# Chapter 2
# Utilizing UML, cTLA, and SRN: An Application to Distributed System Performance Modeling

**Razib Hayat Khan**
*Norwegian University of Science & Technology (NTNU), Norway*

## ABSTRACT

*To meet the challenge of conducting quantitative analysis at the early stage of the system development process, this chapter introduces an extensive framework for performance modeling of a distributed system. The goal of the performance modeling framework is the assessment of the non-functional properties of the distributed system at an early stage based on the system's functional description and deployment mapping of service components over an execution environment. System's functional description with deployment mapping has been specified using UML. To analyze the correctness of the UML specification style, we have used temporal logic, specifically cTLA, to formalize the UML model. We have shown in detail how UML models are formalized by a set of cTLA processes and production rules. To conduct the performance evaluation of a distributed system, the UML model is transformed into analytic model SRN. We have specified an automated model transformation process to generate SRN model from UML, which is performed in an efficient and scalable way by the use of model transformation rules.*

## INTRODUCTION

Distributed software system typically deploys in a resource-limited environment that is mainly constrained by rigorous performance requirements. In a distributed system, system functional behavior is normally distributed among several objects. The overall behavior of the system is composed of the partial behavior of the distributed objects of the system. It is essential to capture the behavior of the distributed objects for appropriate analysis to evaluate the performance related factors of the system at the early design stage. Distributed software systems offer functionalities to the end users by utilizing the distributed components behavior that leads to complex specification as each distributed component provides the partial functionalities of the whole system. Hence, modeling plays an important role in the whole design process of the distributed system to capture partial component behavior efficiently for qualitative

DOI: 10.4018/978-1-4666-8493-5.ch002

and quantitative analysis. As functionality is basically a service spanning over several components, we thus need a specifications describing the collaboration of various distributed components. We therefore introduce a specification style in which a self contained specification block models all functional aspects and collaboration of distributed objects in encapsulated ways that can be reused. We apply UML (Unified Modeling Language) collaborations to express static properties and UML activities to model collaborative detailed behavior as reusable specification of building blocks which in turn provide the facility to build service and adjust the system functions rapidly rather than start the development process from scratch (OMG, 2009a). Reusability makes the developer task easy and faster as system of a specific domain can be built by applying the reoccurring building blocks that are selected from existing libraries. Another advantage of reusable building block utilizing collaboration oriented modeling is that it supports the convergence of Information and Communication Technology (ICT) modeling concept. In particular, we can encapsulate the specific properties of component of certain domain in reusable building block. Thus developers familiar with the concept of different domain of ICT should have less trouble to create sound distributed system using the building block of existing domain (Kraemer & Herrmann, 2006). Besides, considering collaboration diagram helps to compose the system from the subtask of the participants which are relevant for drawing the system's overall functional behavior. Above all, collaboration oriented approach provides a way to group chunks of interaction behavior contributed by the most pertinent actors of the system.

Furthermore, we consider system execution architecture to realize the deployment of the service components. Abstract view of the system architecture is captured by the UML deployment diagram which defines the execution architecture of the system by identifying the system components and the assignment of software artifacts to those identified system components (OMG, 2009a). Considering the system architecture to generate the performability model resolves the bottleneck of system performance by finding a better allocation of service components to the physical nodes. This needs for an efficient approach to deploy the service components on the available hosts of distributed environment to achieve preferably high performance and low cost levels. The most basic example in this regard is to choose better deployment architectures by considering only the latency of the service. The easiest way to satisfy the latency requirements is to identify and deploy the service components that require the highest volume of interaction onto the same physical resource or to choose resources that are connected by links with sufficiently high capacity (Csorba, Heegaard, & Hermann, 2008).

In order to guarantee the precise understanding and correctness of the model, the approach requires formal reasoning on the semantics of the language used and to maintain the consistency of the building blocks and their composition. Temporal logic is a suitable option for that. In particular, the properties of super position supported by compositional Temporal Logic of Action (cTLA) make it possible to describe systems from different view points by individual processes that are superimposed (Hermann, 1997; Hermann & Krumm, 2000). Here, we focus on the definition of cTLA/c, a style of cTLA that allows us to formalize the collaboration service specifications given by UML activities and also to define the formal semantics of deployment diagram. By expressing collaborations as cTLA processes, we can ensure that a composed service maintains the properties of the individual collaborations it is composed from. Moreover, the semantic definition of collaboration, activity and deployment in form of temporal logic is implemented as a transformation tool which produces TLA$^+$ modules (Slåtten, 2007). These modules may then be used as input for the model checker TLC (Yu, Manolios, & Lamport, 1999). The tool also generates a number of theorems, so that collaborations may be analyzed for more advanced properties than simple syntactic checks.