Computer Support for Discovering OSS Processes

Chris Jensen and Walt Scacchi
Institute for Software Research
Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA USA 92697-3425
{cjensen, wscacchi}@ics.uci.edu

ABSTRACT

Large scale open source software (OSS) projects offer a wide range of documentation of the software processes that have enabled their success. Discovering these processes has been shown to be difficult to achieve. This paper describes our experiences with providing computer support for discovering OSS processes from project data. We discuss challenges of collecting and analyzing data from multiple types of project artifacts and how to address them.

1. INTRODUCTION

OSS projects have been shown capable of creating large, widely adopted, high quality, and arguably secure software systems. While research interest in OSS projects remains avid, much of this research examines software development issues, joining and participation, and project governance. Few researchers have examined these issues from a process perspective. What exactly is a process and how are processes discovered? Software process research defines a process as a sequence of partially ordered steps intended to reach a goal (e.g. release a version of a software product, transition to a different role within a project, etc.). [1]. A plethora of process languages define possible orderings (control flow) and what precisely constitutes a step (the constituents sometimes referred to as *process entities*). These languages are most frequently used to *prescribe* what a process should be: what steps should be performed and in what order. There is comparatively little research providing *descriptions* of specific software processes in specific projects: what steps were performed in a particular process and in what order. Similarly, there is little research on how to discover and model process descriptively.

Process discovery generally entails collecting data that informs us about the process under study, composing evidence of the process as individual process steps, and putting those steps in order. Over the past several years, we have been looking at approaches to discovering and modeling OSS processes. Our overall strategy is ethnographic, applying a reference model [2] of known OSS activities, resources, tools, and roles-- the process entities we use in our models-- to identify evidence of process steps from publicly available project artifacts. We have utilized this strategy in the study of multiple types of processes in multiple projects. In particular, we have focused our efforts on requirements and release, role migration, and governance processes in the NetBeans project and the Apache HTTP server project. In doing so, we have identified several challenges in data collection and analysis and our approach to discovery has evolved to address them.

The first discovery approach we used was applied to the requirements and release processes of the NetBeans and Apache projects and conducted in a purely manual fashion: browsing project websites and noting instances that served as record of software requirements being presented, discussed, and asserted and releases being coordinated [3]. To date, we have identified over fifteen types of project artifacts that provide evidence of software processes, including threaded email archives, issue/defect reports, source code, and documentation [4]. Given the large number of artifacts in a typical OSS project web repository, this approach quickly proved intractable. Without computer support there is simply too much data to analyze in order to produce a plausibly valid process model. Thus, our work to provide computer assistance for process discovery has focused on identifying process information from project artifacts and constructing process steps.

In this paper, we discuss two projects we have worked on to provide computer support for process discovery. Both rely on the reference model to provide a mapping between project data and process concepts. Both also enlist the aid of search technology to facilitate the process data elicitation. Since process discovery is a knowledge elicitation activity [5], it made sense to apply a tool geared towards retrieving artifacts with desired knowledge to the research problem. The widespread development and adoption of search technology encouraged this strategy, more so because data evidencing each process step is commonly spread across multiple artifacts/web documents.

For our first project, we tailored an open source search engine (Apache.org's Nutch [6] web crawler and Lucene search engine) to perform detailed analysis of a local set of project artifacts from the Apache HTTP and NetBeans projects. Our goal was to construct high fidelity models using a mix of qualitative and quantitative analysis of the data. However, we encountered several challenges in data collection and analysis. We discuss these next and follow this with a discussion of a tool we have recently built in response as a way forward in computer-supported process discovery.

2. PROCESS DISCOVERY WITH LUCENE

Under this scheme, the reference model provided process concepts used to construct search queries from an index of project artifacts. Process fragments were identified from the query “hits” and assembled into atomic steps, ordered based on contextual clues in the selected artifacts (e.g. date stamps from email messages, versioning repositories, etc.). While the querying stage was easily scripted, it quickly became clear from the results that constructing process steps and putting them in order could not be done reliably without human intervention. Nevertheless, the tool could inform these tasks by providing time and date information, as well as links to documents referenced by data relating to a particular process step.

The objective of using a customized search engine on locally stored data was to provide document-specific analysis of project data in a rigorous fashion, benefiting from the similarities in the document structure (both in terms of file format as well as link structure). The data being local, it could be massaged into a more parsable format and computationally analyzed without imposing on project Web resources (bandwidth). The Lucene/Nutch tool set was chosen as it was particularly well suited to analyzing large numbers of artifacts of varying types, such as is needed for process discovery, was a mature and well regarded project and because it had an active developer and user community that could be consulted if we required technical advice.

With approval from the NetBeans community, we were able to collect project data by crawling their website. Data from Apache was obtained from the source code repository, which also versions documents on the project website. Both methods have advantages and disadvantages. The web crawl had limits set upon it by the project in terms of which pages it could retrieve. Consequently, we were unable to crawl pages from the mailing list archive, one of the most salient data sources. Unlike content retrieved from versioning repositories, however, crawled documents contained project web database content, which was not available for pages retrieved from versioning repository snapshots. Repository snapshots, on the other hand, provided a greater range of document types that we could parse with greater precision. Crawled data were mostly HTML files. We grew concerned that the disparity in available data from the two projects would hinder our ability to perform truly comparative case analysis. As with all data snapshots, there were also issues of up-to-datedness of the data and its storage to consider.

The ability of search technology to assist in retrieval of process evidence presented a second problem: noisy data. Put simply, search results are imperfect. Only a fraction of the returned results contain useful process evidence. Search results also included duplicates: multiple artifacts depicting the same process evidence. Although multiple indicators of process evidence can reinforce process model validity, we came across some cases of data duplication that, if undetected, could have misrepresented the process (e.g. the same document being stored in multiple places in the repository). Perhaps worse than duplicate data, process models are affected by missing or externally located (outside a project repository) data. Gaps in process data, whether due to technical restrictions on data collection such as those described above or otherwise, affect process participants as much as process researchers. As the OSS phenomenon has caught on, more and more organizations have begun contributing to OSS projects, leading to breakdowns in decision-making transparency and questions of project leadership and control [7, 8].

Unlike projects in the FLOSSMOLE and FLOSSMETRICS repositories, Apache and NetBeans each use separate infrastructure for their project repositories. While certain document types (e.g. mbox email archives and message forums) offer useful metadata that can significantly enhance analysis precision (for example, by providing date fields useful for process step ordering), this precision comes at a cost. We had hoped to avoid constructing analysis techniques specific to particular repository document and development artifact types. The volatility of document structure meant that any classifying scheme created would only be applicable to one type of community repository infrastructure. Even within a community, this infrastructure is not guaranteed to be static. Considering the breadth of artifact types, any such classification scheme would be both complex and fragile.

The fragility of project artifact formats is a known problem for those working with OSS repositories, as reported by maintainers of both the FLOSSMOLE repository and the SourceForge Research Data Archive (SRDA) at Notre Dame [9] at a recent workshop on Free/OSS repositories and research infrastructures [10]. The issue is more poignant for analysis of data from disparate infrastructures. It became clear that creating document type-specific analysis was necessary in certain cases (to parse mbox archives used by Apache into discrete messages, for example, since the existing parsers either did not provide sufficient granularity or had fallen out of maintenance).

Ultimately, the method's greatest promise- deep analysis of project artifacts- proved too difficult. We simply lacked the resources in our small project team to perform in-depth analysis of constantly evolving data sets spanning multiple document types. This led us to a more lightweight solution. We discuss this solution next.

3. PROCESS DISCOVERY WITH FIREFOX AND ZOTERO

Our current strategy for providing computer support for process discovery returns to using each project's own search engine to locate process information. We have operationalized the reference model as an OWL ontology with the Protégé ontology editor [11], using only the OWL class and individual constructs to store process concepts and their associated search queries respectively. Secondly, we built a Firefox plugin, Ontology [12], to display the reference model ontology in the Firefox web browser. Next, we enlisted the Zotero citation database Firefox plugin [13] to store process evidence elicited from project data, integrating the two plugins such that each datum added to the citation database from a project artifact is automatically tagged with the selected reference model entities.

The use of a citation database as a research data repository may seem unintuitive. Zotero, however, has proven well suited for our needs. Like many Firefox plugins, Zotero can create records simply from highlighted sections of a web document, though the creation of arbitrary entries (not gleaned from document text selections) is also possible. It can also save a snapshot of the entire document for later review, which is useful given the high frequency of changes of some web documents- changes that evidence steps in a software processes. The tag, note, and date fields for each entry are useful for recording reference model associations and memos about the entry for use in constructing process steps and ascertaining their order.

The plugin integration greatly facilitates the coding of process evidence and provides traceability from raw research data to analyzed process models. As the tool set is browser-based, it is not limited to analysis of a particular data set, whether local or remote. Moreover, the tool set does not limit users to a single ontology or Zotero database, thereby allowing users to construct research models using multiple ontologies describing other (e.g.

non-OSS process) phenomenon and reuse the tool set for analysis of additional data sets. Thus, it may be easily appropriated for grounded theory research in other fields of study.

The elicitation of process evidence is still search driven. Rather than use one highly customized search engine for all examined data repositories, the search task has been shifted back to the organizations of study. This decision has several implications in comparison with the previous approach, both positive and negative. Using an organization's own search engine limits our ability to extract document-type specific metadata, however among the organizations we have studied, their search tools provide greater coverage of document and artifact types than Lucene handled at that time. Furthermore, this approach does not suffer the data set limitations imposed by web crawler exclusion rules. The ability to query the data set in a scripted fashion has been lost, yet some scientists would see this as a gain. The use of computer-assisted qualitative data analysis software (CAQDAS) historically has put into question the validity of both the research method and results [14,15].

We have only just begun exploring Zotero's capability as a research data repository, yet already there is much promise for further advancement. Zotero's design offers hooks into the database and triggers for database manipulation actions (e.g. adding a record to the database). We used this hook to tag records with reference model information when new records are added. Yet, plugins could use these triggers and hooks to externally manipulate Zotero data to insert additional metadata into Zotero records, provide advanced qualitative and quantitative analysis beyond the built-in timeline functionality. One such advancement we have discussed is automatic highlighting of reference model concepts and date/time indicators in web document text with the option to add such references to the Zotero database. This operation would provide further relief for dealing with the large number of project artifacts in need of analysis.

Zotero is an actively developed project and there is ongoing work to add features such as distributed database access, enabling users in multiple locations to edit a shared Zotero database. Resource sharing, in particular, enables Zotero to transition from being an individual knowledge base to a community knowledge base for researchers, OSS project members, and prospective project members. Lastly, Zotero is an open source project, itself, and may be modified accordingly.

4. CONCLUSIONS

The WOPDASD workshop has sought tools to obtain data from multiple data sources. In developing computer support for OSS process discovery, we have examined various tradeoffs trying to examine the breadth of artifacts used with depth of analysis. Our first attempt required much more computer-assistance than was feasible. Our current research direction poses a compromise between computer assistance and human interactivity. We do not believe it to be a perfect balance between computer assistance and human interaction. Nevertheless, it has shown itself both useful and usable in decreasing the effort required to discover and model OSS processes from multiple project data sources and provides a basis to improve upon.

5.REFERENCES

[1] P. Feiler and W. Humphrey. Software Process Development and Enactment: Concepts and Definitions. Second International Conference on the Software Process: Continuous Software Process Improvement, 1993, 28-40

[2] Jensen, C. & Scacchi, W. Guiding the Discovery of Open Source Software Processes with a Reference Model Third IFIP International Conference on Open Source Systems, 11 June, 2007. Limerick, Ireland.

[3] Jensen, C. and Scacchi, W., Simulating an Automated Approach to Discovery and Modeling of Open Source Software Development Processe, Workshop on Software Process Simulation and Modeling (ProSim03), Portland, OR,

[4] Scacchi, W. Free/Open Source Software Development: Recent Research Results and Methods, in M.V. Zelkowitz (ed.), Advances in Computers, 69, 243-295, 2007.

[5] Becker-Kornstaedt, U. F. Bomarius, S. K. (ed.) Towards Systematic Knowledge Elicitation for Descriptive Software Process Modeling. Proceedings of the International Conference on Product Focused Software Process Improvement Kaiserslautern, Germany, September 10-13, 2001 Lecture Notes in Computer Science, Springer, 2001, 2188, 312-325

[6] The Apache Nutch Project, available online at http://lucene.apache.org/nutch [last accessed 23 June, 2008]

[7] Jensen, Chris, Scacchi, Walt, Collaboration, Leadership, Control, and Conflict Negotiation in the NetBeans.org Software Development Community, Hawaii International Conference Systems Science, vol. 38, Kona, HI, 5 Jan., 2005

[8] O'Mahony, S. The governance of open source initiatives: what does it mean to be community managed? Journal of Management and Governance, 2007, 11(2): 139-150

[9] SourceForge Research Data Archive, available online at http://zerlot.cse.nd.edu/mywiki/index.php?title=Main_Page [last accessed 23 June, 2008]

[10] NSF Workshop on FOSS Repositories and Research Infrastructures, 11-12 February, 2008, Irvine, CA

[11] The Protégé Ontology Editor Project, available online at http://protege.stanford.edu/ [last accessed 23 June, 2008]

[12] The Firefox Ontology Plugin project available online at http://rotterdam.ics.uci.edu/development/padme/browser/ontology [last accessed 23 June, 2008]

[13] The Zotero Project, available online at http://www.zotero.org/ [last accessed 23 June, 2008]

[14] Bringer, J. D.; Johnston, L. H. and Brackenridge, C. H. Using Computer-Assisted Qualitative Data Analysis Software to Develop a Grounded Theory Project Field Methods, 2006, 18(3): 245-266

[15] Kelle, U. Theory Building in Qualitative Research and Computer Programs for the Management of Textual Data Sociological Research Online, 1997, 2(2) available online at http://www.socresonline.org.uk/socresonline/2/2/1.html [last accessed 23 June 2008]