Flexible Model-Driven Runtime Monitoring Support for Cyber-Physical Systems

Marco Stadler
Michael Vierhauser
{firstname.lastname}@jku.at
LIT Secure and Correct Systems Lab
Johannes Kepler University Linz
Linz, Austria

Antonio Garmendia
Manuel Wimmer
{firstname.lastname}@jku.at
CDL-MINT/WIN-SE
Johannes Kepler University Linz
Linz, Austria

Jane Cleland-Huang
janehuang@nd.edu
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN, USA

ABSTRACT
Providing adequate runtime monitoring is critical for ensuring safe operation and for enabling self-adaptive behavior of Cyber-
Physical Systems. This requires identifying runtime properties of
interest, creating Probes to instrument the system, and defining
constraints to be checked at runtime. Implementing and setting
up a monitoring framework for a system is typically a challenging
task, and most existing approaches lack support for the automated
generation and setup of monitors. GRuM significantly eases the task
of creating monitors and maintaining them throughout the lifetime
of the system by automatically generating runtime models and
providing support for updating and adapting them when needed.

CCS CONCEPTS
• Software and its engineering → Model-driven software en-
ingineering; • Computer systems organization → Robotics.

KEYWORDS
Runtime Monitoring, MDE, Cyber-Physical Systems

ACM Reference Format:
Marco Stadler, Michael Vierhauser, Antonio Garmendia, Manuel Wimmer,
and Jane Cleland-Huang. 2022. Flexible Model-Driven Runtime Monitoring
Support for Cyber-Physical Systems. In 44th International Conference on
Software Engineering Companion (ICSE ’22 Companion), May 21–29, 2022,
1145/3510454.3528647

1 INTRODUCTION
The design and monitoring of Cyber-Physical Systems (CPS) is be-
coming a recurring task, due to the rapid growth of robotic systems
used for shop floor automation [6], medical device delivery [2], and
search-and-rescue operations [10]. CPS exhibit tight integration be-
tween hardware and software components, and frequently interact
with humans requiring additional precautionary measures to be
taken, to ensure that the CPS adheres to its requirements [12].

Runtime information must be collected, typically using code
instrumentation [4], dedicated data-buses [3], or other services, and
then processed at runtime to check constraints to ascertain whether
the system is behaving as intended, or if deviations from expected
behavior have occurred. Off-the-shelf monitoring approaches often
support application performance monitoring [1], or check Service
Level Agreements [9], but they provide inadequate support for
the instrumentation of custom systems, or for defining complex
temporal or structural constraints. In addition, CPS are typically
running on a long-term basis with continuous need for maintenance
and evolution. These activities require data collection mechanisms,
as well as constraints to be updated and adapted. If the monitoring
infrastructure does not co-evolve with the System under Monitoring
(SuM), constraints become obsolete, and data may become outdated.

Model-Driven Engineering (MDE) has previously been used in
various different domains to model entire systems, including run-
time properties [8]. When a new feature or function is introduced
into the system, the corresponding model is updated and the code
is regenerated. However, the extended application of MDE in the
context of runtime monitoring has fallen short of enabling the
generation and evolution of a complete monitoring platform.

To address these shortcomings, based on challenges we have
identified in earlier work [11], we have created GRuM, a model-
driven framework for Generating CPS Runtime Monitors. With
GRuM we support (i) the generation of a customized monitor-
ing platform with support for data collection and analysis, which
(ii) can be readily extended and updated when changes to the
monitored system occur. GRuM provides a lightweight monitoring
solution that requires modeling only those parts of the system that
are relevant for the monitoring infrastructure.

2 THE GRUM APPROACH
A runtime monitoring platform typically provides functionality
for collecting, aggregating, analyzing, and storing runtime data of
a SuM. In our previous work [7], we have extensively analyzed
existing runtime monitoring approaches and derived a reference
architecture for monitoring systems consisting of three major parts:
The Monitoring Setup, responsible for defining and creating Probes,
the Monitoring Execution, related to runtime capabilities, and finally,
Monitoring Support, related to additional capabilities such as data
persistence or visualization. Based on this reference architecture
GRuM leverages MDE techniques to specify relevant monitoring
properties, and ultimately generate a complete monitoring infra-
structure. However, one of the novel characteristics of our approach
is that for a SuM both, a Set of Probes, as well as a fully customized
Monitoring Platform can be generated. Fig. 1 provides an overview
of GRuM’s architecture.

Monitoring Models: The challenge in creating a Probe [5] to
collect runtime information from a SuM is that each individual
SuM uses different technologies and architectures and consists of
different software and hardware components. GRuM addresses this
challenge by providing a generic approach for describing monitor-
ing properties and their respective monitors. We propose a com-
bination of a Monitoring Meta-Model (MMM) to identify relevant
parts of the system, and a Domain Model Fragment, which instantiates the MMM defining the domain-specific components, agents, and (sub-)properties. The actual elements that should be monitored are specified in the Domain Model Fragment and are linked to the elements of the MMM to generate monitoring code. These links are prescribed in a dedicated Augmented Domain Model (ADM) and are used to determine which code fragments are generated for the elements in the Domain Model Fragment.

**Code Generation:** Based on the information in the ADM and the Domain Model Fragment, GRuM can completely and automatically generate (i) a set of Probes for collecting information, and (ii) an instance of the Monitoring Platform for the SuM. In contrast to other monitoring approaches, GRuM aims to reduce the overhead of manually implementing system-specific Probes by generating technology and language-specific Probes. This way, users only need to manually define the Domain Model Fragment with its properties once. The resulting Probe Set Generators (PSGs), e.g., implemented for a Java-based system, can be reused for other Java-based systems, and code can be regenerated when new properties or types of agents are added to the Domain Model Fragment. A topic-based Message Broker sends runtime data from the SuM to the generated infrastructure. The respective topics and topic subscriptions are also generated automatically based on the ADM. At runtime, the monitoring platform instantiates the Domain Model Fragment as a Runtime Model and updates the model when new data is received. This in turn triggers manually created constraint checks, either via an integrated Model Query Engine, or an external Constraint Engine connected to the platform. A generated middleware component provides a SuM-specific interface, e.g., to attach user interfaces, and serves as an API between the monitoring platform and other applications. Examples for such services include additional Constraint Engines to support specific types of constraints (e.g., temporal constraints or event patterns), databases to store runtime data, or user interfaces for visualizing runtime data with respective constraint violation notifications.

![Figure 1: Architectural overview of the GRuM components.](image-url)

3 CONCLUSION

Creating a runtime monitoring framework requires significant effort, resources, time, and in-depth knowledge about the system. With GRuM, our goal is to ease this task by leveraging MDE techniques and automatically generating a runtime monitoring platform for a given SuM. We have successfully applied GRuM to a Java-based UAV management and control system, and TurtleBot3 robots. Results from our initial evaluation have shown that our model-based GRuM framework can be successfully used to (i) describe relevant properties and to (ii) collect and check runtime data via the automatically generated Monitoring Platform. As part of our ongoing future work, we plan to explore ways to support dynamic reconfiguration of the monitoring platform, and additional evolution support by automatically synchronizing the models with the SuM.

ACKNOWLEDGMENT

The work in this paper is funded by the Linz Institute of Technology (LIT-2019-7-INC-316), the Austrian Science Fund (P 30525-N31), and the US National Science Foundation (CNS 1953162).

REFERENCES


