Micro Process Adherence for Delivering Reliable Software

Vibhu Saujanya Sharma and Vikrant Kaulgud
Accenture Technology Labs, Bangalore, KN, India.
{*vibhu.sharma, vikrant.kaulgud*}*@accenture.com*

***Abstract*—With quality and timely delivery of software becoming ever more vital, ensuring that development processes for building high quality code are adhered to has become important. However the problem project managers face is in verifying if developers are actually adhering to the recommended processes pertaining to activities they perform at their workstations, which can be termed as micro processes. This lack of insight often leads to developers bypassing the development micro processes thereby reducing the quality and reliability of the checked-in code. We present an approach to provide enhanced visibility to the project manager based on capturing the occurrence of relevant activities performed in the developer IDEs and related tools and then analysing the set of events to identify the deviation of the pattern of these set of events from a defined model of micro-processes. We discuss our current prototype implementation and the benefits that we foresee from this work.**

I. Introduction

Software delivery projects use fine-grained processes or micro processes standardizing code reviews, unit testing, code check-in best practices, etc. Definition, deployment and adherence of development micro-processes decide the quality and reliability of software delivery artefacts. Fine-grained activities captured in a micro-process are not amenable for manual reporting, since the sheer number of these activities make manual reporting very time consuming. Today, there is inadequate support for project managers to define such micro processes and automatically check their adherence.

To illustrate the problem, we take an exemplary micro process - "Developer should run a (project-mandated) quality tool and resolve all *Severity-1* issues before code check-in". In a crunch situation, non-adherence of this micro process would result in checked-in code of inferior quality and code review might not happen in a stringent manner. Thus, defects will slip through into production code thereby increasing the failure rate. Since the developer is running the quality tool as part of his development work, he cannot possibly submit a tool report for every tool run. In absence of such reports, how then can the project manager be assured of micro process adherence and be sure of project outcomes?

Here, we briefly introduce the Micro Process Adherence System (MPAS). MPAS allows definition of micro processes spanning multiple development activities and multiple development tools. MPAS aims to provide a mechanism for monitoring micro adherence and for correlating it with project metrics as well as providing developer guidance.

II. Monitoring Micro Process Adherence

MPAS uses a DSL based approach for micro process definition. MPAS DSL uses 'parameterized activity phrases' such as "Run <Tool>" wherein the project manager instantiates the phrase through selecting an actual and supported development tool. For example, in the above exemplary micro process, the instantiated phrase could be "Run PMD", where PMD [1] is well-known Java code quality tool. MPAS DSL supports activity phrases related to typical development activities of code quality assessment, unit testing, code editing, code check-in and manual work activities like code reviews. Furthermore, the MPAS DSL uses temporal sequencing phrases such as "Before" and logical connectivity phrases such as "And" for weaving together the parameterized activity phrases into a micro process definition.

The deployment of a micro process is done using a state-machine. Since the development activities are discrete events, it is intuitive to view a micro process definition as a state-machine. Typical state-machine analytics provide statistics on micro process execution and permit checking of temporal sequencing constraints and state omission constraints. It is possible to focus on two kinds of micro processes: developer-centric and environment-centric. In developer-centric micro processes, only developer telemetry is used. For this a state machine approach provides a lightweight deployment model wherein MPAS is installed on each developer's machine providing real-time feedback and guidance to the developer. For environment-centric micro processes such as those using work-item trackers, MPAS is installed centrally. So a state machine approach is applicable for both distributed as well as centralized adherence checks. Finally, state machines are good for micro process discovery as well.

For automated adherence monitoring, event data is collected in a non-invasive and automatic manner from developer workstation. A similar approach has been used in Hackystat project [2]. As shown in figure 1, the Micro Process Observer co-ordinates this event collection from the development environment into a central event repository. Further, using event signatures from the Event Characteristics database, raw event data is filtered to obtain only relevant information. This filtered event data is analyzed using state-machines for micro-process adherence. Subsequently, reports are generated indicating micro process adherence

levels.

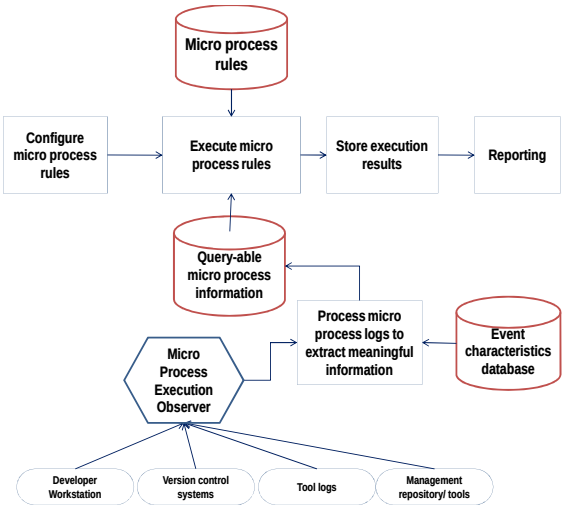

Figure 1. Monitoring Micro Process Adherence

III. Managing Complex Micro Processes

A key aspect of MPA system is the ability to define a conglomeration of activity phrases to create a complex micro process. For example, to achieve the objective of increasing unit testing effectiveness, developers need to write unit tests before a corresponding class is written and they need to run unit tests frequently. This requires a conglomeration of two activity phrases: (i) Unit test-case class has to be created before the corresponding Java Class is created, and (ii) Unit test should be run <frequently> between Java Class edits. With conglomeration, each activity phrase is monitored separately and a combined report is generated. Project manager weights each constituent activity phrase to get a combined score indicating some degree of impact on the stated objective. With this, project managers can continually evolve the complexity of micro process definitions and achieve higher degree of visibility into project dynamics and higher control on project outcomes. Note that conglomeration of activity phrases supports partial adherence levels too.

IV. Discussion and Conclusion

MPAS allows a project manager to monitor adherence to recommended development micro processes resulting in ensuring high fidelity adoption of development micro processes, thereby improving project quality and reliability. This in turn reduces rework, cost as well as reducing schedule variance. From another viewpoint, it allows project managers to quickly identify weak processes and use training / incentives to improve process adherence.

Currently we are engaged in implementing the prototype for MPAS as well as collaborating internally with various project groups to refine and align this effort to be most effective for projects in our organization. This includes collecting, prioritizing and implementing various micro processes that are followed explicitly, or sometimes implicitly, in various projects. Further, we are tracking correlation between micro process adherence levels and project outcomes. This tracking is critical for continuous improvement plans that many projects adopt for improving project efficiency.

Approaches similar to MPAS have been reported in [2] and [3]. While MPAS uses similar principles, it is more focused on modeling and monitoring complex micro processes, real-time feedback and guidance to developers, integration of manual work activities and supporting new tools and repositories like SharePoint.

From the response we have thus far received, MPAS encourages better adherence to recommended micro processes, thus helping them deliver reliable higher quality development artefacts within the stipulated cost and effort.

References

[1] "Pmd," *http://pmd.sourceforge.net/*.

[2] H. Kou, "Ics2005-07-01: Studying micro-processes in software development stream," *Technical Report: Department of Information and Computer Sciences, University of Hawaii*.

[3] F. Schlesinger and S. Jekutsch, "Electrocodeogram: An environment for studying programming," *https://www.mi.fu-berlin.de/wiki/pub/SE/ElectroCodeoGram/lancaster.pdf*.