Automating the Discovery and Modeling of Open Source Software Development Processes

Chris Jensen and Walt Scacchi
Institute for Software Research
University of California, Irvine
Irvine, CA USA 92697
{cjensen, wscacchi}@ics.uci.edu

Keywords: Automated Process Discovery, Process Modeling and Simulation, Open Source Software Development

Introduction

The goal of our work is to develop new techniques for discovering, modeling, analyzing, and simulating software development processes based on information, events, and contexts that can be observed through public information sources on the Web. Our domain examines processes in open source software development (OSSD) projects, such as those associated with the Apache Web server, Mozilla Web browser, and interactive development environments like NetBeans and Eclipse. In our previous work, we demonstrated the ability via manual search and analysis methods to discover (fragments of) process workflows in projects like NetBeans [9] by analyzing the content of their web information spaces, including informal task prescriptions, community structure and work roles, overall project organization, product histories, and communications among community members.

Though this approach netted a wealth of information with which to model, simulate, and analyze OSSD processes, it suffers from a lack of scalability when applied to the study of multiple OSSD development projects and suggests the need for an automated approach to that can more readily facilitate process discovery and modeling.

In our approach, we have been identifying what kinds of OSSD artifacts [12] (source code files, messages posted on public discussion forums, Web pages, etc.), artifact update events (version release announcements, Web page updates, message postings, etc.) and work contexts (roadmap for software version releases, Web site architecture, communications systems in use (email, forums, instant messaging, etc.)) can be observed, detected, extracted, or inferred through automated tools operating across the Web. Though such an approach clearly cannot observe the entire range of software development processes underway in an OSSD project, it does draw attention to what can be publicly observed and modeled at a distance. This is the same challenge that prospective developers or corporate sponsors who want to join a given OSSD project face. As such, we have been investigating what kinds of processing capabilities and tools can be applied to support the automated discovery and modeling of software processes (e.g., for daily software build and periodic release) that are found in many OSSD projects. The capabilities and tools include those for Internet-based event

notification, Web-based data mining and knowledge discovery, and previous process discovery tools.

Previous Work

Event notification systems have been used in many contexts [5, 14]. However, of the systems promising automated event notification, many require the process actant to obtain, install, and use applications on their own machines to detect when events occur. While yielding mildly fruitful results, this approach is undesirable for several reasons, including the need to install and integrate remote data collection mechanisms with local software development tools. Prior work in process event notification has also been focused on information collection from command shell histories, applying inference techniques to construct process model fragments from event patterns, which imparts additional inconvenience on the user and relies on his willingness to use the particular tools that observe events. By doing so, the number of process actants for whom data is collected may be reduced well below the number of participants in the community due to privacy concerns and the hassles of becoming involved.

While process research has yielded a plethora of views of software process models, none has yet been proven decisive or clearly superior. Nonetheless, contemporary research in software process technology, such as Lil Jil [2] and PML [8] argues for graphical or visual navigation representations of software processes. While graphical process representation schemes such as Petri Nets date back to the 1970's [10] with a wide variety of activity and state-chart diagrams in between, it appears they still suffer from a lack of scalability when applied to a process of any reasonable complexity.

Cook [3, 4], utilized both algorithmic and statistical inference techniques with an ultimate goal was to create a single, monolithic finite state machine (FSM) representation of the process. However, it is not entirely clear that a single FSM is appropriate for modeling complex processes. While admitting that some massaging of the event-stream data was necessary to eliminate spurrious events not pertaining to the process in focus, Cook notes the robustness of the Markov model in the face of noisy data. Realizing that the Markov model does not account for previous events in the determination of the next state, Cook experimented with Bayesian Markov transformations using conditional probabilities. Although he disregarded the results as insignificant in comparison to those obtained without the extension, Cadez [1] points out that joint probabilities tend to be more informational.

Approach

Our approach is to obtain process execution event streams by monitoring open source development web information spaces. By examining changes to the information space, we may be able to infer process activities. Work such as WebQuilt [6] demonstrates that such information may be gathered in ways that are unobtrusive to process participants and that minimize the invasiveness found with many autonomous agent based event notification systems. These capabilities enable the capture of event streams from multiple iterations of a process or task [6]. Using data mining and knowledge discovery techniques, we then identify and extract process fragments. In turn, we reconstitute process instances using xPADL, a process architecture [13] description language based on XML and PML [8]. xPADL provides us with a formal description of each process instance which can be transformed (generalized) to realize an enactable model of the process in question, that can then be employed with a process simulator [8, 13].

Modeling and Simulation

Our aim in applying probabilistic techniques to process enactment modeling is an acknowledgement that there is often variation within the instantiation of processes actions, whether due to constraints of a given project or merely a developer's changing preference of a selection of tools, both of which may impact not only the way an action is performed, but the types of actions that are performed. These dependencies and their consequences are taken care of by accounting for, not only the probability of enacting an activity given the action completed immediately prior, but also then entire chain of actions leading up to it. Eliminating chains with an execution rate below a given threshold gives us a basis for simulating action sequences that reflect this variability.

Leveraging existing OSS object modeling tools such as Protege-2000 [11, 7], we were able to develop a xPADL editor and (manual) modeling environment, that can produce xPADL, XML or SQL as its outputs. These tools were successfully used to model software processes observed in a sample of OSSD projects, including the NetBeans project [9]. Subsequently, we believe that outputs such as these can be hand-fed into existing process simulator tools we have previously investigated [8,13].

We are now investigating how best to apply probabilistic data mining and knowledge discovery techniques to deduce likely process fragments, as well as to detect events or updates that may reveal where the current process enactment has failed. Likewise, we are examining how XML data binding tools like Protégé can provide a straightforward segue from our xPADL process descriptions to existing or new process simulation capabilities.

Acknowledgements

The research described in this report is supported by grants from the National Science Foundation #IIS-0083075 and #ITR-0205679. No endorsement implied. Contributors to work described in this paper include John Georgas, who tailored the Protégé tool for use in software process modeling. Mark Ackerman at the University of Michigan Ann Arbor; Les Gasser at the University of Illinois, Urbana-Champaign; John Noll at Santa Clara University; Margaret Elliott, Chris Jensen, Mark Bergman, and Xiaobin Li at the UCI Institute for Software Research; and Julia Watson at The Ohio State University are also collaborators on the research project described in this paper.

References

[1] I. V. Cadez, D. Heckerman, C. Meek, P. Smyth, S. White. Visualization of Navigation Patterns on a Web Site Using Model Based Clustering, *Proceedings of the KDD 2000*, pp 280-284

[2] Cass, A.G., Lerner, B.S., McCall, E.K., Osterweil, L.J., Sutton, Jr., S.M. and Wise. A. Little JIL/Juliette: A process definition language and interpreter, in *Proceedings of the 22nd International Conference on Software Engiineering* (Limerick, Ireland, June 2000), 754-757

[3] Cook, Wolf, Automating Process Discovery through Event-Data Analysis, *Proceedings of the 17th International Conference on Software Engineering*, Seattle, Washington, USA, April 1995, pages 373-386.

[4] Cook, Jonathan, *Process Discovery and Validation through Event-Data Analysis*, Ph.D. thesis, Computer Science Dept., University of Colorado, 1996

[5] David M. Hilbert and David F. Redmiles. An Approach to Large-Scale Collection of Application Usage Data Over the Internet, Technical Report UCI-ICS-97-40, Department of Information and Computer Science, University of California, Irvine, September 1997.

[6] Jason I. Hong, Jeffrey Heer, Sarah Waterson, and James A. Landay. WebQuilt: A Proxy-based Approach to Remote Web Usability Testing To appear in *ACM Transactions on Information Systems*.

[7] Natalya F. Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Fergerson, and Mark A. Musen. Creating Semantic Web Contents with Protégé-2000, *IEEE Intelligent Systems*, March/April 2001

[8] J. Noll and W. Scacchi, Specifying Process Oriented Hypertext for Organizational Computing, *Journal of Network and Computer Applications*, 2001 (24) 39-61

[9] M. Oza, E. Nistor, S. Hu, C. Jensen, and W. Scacchi. A First Look at the Netbeans Requirements and Release Process, available at http://www.ics.uci.edu/~cjensen/papers/FirstLookNetBeans/

[10] Peterson, James L. Petri Nets, *ACM Computing Surveys* (CSUR) September 1997, Volume 9 Issue 3

[11] Protégé Web site: http://protege.stanford.edu/

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

[13] J.S. Choi and W. Scacchi, Modeling and Simulating Software Acquisition Process Architectures, *Journal of Systems and Software*, 59(3), 343-354, 15 December 2001.

[14] Alexander L. Wolf and David S. Rosenblum. A Study in Software Process Data Capture and Analysis, *Proc. Second International Conference on the Software Process*, IEEE Computer Society, Feb. 1993, pp. 115-124.