Chapter 3
Runtime Verification of Distributed Programs

Eslam Al Maghayreh
Yarmouk University, Jordan

ABSTRACT
Ensuring the correctness of a distributed program is not an easy task. Testing and formal methods can play a significant role in this regard. However, testing does not allow for formal specification of the properties to be checked. On the other hand, formal methods verify that a model of a distributed system satisfies certain properties that are formally specified. However, formal methods do not guarantee that a particular implementation of the system under consideration will also satisfy all of the necessary properties. Runtime verification tries to combine some of the advantages of testing and some of the advantages of formal methods. Runtime verification works on a particular implementation of the system and at the same time it allows us to specify the properties of interest formally. In this chapter we have talked about runtime verification of distributed programs. The main components of a runtime verification framework have been presented and discussed in some details. The reasons that make runtime verification of distributed programs a difficult task have been discussed. A summarization of some of the techniques presented in the literature to simplify runtime verification of distributed programs has also been presented.

INTRODUCTION
Runtime verification is a technique that combines formal verification and traditional testing. The main components of a runtime verification framework are shown in Figure 1. The observed behavior (in terms of log traces) of a program can be monitored and verified dynamically to make sure that given requirements (properties)
are satisfied. Such requirements are typically specified by a formalism which can express temporal constraints, such as LTL-formulae (Linear Temporal Logic).

Testing is an ad hoc technique that does not allow for formal specification and verification of the properties that the program needs to satisfy. Formal methods work on an abstract model of the program. Consequently, even if a program has been formally verified, we still cannot be sure of the correctness of a particular implementation. However, for highly dependable systems, it is important to analyze the particular implementation.

Runtime verification is more practical than other verification methods such as model checking and testing. Runtime verification verifies the implementation of the system directly rather than verifying a model of the system as done in model checking. It is also based on formal logics and provides formalism, which is lacked in testing. Figure 2 demonstrates that runtime verification combines the benefits of traditional testing techniques and formal method.

In the literature, the following four reasons were mentioned in order to argue for runtime verification:

1. If you check the model of your system you cannot be confident on the implementation since correctness of the model does not imply correctness of the implementation.
2. Some information is available only at runtime or is convenient to be checked at runtime.
3. Behavior may depend heavily on the environment of the target system; then it is not possible to obtain the information necessary to test the system.
Founding a Field Theory of Work: Re-Organization through Energy Exchange
www.igi-global.com/chapter/founding-a-field-theory-of-work/108810?camid=4v1a

Improving Effectiveness of Process Model Matchers Using Wordnet Glosses
www.igi-global.com/article/improving-effectiveness-of-process-model-matchers-using-wordnet-glosses/247489?camid=4v1a