

Applying Micro Process Analysis to Global Software Development

Shuji Morisaki and Hajimu Iida

Graduate School of Information Science, Nara Institute of Science and Technology

Abstract

We propose applying micro process analysis method to ongoing distributed software development. Micro process analysis firstly identifies process instances by matching logs of software development tools with pre-defined micro (fine-grained) process models. Micro process metrics can be measured from identified process instances. In distributed or multi-sited software development environment, micro process metrics provide project manager with more visibility to precise process execution and also to precise monitoring.

1. Introduction

In multi-site software development, low visibility to the development process easily lead to low software quality or delivery slippage. Though methods and tools to visualize product status and change histories in distant site have been proposed[1][3], such practices are mainly focus to product itself while procedural aspects were not sufficiently studied in realistic situations.

We are proposing retrospective analysis of the fine-grained process as *micro process analysis (MPA)*. This paper proposes applying the micro process analysis to ongoing distributed, or multi-sited, software development in order to provide visibility and opportunities to ask accurate process executions and monitoring. We applied our methods to a commercial multi-sited project data and extracted *process instances* (process fragments identified in an activity log).

2. Micro process analysis

MPA is performed over the activity log, which is a series of events collected through execution of software development. An event is just assumed to be collected and recorded automatically by software development tools, so that every event has a type associated to its data source, such as change management log of intermediate and delivered product. We assume to use log data collected by Project monitoring environment EPM [2]. EPM provides histories of source code revision, bug tracking and

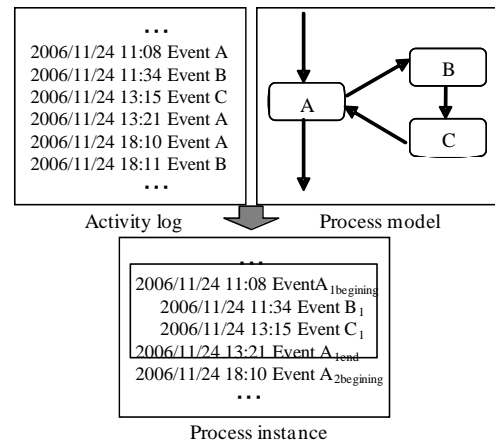


Fig. 1 Process instance extraction from process model and activity log

communications by emails without specific modification to each tool.

We represent a software process as a *grammar* of activity sequences that should be used to parse activity log. By parsing activity log with a process model, matching activity sequences are identified as instances of the specified process. Fig.1 shows an example of an activity log, a software process model described in activity diagram, and identified process instances. The model describes that activity A involves sequential sub-activities B and C. A process instance, starting at 2006/11/24 11:08 and ending 13:21, is identified.

Micro process metrics, such as number of activity omissions or incorrect sequence, can be obtained from identified instances. They are useful for evaluating quality of the process as well as the quality of the logging itself.

3. Analyzing distributed on-going projects

Micro process analysis also provides valuable opportunities to observe and improve remotely distributed processes at low cost. Fig. 2 shows the procedures of on-time analysis in distributed projects. First, project manager and development sites agree to follow the prescriptive process models. Agreed process models are distributed to each site. Second, each site collects activity log and sends them to the project manager periodically. Process instances are extracted

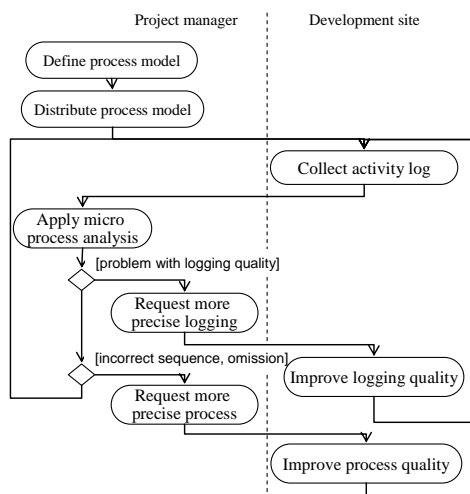


Fig. 2 Activity diagram of multi-site process analysis
 from those activity logs. If project manager recognizes degrade of process quality such as omissions of activity, or incorrect sequence, s/he can ask the site members to refer the specified process model, or s/he may reconsider the process model itself. Project manager can also recognize degrade of collected activity log, e.g. many logs are recorded at one time (not on-time), or many mismatch of begin/end of the activities are found. In such cases, manager should check tool settings or operational prescription for activity logging in the target site.

3 Example Study

As an example, we applied the micro process analysis to a part of logs of completed commercial software development project that follows traditional water fall model. The software was developed by five companies in six distributed development sites. The activity logs collected from one of those sites is used for MPA. Process model was preliminary defined as follows and committed by those companies: When a developer finds a bug, s/he registers statements of bug. A unique identifier is automatically assigned to the bug and status is set to be “get started” by GNATS, a bug tracking tool. The assigned developer checks out source code from repository to fix the problem if necessary. Then developer locates and corrects the defect and confirms the fix by testing. If the developer decides that the bug is fixed, s/he checks in the modified source code to the repository. The bug identifier is manually specified as a comment of checking in. Finally the bug status is set to “resolved” using GNATS.

Fig. 3 is a visualized summary of the process instances generated by our extraction tool. Vertical axis and horizontal axis represents date time and bug identifier

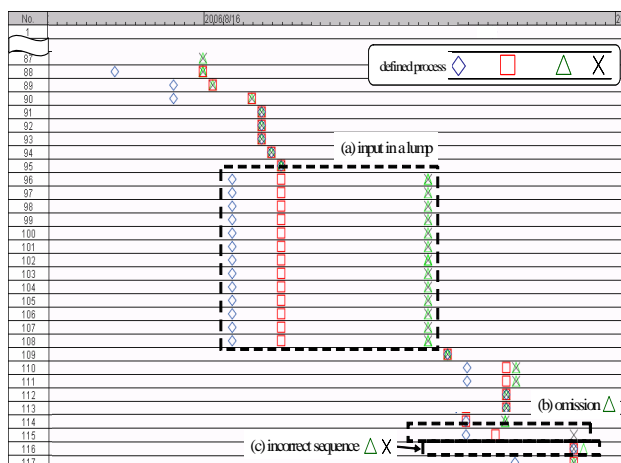


Fig. 3 Visualized summary of identified process instances

respectively. The process instances are plotted according to bug ID and date of the start time. Events in each process instance are plotted according to time stamp. Diamonds, squares, triangles and Xs represent the steps in the debugging process.

4. Conclusion

We confirmed that micro process analysis can extract process instances and measure micro process metrics such as omissions of activity or incorrect sequence. We expect that process instances can be extracted and visualized during project is in progress.

And we believe that extracting and visualizing process instances lead to increase visibility and opportunities to improve quality of process executions and logging especially in distributed software development.

Acknowledgement

Part of this research is supported by the Comprehensive Development of e-Society Foundation Software program of the Ministry of Education, Culture, Sports, Science and Technology of Japan. The authors would like to thank members of EASE project and members of COSE project.

1. Froehlich J. and Dourish P., “Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams,” Proceedings of International Conference on Software Engineering, pp.387-396 (2004)
2. Ohira M., et al., “Empirical Project Monitor: A Tool for Mining Multiple Project Data,” Proceedings of International Workshop on Mining Software Repositories, pp. 42-46, (2004)
3. Zimmermann T., et al., “Mining Version Histories to Guide Software Changes,” Proceedings of International Conference on Software Engineering, pp.563-572, (2004)