

Feb 10, 2014
Luca Ponzanelli:
last status + cleaning

Jun 13, 2014
Luca Ponzanelli:
Added SQLServer Cla
Tables

Jun 13, 2014
Luca Ponzanelli:
Fixed Schema and Ta

Jun 17, 2014
Luca Ponzanelli:
added weka classifier

Jun 17, 2014
Andrea Mocci:
Cose nuove

Jun 17, 2014
Andrea Mocci:
Cose nuovissime

Jun 17, 2014
Andrea Mocci:
Cose nuoverrime

Jun 18, 2014
Andrea Mocci:
CorrectStuff

Jun 18, 2014
Andrea Mocci:
confusion matrix com

Differences between commit of 6/13/14 and commit of 6/17/14

To:

Cc/Bcc:

Subject: Differences between commit of 6/13/14 and commit of 6/17/14

Added files:

- Flag.scala ,
- FlagDismissReason.scala ,
[...]
Deleted files:
- PostMetric.scala ,
- SuggestedEdits.scala ,
[...]
Edited files:
- Post.scala ,
[...]

Fig. 6. Sharing the Diff View

IV. Related Work

Researchers in software engineering proposed visualizations for many aspects of software systems, including the static structure [4] and of the evolution of repositories, like in the case of CVSScan [5]. Similarly, Riva *et al.* proposed an approach to analyze the stability of the architecture [6] by using colors to depict the changes over a period of releases. Rysselberghe and Demeyer proposed a simple visualization of the evolution of software systems based on information in version control repositories [7]. Similarly, Taylor and Munro [8] proposed *revision towers* to visualize CVS data.

Pioneering work on 3D visualization was proposed by Reiss [9]. Since then, many 3D approaches have been proposed, like Knight *et al.*'s *Software World* [10], use a city metaphor, while Marcus *et al.*'s *sv3d* [11] use a similar 3D metaphor to visualize single versions of software systems. Langelier *et al.*'s *Verso* [12] used 3D visualizations to display structural information, representing classes as boxes with metrics mapped on height, color and twist, and packages as borders around classes.

V. Conclusion

We presented URBANIT, an iPad application paired with a web application that provides basic but effective mechanisms to visualize and navigate `Git` repositories using the city metaphor, providing basic analyses like changing the visualized version and exploring the repository evolution, checking the evolution of a selected file, and comparing two selected versions and sharing the diff view. URBANIT is an attempt to port software visualization and basic evolution analyses in a portable and frequently available environment, that we believe would be effective especially in the context of highly distributed and collaborative environments.

A. *Limitations and Future Work*

One of the main limitations of URBANIT is the ability to work only on public `Git` repositories (*i.e.,* repositories that do not require authentication to be cloned). At the actual state, URBANIT does not support branches in `Git` and analyzes only the main branch. We also plan to support more and diverse version control systems, like Subversion[5] and Mercurial[6], and provide automatic notifications for repository changes.

URBANIT is not tied to source code files, instead it visualizes any kind of data inside repositories. As part of our future work we will investigate how to conveniently analyze the evolution of different categories of repositories, *e.g.,* a repository containing the set of files composing a scientific paper (*i.e.,* LaTeX files and images). This includes the development of novel evolutionary or diff views to better present particular types of data, *e.g.,* text files.

Acknowledgements

We gratefully acknowledge the financial support of the Swiss National Science foundation for the project "HI-SEA" (SNF Project No. 146734).

References

[1] R. Wettel, M. Lanza, and R. Robbes, "Software systems as cities: A controlled experiment," in *Proceedings of ICSE 2011 (33rd International Conference on Software Engineeering)*. ACM, 2011, pp. 551–560.

[2] A. Sarma, G. Bortis, and A. van der Hoek, "Towards supporting awareness of indirect conflicts across software configuration management workspaces," in *Proceedings of ASE 2007 (22nd IEEE/ACM International Conference on Automated Software Engineering)*, 2007, pp. 94–103.

[3] R. Wettel and M. Lanza, "Program comprehension through software habitability," in *Proceedings of ICPC 2007 (15th IEEE International Conference on Program Comprehension)*, 2007, pp. 231–240.

[4] Telea, Maccari, and Riva, "An open visualization toolkit for reverse architecting," in *Proceedings of IWPC 2002 (International Workshop on Program Comprehension)*. IEEE CS, 2002, pp. 3–13.

[5] L. Voinea, A. Telea, and J. J. van Wijk, "CVSscan: visualization of code evolution," in *Proceedings of Softviz 2005 (ACM Symposium on Software Visualization)*, 2005, pp. 47–56.

[6] M. Jazayeri, H. Gall, and C. Riva, "Visualizing software release histories: The use of color and the third dimension," in *Proceedings of ICSM 1999 (16th IEEE International Conference of Software Maintenance)*. IEEE CS Press, 1999, pp. 99–108.

[7] F. Van Rysselberghe and S. Demeyer, "Studying software evolution information by visualizing the change history," in *Proceedings of ICSM 2004 (20th IEEE International Conference of Software Maintenance)*. IEEE Computer Society Press, 2004, pp. 328–337.

[8] C. Taylor and M. Munro, "Revision towers," in *Proceedings of VISSOFT 2002 (1st International Workshop on Visualizing Software for Understanding and Analysis)*. IEEE Computer Society, 2002, pp. 43–50.

[9] S. P. Reiss, "An engine for the 3D visualization of program information," *J. Vis. Lang. Comput.*, vol. 6, no. 3, pp. 299–323, 1995.

[10] C. Knight and M. C. Munro, "Virtual but visible software," in *Proceedings of IV 2000 (IEEE International Conference on Information Visualization)*, 2000, pp. 198–205.

[11] A. Marcus, L. Feng, and J. I. Maletic, "3D representations for software visualization," in *Proceedings of Softvis 2003 (ACM Symposium on Software Visualization)*. IEEE, 2003, pp. 27–37.

[12] G. Langelier, H. A. Sahraoui, and P. Poulin, "Visualization-based analysis of quality for large-scale software systems," in *Proceedings of ASE 2005 (120th IEEE/ACM International Conference on Automated Software Engineering)*. ACM, 2005, pp. 214–223.

[5]See https://subversion.apache.org
[6]See https://mercurial.selenic.com