Guiding the Discovery of Open Source Software Processes with a Reference Model

Chris Jensen and Walt Scacchi
Institute for Software Research, Bren School of Information and Computer Sciences, University of California. Irvine, CA 92697-3440 USA
{cjensen, wscacchi}@ics.uci.edu
WWW home page: http://www.ics.uci.edu/~{cjensen, wscacchi}

Abstract. This paper describes a reference model for open source software (OSS) processes and its application towards discovering such processes from OSS project artifacts. This reference model is the means to map evidence of an enacted process to a classification of agents, resources, tools, and activities that characterize the process.

Keywords. Reference model, open source, process discovery

1 Introduction

OSS community observers and novice members often face the challenge of determining how the community works and, equally important, how to properly contribute. Process discovery aims to answer these questions. As researchers and individuals wishing to understand and/or participate in OSS communities, process discovery can be a time consuming task as each community has its own way of doing things. Much of this time is spent collecting and coding evidence, as in any qualitative data analysis. As with errors committed early in the software development lifecycle, data miscoded early in the overall research process has a high cost. We use a reference model based approach for process discovery to assist in coding process evidence to reduce the risk and the cost associated with such coding errors, as well as to more completely articulate discovered OSS processes [1, 2, 3]. The discovery and codification of OSS projects is also a prerequisite to continuous improvement of these processes.

Our reference model serves as a guide to coding process data collected from project artifacts. However, as our reference model is based on a process meta-model, [4], the reference model can serve as a guide to help in coding and modeling OSS processes in forms suitable for automated analyses [9]. Our hypothesis is that a reference model-based approach provides a systematic approach to the problem, giving us consistent and satisfactory results when applied to multiple projects and different types of processes.

In the remainder of this paper, we define a reference model for OSS processes and demonstrate its use in guiding process discovery. We give examples of its usage and conclude with further research opportunities.

2 Reference Model Composition

Our reference model provides a mapping between process evidence discovered by searching the project Web and a classification scheme of process attributes. To do this, we must to determine what aspects of the process we wish to discover. These can be defined by our process meta-model. This meta-model is neither specific to our domain (OSS development processes), nor software processes. The meta-model we use is that of Mi and Scacchi [4]. It establishes a vocabulary of process modeling objects, attributes, relations, and constraints that describe processes. This meta-model defines processes by hierarchical decomposition and by precedence (partially ordered resource flows): processes are composed of tasks (sets of related actions) and atomic actions. Each action is defined in terms of the following process entities: agents in different roles participate in the process activity using tools that require (consume) or provide (produce) resources (artifacts). Tool-based actions are coded using tool invocation scripts/methods that may contain narrative description[s] of the action, or “HTML markup specifying a form to be completed as part of the task. We find these to be a minimal set of concepts necessary to describe a process [9, 10]. This meta-model is augmented with control-flow grammar in the process modeling language we use to formally represent software processes, showing the order in which activities are instantiated [cf. 9].

In practice, we have a dictionary of terms observed in these projects, correlated with known associations from the taxonomy of actions, agents, tools, and resources we constructed based on the original case studies. Tool invocation scripts have been left to downstream process discovery tasks due to their complexity.

Whereas existing ontologies for OSS development (e.g. what Simmons and Dillon [5] presented at OSS 2006 and Rothfuss's [6] framework for open source projects) provide a classification of OSSD concepts, they lack mappings to known process entity instances required to discover software processes. It is worth noting that an individual term in our dictionary may correspond to several different types of process entities in the taxonomy, both within a main branch (e.g. tool) and across main branches (e.g. tool or resource) depending on the context of the term's usage. As an example, consider the term “report” in Figure 1.

```
Term: report
  Known Actions: testing, defect reporting, logging
  Known Agents: none
  Known Resources: whitepapers, test results, web
     server logs, defects, feature requests
  Known Tools: defect repository, test suite
```

Figure 1. Example reference model mapping

There are many OSS artifacts that encode process information and their presence varies by community. Some of the more common artifact include webpages [10], chat transcripts [7], defect reports, source repositories, development and community infrastructure tools [8], and development resources, including process fragment descriptions (e.g., How-To guides, FAQs, etc.) [7]. These artifacts encode data useful for process discovery in four dimensions: s*tructure* (how project-related software development artifacts are organized), *content* (types of artifacts and information they contain), *usage patterns* (user interaction within the community web), and *update patterns* (content updates, including initial creation and deletion).

OSS artifacts vary along these four dimensions over time, and this variance is the cause or consequence of process events. By observing patterns of patterns of reference model attributes within a process iteration and how these patterns change across iterations, we can discover different types of processes and their evolution over the life of a project. We give some examples of this in the next section where we discuss our experiences in applying our reference model in the course of process discovery.

3 Experiences in Reference Model Based OSS Process Discovery

We have applied this technique to discovery of three different types of processes in and around the NetBeans IDE, Apache HTTPD server, and Mozilla projects for a total of seven case studies. We have framed our research with the perspective of an observer or would-be contributor [9] and have, thus, limited ourselves to data that is publicly available via their respective online project portals. Consequently, we have found that access to all four artifact dimensions that encode process data has been limited, especially usage patterns. These limitations have varied across projects, but also between artifacts within a project. Although our resulting process models may have been more precise if not for these limitations, each study, nevertheless, yielded an overwhelming amount of data to examine. In this section, we describe how we have used our reference model in each of the three types of processes we looked at.

3.1 Development Processes

Surveys of development processes in Apache, Mozilla, and NetBeans seeded our initial reference model. These studies examined the requirements and release processes in NetBeans, the Apache server's release process, and the Mozilla testing cycle, circa 2002. The processes were discovered without an explicit reference model, but rather drew on our knowledge of the process meta-model [4] and the projects under study. From these studies we constructed a taxonomy of types of agents, tools, resources, and actions observed and built our reference model dictionary using instances of these entities from project data, as discussed above. Successive process discovery case studies have further enhanced the reference model's precision and relevance in guiding subsequent OSS process discovery,

3.2 Interorganizational Process Communication

In our second case study, we examined interorganizational process communication [10] across several projects (including, but not limited to NetBeans, Apache, and Mozilla) that form a software ecosystem. We said processes that communicate activities or synchronize resources across OSS projects are "*integrative* if they identify compatibilities or potential compatibilities between development projects" (i.e. enables external stakeholders to continue following their internal process as normal) or *conflictive* if "the degree of accommodation or adaptation becomes too great" [10].

In the course of discovering communicating processes, we examined relationships that synchronize resources shared between projects. Some of the most common relationships were integration of tools and libraries produced by one project and used by another and the participation of individuals on several projects, sometimes called linchpin developers [11]. The reference model produced in the course of discovering development processes provided insufficient detail in terms of proper names of specific tools and resources (especially shared libraries) necessary to observe the types of interorganizational relationships that precipitated integration and conflict. Moreover, discovering interorganizational processes requires tracking project specific vocabularies in order to identify instances of process communication between projects. Specifically, we had to denote producers and consumers of specific libraries and development tools, as well as project contributors. As a complicating factor, process communication often occurs outside project web portals, in other channels, both public and private. Some of the richest data available regarding interorganizational process communication between the NetBeans project and the Eclipse IDE project (see http://www.eclipse.org), for example, came from interviews and reports regarding private meetings between the governing body of the Eclipse project and the management of SUN Microsystems, employer of many developers and project leaders for the NetBeans project. Nevertheless, evidence of process communication, and more so the extent to which it was integrative or conflictive, was difficult to observe.

3.3 Role Migration Processes

Our third set of case studies [7] involved observing OSS project participants changing roles. We observed several different tracks of participation, ranging from source code development to community governance and how developers change roles over the course of their participation (or career) within an OSS project.

The key to observing role migration lies in changes in the activity patterns for individual project participants. Using our original reference model as a basis, we saw that the types of actions an individual is associated with have shifted to other branches of the action taxonomy over time. A shift to a closely related branch indicated a migration along the same participation track while a shift to a distant branch indicated a shift to a different track. This technique also allowed us to determine individuals with multiple roles, as well as to detect adoption and abandonment of such roles over time. Using this logic, if many participants show the same migration patterns near the same point in time, we may infer a shift in the participation track-specific process. In this event, we would not be surprised to find shifts in the resources and tools along that track, though may not always be the case.

4 Discussion and Future Work

Becker-Kornsteadt [12] gives a detailed model of process discovery, used in conjunction with the Spearmint project. Abstracting the problem to three stages: data collection, data analysis, and data presentation, the reference model offers only a partial solution to the data analysis stage. After data is coded, it must be assembled into a set of actions (composed of sets of participating agents, tools, and resources required and created by the action, and invocation scripts, as appropriate). Lastly, to produce a process description, this set of actions must be arranged in a fashion representing the (partial) ordering in which they took place [13].

We have looked towards automation to reduce the substantial effort required for process discovery. Automatically coding process evidence is a tantalizing objective, given the advances in machine learning. The example in Figure 1 demonstrates the importance of context in precisely mapping process evidence to classification, suggesting that full automation is either not yet attainable or has a high cost. With this consideration, we have pursued an interactive direction, based on search technology. Our strategy is to query project artifacts using terms from our reference model and, where a term may have multiple mappings, allow the human process discoverer to make a determination as for which is accurate. This approach significantly reduces the effort in locating process evidence within project data and partially reduces the effort associated with data coding. Further, the approach is well suited for artifact-specific analysis and tracking timestamps associated with each document. These temporal signatures are necessary for establishing action-task-subprocess-process composition, as well as partial ordering of events. These efforts are in progress. Other research efforts [14, 15] have sought automated OSS process discovery, though these appear to focus on structure, usage patterns, and update patterns and less on their often semi and unstructured content.

The reference model cannot eliminate process discovery risk altogether, due to the variance in lexica between projects, however its usefulness as heuristic can be improved through updates and process (and project) specific tailoring. In this paper, we discussed using reference modeling in discovering OSS processes. We discussed the composition of such a reference model. We looked at where to find process data to apply the model to. We saw how it was used and evolved over the course of several case studies, and further opportunities for reference modeling to lower the risks and costs of process discovery.

Acknowledgments

The research described in this paper has been supported by grant #0534771 from the U.S. National Science Foundation. No endorsement implied. Mark Ackerman at University of Michigan, Ann Arbor; Les Gasser at University of Illinois, Urbana-Champaign; John Noll at Santa Clara University; Margaret Elliott and others at the UCI Institute for Software Research are collaborators on the research described here.

References

1 S. Koch, S. Strecker, and U. Frank, Conceptual Modelling as a New Entry in the Bazaar: The Open Model Approach. *Proc. Second Intern. Conf. on Open Source Software*, 8-10 June, 2006 Como, Italy (Eds) E. Damiana, B. Fitzgerald, W. Scacchi, and M. Scotto, p. 9-20

2 J. vom Brocke and C. Buddendick, Reusable Conceptual Models - Requirements Based on the Design Science Research Paradigm, *1st Intl. Conf. on Design Science Research in Information Systems and Technology*, Poster Paper, Eds: A. Hevner, Claremont, CA, USA, 24-25 Feb, 2006.

3 C. Jensen and W. Scacchi, Applying a Reference Framework to Open Source Software Process Discovery, in *1st Workshop on Open Source in an Industrial Context*, Anaheim, CA October 2003.

4 P. Mi, and W. Scacchi, A Meta-Model for Formulating Knowledge-Based Models of Software Development, *Decision Support Systems*, **17**(4), 313-330 (1996).

5 G. Simmons and T. Dillon, Towards an Ontology for Open Source Software Development. *Proc. Second Intern. Conf. Open Source Software*, 8 June, 2006 Como, Italy (Ed) E. Damiana, B. Fitzgerald, W. Scacchi, and M. Scotto, p. 65-76

6 G. Rothfuss, A Framework for Open Source Projects. Master Thesis in Computer Science, Department of Information Technology, University of Zurich, 2002.

7 C. Jensen and W. Scacchi, Process Modeling Across the Web Information Infrastructure. Software Process: Improvement and Practice, Special Issue on ProSim 2004, 10(3), 255-272 (2004).

8 T. Halloran and W. Scherlis, High Quality and Open Source Software Practices, *2nd Wrkshp. on Open Source Software Engineering*, Orlando, FL, 25 May, 2002.

9 W. Scacchi, Understanding the Requirements for Developing Open Source Software Systems, *IEE Proceedings- Software*, **149**(1) 25-39 (2002).

10 C. Jensen and W. Scacchi, Process Modeling Across the Web Information Infrastructure. Software Process: Improvement and Practice, Special Issue on ProSim 2004, 10(3), 255-272 (2004).

11 G. Madey, V. Freeh, and R. Tynan, Modeling the F/OSS Community: A Quantitative Investigation, in S. Koch (ed.), *Free/Open Source Software Development*, (Idea Group Publishing, Hershey, PA. 2005), pp. 203-221.

12 U. Becker-Kornstaedt, Towards Systematic Knowledge Elicitation for Descriptive Software Process Modeling, *Intl. Conf. on Product Focused Software Process Improvement,* Kaiserslautern, Germany, 10 Sep, 2001 (Ed) S. Bomarius, Lecture Notes in Computer Science, Springer, 2188, pp. 312-325 (2001)

13 P. Feiler and W. Humphrey, Software Process Development and Enactment: Concepts and Definitions. *2nd Intl. Conf. on the Software Process: Continuous Software Process Improvement*, 28-40 (1993)

14 R. Sandusky, Software Problem Management as Information Management in a F/OSS Development Community. *Proc. 1st Intl. Conf. on Open Source Systems*, 11-15 Jul, 2005, pp. 44-49

15 Y. Liu, E. Stroulia, and H. Erdogmus, Understanding the Open-Source Software Development Process: A Case Study with CVSChecker. *Proc. 1st Intl. Conf. on Open Source Systems*, 11-15 Jul, 2005, 154-161