Proc. Int. Workshop on Intelligent Networks (Lappeenranta, Finland; Aug. 1993)

A Framework for Evaluating the Performance of IN Services

Kimmo E. E. Raatikainen
University of Helsinki, Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23), SF-00014 University of Helsinki, FINLAND
email: Kimmo.Raatikainen@Helsinki.fi
fax: *int* + 358 0 708 4441

From the point of view of computer science, the Intelligent Network can be interpreted as a platform for implementing distributed real-time applications. The requirements for reliability, availability, and performance of IN services are stringent. These requirements challenge the hardware and the software development methods. In this paper we examine the possibilities and implications of applying the software performance engineering approach for IN services. We briefly motivate why the IN architecture should also have a data model and review some existing techniques for evaluating the performance of database operations and signalling networks.

1. Introduction

From the point of view of computer science, the Intelligent Network can be interpreted as a platform for implementing distributed real-time applications. In the same way as for most real-time applications the requirements for IN services regarding reliability, availability, and performance are stringent. Typically, the downtime allowed is only a few hours in a year, only one transaction in ten thousand is allowed to fail, thousands of transactions must be processed in a second, and 99 per cent of the transactions must be completed in fractions of second.

Such requirements challenge the hardware and the software development methods. It is essential that the reliability, availability, and performance of the service can be evaluated when the service is still in the planning phase. In this paper we present a framework for evaluating the performance characteristics of IN services that are in the development phase.

We start by briefly reviewing the basic methodology of software performance engineering. In Section 3 we outline a description of IN services that supports performance predictions. In Section 4 we briefly motivate why the IN architecture should also have a data model. In Section 5 we concentrate on network configuration planning. In Section 6 we discuss some techniques for evaluating the performance of database operations and signalling networks.

2. Software Performance Engineering

The fundamental methodology of Software Performance Engineering (SPE) was established during the 1980's. SPE was one response to the "software crisis" that resulted from the expansive growth of the complexity in software systems. Software Performance Engineering is an approach for constructing software systems to meet responsiveness and throughput goals. With it, one models software requirements and designs, and evaluates whether predicted performance metrics meet the specified goals. If not, alternatives are proposed, assessed, and adopted.

The process continues through the whole lifecycle of the software system, that from specification of requirements to maintenance. As the process continues more precise models of the software are developed and more precise predictions of performance are attained.

Those who are familiar with the practice of performance analysis know that the workload characterization is the most challenging and cumbersome part of any performance study. For example, if we are using a product-form queueing network model with multiple classes of customers to predict various performance indices, we need the following three descriptions (Lazowska [1984, Ch. 4.4]):

customer description: The number of *customer classes* C and for each class c its *workload intensity* λ_c.

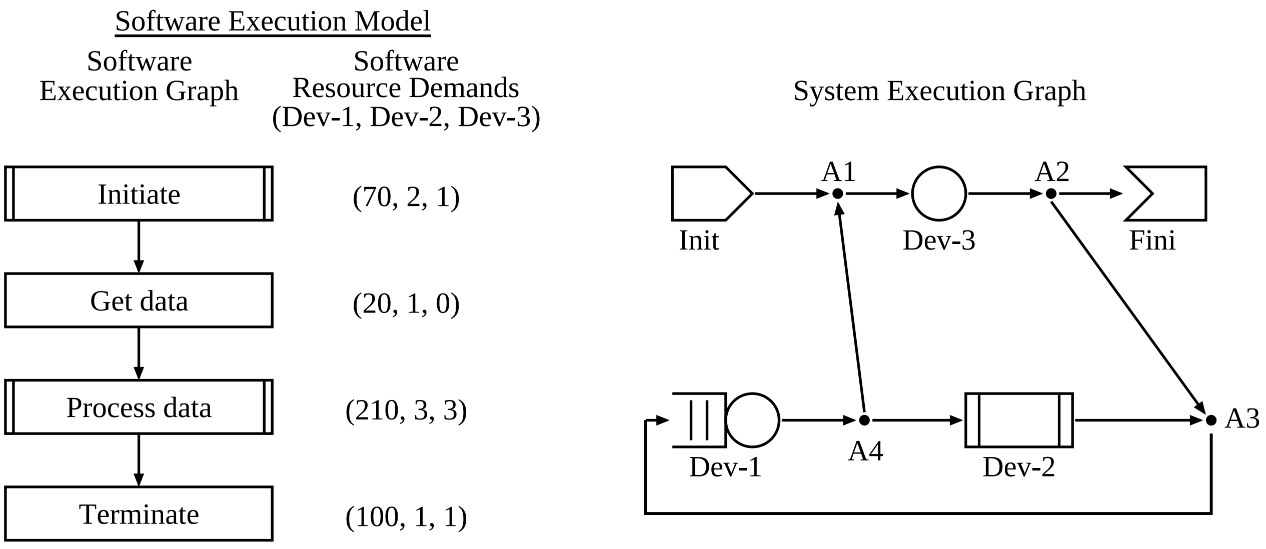


Figure 1: Examples of Software and System Execution Models

center description: The number of *service centers* K and for each center its *type* (queueing or delay).

service demands: For each class c and center k the *service demand* $D_{c,k}$.

When the numbers are available, it is only elementary arithmetics to obtain the predictions for the system response time, system throughput, distribution of active load, device utilizations, device queue lengths etc.

If our actual system is so complex that a product-form queueing network model is insufficient, the calculations may be quite complicated. However, it is "only mathematics" what we have the three descriptions. Typically, the customer and center descriptions remain quite simple even in complex systems. The challenging part is the characterization of service demands. For example, in simulation studies if we are interested in the quantiles of response times, we must model the joint distribution of service demands; see e.g. Raatikainen [1992b].

The main contribution of the SPE approach is the systematic way that can be followed to obtain the service demands. The approach is fully described in the textbook by Connie Smith [1990]. In SPE we work with two different kinds of models: *software execution models* and *system execution models* (see Figure 1).

The software execution models are used to characterize the structure of the software (*software execution graph*) and the resource demands of software modules. The software resource demands are specified in a manner that does not depend on the execution platform. Typically, the demands are specified as the number of statements in a high-level programming language and as the number of io-operations.

The system execution model describes the hardware that executes the software (*system execution graph*). In addition, the system execution model gives the translations from the numbers of statements and io-operations into execution times.

Suppose that in the example of Figure 1 *Dev-1* is a processor, *Dev-2* is a disk, and *Dev-3* is a terminal device. Let us further assume that each software module is implemented as a subroutine. When we translate the software resource demands into execution times, we must take into account the flow control (i.e. subroutine calls) and the fact that each disk and terminal operation also consumes processor time.

Table 1 provides an example of the translation matrix for obtaining hardware resource consumptions from the software resource consumptions. For example, module *Initiate* consumes 93[1] units of work in the processor (*Dev-1*), when the flow control and the processor demands of disk and terminal operations are taken into account. Hardware resource consumptions in the example are given in Table 2.

One may wonder why we use the translation matrix instead of directly taking into account the processor de-

[1] $3 + 70 \times 1 + 2 \times 5 + 1 \times 10 = 93$

Table 1: Hardware Resource Consumptions per Software Resource Request

Hard-ware resource	Hardware resources per software resource			
	Flow control	Component		
		Dev-1	Dev-2	Dev-3
Dev-1	3	1	5	10
Dev-2	0	0	1	0
Dev-3	0	0	0	1

Table 2: Hardware Resource Consumptions in the Example of Figure 1

Software Module	Device		
	Dev-1	Dev-2	Dev-3
Initiate	93	2	1
Get data	28	1	0
Process data	258	3	3
Terminate	118	1	1
Σ	497	7	5

mands of io-operations. The reason is that changes in the system software affect only the translation matrix and not the software resource demands.

Suppose that our processor executes 1 high-level statement in a millisecond and that each terminal-operation (requires human action) take Z seconds. Let us further assume that an disk-operation takes

- no time if the data request is in the main memory buffer,
- 5 milliseconds if the data is in the internal buffer of the disk, and
- 50 milliseconds otherwise.

When q denotes the probability that the data cannot be found in the main memory buffer and p denotes the conditional probability that the data is in the disk buffer when it is not in the main memory buffer, we have the following parametrized service demands:

$$\begin{aligned} D_1 &= 0.497\ sec \\ D_2(p,q) &= 7q(0.050 - 0.045p)\ sec \\ D_3(Z) &= 5Z\ sec \end{aligned}$$

When the devices are also used by other software, we need to know their service demands and workload intensity. Then we can evaluate the performance predictions under various conditions, that is under different values of p, q, Z, and the workload intensity λ. Typically we are interested in the best case ($q = 0$) and in the worst case ($q = 1$, $p = 0$).

Another usual situation is that we state an upper bound for the response time, say $R^\star$, and we determine the set of tuples (p, q, Z, λ) for which the response time is at most $R^\star$. When the set is determined, we subjectively examine its feasibility.

3. Description of IN Services

The IN Conceptual Model or INCM shown in Figure 2 is described in CCITT recommendation Q.1201[2]. It is an architectural tool for the design and description of IN architecture. The SPE approach can quite easily be modified to match the IN Conceptual Model.

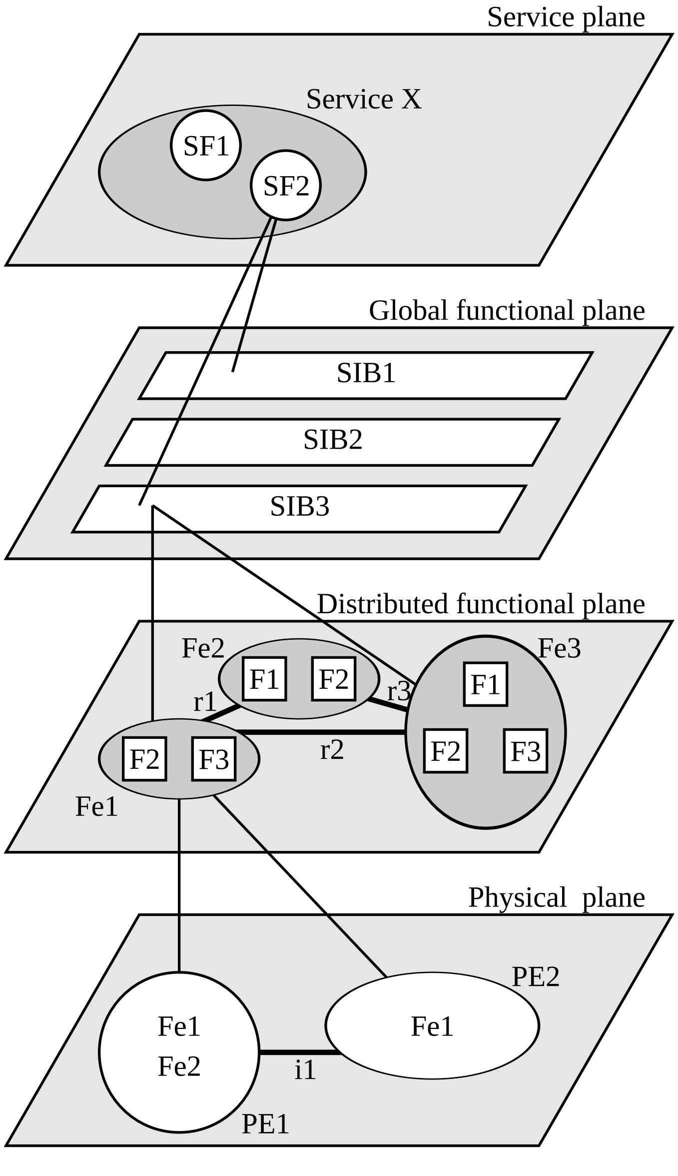
At the service plane the service is described through service independent building blocks (the *SIB*'s) and reusable service features (the *SF*'s). This description defines the global service logic (*GSL*) that is independent from the underlying physical network. Therefore, GSL corresponds to the software execution graph.

The IN functional architecture (Figure 4) corresponds to the distributed functional plane of INCM. The functional entities (the *FE*'s) provide FE actions (the *F*'s in Figure 2). The information flows between the functional entities (the *r*'s in Figure 2) or "relationships" (as referred in CCITT terminology) are the other component of the description of the IN functional architecture.

The functional entities and relationships are described in a way that does not depend on the physical implementation and deployment of the network. The SIB's for the IN capability set 1 (IN CS-1) on the global functional plane are realized on the distributed functional plane by a sequence (= serial sequence) of functional entity actions and resulting information flows. When the SIB's for the management services are specified, the description must also contain parallel execution; at least *fork* and *join* operations.

IN CS-1 contains 14 SIB's that include algorithm, charge, compare, translate, basic call process, among others. In principle many services other than Freephone, UPT, VPN, and others described in CCITT Recommendation Q.1211 could be specified using these

[2] CCITT [1992b]; see also Garrahan et al. [1993] or Appeldorn et al. [1993].



SFi	Service Feature i
SIBi	Service Independent Building Block i
Fei	Functional Entity i
Fi	Functional Entity Action i
ri	Relationship i
PEi	Physical Entity i
ii	Interface i

Figure 2: The IN Conceptual Model

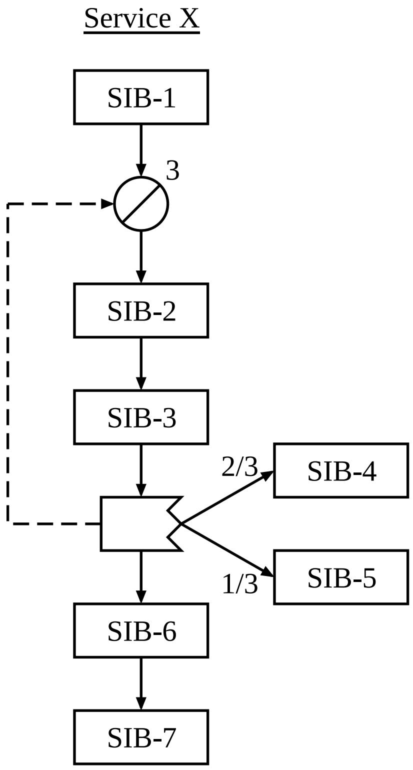


Figure 3: Software Execution Graph for Service X

SIB's. Unfortunately the application programming interface was not specified in IN CS-1. Another shortcoming, from the point of view of performance engineering, is that the translations from SIB's into functional entity actions are unspecified.

Successful and easy performance engineering of IN services requires that we can automatize the translation from the service description into the numbers of FE action requests and into volumes of information flow between different functional entities. Figure 3 gives the software execution graph for a hypothetical Service X. Table 3 demonstrates the translation from the SIB's into the numbers of requests of actions in different functional entities. This translation can be interpreted as the first part of Software Resource Demands in SPE terminology. The difference, however, is that functional entity actions are logical resources. The second part of the demands is the volumes of information flows in the relationships.

In the example Service X first uses SIB-1. Then it executes a loop three times. The loop consists of invokes

Table 3: Numbers of Functional Entity Actions that Each SIB Requests in Service X

	Requests of Functional Entity Actions																
	CCAF		CCF/SSF					SCF				SDF			SRF	
	#1	#2	#1	#2	#3	#4	#5	#1	#2	#3	#4	#1	#2	#3	#1	#2
SIB-1	1		1	1		1		1		2		1-n_{11}			1	
SIB-2				2			2	1	2		3		3-n_{22}			1
SIB-3					1	1				3				1-n_{33}		
SIB-4				2			2		1		3	3-n_{41}		2-n_{43}		
SIB-5			1			1				1					2	
SIB-6		1		1	2		2		1	1		3-n_{61}	3-n_{62}	1-n_{63}		1
SIB-7	1	1	1		1		2			2	1		1-n_{72}	1-n_{73}	1	1

of three SIB's: SIB-2, SIB-3, and either SIB-4 or SIB-5. The probability of executing SIB-4 is 0.67 and SIB-5 is 0.33. Finally, the service uses SIB-6 and SIB-7.

From Table 3 we can see that SIB-4 uses the action #2 and #5 in Call Control Function/ Service Switching Function twice. In Service Control Function SIB-4 uses the action #2 once and the action #4 three times. The usages of actions in Service Data Function are parametrized. The parametrization is needed when the amount of work that the action performs depends on the input parameters. In Service Data Function SIB-4 uses the action #1 three times and the action #3 twice. The amounts of work that the actions perform are characterized from the parameters n_{41} and n_{43}, respectively.

4. Data Model for IN Services

The IN services rely heavily on information in databases; see e.g. Robrock [1991]. This suggests that the IN should have a common data model. Eventually the data model should have several planes in a manner similar to INCM.

The global data model is similar to the global functional plane in the sense that all pieces of service data can be viewed as a monolithic entity. The distributed data model[3] should correspond to the distributed functional plane in the sense that each functional entity has a private view of the global data model that contains the data accessed by the actions of this functional entity.

The object-oriented approach adopted for the architecture of the telecommunications management network (TMN) — see e.g. CCITT [1992a] and Raman [1993]— suggests that the data models should be object oriented. The implication of the object-oriented approach is that in addition to the FE actions the functional entities should contain functional entity data elements. The relationships are no longer abstract information flows but requests to actions in other FE's to deliver, receive, and/or update pieces of information. However, the differences between the relationships and requests of actions are more in the syntax than in the semantics.

The need for data models becomes obvious when we want interoperability between database management systems provided by different vendors; see e.g. Aiken et al. [1991]. The demand of globalizing the IN services[4] requires that different databases can co-operate. The data models also clarify the performance engineering and network configuration planning.

5. Network Configuration Planning

The IN physical architecture, Figure 5 shows IN CS-1 physical architecture alternatives, describes how the functional entities are realized. The interfaces (the i's in Figure 2) provide the means for realizing the information flows.

One or more functional entities from the distributed functional plane may be realized in one physical entity on the physical plane. Moreover, one functional entity may have different realizations in several physical entities or may be realized through several co-operating physical entities. These three strategies of realizing entity actions may be labelled as *centralization*, *replication*, and *distribution*. It should be noted that in one

[3]The term distributed data model is misleading in the same way as the term distributed functional plane. The distribution is actually logical partitioning. The database and functions may locate in a single node.

[4]see e.g. Wyatt et al. [1991], Sykes and Visser [1992], and Fujioka et al. [1993].

Functional Entities	
CCAF	Call Control Agent Function
CCF	Call Control Function
SSF	Service Switching Function
SCF	Service Control Function
SDF	Service Data Function
SRF	Specialized Resource Function
SMF	Service Management Function
SCEF	Service Creation Environment Function
SMAF	Service Management Access Function

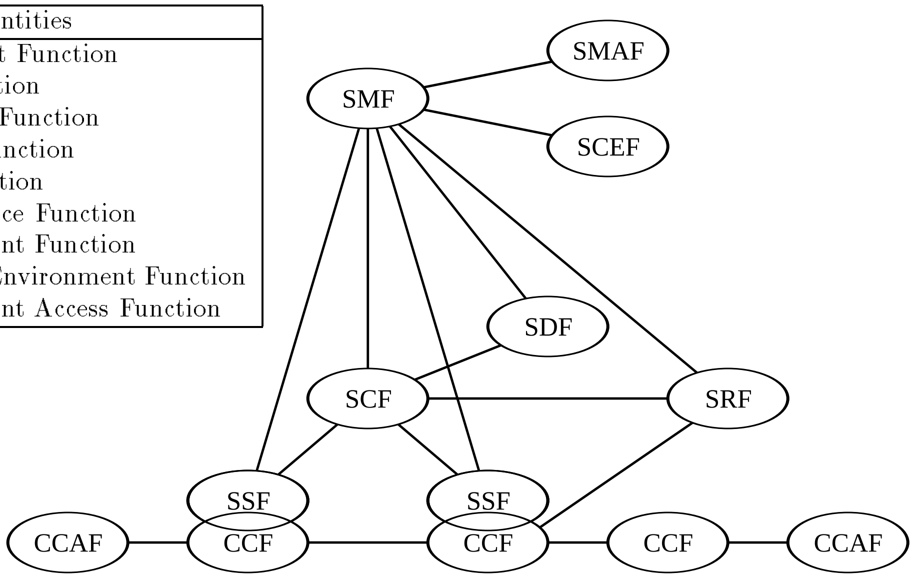


Figure 4: The IN Functional Architecture

network we may have both centralized, replicated, and distributed realizations.

In network configuration planning we focus on the mapping from the functional entities into physical entities and on mapping from relationships into interfaces. CCITT IN CS-1 concentrated on the service functionality. Therefore, the effects that centralization, replication, and/or distribution of service logic have upon performance can quite easily be modelled for the performance engineering of IN services.

The analysis of database operations is essentially more complicated. One central reason is that we do not yet have, in the IN framework, a suitable conceptual data model. When we are going to examine the effects of centralizing, replicating, and distributing the service data, we must develop a data model. From the point of view of performance engineering it is essential that the data model

1. provides a monolithic view of all pieces of data so that the service developer need not pay attention to details of data access in early stages of the development project
2. encapsules the data so that the data integrity and security can easily be maintained
3. allows flexibility in centralizing, replicating, and distributing the service data
4. does not seriously restrict different implementations

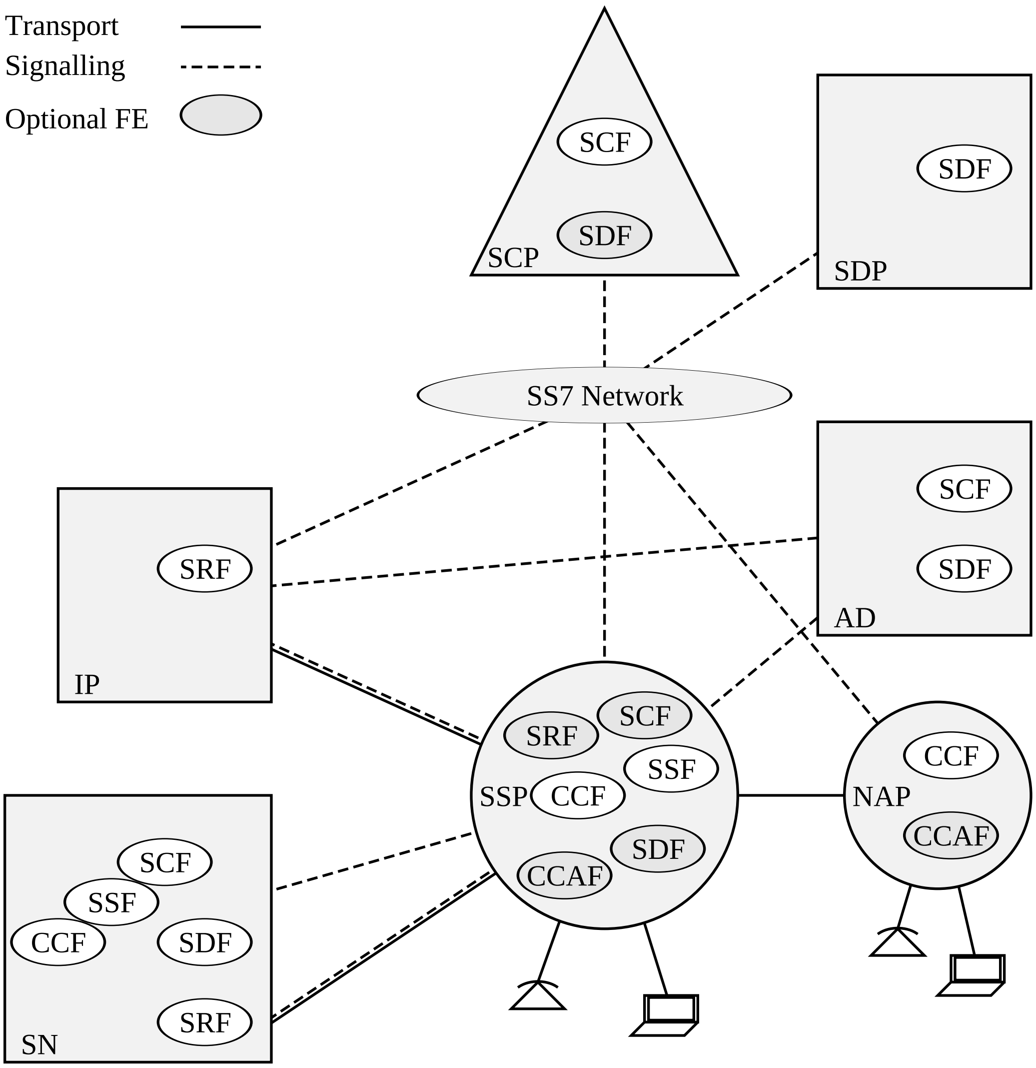
The objectives of SPE activities that corresponds to the network configuration planning are 1) to obtain the system execution graph and 2) to obtain the translations from the Software Resource Demands into resources demands of the actual physical entities.

6. Review of Performance Evaluation Techniques

In the previous sections we treated the workload characterization for software performance engineering. In the current section we briefly discuss some techniques that can be applied in the performance evaluation of IN services. We restrict the discussion to Signalling System No. 7 Network and to database operations. We believe that they are the most important areas for successful performance engineering.

Performance Modeling of SS7 Network

Based on our current knowledge we believe that the layered performance modeling approach described by Willmann and Kühn [1990] and the queueing network modeling approach introduced in Conway [1990] meet the



Physical Entities	
SN	Services Node
SSP	Service Switching Point
NAP	Network Access Point
IP	Intelligent Peripheral
AD	Adjunct
SDP	Service Data Point
SCP	Service Control Point

Functional Entities	
CCAF	Call Control Agent Function
CCF	Call Control Function
SSF	Service Switching Function
SCF	Service Control Function
SDF	Service Data Function
SRF	Specialized Resource Function
SMF	Service Management Function
SCEF	Service Creation Environment Function
SMAF	Service Management Access Function

Figure 5: The IN Physical Architecture

targets of performance engineering as long as we are only interested in the steady-state mean values of the common performance indices. However, the two papers mentioned above only present demonstrative example applications of the proposed methods. Therefore, additional validation studies should be conducted; hopefully with real measurement data.

If we are interested in performance indices that are not steady-state means, we need techniques which are more complex or more time-consuming. If we are interested in the quantiles of the steady-state response time distribution in a queueing network model, we can use the approximate method introduced in Raatikainen [1989] or the estimation method introduced in Conway and O'Brien [1992]. However, the analytic or approximative methods may be insufficient, if we need detailed evaluation. In such cases we must carry out simulation studies.

During the recent years we have developed simulation methods for advanced output analysis. These methods include

- percentile estimation (Raatikainen [1990]),
- simultaneuos estimation of several means (Raatikainen [1993a]),
- simultaneuos estimation of several proportions (Raatikainen [1992b]),
- precision assessment in interactive simulation (Raatikainen [1992c]),
- estimations of dynamic properties in queueing system (Raatikainen [1993b, 1993c]).

All those methods can easily be incorporated into SPE practice.

Performance Modeling of Database Operations

Our survey study on the topic is still unfinished. The performance of database operations is a wide question. During the last five years hundreds of research papers have been published in journals and proceedings of conferences. Unfortunately we have not found any state of the art surveys. In order to achieve the proposed targets of the data models in Section 5, we must address topics like real-time transaction processing, memory-resident databases, distributed databases, and replication of data, among others.

The recent literature is full of isolated techniques for specific subareas in performance modeling of database systems. Our challenging task is to find those methods that can successfully be incorporated into the software performance engineering of IN services. We consider the layered database performance model, which is quite similar to the IN Conceptual Model, introduced by Sevcik [1981] as an initial framework. The layered model is redefined in Casas and Sevcik [1987] and briefly summarized in Casas and Sevcik [1989]. This framework may be developed to match with the proposed data model.

Summary

We have outlined a framework for applying the established software performance engineering approach in developing IN services. We have identified three key areas for future research:

1. Mappings from IN services through SIB's into functional entity actions and relationships on the distributed functional plane of INCM
2. Conceptual data model that clarifies the service data in the same way as INCM clarifies the functionality of IN services
3. Development (or selection) of performance evaluation methods for data access of IN services

The final challenge is to provide the SPE tool that meets the targets discussed in this paper and that is compatible with the IN service development environments.

References

Aiken, J. A; Parker, S. T.; and Woodwell, D. R. (1991) Achieving Interoperability with Distributed Relational Database. *IEEE Network Mag. 5,* 1 (Jan.): 38–45.

Appeldorn, M.; Kung, R.; and Saracco, R. (1993) TMN + IN = TINA. *IEEE Commun. Mag. 31,* 3 (Mar.): 78–85.

Casas, I. R. and Sevcik, K. C. (1987) Structure and Validation of an Analytic Performance Predictor for System 2000 Databases, In *Proc. of the 5th Canadian Information Processing Society Conf.* (Edmonton, Canada; Nov.).

Casas, I. R. and Sevcik, K. C. (1989) A Buffer Management Model For Use In Predicting Overall Database System Performance. In *Proc. of the 5th Conf. on Data Engineering* (Los Angeles, Calif.; Dec.): 463–469.

CCITT (1992a) *Recommendation M.3010 — Principles for a Telecommunications Management Network.*

CCITT (1992b) *Draft Recommendations Q.120x, Q.121x — Intelligent Network.*

Conway, A. E. (1990) Queueing Network Modeling of Signaling System No. 7. In *Proc. of the IEEE GLOBECOM'90.* (San Diego, Calif.; Dec.): 552–558.

Conway, A. E. and O'Brien, D. E. (1992) Estimating Response Time Distributions in Queueing Networks. In *Performance of Distributed Systems and Integrated Communication.* T. Hasegawa, H. Takagi, and Y. Takahashi, eds. IFIP Transactions C-5. North-Holland, Amsterdam, The Netherlands: 225–244.

Fujioka, M.; Kikuta, H.; Sakai, S.; and Wakahara, Y. (1993) Globalizing IN for the New Age. *IEEE Communications Magazine 31*, 4 (Apr.): 54–60.

Garrahan, J. J.; Russo, P. A.; Kitami, K; and Kung, R. (1993) Intelligent Network Overview. *IEEE Commun. Mag. 31,* 3 (Mar.): 30–36.

Lazowska, E. D.; Zahorjan, J.; Graham, S. G.; and Sevcik, K. C. (1984) *Quantitative System Performance: Computer System Analysis Using Queueing Network Models.* Prentice-Hall, Englewood Cliffs, N. J.

Raatikainen, K. E. E. (1989) Approximating Response Time Distributions. *Performance Evaluation Review 17,* 1 (May): 190–199.

Raatikainen, K. E. E. (1990) Sequential Procedure for Simultaneous Estimation of Several Percentiles. *Transactions of the Society for Computer Simulation 7,* 1 (Mar.): 21–44.

Raatikainen, K. E. E. (1992a) *Simulation Estimation of Proportions.* University of Helsinki, Dept. of Computer Science (Feb.).

Raatikainen, K. E. E. (1992b) Modeling Service Distributions in Queueing Network Simulation. *Simulation 59,* 2 (Aug.): 116–126.

Raatikainen, K. E. E. (1992c) *Controlling the Precisions of Estimates in Interactive Simulations.* University of Helsinki, Dept. of Computer Science, (Oct.).

Raatikainen, K. E. E. (1993a) A Sequential Procedure for Simultaneous Estimation of Several Means. *ACM Trans. on Modeling and Computer Simulation 3,* 2 (Apr.): in press.

Raatikainen, K. E. E. (1993b) Accuracy of Estimates for Dynamic Properties of Queueing Systems in Interactive Simulation. *Int. J. of Computer Simulation,* to appear.

Raatikainen, K. E. E. (1993c) *Simulation Estimates of Dynamic Properties in Queueing Systems.* University of Helsinki, Dept. of Computer Science, (July).

Raman, L. (1993) CMISE Functions and Services. *IEEE Commun. Mag. 31,* 5 (May): 46–51.

Robrock II, R. B. (1991) The Intelligent Network — Changing the Face of Telecommunications. *Proceedings of the IEEE 79,* 1 (Jan.): 7–20.

Sevcik, K. C. (1981) Data Base System Performance Prediction Using an Analytic Model. In *Proc. of the 7th Int. Conf. on Very Large Data Bases* (Cannes, France; Sept.): 182–189.

Smith, C. U. (1990) *Performance Engineering of Software Systems.* SEI Series in Software Engineering. Addison-Wesley, Reading, Mass.

Sykes, J. S. and Visser, E. J. (1992) Globalization of Intelligent Network Services. *AT&T Technical Journal 71,* 5 (Sep./Oct.): 6–12.

Willmann, G. and Kühn, P. J. (1990] Performance Modeling of Signaling System No. 7. *IEEE Commun. Mag. 28,* 7 (July): 44–56.

Wyatt, G. Y.; Barshefsky, A.; Epley, R. V.; Kaplan, M. P.; and Krishnan, K. P. (1991) The Evolution of Global Intelligent Network Architecture. *AT&T Technical Journal 70,* 3–4 (Summer): 11–25.