ReMinds-CMT: An Interactive Tool Supporting Constraint Mining for Requirements Monitoring

Thomas Krismayer Peter Kronberger Rick Rabiser Paul Grünbacher

Christian Doppler Laboratory MEVSS
Institute for Software Systems Engineering
Johannes Kepler University Linz, Austria
thomas.krismayer@jku.at

Abstract

[Context and Motivation] Existing approaches for specification mining and process mining allow to automatically identify requirements-level system properties, which can be used for describing, verifying, or monitoring systems. We have developed an approach that can mine constraint candidates from event logs to support requirements monitoring. **[Question/Problem]** However, the usefulness of mining approaches is currently limited because of (i) weak support for adjusting the algorithms and settings to the current problem, (ii) the high number of properties mined from complex systems, and (iii) the typically high false positive rate. **[Principal Ideas/Results]** In this paper, we present ReMinds-CMT, a tool that guides domain experts throughout the mining process. **[Contributions]** The tool allows users to experiment with different thresholds and configurations of our mining, grouping, filtering, and ranking algorithms to ease the selection of useful constraints. We demonstrate the tool's features using constraints mined from event logs of a complex cyber-physical system controlling unmanned aerial vehicles.

1 Introduction

Researchers from different areas have proposed approaches to (semi-)automatically extract requirements-level system properties, e.g., in the form of constraints, invariants or validity rules, by analyzing source code or outputs of software systems. For instance, in the field of specification mining [6] static approaches [10, 14] use the source code of programs to detect invariants, while dynamic mining approaches [2, 5] analyze the output of programs, e.g., log statements, to derive specifications. Approaches in the area of process mining [7, 11] automatically generate models of existing processes, e.g., in the form of Petri nets, by analyzing (event) logs. In our own research we have developed an approach [3, 4] for mining different types of constraints from event logs recorded from software systems to support requirements monitoring [9, 12].

The usefulness of such mining approaches is typically challenged by the high number of mined properties and the high number of false positives. Constraints are only considered for monitoring if they describe behavior domain experts considers as relevant. This means that experts potentially need to review many constraint candidates to select the ones that really need to be monitored. For this purpose tool support is required. In this paper, we present ReMinds-CMT, a tool that supports domain experts throughout the mining process, e.g., in filtering, grouping, and ranking constraint candidates. It allows users to experiment with different parameters, configurations and algorithms to eventually select constraints for requirements monitoring. ReMinds-CMT is

a stand-alone tool, not an extension of ReMinds: events recorded with ReMinds [13] are just one possible input. However, it currently uses the ReMinds DSL to output the constraints. We demonstrate our tool using examples from a real-world cyber-physical system controlling unmanned aerial vehicles, i.e., drones [1].

2 Our event-based constraint mining approach

Our constraint mining approach [3, 4] analyzes events and data recorded from systems (e.g., through monitoring) and derives candidates for constraints that can be used to check the compliance of these systems at runtime.

The input for our approach are *event logs* comprising recorded events and associated data, usually resulting from multiple runs of different (sub-)systems completing different tasks. They contain multiple *event sequence types* (patterns of multiple event types that have to occur in a given order) and *event sequence instances* (concrete events matching an event sequence type). The information stored in event logs also has implications on the types of constraints that can be mined. The minimal input needed for our mining approach is an event log containing timestamped events. Event logs can be produced by monitoring tools such as ReMinds [13], but also by standard logging tools used in most software systems today.

We use the ReMinds constraint DSL [8] to represent the mined constraints. Constraints in this DSL contain a specific trigger event type to initialize their evaluation. Constraints are checked at runtime as soon as an event of this type is encountered in the stream of events produced by the monitored system.

In the context of automation software and cyber-physical systems we have encountered the following types of constraints: *Temporal constraints* define a sequence of events that has to occur in a specific or arbitrary order and (optionally) within a specified amount of time. Such constraints typically describe a specific task of the monitored system, which consists of several steps, e.g., for the drone example a temporal constraint could check that a `start` event (the drone started flying a route) is followed by `waypoint` events and eventually an `end` event (the drone finished flying the route). Temporal constraints can be mined from an event log containing timestamped events without any further input. *Value constraints* specify the valid content of one or several event data elements – either as one explicit value (e.g., $x == 1$) or with thresholds (e.g., $5 < x \leq 10$). Often such data elements are grouped hierarchically, e.g., coordinates (`X`, `Y`, `Z`) are grouped under an element `location` sent as part of `state` events containing the status of the drone. *Hybrid constraints* combine temporal and value constraints. They include multiple events that have to occur in a given order and/or time, like in a temporal constraint, and additionally check event data elements of at least one of these events, like in a value constraint. Hybrid constraints can also check the relation between multiple event data elements, potentially related with multiple different events, e.g., to ensure that a certain ID remains unchanged throughout one particular event sequence. For the drone example (cf. Figure 1) a hybrid constraint could check that after a `handshake` event a `state` event occurs and that the `state` event contains the data element `groundspeed` with value 0, which checks that a new drone connecting to the system by performing a handshake is waiting for commands on the ground.

3 ReMinds-CMT

For complex software systems our mining approach potentially detects a large number of constraint candidates. While many candidates can be removed from the list automatically, e.g., by removing duplicates, the selection of constraint candidates cannot be fully automated and relies on domain knowledge. We have thus developed ReMinds-CMT, an interactive tool guiding domain experts through the mining process and supporting them in selecting the relevant constraints from the candidates. The tool provides a wizard-based interface (cf. Figure 1) supporting end users throughout the mining process. It provides an interface to the algorithms of our approach and allows users to select the event logs to be analyzed, to (optionally) configure all stages of the mining process, and to review the constraint candidates for selection. Users can experiment with different filtering, ranking and grouping algorithms and can also search in the list of candidates (cf. options on top of the screenshot shown in Figure 1). The tool further allows users to fine-tune constraint candidates, e.g., change the value or operator used in a value constraint (cf. the spinner and combo box controls after each data check and after each 'within' command for the constraints shown in Figure 1). It is possible to select individual constraints or whole groups of constraints, which are then exported to a monitoring tool such as ReMinds [13].

When designing ReMinds-CMT we put special emphasis on flexibility and extensibility. The UI is therefore implemented in a two-level fashion: the outer level contains basic UI elements like the menu bar, whereas the inner level contains the individual pages as described below. This allows for easy exchange and extension of wizard pages. Further, regarding the extensibility in the model part of the application, we implemented all algorithms – such as the filtering, ranking, and grouping algorithms – in Java and provide an API that allows to

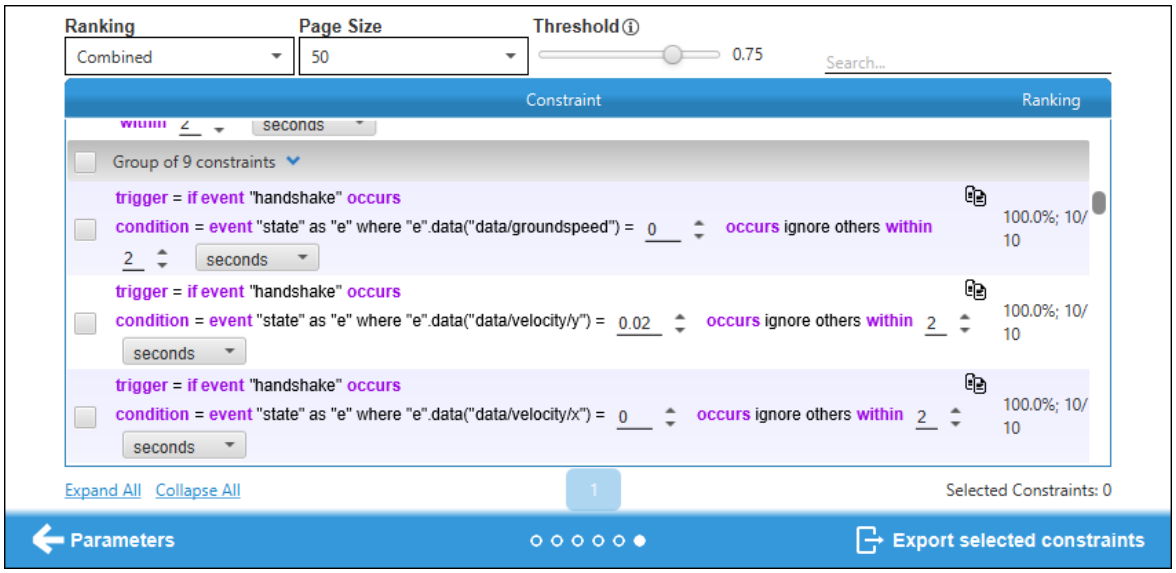

Figure 1: ReMinds-CMT: interactive tool support for selecting constraints.

configure thresholds and factors influencing the calculations. The tool solely relies on this API – meaning that all algorithms can easily be replaced and new algorithms or grouping factors can easily be added by implementing an interface.

Input file selection. The first page of the tool allows the user to choose one or multiple event logs that are used as input for the mining algorithms. The user also has to define how the logs define the event types (e.g., waypoint event, handshake event, etc.), timestamps, and sources of the events (e.g., a particular drone). For event logs recorded with our own monitoring tool ReMinds, this can be parsed automatically.

Input filters. Our tool allows to define filters on the events from the input event list read from the event logs. Options for these filters include filters on event types, sources, and time frames. Multiple filters can be combined and all filters can be defined inclusively or exclusively. Users can view statistics on how many events of the input list are filtered. Additionally, the tool shows examples to illustrate which kind of events pass the filter and which do not.

Configure algorithms. The tool then provides the possibility to choose and configure the mining algorithm, i.e., to skip certain parts of the selected algorithm. For example, the user can choose not to mine any constraints based on intervals calculated from the distribution of the observed event data items.

Constraint filtering. The tool allows to select and configure filters on the mined constraint candidates. Examples for such filters include filters on constraint types, accuracy (i.e., number of positive evaluations of a mined constraint in the event log), and constraints on specific event data items.

Grouping. To support the user during the selection of useful constraints from the (potentially very long) list of mined constraint candidates the tool groups the candidates (cf. Figure 1) based on multiple factors: the trigger event type (e.g., handshake event), the constraint type (e.g., hybrid constraint), the event sequence (e.g., `start`–`waypoint`–`end`), the event data item names (e.g., `status`, `groundspeed`), and the event data item values (e.g., 0, 'active'). The groups are built such that constraints with high similarity – calculated as the weighted average of all factors – are grouped together. Figure 1 shows several constraints concerning handshake events and state events which have been automatically grouped. The weights for all factors and the threshold for grouping constraints are configurable.

Ranking. The user can select the ranking from a list of provided strategies (cf. Figure 1 top left). The tool ranks the constraint candidates within each of the groups first and selects the highest ranked constraint from each group. These constraints are then ranked to estimate the order of the groups.

Our tool supports the following ranking strategies [3]: *(i)* The first algorithm ranks constraints based on *accuracy*, calculated as the percentage of checks evaluating to true, among all evaluations of the given constraint for the input event log(s). *(ii)* The second strategy ranks constraints primarily based on their *type*. Temporal constraints depict the behavior of software systems and are often less specific. They are ranked first, followed by hybrid constraints, and value constraints. *(iii)* The third, *combined* ranking strategy sorts the constraints based on their average rank from the accuracy-based strategy and the type-based strategy. A fourth ranking strategy focuses on (iv) *evaluations*: it combines the accuracy (strategy (i)) with the relative number of positive evaluations, i.e., the number of positive evaluations for the given constraint candidate divided by the highest number of positive evaluations for any constraint candidate. Additional ranking strategies can be defined relying on the provided API.

Adapt and select constraints. Finally, the user can adapt the mined constraints and select the ones that should be exported. Possible adaptations include changing the duration and the thresholds of constraints and changing the used operators. For example, the maximum speed for a drone in a mined constraint is set close to the maximum speed encountered in the input data (6.1 m/s). However, in other scenarios drones can safely fly faster than this maximum and therefore this threshold can be increased before exporting the constraint.

4 Conclusions

We have presented ReMinds-CMT, an interactive tool that supports domain experts in mining constraints for requirements monitoring. The tool is flexible and extensible, e.g., it supports different input formats and new pages can be added in just a few steps. It allows end users to effectively mine constraints and customize the mining algorithm to their use case. So far, we have used our tool for constraints mined from event logs of two complex real-world systems [3] – a plant automation software system and a cyber-physical system controlling unmanned aerial vehicles. We recently presented our tool to several potential users of our industry partner during a workshop and received very positive feedback. In future work we plan to evaluate our tool for further systems and extend it based on feedback we will receive.

Acknowledgments

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, and Primetals Technologies is gratefully acknowledged.

References

[1] Cleland-Huang, J., Vierhauser, M., Bayley, S.: Dronology: An incubator for cyber-physical systems research. In: Proceedings of the 40th Int'l Conf. on Software Engineering: New Ideas and Emerging Results. pp. 109–112. ACM (2018)

[2] Gabel, M., Su, Z.: Javert: fully automatic mining of general temporal properties from dynamic traces. In: Proceedings of the 16th ACM SIGSOFT Int'l Symposium on Foundations of Software Engineering. pp. 339–349. ACM (2008)

[3] Krismayer, T., Kronberger, P., Rabiser, R., Grünbacher, P.: Supporting the Selection of Constraints for Requirements Monitoring from Automatically Mined Constraint Candidates. In: 25th Int'l Working Conf. on Requirements Engineering: Foundation for Software Quality. Essen, Germany (2019)

[4] Krismayer, T., Rabiser, R., Grünbacher, P.: Mining Constraints for Monitoring Systems of Systems. In: 34th ACM/SIGAPP Symposium On Applied Computing. Limassol, Cyprus (2019)

[5] Lemieux, C., Park, D., Beschastnikh, I.: General LTL Specification Mining (T). In: Proceedings of the 30th IEEE/ACM Int'l Conf. on Automated Software Engineering. pp. 81–92. IEEE (2015)

[6] Lo, D., Khoo, S.C., Han, J., Liu, C.: Mining Software Specifications: Methodologies and Applications. CRC Press (2011)

[7] Maita, A.R.C., Martins, L.C., Paz, C.R.L., Rafferty, L., Hung, P.C.K., Peres, S.M., Fantinato, M.: A systematic mapping study of process mining. Enterprise Information Systems **12**(5), 505–549 (2018)

[8] Rabiser, R., Thanhofer-Pilisch, J., Vierhauser, M., Grünbacher, P., Egyed, A.: Developing and Evolving a DSL-Based Approach for Runtime Monitoring of Systems of Systems. Automated Software Engineering **25**(4), 875–915 (2018)

[9] Robinson, W.: A requirements monitoring framework for enterprise systems. Requirements Engineering **11**(1), 17–41 (2006)

[10] Shoham, S., Yahav, E., Fink, S.J., Pistoia, M.: Static Specification Mining Using Automata-Based Abstractions. IEEE Transactions on Software Engineering **34**(5), 651–666 (2008)

[11] Van Der Aalst, W., Adriansyah, A., De Medeiros, A.K.A., Arcieri, F., Baier, T., Blickle, T., Bose, J.C., van den Brand, P., Brandtjen, R., Buijs, J., et al.: Process mining manifesto. In: Business Process Management Workshops. pp. 169–194. Springer (2012)

[12] Vierhauser, M., Rabiser, R., Grünbacher, P.: Requirements Monitoring Frameworks: A Systematic Review. Information and Software Technology **80**(December), 89–109 (2016)

[13] Vierhauser, M., Rabiser, R., Grünbacher, P., Seyerlehner, K., Wallner, S., Zeisel, H.: ReMinds: A Flexible Runtime Monitoring Framework for Systems of Systems. Journal of Systems and Software **112**, 123–136 (2016)

[14] Weimer, W., Necula, G.C.: Mining Temporal Specifications for Error Detection. In: Proceedings of the 11th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. pp. 461–476. Springer (2005)