Towards Adapting Non-Standard System Execution Traces for Validating Enterprise DRE System QoS Properties

T. Manjula Peiris and James H. Hill
Dept. of Computer and Information Science
Indiana University-Purdue University Indianapolis
Indianapolis, IN USA
Email: {tmpeiris, hillj}@cs.iupui.edu

***Abstract*—System execution traces are useful artifacts for validating enterprise distributed real-time and embedded (DRE) system quality-of-service (QoS) properties, such as end-to-end response time, throughput, scalability, and reliability. With proper planning during development phase of the software lifecycle, it is possible to ensure such traces contain the required properties to simply their analysis for QoS validation. In some case, however, it is *hard* to ensure system execution traces contain the necessary properties, such as with externally developed DRE system components. Consequently, this makes it *hard* to analyze such system execution traces for validation of QoS properties.**

This work-in-progress paper provides two contributions for analyzing system execution traces for enterprise DRE system QoS validation. First, this presents a methodology called SETAF for adapting non-standard system execution traces for analysis of QoS properties. Secondly, this paper presents preliminary results from applying SETAF to externally developed applications and analyzing its QoS properties. Initial results show that is possible to analyze non-standard system execution traces for validate QoS properties without modifying the applications existing source code.

***Keywords*-QoS validation, non-standard system execution traces, adaptation, patterns, dataflow**

I. Introduction

Current trends and challenges. System execution traces [3], [7], [8], *i.e.*, a collection of log messages, are useful artifacts for analyzing soft real-time enterprise distributed real-time and embedded (DRE) quality-of-service (QoS) properties, such as scalability, throughput, and end-to-end response time. A benefit of using system execution traces for QoS validation is that they provide a comprehensive view of the system's behavior and state throughout its execution lifetime as opposed to a single snapshot of the system at a given point in time, such as a global snapshot [10] that can be hard to analyze such concerns throughout an enterprise DRE system's execution lifetime. Likewise, they provide DRE system developers and testers with a rich set of data for analyzing data trends associated with a QoS given property, *i.e.*, how a given QoS property changes with respect to time, such as viewing how latency changed over the system's execution lifetime or points in the execution lifetime where end-to-end response time missed its deadline.

The UNITE [5], [6] tool describes a methodology for validating QoS properties using system execution traces. UNITE applies relational database theory [1] and dataflow models [2], [9] to analyze different QoS properties. UNITE uses these techniques because most system execution traces describe different but related events which happen in different points in time with some keywords. For example, a system execution trace may contain messages that dictate the sending/receiving of an event along with a timestamp for each occurrence of the message. By searching for the send/receive message keywords and using a dataflow model to show their relation, it is possible to mine the system execution trace for such messages and analyze event latency within a enterprise DRE system. More importantly, such analysis can take place irrespective of system complexity, composition, and implementation because the dataflow model is at a higher level-of-abstraction than the concrete system and system execution traces are platform-, technology-, and language-independent artifacts.

Although it is possible to validate QoS properties via system execution traces, the system execution traces must contain several properties, such as identifiable keywords (as described above). Moreover, the dataflow model, which is used to analyze the system execution trace, must also contain several properties, such as identifiable log message formats, *i.e.*, a regular expression that represents similar log messages, and unique relations between different log formats. If planned early enough in the software lifecycle, it is possible to ensure such properties exist in both the dataflow model and system execution trace. Unfortunately, it is not possible to always ensure system execution traces from enterprise DRE systems contain the required for analysis and QoS validation, such as with externally developed software components and systems. It is therefore critical to develop methodologies that will enable non-standard system execution traces (and their dataflow model) to undergo analysis and QoS validation.

Solution approach → Adapt system execution traces using patterns. The adapter software design pattern [4] is a pattern that enables one object to adapt to the expected interface that another object expects. More importantly, this pattern does not require modification of the two original objects. In the context of analyzing non-standard system execution traces

to validate enterprise DRE system QoS properties, the adapter pattern can be used to *adapt* the dataflow model and system execution trace to contain the required properties for QoS validation. The main challenge, however, is determining how either the system execution trace and dataflow model must be adapted so that either contains the necessary properties to support analysis and QoS validation.

This work in progress paper therefore presents the initial work on *System Execution Trace Adaptation Framework (SETAF)*, which is framework used to adapt system execution traces and dataflow models so they contain the required properties for analysis and QoS validation. DRE system developers and testers use SETAF by first analyzing the system execution trace to identify the pattern for adapting their system execution trace. They then specify the adaptation pattern as a high-level adaptation specification. UNITE then uses SETAF and adaptation specification to inject the required properties at run-time into the analysis of the system execution trace. This process ensures the system execution trace analysis is valid. Initial results for applying SETAF to an open-source software project show that SETAF is able to adapt non-standard system execution traces without requiring modification to the original source code that generates the system execution trace.

Paper organization. The remainder of this paper is organized as follows: Section II provides a brief overview of UNITE; Section III presents the initial design and methodology of SETAF; Section IV presents the preliminary results for applying SETAF to an open-source project; and Section V presents concluding remarks and future research directions.

II. Brief Overview of UNITE

UNITE is a methodology and tool for analyzing system execution traces and validating QoS properties. DRE system developers and tester use UNITE by first generating a system execution trace what consists of a set of log messages. For example, Listing 1 illustrates a portion of a system execution trace generated by a enterprise DRE system.

```
activating Sensor...
...
Sensor sent message A.1 at 2345
Config received message A.1 at 2347
...
Sensor sent message A.2 at 2376
Sensor sent message A.2 at 2379
Config received message A.2 at 2378
...
Shutting down system...
```

Listing 1. Portion of an example system execution trace.

As shown in Listing 1, the send and receive messages can be used to calculate event latency. To perform such analysis using UNITE, DRE system developers and testers identify what log messages they want to extract using a *log format*. This log format captures both the static and variable portions of the log messages. More importantly, the variables identify what portion of the log message to extract for usage in QoS validation. For example, Listing 2 highlights the log formats for the send and receive messages in Listing 1, respectively.

```
LF1: Sensor sent message {STRING type}.
{INT msgid} at {INT sent}

LF2: Config received message {STRING type}.
{INT msgid} at {INT recv}

Relation:
  LF1.type = LF2.type
  LF1.msgid = LF2.msgid
```

Listing 2. Dataflow model for analyzing system execution trace.

After defining the log formats for extracting metrics of interest from a system execution trace, DRE system developers and testers then define a dataflow model that captures the relationship between the different log formats. This is necessary because enables reconstruction of execution flows in the system (1) irrespective of system complexity and composition and (2) without a need for a global clock to ensure causality [10] because the relations between the log formats preserve causality.

```
AVG(LF2.recv − LF1.sent)
```

Listing 3. Expression for analyzing event latency using UNITE.

Finally, DRE system developers and testers define an expression that validates a given QoS property based on the variables in the log format. For example, Listing 3 highlights the expression evaluating average event latency. UNITE then uses the dataflow models and expression to mine the system execution trace and evaluate the provided expression. Likewise, if the aggregation function (*i.e.*, `AVG`) is removed from the expression, then UNITE will present the data trend for the QoS property undergoing analysis.

III. The Design and Functionality of SETAF

This section describes the current design and functionality of SETAF. This section also uses concrete examples to illustrate concepts realized in SETAF.

A. Challenges Associated with Analyzing Non-standard System Execution Traces

Section II provided a brief overview of UNITE and its technique for analyzing system execution traces to validate enterprise DRE system QoS properties. In order to ensure such validation can occur, however, it is necessary that the values in each relation is unique. For example, as shown in Listing 1, the relation between the event ids is *always* unique. If the relations between log formats is not unique, then there is high probability that the analysis will yield incorrect results.

```
Started doing task A at 12.00
Finished doing task A at 12.01
Started doing task A at 12.02
Finished doing task A at 12.03
```

Listing 4. Portion of a system execution trace that contains a non-standard dataflow model.

For example, Listing 4 illustrates an example system execution trace where the dataflow graph will not have unique relations between the log format. This is because it is *hard* to know start/finish messages are associated with one another

without human intervention. Moreover, when an example similar to the one present in Listing 4 is analyzed by UNITE, it will yield incorrect results because it is *hard* to determine correct causality between similar log messages.

With proper planning early in the software lifecycle, it is possible to ensure generated system execution traces have unique relations to facilitate proper analysis. This, however, is not alway possible—especially when analyzing system execution traces generated by third-party systems and their components. Although such non-standard system execution traces may not contain unique relations, the existing relations can be exploited (or adapted) to enforce a unique relation. For example, in Listing 4, although the relation is not unique, it can be adapted to be a unique relation by adding an id to each log message. This will ensure that UNITE analyzes the dataflow model and evaluates the expression correctly. The next section therefore explains how SETAF enables such adaptation of non-standard system execution traces.

B. On Adapting Non-Standard System Execution Traces

As explained in Section III-A, dataflow models that do not contain unique relations for analyzing system execution traces can be adapted to contain unique relations. Unfortunately, it is not possible to adapt each non-standard dataflow model in the same manner to enforce a unique relation. This is because the dataflow model is associated with the given system that generates the system execution trace is used to analyze. A dataflow model therefore can only be reused for different executions of the same system.

Because of this fact, DRE system developers and testers use SETAF by first manually analyzing the non-standard system execution trace. Through this analysis, the DRE system developers and tester identify an *adaptation pattern* for adapting the dataflow model to contain unique relations. More specifically, the adaptation pattern contains details about what variables (and values) needed to be added to each log format (and log message) to enforce a unique relation between each log format that does not contain a unique relation.

Using the example present in Listing 4, DRE system developers then define the adaptation pattern specification that is used by SETAF to adapt the system execution trace. For this example, each log message that represents a start task is proceeded by a finish task before another start task occurs. Using this domain knowledge of the system execution trace, Listing 5 highlights the adaptation pattern specification DRE system developers and testers write to ensure proper analysis of a non-standard system execution trace.

```
Columns:
  LF1.uid, LF2.uid

Init:
  let i = 0;

On LF1:
  LF1.uid = i;

On LF2:
  LF2.uid = i ++;
```

Listing 5. Example of an adaptation pattern specification in SETAF.

As illustrated in Listing 5, first DRE system developers and testers specify what data points need to be added to each log format, *e.g.*, `uid` for `LF1` and `LF2`. DRE system developers and testers then define the initial state of the adaptation pattern. Finally, they define how to adapt each log format (and log message) so that the relations in the dataflow graph are unique. In this example, the `uid` variable is assigned the current value of `i` in both LF1 and `LF2`. In `LF2`, however, the state variable `i` is incremented. This will ensure the next occurrence of `LF1` is differentiated from the previous occurrence of `LF1`, as well as `LF2`. Finally, UNITE analyzes the system execution trace and uses SETAF to adapt its analysis of the system execution trace at run-time to ensure valid analysis, and reconstruction of the different execution flows.

IV. Preliminary Results for Applying SETAF to Apache ANT

This section presents preliminary result for applying SETAF to an example application that contains non-standard system execution traces.

A. Experimental Setup

Section III discussed SETAF's technique for adapting non-standard system execution traces for QoS validation. To determine initial validity of SETAF's technique, we applied SETAF to several Java-based open-source applications, *e.g.*, ANT, Tomcat Web Server, and ActiveMQ JMS Broker. We selected Java-based applications because most standard Java-based applications use log4j (http://logging.apache.org/index.html) to generate system execution traces. It is therefore possible to use UNITE's log4j appender to intercept log messages and store them in a database that is analyzable by UNITE.

One such open-source application that we have analyzed is ANT (http://ant.apache.org), which is a build engine primarily used to build Java applications. To setup the experiment, we first executed ANT to generate a system execution trace. Next, we analyzed the generated system execution trace to identify an adaptation pattern. After we identified the adaptation pattern, we defined an adaptation pattern specification for SETAF. Finally, we used UNITE and SETAF to analyze ANT's generated system execution trace. All experiments were conducted on a Intel core 2 Duo 2.1 GHz processor, with 3GB memory and running 32-bit Windows Vista operating system.

B. Experimental Results

Table I highlights the results for using UNITE to analyze a system execution trace generated by ANT without applying SETAF. As illustrated in this table it is correlating `startTime` and `finishTime` of different ANT tasks. For example if we take the second row of the table an ANT Task named "property" has started at 1500 and finished at 1704. The problem with this table is for some entries (*e.g.*, first and third rows) the `startTime` is greater than the `finishTime`. This is because of the non-uniqueness in the dataflow model used

to reconstruct in the dataset from the non-standard system execution trace. Because of the non-uniqueness in the dataflow

TABLE I
TABLE RECONSTRUCTED BY UNITE WITHOUT ADAPTATION PATTERN SPECIFICATION.

startTime	LF1.task	finishTime	LF2.task
1500	property	860	property
1500	property	1704	property
1516	available	1511	available
1516	available	1518	available

model used to reconstruct in the dataset from the non-standard system execution trace, it resulted several negative values for the evaluation time of different ANT tasks as illustrated below in Table II.

TABLE II
RESULTS FOR ANALYZING RECONSTRUCTED TABLE IN UNITE WITHOUT ADAPTATION SPECIFICATION.

Task	Time (msec)
available	-630.333333333333
delete	0.0
macrodef	140.0
mkdir	-25.125
path	297.0
patternset	-9.76923076923077
property	-241.4
Total evaluation time (sec)	0.345873

Using the adaptation specification defined for ANT, which is similar to the one illustrated in Listing 5, it is possible to improve the results presented in Table I and Table II. Table III therefore highlights the dataset reconstructed by UNITE after using SETAF to apply the adaptation pattern to the reconstruction process. As shown in this table, `startTime` and `finishTime` are now correlated correctly because of the unique id added by SETAF. In this table, all the values of `LF1.startTime` is not greater than the `finishTime`. So it is a correct mapping table.

TABLE III
IMPROVED TABLE RECONSTRUCTION USING SETAF AND UNITE.

LF1.uid	LF1.task	startTime	LF2.uid	LF2.task	finishTime
1	property	766	1	property	860
2	property	1500	2	property	1704
3	available	1500	3	available	1511
4	available	1516	4	available	1518

Likewise, Table IV illustrates the updated final results for analyzing task execution time after adapting the UNITE's analysis at runtime using SETAF. As shown in this table all the evaluation times for different ANT tasks have positive values, which is the expected analysis results.

TABLE IV
FINAL RESULTS FOR ADAPTING UNITE'S ANALYSIS USING SETAF.

Task	Time (msec)
available	93.6666666666667
delete	55.0
macrodef	79.0
mkdir	2.0
path	390.0
patternset	6.0
property	17.975
Total evaluation time (sec)	0.59851

V. CONCLUDING REMARKS

This work-in-progress paper presented initial work on SETAF, which is a technique and tool that adapts non-standard system execution traces for QoS validation. SETAF operates by applying adaptation patterns to system execution traces to ensure correct analysis. Preliminary results from applying SETAF to an open-source project highlight that it is possible to perform such adaptation at run-time without modifying the original source code to ensure the generated system execution traces contain the necessary properties for QoS validation. Future research will focus on applying this technique to other applications—including large-scale enterprise DRE systems—to further validate the technique.

REFERENCES

[1] P. Atzeni and V. D. Antonellis. *Relational Database Theory*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1993.

[2] J. T. Buck. A Dynamic Dataflow Model Suitable for Efficient Mixed Hardware and Software Implementations of DSP Applications. In *Proceedings of the 3rd International Workshop on Hardware/software co-design*, pages 165–172, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[3] F. Chang and J. Ren. Validating System Properties Exhibited in Execution Traces. In *Proceeding of the 22nd IEEE/ACM International Conference on Automated Software Engineering*, pages 517–520, New York, NY, USA, 2007. ACM.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, 1995.

[5] J. H. Hill. Data Mining System Execution Traces to Validate Distributed System Quality-of-Service Properties. In D. A. S. Kumar, editor, *Knowledge Discovery Practices and Emerging Applications of Data Mining: Trends and New Domains*. IGI Global, 2010.

[6] J. H. Hill, H. A. Turner, J. R. Edmondson, and D. C. Schmidt. Unit Testing Non-functional Concerns of Component-based Distributed Systems. In *Proceedings of the 2nd International Conference on Software Testing, Verification, and Validation*, Denver, Colorado, April 2009.

[7] N. Joukov, T. Wong, and E. Zadok. Accurate and Efficient Replaying of File System Traces. In *FAST'05: Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies*, pages 25–25, 2005.

[8] D. Narayanan. End-to-end Tracing Considered Essential. In *Proceedings of High Performance Transactions Systems*, September 2005.

[9] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and P. Wohed. On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling. In *Proceedings of the 3rd Asia-Pacific Conference on Conceptual modelling*, pages 95–104, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[10] M. Singhal and N. G. Shivaratri. *Advanced Concepts in Operating Systems*. McGraw-Hill, Inc., New York, NY, USA, 1994.