Using Process Mining Metrics to Measure Noisy Process Fidelity

Chris Thomson, Marian Gheorghe
Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello, Sheffield S1 4DP
{c.thomson, m.gheorghe}@dcs.shef.ac.uk

Background: **When teams follow a software development process they do not follow the process consistently. We need a method to measure their fidelity to that process.** ***Objective:*** **To evaluate Rozinat and Aalst's metrics for process conformance to a state based model on noisy data (Rozinat and van der Aalst, 2008).** ***Method:*** **We instructed 14 teams that were developing a software system using Extreme Programming (XP) to record the events of their project (for example writing code, or testing). We calculated the values of the proposed metrics by comparing the data collected to a process model of XP.** ***Results:*** **13 teams recoded data that we treat as a multiple case study. The fitness metric gave varying results over the teams that corresponded to the number of event types used in the correct order. The appropriateness metrics measured the same values for all teams.** ***Conclusion:*** **The fitness metric is useful for measuring fidelity, but the appropriateness metrics do not measure over fitting well with noisy data. In addition neither metric gave useful information about other aspects like iteration.**

process fidelity, empirical, multiple case study, metric, evaluation

1. INTRODUCTION

There are two main methods that can be explored for the systematic measurement of process fidelity. The first is via a subject reported questionnaire such as Shodan (TR-2003-17, 2003), the second method is via a log of the events that take place during the project. In this paper we will concentrate on the use of a log, and more precisely a log that contains events reported and that does contain errors. The purpose of this investigation is to see if metrics proposed to assess the quality of mined process models can also be used to assess fidelity to process, with data from real software engineering projects.

Many business processes result in well formed sequences of patterns of events that describe various activities required to perform a task. This has spawned process mining as an important research area, as it is normally straightforward to identify patterns of behaviour from logs of this data. Given that we have both a representation of the desired process as a log of events, there are two ways of assessing the fidelity of this process to the prototype model. The first method would be to generate a model from the log using the process mining techniques and then to compare the two models using a state based verification technique. However such a technique will have a high margin of error as the mining technique will have to discard some of the details in the logs to form a consistent model because the data is noisy. The second method is to directly compare the log against the prototype model

Recently a metric has been proposed to evaluate the generated model by replaying the events in a task through a model as represented by a Petri net (Rozinat and van der Aalst, 2008). Clearly these metrics assume that the proposed model bears significant resemblance to the log data, as they are designed to validate models generated by the log data. Therefore if we wish to use them as measurements of conformance to some prototype model then a certain amount of empirical work should be done to check their validity. Hence we collected logs of events for 14 software development teams, calculated these metrics, and compared them to the raw data.

2. BACKGROUND

Processes are frequently expressed as a form of directed transition system (Rubin et al., 2007, Cook and Wolf, 1998, Soto and Münch, 2007). These are composed by events or activities. Approaches to forming these from observations vary based on the type of events captured. Principally there are three observation approaches discussed in the literature: in the first case the developers document the process in detail as they complete it (Leszak et al., 2002) although this is maybe unreliable (Soto and Münch, 2007); secondly a research group observes the team, perhaps recomposing the process by interview (Krishnan and Kellner, 1999, Leszak et al., 2002); lastly the final method involves the analysis of logs post hoc (Colombo et al., 2006, Rubin et al., 2007). In the former instance some knowledge of the process model and the relation between the states is known, whereas in the latter two all that is available is the event series in the timeline. In the later case where the data is reconstructed, business rules must be identified in order to associate a log entry with the activity stereotype.

In order to rebuild the process model, such as to present a typical generic model such as "the waterfall model", there are two principal difficulties (Cook and Wolf, 1998): the first is to identify the exceptional cases where an activity was completed out of the cycle that was more generally followed, and to identify issues pertaining to concurrent work in models where team members follow independent lifecycle iterations. Typically the former issue is dealt with via a pruning technique (selectively ignoring the exceptional cases) (Colombo et al., 2006), whereas

the latter is resolved by following the products through the process with some notion of implicit concurrency either random or through communicating machines (Rubin et al., 2007, Cook and Wolf, 1998). Techniques and tools to address these difficulties are available, but current implementations focus on this specific application, thus they attempt to build a generic model for the whole lifecycle. The literature seems to implicitly favour the fact that a process is fixed over a time period, although techniques for managed process evolution have been discussed (Ahmed-Nacer, 2001) and a method has been presented to identify changes (Soto and Münch, 2007).

Other recent work has sought to measure process conformance, having obtained a sequence of events, by comparing the models obtained from observations to theoretical models. This work is in the early stages having come from the foundations of methods that simply tested conformance. Fitness and appropriateness are proposed as metrics that can be assessed by incrementally replaying the events and measuring unused features of the model respectively (Rozinat and van der Aalst, 2008). Two further metrics precision (how many invalid steps occur) and recall (how many steps are enabled) have also been defined from a machine behaviour perspective (Alves et al., 2008).

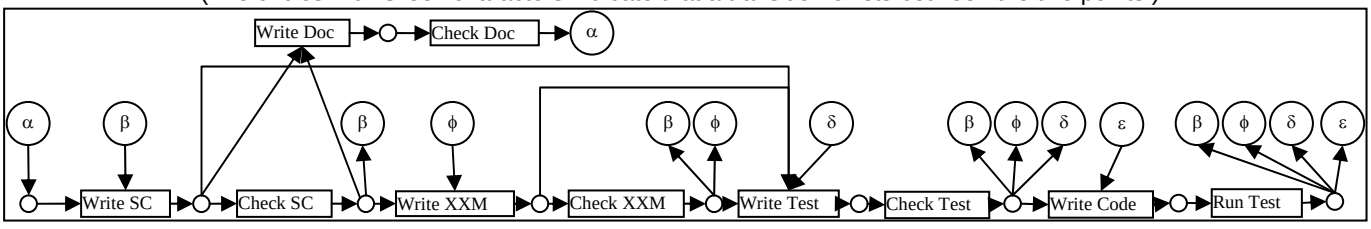
In this paper we focus on the method proposed by Rozinat and Aalst which represents the process model as a Petri net (Rozinat and van der Aalst, 2008). We will refer to the log of a series of events as a task. When the task is executed a token is produced at the start state, then when an event is executed if an edge with the same name is enabled (that is to say there is a token on the preceding state) then the token is consumed and one produced at each connected state. The fact that an edge may lead to more than one state allows for parallel execution. If there is no enabled edge matching the event then one which has the same name is chosen at random and the same process is followed, without the consumption.

The method described by Rozinat and Aalst is two fold based on two metrics fitness and appropriateness (Rozinat and van der Aalst, 2008). Fitness is a measurement of the extent to which the tasks can be fitted to a Petri net taking into account the number of times tokens are produced but not consumed, and the number of missing tokens where they had to be introduced. Appropriateness is a measurement of over fitting, or in other words how much of the process model was not used by the tasks executed. Behaviour appropriateness measures if the model allows too many potential paths. This is achieved by calculating all possible orderings of events in both the model and the log, and discounting those that always or never follow each other. The metric calculates the remaining set (those states which sometimes follow each other) by considering the difference between the model and log sets. The calculation is based on a count of alternate duplicate events and unnecessary invisible events that could be removed without otherwise affecting the behaviour of the model. The alternate duplicate events are those that never happen together in one execution sequence. This measurement is made on the model only and so will not be affected by changes to the logs.

3. METHOD

The data was collected from a study of 14 teams undertaking a software development project within the Sheffield Software Engineering Observatory (CS-09-01, 2009). The teams were assigned clients randomly, with 3-4 teams working with each of the four clients. This corresponds to a randomised complete block experimental design (Ostle and Malone, 1988). The clients wanted a software product to use in their business, and each selected one of the delivered products to put into use. The teams were taught how to use a modified form of Extreme Programming (XP) at the start of the project (Holcombe, 2008). We informed the teams that we would be studying their approach to XP and provided them with a diagram (as shown in figure 1) that showed the process model involved (expressed as a Petri net). We asked the teams to self report the process that they followed by keeping a log of events on their story cards, therefore each story card represented a task. They were permitted to use any of the event names shown in the diagram or add their own.

FIGURE 1: The prototype process model for Extreme Programming.
(The circles with Greek characters indicate that a transition exists between the two points.)



We collected the diary of events from the story cards at the end of the project and processed them by hand into a XML file. We used the ProM toolkit (van der Aalst et al., 2007) provided by Rozinat and Aalst to calculate their metrics for our data following their instructions (Rozinat and van der Aalst, 2008). We calculated the advanced behaviour appropriateness and fitness metrics against the model presented in Figure 1 for each team. As already noted the advanced structural appropriateness metric would return the same value if the model was not changed so for each team if they did not report one of the event types we set this as an invisible event in the model to calculate this metric. The advanced structural appropriateness value for the unaltered model is 1. Lastly in order to interpret these metrics we produced Petri net diagrams for each team that described their process by using the heuristic mining tool in ProM. This algorithm attempts to form a general model of the process followed, excluding exceptional cases (Weijters et al., 2006). In all cases the tools were run with their default settings.

3.1 Threats to Internal Validity

The teams working with different clients developed different software products, but all had equal motivation to follow the process. In order to address the issue of interaction the teams were aware that this was a competition; equally the client was told not to share ideas between teams. To mitigate the effects of the teams' maturing we allowed the members to self select, hence some members in each team had worked together before. The team members may have learnt, altering their behaviour during the course of the observations but we have no obvious way to measure this.

3.2 Threats to External Validity

The developers were novice users of XP although they had completed a previous team software development project. The task was representative of other small web based development projects where there is a development team of 4-6 members spending 120 hours per person. The developers worked at home and in a university laboratory, having access to a range of professional software tools.

4. RESULTS

The calculated metrics for each team's logs are shown in Table 1 below. Team 10 did not record any events and so is not described further in the analysis; their missing data appears to be an oversight as they did deliver a final solution. The table clearly shows three important points, the first is that all the teams took advantage of the orderings of events within the model, as in all cases the model had a behaviour appropriateness of 1. Secondly no team's model had a structural appropriateness of 1; this indicates that the events that were marked as invisible as the teams did not record those events were not necessary to describe their process. Lastly teams 3 and 6 only reported a small number of tasks (1 and 5 respectively) this indicates that they provided a low sample of tasks compared to the other teams and the measurements made on those teams should be treated with caution.

TABLE 1: Calculations of the fitness and appropriateness metrics for each team.

Team	Number of tasks	Fitness	Behavioural Appropriateness	Structural Appropriateness
1	16	0.33	1	0.9
2	14	0.50	1	0.7
3	1	0.42	1	0.6
4	36	0.40	1	0.7
5	18	0.58	1	0.8
6	5	0.62	1	0.7
7	35	0.30	1	0.6
8	26	0.64	1	0.9
9	17	0.47	1	0.9
11	16	0.50	1	0.9
12	27	0.52	1	0.8
13	26	0.40	1	0.7
14	13	0.44	1	0.8

The fitness metric provides the most useful and interesting details, there is a clear differentiation between the teams, suggesting that some teams followed the prescribed method more closely than others. To validate the results we inspected the raw data and the models produced by ProM's heuristic miner three of which are shown in Figures 2, 3 and 4. The differences between the processes of team 8 (Figure 2) and team 7 (Figure 3) are clear. Team 8 recorded instances of most of the events but following a fairly linear path, only iterating once the code was written. In contrast the model for team 7 is more confusing. The model also indicates that other events have been not recorded, including some iteration when writing code. This is explained by the fact that there are instances of "Write code" repeated in some tasks and in others where this is interspersed with "Write story card". The model also shows that the "Check story card" event is out of order; with the team validating it only after the code is complete in some cases. Thus the fairness metric seems a good interpretation of the differences between these models.

FIGURE 2: A model for team 8, found by heuristic mining.

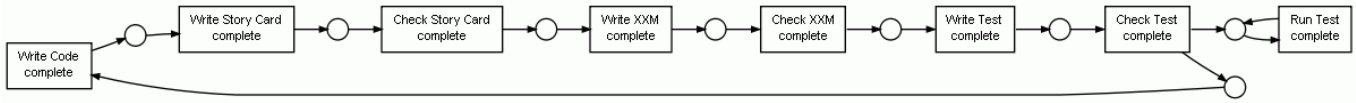
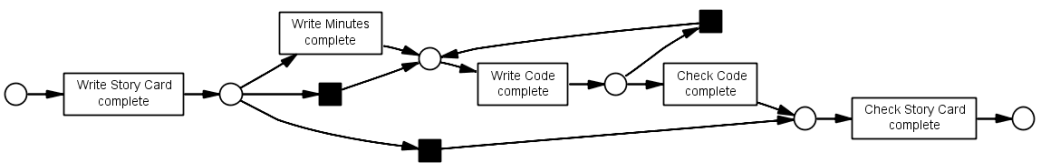


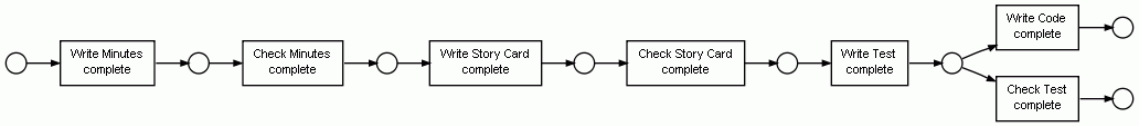
FIGURE 3: A model for team 7, found by heuristic mining.



Team 2 scored in the middle of the observed range, but the model they followed shows a deficiency of these metrics. Team 2 followed most of the basic process in the correct order, but did not have any iteration. If we were

measuring a waterfall type process this might not be important, whereas an iterative function is an important part of agile design. This might be due to a limitation of the behavioural appropriateness metric, as the noisy nature of the data can produce examples of uncommon orderings for a team, thus causing the metric to tend towards 1.

FIGURE 4: A model for team 2, found by heuristic mining.



5. DISCUSSION AND CONCLUSIONS

This study looked at three metrics which have been proposed to assess models derived from process mining as a way to assess process fidelity. The metrics were calculated by using logs of events collected from 13 software development teams working on similar projects and using the same process. Of the three metrics considered, the fitness provided the most appropriate information, it had variance over the teams studied and the magnitude reflected how completely the teams had followed the process in terms of ordering and variety of the events.

Behavioural appropriateness gave the same score to each team and thus was not very useful. This the definition of the metric suggests why, the metric is based on comparing the ordering of pairs of events between the model and log, noting if they sometimes, always or never pair. The noisiness of the data ensured that for most teams all of the pairs were present and so the metric returned 1. This indicates that the model is not over generalised, but it means that this is not a good measure of how well a log, in general, utilises all of the process.

Structural appropriateness was measured by marking event types that did not occur in the log as invisible events in the model. This gave a measurement of the number of events not present (as expected). This does give a measure of the utilisation of the process, but is of limited use as it does not take into account the noisiness of the data. A single occurrence of an event is the same as many in calculating the metric, which could be misleading if the sample size is large.

Lastly none of the metrics seemed to measure the iterative nature of the model well. As this is an important part of the agile process model it is something useful to measure. For future work we propose the development of metrics to measure the iterative nature of the log with respect to the model and an appropriateness metric which takes into account the frequency with which an event or event pairing occurs in the log.

This work was supported by an EPSRC grant: EP/D031516 - the Sheffield Software Engineering Observatory.

REFERENCES

AHMED-NACER, M. (2001) *Towards a new approach on software process evolution*. *Computer Systems and Applications, ACS/IEEE International Conference on. 2001.*

ALVES, VAN DER AALST, W. M. P. & WEIJTERS, A. (2008) Quantifying process equivalence based on observed behavior. *Data & Knowledge Engineering,* 64**,** 55-74.

COLOMBO, A., DAMIANI, E. & GIANINI, G. (2006) Discovering the software process by means of stochastic workflow analysis. *Journal of Systems Architecture,* 52**,** 684-692.

COOK, J. & WOLF, A. (1998) Discovering models of software processes from event-based data. *ACM Trans. Softw. Eng. Methodol.,* 7**,** 215-249.

HOLCOMBE, M. (2008) *Running an Agile Software Development Project*, Wiley.

KRISHNAN, M. S. & KELLNER, M. I. (1999) Measuring process consistency: implications for reducing software defects. *Transactions on Software Engineering,* 25**,** 800-815.

LESZAK, M., PERRY, D. & STOLL, D. (2002) Classification and evaluation of defects in a project retrospective. *Journal of Systems and Software,* 61**,** 173-187.

OSTLE, B. & MALONE, L. (1988) *Statistics in Research: Basic Concepts and Techniques for Research Workers*, Iowa State Press.

ROZINAT, A. & VAN DER AALST, W. M. P. (2008) Conformance checking of processes based on monitoring real behavior. *Information Systems,* 33**,** 64-95.

RUBIN, V., GÜNTHER, C., VAN DER AALST, W., KINDLER, E., VAN DONGEN, B. & SCHÄFER, W. (2007) Process Mining Framework for Software Processes. *Software Process Dynamics and Agility,* LNCS, 4470/2007, 169-181, Springer-Verlag, Berlin.

SOTO, M. & MÜNCH, J. (2007) Focused Identification of Process Model Changes. *Software Process Dynamics and Agility,* LNCS, 4470/2007, 182-194, Springer-Verlag, Berlin.

CS-09-01 (2009) The Sheffield Software Engineering Observatory Archive: Six Years of Empirical Data Collected from 73 Complete Projects. Department of Computer Science, University of Sheffield, UK.

VAN DER AALST, W., VAN DONGEN, B., GÜNTHER, C., MANS, R., DE MEDEIROS, A., ROZINAT, A., RUBIN, V., SONG, M., VERBEEK, H. & WEIJTERS, A. (2007) ProM 4.0: Comprehensive Support for Real Process Analysis. *Petri Nets and Other Models of Concurrency - ICATPN 2007.* Springer.

WEIJTERS, A. J. M. M., VAN DER AALST, W. M. P. & DE MEDEIROS, A. K. A. (2006) *Process Mining with HeuristicsMiner Algorithm*. *BETA Working Paper Series.* Eindhoven University of Technology.

TR-2003-17 (2003) The Extreme Programming Evaluation Framework. Department of Computer Science, North Carolina State University, USA.