# An Architectural Approach to Performance Issues – from Experiences in the Development of Network Management Systems -

*Natsuko Noda, Tomoji Kishi*
*Software Design Laboratories, NEC Corporation*
*(Igarashi Building) 11-5, Shibaura 2-chome,*
*Minato-ku, Tokyo 108-8557, JAPAN*
*TEL +81 3 5476 1089, FAX +81 3 5476 1113,*
*e-mail n-noda@ccs.mt.nec.co.jp*

**Abstract**

For the last few years, we have been consulting a project which develops Network Management systems (NMSs). In NMS development, requirements about performance are rigid and various performance problems arise. To avoid these problems, we try to establish a performance design method in which we estimate the system performance based on the actual measurement. In this method, it is important which modules should be chosen as measurement targets to estimate the whole system performance accurately. In order to identify target modules for the measurement properly, it should be considered what relationships the modules have with other modules in view of performance. That is, we should identify measurement targets considering the software architecture from a performance perspective.

In this paper, first we define the software architecture for the performance design, and then we describe a method for identifying the appropriate target modules to estimate performance. For this method, we also introduce the linkage/dependency graph to capture the architecture.

**Keywords**
**Software architecture, performance issues, design method, network management system**

# 1 INTRODUCTION

For the last few years, we have been consulting a project which develops network management systems (NMSs). This team has been developing many NMSs for the last ten years. A NMS is the system which monitors and controls equipment connected to networks. Developing NMSs involves addressing rigid requirements about performance. For example, a NMS must have enough performance in processing alarms from the network equipment. Since it controls a large number of equipments and they would raise alarms at the same time, it has to deal with these alarms at a time. If an alarm is not properly handled, the equipment keep raising alarms and that could cause another equipment to raise alarms. Therefore, a number of alarms to be processed by a NMS at a time is clearly defined.

It is difficult to design software to meat its performance goals (to *design performance*), and numerous problems invariably arise. Frequently, it is found at the end of the development that the system does not show the desired performance such as the response time and the throughput.

In order to avoid these performance problems, we try to establish an effective method for designing performance. We intend to make good use of performance measurement for this method, because it is difficult to understand performance concretely in figures without measuring. However, it is impractical to find problems ultimately when the system is completed and measured about its performance. In our method, we measure performance partially in design process, i.e., we implement some modules, which seems to be significant for the system performance, run them, and then measure their performances, in order to estimate the whole system performance. We call this approach for designing performance, *measurement approach*.

In the measurement approach, it is quite important to determine appropriately which modules to be the measurement target. To identify the proper measurement target, it is required to understand software structure in view of performance. In this paper, we propose the technique to identify the measurement target modules, in which we use linkage/dependency graph that represents software architecture from a performance perspective.

# 2 MOTIVATIONS

Many development teams actually have tried the measurement approach. However, it does not always give us an accurate prediction. The followings are such typical examples:

- When we had implemented a part of a system and measured its performance, it had showed the sufficient performance. However, it showed much lower performance, when it was put into the entire system. In fact, this function used a channel, which was used by many other functions, and all these functions scrambled that channel.
- A process of a communication from A to B consists of two communication processes: a communication from A to C and from C to B. To know the time

for the process of the communication from A to C, the time for those two consisting communication processes had been measured. However, the real communication time from A to B was much longer than the sum of these measured time. In fact, since these two communication processes used different protocols, a transformation of protocols was required in C and it took long time.

As the above examples show, in order to get accurate performance prediction, it is significant to choose modules properly which are implemented and measured about its performance. In this paper, we call this set of modules (to be implemented and measured about its performance) "target module set" for the measurement approach.

It is not obvious which modules make a proper target module set. To identify it, it is important to consider the structure of software, i.e., software architecture[Gar95_1]. Especially, we believe the followings to be significant:

- Whether relation between the performance of the entire system and that of the set can be understood.
- Whether the performance of the set does not change, when it is put into the entire system.

## 3 GOALS OF OUR STUDY

In order to make the measurement approach for designing performance work well, we try to establish an effective method in which we can identify target module sets and use them to estimate the whole system performance. For this purpose, first we define the software architecture from the performance points of views. Then we describe the method for identifying the appropriate target module sets by utilizing this architecture.

## 4 THE ARCHITECTURE FOR THE PERFORMANCE DESIGN

In this section, we define the architecture for the performance design. It should be noticed that this architecture does not directly represent performance. Our aim is not to model performance itself, but clarify the structure to identify modules to be measured for the performance design, because we think it is difficult to understand performance without measuring.

Since we are interested in the performance with respect to certain service, we explicitly express the relationship between performance and service in our definition. In this paper, service is the behavior provided by the given part of a system[Boo94].

The architecture is defined by means of a type of component (*serviceable module set*) and two types of connector (*dependency*, *linkage*):

- **Serviceable module set**:
  A serviceable module set for service S is a set of modules that are required to realize S. Note that resources, such as channels and queues, are also included

in a serviceable modules set, since they are also necessary to implement the service. In this paper, M(S) denotes a serviceable module set for service S.

- **Dependency**:
  A dependency between two serviceable module sets M(S1) and M(S2) is a relationship between M(S1) and M(S2) in which the behavior of the modules in M(S1) influences the behavior of the modules in M(S2).
  - **Strength of dependency**:
    Indicates the degree to which the behavior of the modules in M(S1) has the influence upon the behavior of the modules in M(S2).
- **Linkage**:
  A linkage between two serviceable module sets M(S1) and M(S2) is a relationship between M(S1) and M(S2) in which M(S1) and M(S2) collaborate with each other to realize another service S.
  - **Strength of linkage**:
    Indicates the degree to which the linkage has the influence upon the performance of S.

According to the above definition, a dependency is a directional relationship. However, in most cases, if M(S1) influences M(S2), M(S2) also influences M(S1). Thus, in this paper we ignore the direction of dependencies.

If there is a dependency between two serviceable module sets M(S1) and M(S2), the behavior of M(S1) is influenced by the behavior of M(S2) and then M(S1) does not show the same performance, when it is put into the entire system. On the other hand, if there is a linkage between M(S1) and M(S2), there exist collaboration between M(S1) and M(S2) to realize another service S and this collaboration makes it difficult to estimate the performance of M(S). For example, if there are mismatches of data model or control model[Gar95_2] between M(S1) and M(S2), that may require transformation process we cannot ignore when we consider the performance, and the performance of M(S) may not be easily estimated based on the those measured performances. Thus, in the measurement approach, it is significant to understand these relationships.

It should be noticed that this architecture is intended to express the software structure in a view of performance with respect to certain service. It may be different corresponding to each service and strength of linkage/dependency is not constant.

## 5 USING LINKAGE/DEPENDENCY GRAPH FOR PERFORMANCE MEASUREMENTS

In this chapter, we introduce the linkage/dependency graph to represent the architecture, then, we describe how we identify appropriate target module sets using the graph and estimate performance with the identified target module sets.

### 5.1 Graph Definition

A linkage/dependency graph for service S (i.e., a linkage/dependency graph with respect to the performance of S) is the graph of which constructs are defined as follows:

- **Node**:
  Denotes a serviceable module set corresponding to a service which is realized by the system. N(S) represents a node that denotes a serviceable module set M(S).
- **Linkage arc**:
  Denotes a linkage between two serviceable module sets, each serviceable module set is denoted by the node at the end of the arc and is for a service included in S, namely each service is a sub-service of S. Since this graph is constructed for the particular service S, a linkage related to realizing other services than S is not denoted as an arc.
- **Dependency arc**:
  Denotes a dependency between two serviceable module sets, each serviceable module set is denoted by the node at the end of the arc.
- **Weight of linkage/dependency arc**:
  Represents the strength of linkage/dependency which the arc denotes. Weight is not less than 0 and not more than 1.

An example of a linkage/dependency graph is shown in Figure 1(b).

### 5.2 Graph Construction

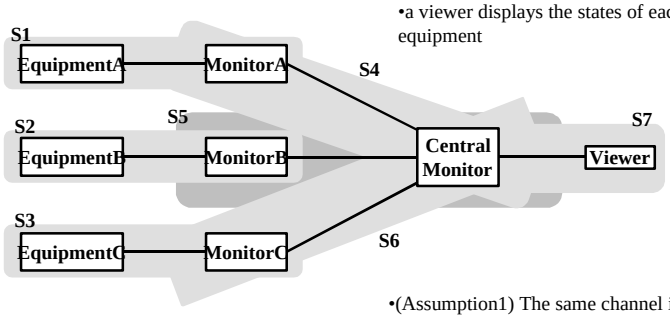
In this section, we discuss on how to construct the graph.

In order to construct a graph which expresses completely accurate architecture, we have to know completely about the precise nature of performance, such as hardware behavior and performance. In most cases, it is quite difficult to have such precise knowledge. However, for the performance design, it is not always required to build a complete graph. Our intention is to build a graph that can be believed to be the approximation of the architecture and can be utilized to find target module sets for the performance measurement.

In this section, we show a very simple construction method based on our experiences. This method can be used without detailed knowledge about performance.

We describe our construction method with an example, simple NMS. The simple NMS is described in Figure 1(a).

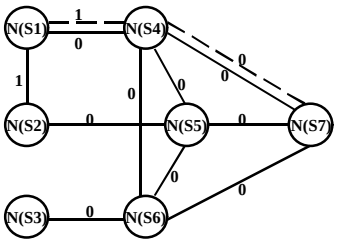
*outline*
•monitors three types of equipment
•three types of monitor servers, each type one monitors each type of equipment
•a centralized monitor server monitors three monitor servers
•a viewer displays the states of each monitored equipment




•S1, S2, S3: EquipmentA(B,C) notifies alarms to MonitorA(B,C)
•S4, S5, S6: CentralMonitorServer observes MonitorA(B,C)
•S7: Viewer shows the information CetralMonitorServer has

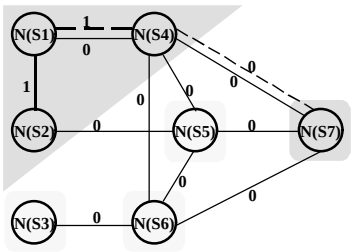
•(Assumption1) The same channel is used in the following two services: the service notifying alarms from EquipmentA to MonitorA and the service notifying alarms from EquipmentB to MonitorB.
•(Assumption2) In order to use the middleware, the service notifying alarms from each equipment to each monitor uses the protocol and data structure specific to the middleware.
•(Assumption3) The protocol and data structure used by the CentralMonitorServer is different from those specified by the middleware.

(a) Simple NMS




Dependency arc
Linkage arc

(b) Linkage/Dependency Graph for S of Simple NMS




(c) Identifying the Target Module Sets

Figure1 Simple NMS

Let S be the service whose performance we are to estimate and for which we are to construct a linkage/dependency graph. About this NMS, let S be the service that Viewer shows that an error has occurred in EquipmentA.

1. Define serviceable module sets and make nodes corresponding to them.

2. For every pair of nodes, define a dependency arc, if two serviceable module sets corresponding to the two nodes share the same modules.

   In this step, we have to be careful not to overlook the sharing of resources, such as channels or queues, because sharing of the resources could make the dependency between entirely different services, and sometimes causes serious problems. Because S1 and S2 use a same channel, a dependency arc is drawn also between N(S1) and N(S2).

3. Give weight [1,0] to every dependency arc; that is, identify the synchronization at the shared modules and give weight 1 to the dependency arc where the synchronization is identified. To the rest of the dependency arcs, give weight 0.

   Here, synchronization means the concurrency semantics of an operation, required to share modules. About the NMS, the synchronization is required when S1 and S2 use the same channel, i.e., one (S1 or S2) have to wait until the other completed to use the channel. This would decreases each performance. Hence the weight of the dependency arc between N(S1) and N(S2) is decided as 1.

4. For every pair of nodes corresponding to the sub-services of S, define a linkage arc, namely, if there exists a linkage between two serviceable module set corresponding to the two nodes, define a linkage arc.

   Among all the serviceable modules sets for the sub-service of S, namely M(S1), M(S4) and M(S7), there is a collaboration between M(S1) and M(S4), and between M(S4) and M(S7). Therefore, we draw a linkage arc between N(S1) and N(S4), and between N(S4) and N(S7) as well

5. Give weight [1,0] to every linkage arc; that is, if data model and/or control model used in each serviceable module set, which participates in the linkage, is different, give weight 1 to the linkage arc. To the rest of the dependency arcs, give weight 0.

   If there are differences in data/control models[Gar95_2], the transformation one model to another (i.e., the transformation of the data type, or the transformation of the protocol) is required. Since this kind of transformation tends to be complicated or to take a long time to process, the collaboration between those two serviceable module sets can influence the performance of service S. About

the NMS, M(S1) and M(S4) use different data and control model each other. Therefore, we give the weight 1 to the arc between N(S1) and N(S4).

### 5.3 Using Graph for Measurement

Based on the constructed graph, we identify appropriate target module sets, and estimate entire system performance based on measurements of those identified target module sets.

How to identify target module sets for the performance measurement of service S is as follows. Decide the threshold value of weight, and remove arcs whose weight is less than the threshold (and leave arcs whose weight is not less than the threshold). Then we can find one or more groups of nodes, in which every node is connected each other by arcs. These groups are isolated each other (see Figure 1(c)). Among them, pick only groups which contain one or more serviceable module sets for sub-services of S. Because weight of each arc indicates strength of linkage or dependency, every serviceable module set in each group is related to each other by stronger linkage and/or dependency. Furthermore it is not strongly related to any serviceable module set in another group. Therefore, each group of serviceable module sets, represented as a group of nodes on the graph, can be appropriate target module set for S.

After identifying the target module sets for S, we implement each module set and measure the performance. When there are linkages in the target module set, implement serviceable module sets related each other by those linkages and realize higher service. On the other hand, when there are dependencies in the target module set, implement modules and measure the performance of the sub-service of S.

## 6 CONCLUSION

In this paper, we have defined the software architecture for the performance design and proposed the performance design method capturing this architecture. In this method, we identify proper target modules to measure the performance using the linkage/dependency graph and estimate performance of the entire system based on the actual measurement.

Since it is unrealistic to expect that we have perfect information to decide the design direction, especially when we develop such a large system as NMS, this method is expected to be used effectively.

## 7 REFERENCES

[Boo94] Booch, G., Object-Oriented Analysis and Design, Benjamin/Cummings, (1994).

[Gar95_1] Garlan, D. et al., Introduction to the Special Issue on Software Architecture, IEEE Transaction on Software Engineering, Vol.21, No.4. April (1995).

[Gar95_2] Garlan, D. et al., Architectural Mismatch: Why Reuse is So Hard, IEEE Software, Nov. (1995).