TheATLAS

MODULE SERIES

TRANSDISCIPLINARY ENGINEERING & SCIENCE

PERFORMANCE DESIGN AND IMPROVEMENT OF COLLABORATIVE SYSTEMS BY THE APPLICATION OF KNOWLEDGE-BASED QUALITATIVE REASONING

Tod Gonsalves
Kiyoshi Itoh
Ryo Kawataba

ISSN: 1933-5423

TAM-Vol.1-No.1, 2005

The Academy of Transdisciplinary Learning & Advanced Studies
TheATLAS Publications

TheATLAS Book Series on Transdisciplinary Engineering & Science

SERIES EDITOR-IN-CHIEF
A. ERTAS

TRANSDISCIPLINE: Integrating science and engineering principles

"...Today, complexity is a word that is much in fashion. We have learned very well that many of the systems that we are trying to deal with in our contemporary science and engineering are very complex indeed. They are so complex that it is not obvious that the powerful tricks and procedures that served us for four centuries or more in the development of modern science and engineering will enable us to understand and deal with them. We are learning that we need a science of complex systems, and we are beginning to construct it..."

Nobel Laureate Herbert A. Simon
Keynote Speech, 2000 IDPT Conference

TheATLAS Publishing

ISSN: 1933-5423

Published in the United States of America by

TheATLAS

www.theatlas.org

Abstract

In Collaborative Systems, several unforeseen problems arise when people come together to build a product or when people having different perspectives, share resources and collaborate in building a system that offers service to end-users. Performance evaluation, benchmarking and improvement of such systems are often neglected due to lack of systematic framework and tools. Conventionally, system performance is evaluated and improvement carried out, much after the implementation of the system. This is an extremely difficult, laborious and costly task. Our strategy is to propose performance estimate at the requirement analysis and design phase itself, long before the implementation phase. In this paper, we introduce novel tools for modeling the system, evaluating the performance of the system and for improving and enhancing its performance. The stages of system modeling, evaluation, performance design and performance improvement share a common integrated knowledge base. System modeling is done by using the Multi-Context Map (MCM) technique, system evaluation is done with GPSS simulation, and improvement is accomplished by the Expert System reasoning approach with Qualitative knowledge-based rules. MCM captures the workflow in a collaborative system wherein collaborators interact with each other through the exchange of Token, Material and Information (TMI). This triple-input-triple-output is what distinguishes contexts in MCM from ordinary single-input-single-output servers in queueing networks. These three additional interactions present a formidable challenge to improving the system quantitatively. In order to overcome the computational complexity, we make use of Qualitative Reasoning (QR) in our Expert system (ES). At the system performance evaluation stage, the ES displays to the user the inherent bottlenecks in the system upon analyzing the performance-knowledge acquired from GPSS simulation. At the performance improvement stage, the ES inference engine refers to the structural-knowledge of the system and by applying the heuristics from the expert's knowledge, draws the parameter-tuning plan to resolve the bottlenecks in the system resulting from underflow, overflow or non-uniformity in TMI flow. The integrated environment of model building, performance evaluation and performance improvement is semi-automatic by design, and as demonstrated by the successful application to the performance design and improvement of the benchmarking system, can be extended to any real life Collaborative System (CS).

Keywords

Collaborative Systems, bottlenecks, performance evaluation, performance improvement, qualitative reasoning, knowledge-base, expert system.

1 Introduction

Testing and evaluating the performance of the system is an important stage in the development of the system; however, it is often ignored for lack of time, tools or both. James Cooling observes that while designing systems “designers are (blindly) optimistic that performance problems – if they arise – can be easily overcome” [1]. The designers test the performance of the system after the completion of design and implementation stages and then try to remedy the problems. The main difficulty with this “reactive approach”, Cooling states, is that problems are not predicted, only discovered. It is the concern of this study to ***predict*** the operational problems and suggest a viable solution while designing the systems. By performance design, we mean that the performance requirements and performance characteristics are already incorporated in the system at the design stage of the system. We give an estimate of the system performance, as it were, at the requirement analysis stage itself of the life cycle of system development. In this study we outline a performance-improvement strategy and introduce a novel tool that diagnoses the problems in the system operation and facilitates the improvements in it.

System modeling is done by the MCM technique. MCM is a descriptive model of Collaborative Systems, wherein people with different skills collaborate to offer service to end-users. The collaborators working in different contexts within the system interact with each other through the exchange of Token, Material and Information (TMI). Service at a context begins when TMI from the preceding contexts assemble at that context, and on receiving service, TMI proceed to the next junction or context in the MCM. An MCM queueing network captures in detail the flow of TMI in the system. The flow of TMI, however, is not always smooth and unobstructed. It ranges from shortage (underflow) to abundance (overflow). At times, service at a context is delayed due to lack of synchronization, i.e., the entity with the highest arrival rate arrives at the context and patiently waits for the arrival of the slower entities before servicing can begin at the context. These and a similar host of phenomena give rise to what is known as bottlenecks in network analysis parlance.

The conventional quantitative approach to resolve the bottlenecks would be to write a set of equations and solve them simultaneously. But the number of equations expressing the interrelationships between the nodes in the network increase exponentially with the number of nodes, N, in the network. In case of contexts in MCM that interact with one another through the exchange of TMI, the number of equations would increase in gigantic powers of 3N. To overcome the computational complexity we make use of Qualitative Reasoning (QR). QR, also known as naïve Physics or common sense Physics, is a branch of AI that tries to emulate the mind of the human expert in the way it tackles a problem. The human expert, when faced with a complex problem, does not construct and solve quantitative equations in his mind, but uses his experience and intuition in dealing with the problem at hand. QR has been applied to solve a diversity of problems exhibiting qualitative behaviors such as electronic circuits, economic systems, etc.[2], [3], [4]. We use the techniques of QR in establishing the qualitative rules that drive the inference engine of the resolving ES.

The entire design approach from modeling to improvement is an enterprise in knowledge engineering. The heart of the project is the three-layered integrated knowledge base. The top layer contains the hard facts of the contexts and junctions and their inter-connections in the MCM network structure which is extracted from the MCM drafter. The process of extraction being automatic, eliminates the need for any additional external tool of knowledge acquisition. The middle layer contains the data obtained from GPSS simulation. These data are processed and arranged in a way that facilitates easy access and interpretation by the ES. The third layer, which is the core of the knowledge-base are the qualitative rules that have been skillfully systematized and elaborated by distilling knowledge from the human experts in the Collaboration domain.

In this short paper, we first present the overview of collaborative systems and discuss the key issues in the core system development life cycle. Then we give a brief description of the MCM modeling methodology followed by a practical example which further illustrates the MCM technique. Then we present a scheme for defining and classifying bottlenecks that may generally occur at the contexts and junctions of MCM. Then we discuss in depth the bottleneck-resolving qualitative rules that make up the knowledge base of the ES. Resolving a bottleneck is an extremely difficult job because of the numerous constraints involved. Sometimes system requirements are such that key parameters that need to be tuned to resolve a neck may not be altered. At times improving a particular bottleneck is at the expense of normally operating contexts and the result of the improvement at one place is nothing short of incurring the risk of generating new and fresh bottlenecks. To overcome this difficulty the ES checks for feasibility and advisability as guidelines for the application of qualitative rules. After discussing these guidelines we trace the propagation of effects on resolving a bottleneck in the downstream of the improved bottleneck. The knowledge of the propagation of these effects enables the ES to determine the advisability strategy. Finally, we discuss the results obtained by applying the ES to resolve bottlenecks inherent in a general Collaborative Engineering benchmarking system.

2 Collaborative Systems

2.1 Collaborative Systems defined

Collaborative Systems are systems whose purpose is collaboration, undertaken to offer service. These systems exhibit a predominant client-server property. The following are all examples of Collaborative Systems: Business firms collaborating to enhance their business prospects, doctors and nurses in a clinic offering medical service to patients, teachers and staff members in a school offering educational service to students, manager and tellers in a bank offering financial service to customers. In fact, computer systems also qualify to be Collaborative Systems because the hardware subsystems such as CPU, memory, I/O devices 'collaborate' via the operating system to offer software programs executing service to the user. Another special class of systems engaged in collaboration is collaborative engineering systems. Their goal is carrying on engineering (or providing 'engineering services') through collaboration.

2.2 Types of Collaborative Systems

Our definition of Collaborative Systems is flexible to include a wide variety of Collaborative Systems. We may loosely divide the Collaborative Systems into the following three classes.

Inter-Collaborative Systems

These are large-scale systems and the scope of collaboration is wide. A group of hospitals or companies engaged in collaboration perpetually or for some phase in their developmental history are examples of Inter-Collaborative Systems. This kind of large-scale collaboration is usually between organizations. Each collaborating organization may have priority goals for itself more sharply focused than the goal of the collaboration as a whole. As shown in Fig. 1, the collaborating organizations are

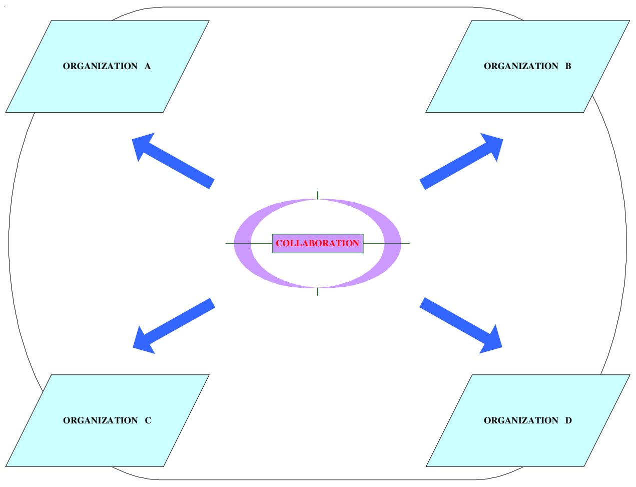


Fig. 1 Inter-Collaborative Systems.

not entirely enveloped by the system boundary, because some of the systems could have collaboration with other organizations outside this system.

Our approach to system development via modelling, performance evaluation and performance improvement can be extended to any of these classes, but we focus our attention on type II Collaborative Systems. The intra-collaboration in these systems is represented by the MCM model.

Intra-Collaborative Systems

Functioning as a unit, an intra-Collaborative Systems is more closely knit, and all its activities are directed towards one particular well-defined goal. A hospital, for instance, is an intra-collaborative system. The diagnosis units, prescription units, surgery units, etc., in the hospital collaborate with one another within the framework of that hospital to offer service to patients. The boundary of the system (Fig. 2) encompasses the entire collaborating subsystems.

Our approach to system development via modelling, performance evaluation and performance improvement can be extended to any of these classes, but we focus our attention on type II Collaborative Systems. The intra-collaboration in these systems is represented by the MCM model.

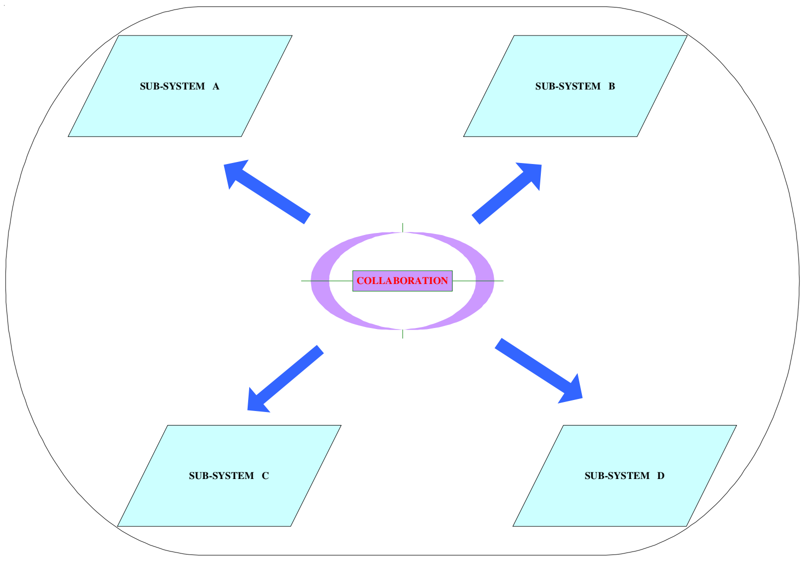


Fig. 2 Intra-Collaborative Systems.

2.3 System Development Life Cycle

Information Systems can be seen as subsystems in larger organizational systems, taking input from, and returning output to, their environmental organizations. Any information system has a developmental life cycle. Different models of System Development Life Cycle (SDLC) have been proposed as a methodology for systems development in many organizations. The traditional SDLC, consisting of planning, analysis, design, implementation, operation & support phases, is a waterfall model where the result of each phase flows sequentially into the next phase [5]. Some people consider the life cycle to be a spiral in which we constantly cycle through the phases at different levels of detail [6].

In this study, we are not dealing with the entire SDLC. Our main concern is the performance design of Collaborative Systems. This being the case, we shall focus our attention on the (requirements) analysis and design of Collaborative Systems. That these phases are of prime importance in SDLC is a fact that can hardly be over-stressed. Itoh and Kawabata highlight the importance of these two phases in their 'crank model' [7]. In the traditional SDLC, evaluation and improvement of the system performance, if at all it takes place, comes at the end of the life cycle. However, it is extremely difficult to modify the performance of an already established system and it is very expensive. It is our strategy to carry out the performance evaluation and improvement in the requirements analysis and design phase itself. Our expert system can evaluate the performance and improve upon it before making a final decision regarding the system design. The performance estimate made by the expert system can be included in the design of the system.

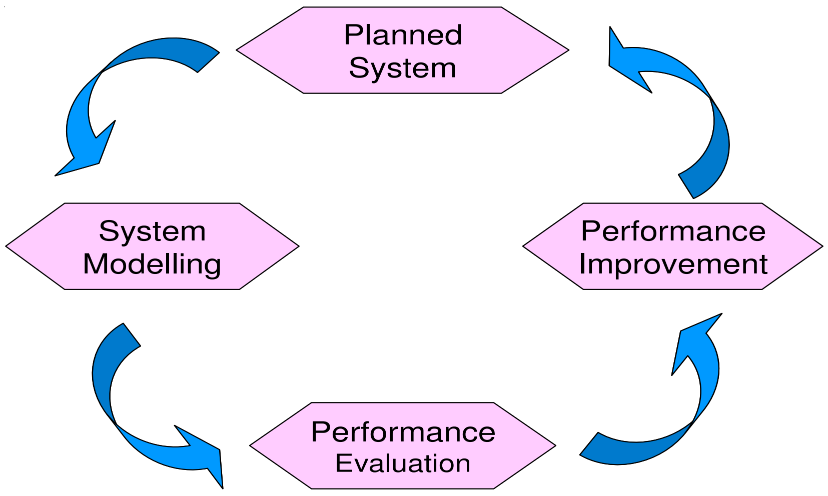


Fig. 3 Core Development Cycle.

To achieve this purpose, we insert 'modelling-performance evaluation-performance improvement' core cycle inside the SDLC as shown in Fig. 3.

2.4 Performance Design Strategy

Bottom-up approach

In the performance design strategy, there could be two distinct approaches, the bottom-up and the top-down approach. In the traditional methodology for system development, the bottom-up approach is commonly used. The system is designed after the completion of the initial phase of requirement analysis and then implemented. The performance of the implemented system is evaluated and then there is another fresh new cycle of planning, design and implementation of the improved system. (Note that there is no implicit mention of performance evaluation and improvement in the early phases of the SDLC). Bottom-up approach (Fig. 4) also applies to the performance improvement of already existing 'hard' systems. Since these systems

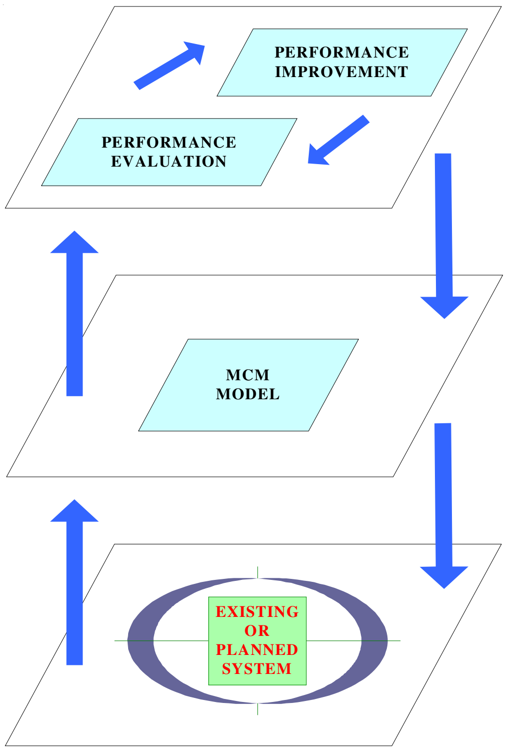


Fig. 4 Performance Design: Bottom-Up Approach.

are already existing and functioning, the only way to improve their performance is through simulation.This would imply the bottom-up approach of modeling, performance evaluation and performance-improvement as shown in Fig. 5.

Top-down approach

What we propose in this study is a top-down approach (Fig. 5). The modeling, performance-evaluation, performance-improvement cycle is included in the requirements analysis and design phase of the system. We start with the MCM model of the proposed system. By simulation we observe system operation and evaluate its performance. Then we invoke the expert system to improve the performance of the system. From the lessons learnt we further simulate the operation of the system and repeat the two operations till we are satisfied with the performance of the system under the given constraints. Then we make functional changes in the MCM model if needed, and again simulate the system performance. The outcome is an optimally operating planned system that may be implemented.

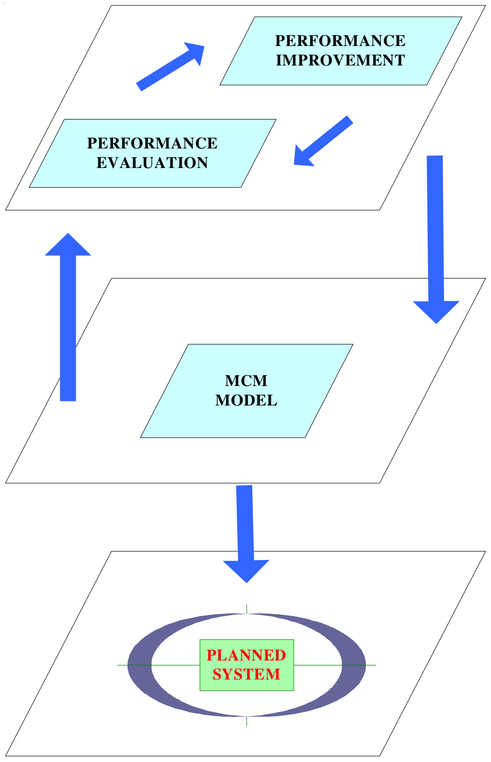


Fig. 5 Performance Design: Top-down approach.

3 System modeling using MCM technique

MCM is an ingenious modeling technique developed by the authors to model the real world systems engaged in collaboration. MCM is a comprehensive way of representing the workflow in Collaborative Systems [8]. "Context" as explained below, is a conceptual interface wherein the exchange of the essential ingredients of Collaborative Systems, viz. token, material and information takes place. Token, material and information flow from context to context. The linking of the various contexts through arcs that represent the flow of token, material and information, gives rise to a topology known as the Multi-Context Map (MCM).

3.1 Contexts in MCM

Collaboration necessarily implies the presence of actors (collaborators), action (task) and "object upon which the action is performed". Context is a conceptual location that facilitates the process of collaboration among the collaborators. It could also symbolically represent the physical location of the collaboration activity. The left-hand perspective of the context represents the "requestor" of the activity and may include person or persons, organization, equipment, etc., while the right-hand perspective represents the "performer" of the activity. Context, in a word, is the combination of these two perspectives (Fig. 6).

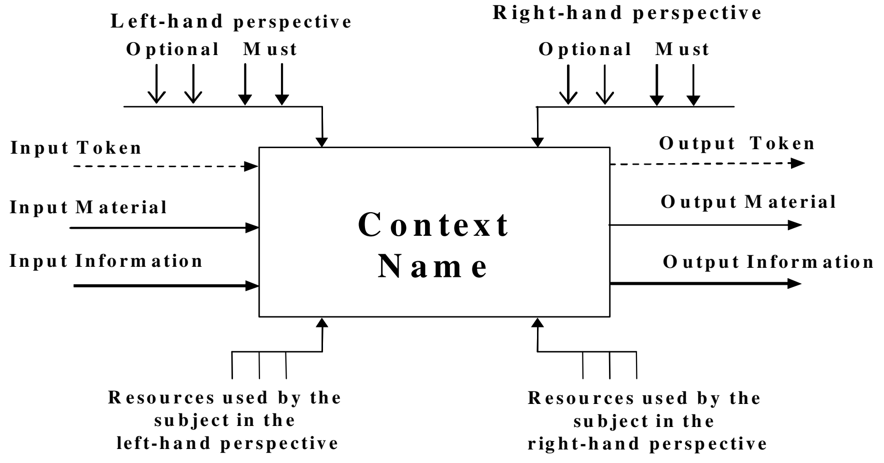


Fig. 6 Context in MCM.

We can imagine an interface sandwiched in between the two perspectives of the context; these two perspectives interact to carry on collaboration by passing the "object upon which the action is performed" through the interface. In the MCM terminology the object that is passed through the interface is called “material”. It could be a piece of machinery, documents, persons, etc., i.e. anything from the physical world that usually has mass or weight. Material in MCM corresponds to the “transactions” in the domain of discrete event simulation. Associated with the material is the “information” – a set of properties of the material. In most practical cases, information is the documentation that accompanies the material as the latter is processed at the context. Another important feature of the interface is that it produces “token”. Token is a collection of facts which are mutually recognized and acknowledged by the two collaborating parties. Tokens can be transmitted to different contexts as telecommunication or as face-to-face communication.

The context, then, is the basic unit in collaboration work. The actors in the context are the collaborators, the action is the service that is being offered and the “object that is acted upon” is the material with its accompanying information. Finally, the token completes the picture of the collaboration between the right-hand and the left-hand perspectives of the context. MCM is the collective linking of contexts. The flow of token, material and information (hereafter, TMI) from context to context is governed by a set of junctions, whose properties and functions are described below.

3.2 Junctions in MCM

Junctions are interspaced among the contexts in MCM and perform the function of directing TMI flow from the output of a context in the upstream to the input of the related contexts in the downstream. They come under two major categories: divergence junctions and convergence junctions. Each category is further divided into individual types of junctions. It should be noted that since T, M and I do not interact with one another while in transit from one context to another, do not share any junction. A single junction strictly handles one and only one of T, M or I.

3.2.1 Divergence junctions (Fig. 9 & Fig.10)

These junctions have a single input and multiple outputs. Their job is to “diverge” as it were, token or material or information arriving from a previous context or junction and direct it towards the junctions or contexts in the next stage of the MCM. Individual members of this category are:

- **Duplication junction (Du)**

  The duplication junction makes identical copies of the entities that arrive at the input and direct the copies to their respective destinations via the output. Since physical material cannot be duplicated upon receiving service at the context, duplication junctions for material do not exist.

- **Decomposition junction (De)**

  Material and information sometimes fragments into constituent part as and when service demands. This aspect is taken care of by the decomposition junction. However, since token represents a fact, it cannot be decomposed. Hence, decomposition junctions for the token do not exist in the junctions repertoire.
- **Branch junction (Br)**

  The branch junction performs the task of providing a forked route to the entities. TMI are all capable of traversing the branch junctions.

3.2.2 Convergence junctions (Fig. 11 & Fig.12)

These junctions have multiple inputs and a single output. Their job is to "converge" as it were, token or material or information arriving from different junctions and direct it towards a single context in the next stage. Individual members of this category are:

- **Synchronization junction (Sy)**

  This junction times the arrival of the different inputs. The faster arrivals wait for the slower ones before emerging as a single output to the next stage.
- **Serialization junction (Se)**

  This junction acts as a collector that collects all the inputs and lets them out as a single unified output.

3.3 MCM example

We consider the real world example of a general clinic to illustrate the MCM modeling technique (Fig. 7). The clinic is a Collaborative System wherein doctors, nurses and other personnel collaborate to offer service to patients. Reception, Diagnosis, Prescription, Medical Examination, Accounts are the contexts that are linked together to form the MCM representing the medical system. The patients who enter the clinic to receive medical service are the "material" that flow through the contexts, the case-papers and their contents are the "information" and the token is the implicit or explicit communication between the left-hand and right-hand perspectives of the context, that relay the fact that "the left-hand perspective's task is over; the right-hand perspective may begin his or her task". The branch junctions indicate that some patients (accompanied with their respective information and implicit tokens) will go for a medical test (X-ray, blood test, etc.) after diagnosis, some will go to the prescription counter while others will go straight to the accounts counter before leaving the clinic. The serialization junctions show that all patients have to visit the accounts counter on their way out of the clinic. At the bottom of each context is a list of resources (including the personal) that are required to perform the task associated with each context.

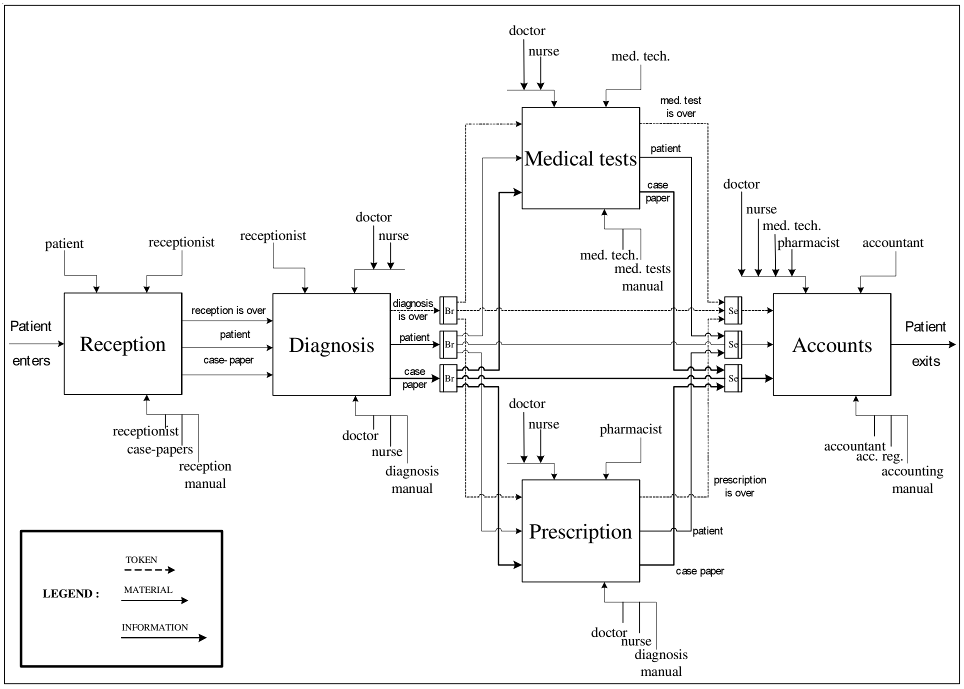


Fig. 7 MCM model of a general clinic.

4 Bottlenecks in MCM

MCM is a network of contexts and junctions. Junctions direct the flow of TMI through the exchange of which the contexts interact with one another. Due to the limitations on the service time of each context and due to the problems in the flow of TMI, bottlenecks can occur at the contexts and at the junctions. Conventionally, bottlenecks are locations of congestions in a network. However, we consider any situation of the context (server) that does not allow the context to be operating in its optimum range as a bottleneck. Thus, under-utilization of the context is also a bottleneck. In this section we formally define the bottlenecks that can occur at these contexts.

4.1 Definition of Context bottlenecks

The context can be functionally compared to a M/M/1 server. The only difference is in the number of input and output limbs. While a conventional M/M/1 server [9] has one input and one output, the context has three inputs and three outputs that correspond to the flow token, material and information flow (Fig. 8).

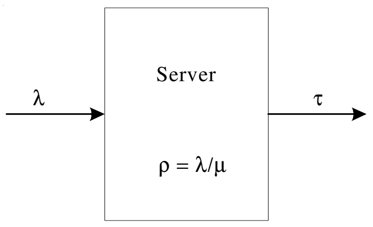


Fig. 8a Single M/M/1 server.

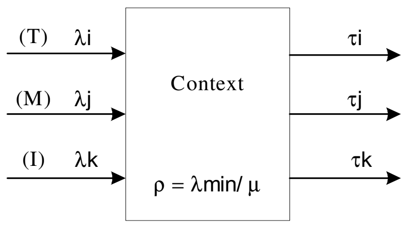


Fig. 8b 3i/p & 3o/p MCM context.

For a single M/M/1 server the general notations are:

λ : average arrival rate of entities for service
μ : average servicing rate of server
τ : average throughput of server
ρ : average utilization rate of server
q : average queue length of entities in front of server
r : branching probability of entities at branch junction

We employ the same notations for the context, too. In the case of the context, however, no service can begin without the arrival of all three of λi, λj, λk. Service will begin only when the slowest of the three entities has arrived at the context. The synchronization of the three inputs at the junction, therefore, imposes the condition,

$$\lambda = \min(\lambda i, \lambda j, \lambda k)$$

With this, the utilization rate of the context becomes :

$$\rho = \lambda min \ / \mu$$

This implies that with the increase in λmin or with the decrease in μ, ρ increases steadily. From expert's heuristics (failure-to-safety aspect) $\rho > 0.7$ is an indication of the possibility of a bottleneck. As λ rises, in addition to rising ρ, TMI queues begin to develop in front of the contexts; i.e. T, M and I arrive independently at the context, form a TMI set and wait for service. After sufficient lapse of time, the TMI sets develop TMI queues; $q > 1.0$, is another landmark indicating the possibility of bottleneck, from the failure-to-safety aspect [10]. On the other hand, ρ

< 0.3 is an indicator of underflow of TMI or under-utilization of the resources in collaborative system. Finally, what distinguishes contexts in MCM from ordinary M/M/1 servers is the three-input-three-output TMI scheme. Due to lack of uniformity in the flow of TMI at a given context, a state of imbalance is created. In other words, the time-lag in the arrivals of T, M and I results in independent T-queue, M-queue and I–queue. This imbalance, in addition to overflow and underflow, is another aspect of a bottleneck.

Summarizing the above measurable parameters, we formally define a bottleneck to be a context that has at least one of the following characteristics.

1. ρ is low ($\rho < 0.3$)
2. ρ is high ($\rho > 0.7$)
3. TMI imbalance (at least one of qi, qj, qk > 1.0)
4. (TMI) q is long (q > 1.0)

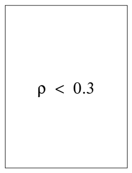
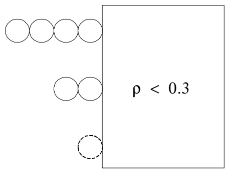
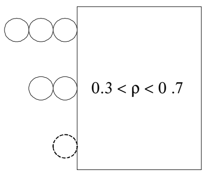
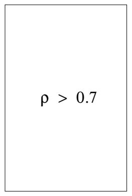
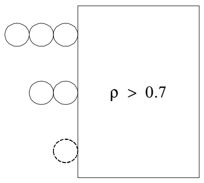
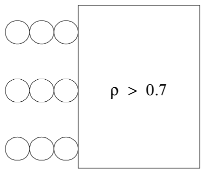
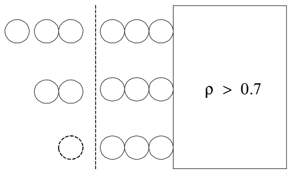
4.2 Definition of Junction bottlenecks

MCM is a network of contexts and junctions. Contexts are servers in nature, thereby sharing the properties of servers and are subject to bottlenecks under adverse conditions. Junctions, too, in peculiar cases, can become locations of bottlenecks. Junctions, in general, are passive entities. They allow TMI to pass through them according to the pre-established rules. There are no ρ and μ concepts defined for junctions, hence no possibility of bottleneck formation. Synchronization junction (Sy), however, has to be seen in a different light. Sy synchronizes two or more entities. The entities with a higher arrival rate have to wait at the junction for the entities with lower arrival rates. Depending on the difference in the arrival rates, this may induce queues at the Sy junction. Sy, therefore, has the possibility of becoming a bottleneck due to imbalance. This is indicated by q > 1.0 for token or material or information.

4.3 Classification of bottlenecks

As seen above, there are three main causes of bottlenecks: TMI overflow, TMI imbalance and TMI underflow. Due to these causes, bottlenecks appear as high utilization of the context, low utilization of the context or as long queues in front of the contexts or as combination of these. We classify the bottlenecks that can occur at a context as follows: If a bottleneck has only one of these characteristics, it is a *simple* bottleneck; if it has two of the above characteristics, it is a *compound* bottleneck; a combination of three characteristics results in *complex* bottlenecks, while combination of all the four characteristics is not possible. Furthermore, the Sy junction bottleneck is always a simple bottleneck due to imbalance. (Refer to Table 1; the circles represent the T,M,I queues in front of the contexts).

Table 1 Bottleneck types.

	SIMPLE	COMPOUND	COMPLEX
UNDERFLOW	 ρ - low	 ρ - low & TMI - imbalance	Bottleneck Not Possible
IMBALANCE	 TMI - imbalance	Bottleneck Not Possible	Bottleneck Not Possible
OVERFLOW	 ρ - high	 ρ - high & TMI - imbalance  ρ - high & Q - high	 ρ - high & Q - high TMI - imbalance

5 Resolution of bottlenecks

By analyzing the GPSS simulation data, the ES checks for bottlenecks at the contexts and junctions and displays them to the user. When the user selects a particular bottleneck for resolving, the ES makes extensive use of the knowledge base to draw a parameter tuning plan that will resolve the bottleneck in question. The inference engine of the ES uses the following qualitative rules in coming to its conclusion. First it applies the "context rules" to the bottleneck context; but the bottleneck cannot possibly be resolved by changing the context (i.e. local) parameters; so the inference engine moves upstream and checks for junctions that may be the source of the bottleneck; it applies the "junction rules"; if the resolution is not achieved at this stage, then it moves on to the previous contexts, applies the rules and so on. Moreover, at each stage the inference engine determines the mini-structure in which the context is located and thereby applies the "mini-structure rules". The nature of these rules and the instances of their application is discussed below.

5.1 Knowledge-Based Qualitative Rules

The rules which control the functioning of the inference engine of the ES are qualitative in nature. We intuitively know that the variables of a context are related by some kind of a monotonic relationship. It is natural to think, for instance, that ρ can be decreased by increasing μ or decreasing λ. Knowledge representation in the form of qualitative rules is a way of concretizing these intuitive qualitative relations [11],[12]. When the value of a certain parameter is found to be high, the rule simply states, "increase or decrease the controlling parameter". Again, when the value of a certain parameter is found to be low, the rule simply states, "increase or decrease the controlling parameter". There are no quantitative calculations performed. The landmark values are sufficient to drive the inference engine. Below we group the rules into three categories depending on when and how they are applied as the ES goes about resolving a given bottleneck. Further, by intuition we know that in the case of overflow, one input could be high, two inputs could be high or all three inputs could be high. Table 2 summarizes this qualitative way of thinking.

5.1.1 Knowledge-based rules for contexts in isolation

Depending on the state of q and ρ, increase or decreannse λ and/or μ. In this set of rules, there are straight forward rules like Rule 1 which states that all inputs should be reduced when ρ is high; however, there are also subtle rules like Rule 4, Rule 5, etc., which state that even the low inputs have to be decreased when ρ is high.

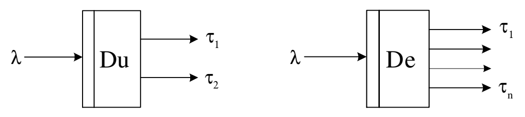
Table 2. Qualitative rules for a context (Rules 1-10) .

Rule	Symptom				State of	Solution			
	qi	qj	qk	ρ	λi, λj, λk	λi	λj	λk	μ
1	1	1	1	H	overflow	↓	↓	↓	↑
2	0	1	1	H	overflow/imbalance	↓	↓	↓	↑
3	0	1	1	M	imbalance	○	↓	↓	○
4	0	1	1	L	underflow/imbalance	↑	○	○	↓
5	0	0	1	H	overflow/imbalance	↓	↓	↓	↑
6	0	0	1	M	imbalance	○	○	↓	○
7	0	0	1	L	underflow/imbalance	↑	↑	○	↓
8	0	0	0	H	overflow	↓	↓	↓	↑
9	0	0	0	M	optimum flow	○	○	○	○
10	0	0	0	L	underflow	↑	↑	↑	↓

H: High; M: Medium; L: Low; ↑: Increase parameter;
↓: Decrease parameter; ○: parameter is OK.

5.1.2 Knowledge-based rules for junctions

Rules 11-12: Du, De junctions



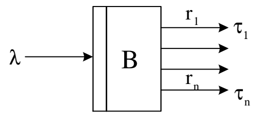
$\tau_1 = \lambda,\ \tau_2 = \lambda, \ldots\ldots \tau_n = \lambda$

Fig. 9 Du & De junctions.

These two junctions are passive, allowing the input to appear as output without any modification. The rule states:

i/p to Sy is not the least	→	no effect on o/p
i/p to Sy is one of the least	→	no effect on o/p
i/p to Sy is the least	→	o/p will increase

Rule 13: Br junction



branching fractions : $r_1, r_2, \ldots\ldots\ldots r_n$

output : $\tau_1 = \lambda^* r_1,\ \tau_2 = \lambda^* r_2 \ldots\ \tau_n = \lambda^* r_n$

Fig. 10 Br junction.

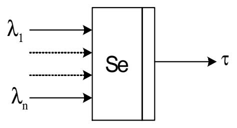
a. If $r_1, r_2 \ldots r_n$ are modifiable then,

Decrease r_1 to decrease τ_1 : $d\tau_1 = -\ \leftarrow\ dr_1 = -$
Increase r_1 to increase τ_1 : $d\tau_1 = +\ \leftarrow\ dr_1 = +$
Decrease r_2 to decrease τ_2 : $d\tau_2 = -\ \leftarrow\ dr_2 = -$
Increase r_2 to increase τ_2 : $d\tau_2 = +\ \leftarrow\ dr_2 = +$
……………
Decrease r_n to decrease τ_n : $d\tau = -\ \leftarrow\ dr_n = -$
Increase r_n to increase τ_n : $d\tau_n = +\ \leftarrow\ dr_n = +$

b. If r1,r2…rn are not modifiable then,

Decrease λ to decrease $\tau_1 \ldots \tau_n$: $d\tau_1 \ldots d\tau_n = -\ \leftarrow\ d\lambda = -$
Increase λ to increase $\tau_1 \ldots \tau_n$: $d\tau_1 \ldots d\tau_n = +\ \leftarrow\ d\lambda = +$

Rule 14: Se junction



$\tau = \mathrm{sum}(\lambda_1, \lambda_2, \ldots\ldots \lambda_n)$

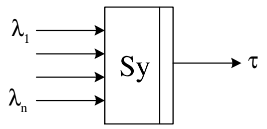
Fig. 11 Se junction.

The Se junction acts as an accumulator. Its output is equal to the sum of all the inputs.

a. To decrease the output, decrease the input with highest value.
b. To decrease the output, decrease the input with higher values.
c. To decrease the output, decrease the appropriate number of inputs (with high or low values).
d. To increase the output, increase the input with lowest value.
e. To increase the output, increase the input with lower values.
f. To increase the output, increase the appropriate number of inputs (with high or low values).

The choice of the rules from each of the above two sets (a-c) and (d-f) will be decided by the feasibility and advisability conditions (section 4.2).

Rule 15: Sy junction



$\tau = \min(\lambda_1, \lambda_2, \ldots\ldots \lambda_n)$

Fig.12 Sy junction.

The inputs with higher arrival rates wait for those with lower arrival rates, thus forming queues at the Sy junction. Output is determined by the input with lowest arrival rate.

To change the output:

a. To decrease the output, decrease the input with lowest value.
b. To increase the output, increase the input with lowest value.

To resolve the queues:

c. Decrease the inputs with higher values.
d. Increase the inputs with lower values (this will, however, increase the output).

5.1.3 Knowledge-based rules for contexts and mini-structures

The rules enlisted in the above sections hold good for contexts and junctions in isolation. However, these rules do not hold for certain configurations in which the context finds itself in the network. For this reason the ES has to first determine the mini-structure (i.e., the sub-structure in the immediate vicinity of the context in question) and then apply the appropriate rules. This mini-structure knowledge is explicitly obtained from the MCM dafter and is structured in the form of mini-structure rules. In all, there are twelve mini-structure rules that govern performance improvement of MCM[13]. For the sake of brevity, here we list just a few of the representative mini-structure rules and indicate how they are applied to solve overflow bottlenecks. The ES employs the same rules to resolve underflow and imbalance bottlenecks, too, albeit with minor variations.

Rule 16: Contexts in tandem (Fig. 13a)
In case of directly linked contexts, there are no junctions in between. The problem of $\rho 2$ being high can be solved only by decreasing τin when $\mu 2$ cannot be modified. The rule states:

decrease τ_{in} to decrease ρ_2 : $\delta\rho_2 = - \quad \leftarrow \quad \delta\tau_{in} = -$

Rule 17: Contexts in loop (Fig. 13b)

When C2 or C3 is a bottleneck, the rules for contexts in isolation would state that τ_{12} or τ_{13} be changed; however, as long as tin remains unaltered, the input to the loop will be constant; therefore mere changes in τ_{12} or τ_{13} will not rectify these bottlenecks. This mini-structure rule overrides Rule no.13 (rule for branch junctions) and states:

To resolve C2 bottleneck, do not change r
To resolve C3 bottleneck, do not change (1-r)

Rule 18: Bottleneck context within a loop (Fig. 13c)

When C2, which is inside the loop, becomes a bottleneck, the only way to resolve it is to change the input to the loop. The rule states:

decrease τ_{in} to decrease ρ_2 : $\delta\rho_2 = - \leftarrow \delta\tau_{in} = -$

Rule 19: Bottleneck context immediately after the loop (Fig. 13d)

When C3, which is immediately outside the loop, becomes a bottleneck, the output of the loop has got to be decreased to decrease the input to C3. The rule states :

decrease τ_{in} to decrease ρ_3 : $\delta\rho_3 = - \leftarrow \delta\tau_{in} = -$

Rule 20: Complex mini-structure-1 (Fig. 13e)

When C3 is a bottleneck, it cannot be improved by manipulating the branch outputs. The only way is to change the external input to the network. The rule states:

decrease λ_{in} to decrease ρ_3 : $\delta\rho_3 = - \leftarrow \delta\lambda in =$

Rule 21: Complex mini-structure-2 (Fig. 13f)

When C4 is a bottleneck, three distinct cases arise:

Case 1: $\tau_{24} >> \tau_{14}$ decrease τ_{24} to decrease ρ_4 : $\delta\rho_4 = - \leftarrow \delta\tau_{24} = -$
Case 2: $\tau_{24} < \tau_{14}$ decrease τ_{14} to decrease ρ_4 : $\delta\rho_4 = - \leftarrow \delta\tau_{14} = -$
Case 3: $\tau_{24} \geq \tau_{14}$ decrease τ_{in} to decrease ρ_4 : $\delta\rho_4 = - \leftarrow \delta\tau_{24} = -$ or
decrease τ_{14} to decrease ρ_4 : $\delta\rho_4 = - \leftarrow \delta\tau_{14} = -$

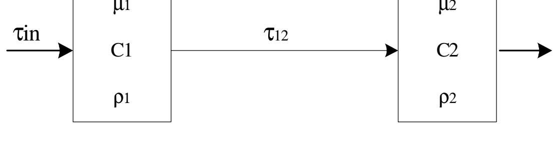


Fig.13a Contexts in tandem.

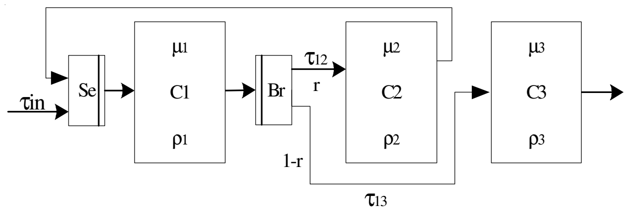


Fig.13b Looped bottleneck.

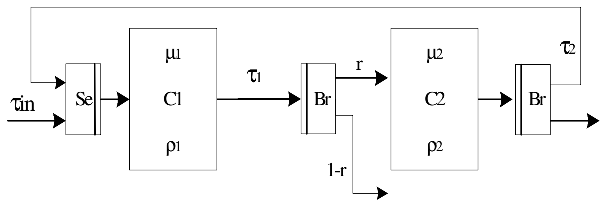


Fig.13c Bottleneck inside the loop.

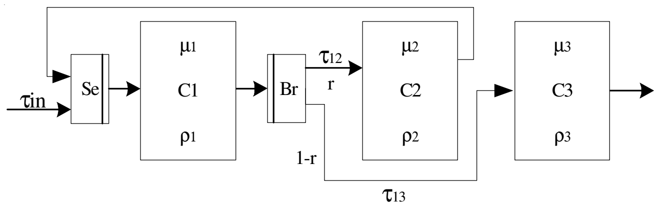


Fig.13d Bottleneck outside the loop.

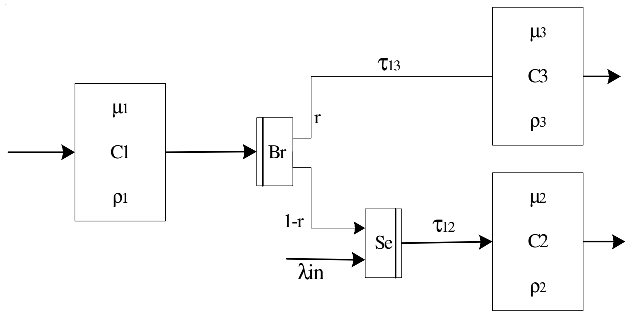


Fig.13e Complex mini-structure-1.

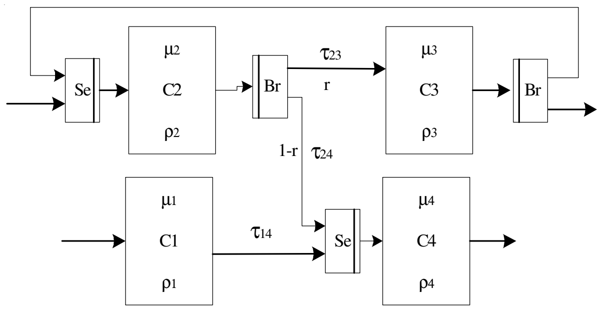


Fig.13f Complex mini-structure-2.

5.2 Guidelines for applying knowledge-based qualitative rules

The qualitative rules listed in the above section cannot be arbitrarily applied. There are conditions that restrict the application of these rules. The ES has to check the upstream sub-structure for the feasibility condition and the downstream sub-structure for the advisability condition before finalizing the tuning plan. (ES has access to the entire structure of the system in its knowledge base which makes feasibility queries and advisability suggestions possible). The section below discusses the feasibility and the advisability conditions.

5.2.1 Feasibility condition

The ES does not have total liberty in working out its strategy. Often it is not possible to resolve a context bottleneck, because the nature of the system is such that the context parameters cannot be changed. In such cases, the ES has to traverse upstream looking for parameters in the upstream contexts that can be changed so as to effect a change at the desired context. If C0i is the bottleneck to be resolved then the upstream substructure of C0i consists of all the contexts and junctions the throughput of which directly or indirectly affects C0i. The ES collects a set of parameters in the upstream by changing the values of which the bottleneck at C0i can be resolved. But the system may be such that certain parameters cannot be changed. By interacting with the user, ES determines which of the parameters are open to change. Let us call this parameter set as the feasible set.

5.2.2 Advisability condition

In resolving the bottlenecks, more important than feasibility condition, is the advisability condition. At times, it may be feasible to change a parameter, but ES may judge that it is not advisable to change it for fear of exerting bad influence on downstream contexts. The downstream sub-structure consists of all the contexts and junctions that are affected by the changes made at C0i and, in addition, all the contexts and junctions that are affected by the changes made in the upstream of C0i to resolve the bottleneck at C0i. If the changes made in C0i and its upstream sub-structure will end up in creating fresh bottlenecks or worsen the existing ones in the downstream sub-structure, then the *advisability* condition will prompt the ES to issue a warning to the user. Advisability means coming up with a sound strategy for resolving such that changes made in the network are minimum and consequently undesirable influence on junctions and contexts operating in normal conditions are kept to the minimum. To work out the advisability, the ES needs to work out in detail the propagation of effects (described in the next section) on resolving a chosen bottleneck. The tuning plan is a combination of the parameters that are in the intersection of the feasible and the advisable sets. After the execution of the tuning plan, C0i is improved to the degree in which the parameters in the above set are increased or decreased.

5.3 Propagation of effects

Bottlenecks are resolved by making necessary changes either in the parameters of the bottleneck-context or in the parameters of the context(s) and/or junction(s) that are located in the upstream of the bottleneck-context. The effect of these changes propagates through the MCM network and appear at related contexts and junctions. Sometimes it can adversely affect the normally functioning contexts. The ES needs to enlist the elaborate path followed by the effects on improvement, to notify the user about the advisability of resolving a particular bottleneck. These effects propagate through the junctions and contexts of the MCM network.

5.3.1 Propagation through Junctions

Except for the Sy junction, all junctions are passive entities that are transparent to the flow of TMI. Consequently an increase/decrease in the TMI entities at the input of these junctions will respectively result in an increase/decrease in the corresponding output. As a result, the changes made in the parameters of a context will propagate unscathed through these junctions to the contexts in the next stage.

Sy junction, however, may, at times, act as a block to the propagation of the effects. The output of the Sy junction is equal to the minimum of the inputs. Hence, as long as the changed input is not the previously existing minimum input, it will not propagate through this junction. The opaque property of Sy junction, although desirable, cannot suppress the inevitable prolongation of queues due to the increased inputs. Summarizing the propagation of effects of resolving a bottleneck, on the Sy junction, we have:

i/p to Sy is not the least	→	no effect on o/p
i/p to Sy is one of the least	→	no effect on o/p
i/p to Sy is the least	→	o/p will increase

5.3.2 Propagation through Contexts

When contexts parameters (λ or μ) are changed, the three outputs of the context will change in accordance with the following general principles.

Case 1: $\lambda > \mu$, implies, $\rho \sim 1$

The context is necessarily an overflow bottleneck. The value of τ can be changed only by changing the value of μ. When it is not feasible to modify μ, further increase in λ will cause the context to go in saturation and no effect will be propagated past the context.

Case 2: $\lambda < \mu$, $\tau = \lambda$

The throughput of the context is equal to its input. As long as $\lambda < \mu$, the changes in the value of λ will be totally reflected in the value of τ. Let $\mathbf{C_{dw1}}$ denote the first context in the downstream of the MCM that will be immediately affected by the changes made in the parameters of C_{0i}. Taking as an example just one transmitting limb from C_{0i} to $\mathbf{C_{dw1}}$, we can infer the following from the above guiding principles:

Case 1: $\mathbf{C_{dw1}}$ is a normal context (Fig. 14)

Imbalance is created by the increased output from C_{0i} (increased λ is shown by black circles)

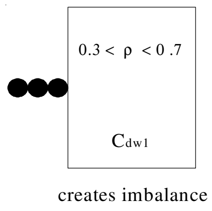


Fig.14 Effect of resolving a bottleneck on normal context in downstream.

Case 1: $\mathbf{C_{dw1}}$ is a normal context (Fig. 15)

Increase in the throughput of C0i creates the following conditions:

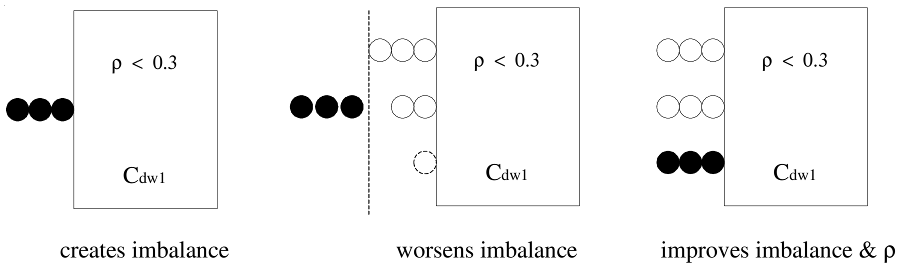


Fig. 15 Effect of resolving a bottleneck on an underflow bottleneck context in the downstream.

1. Imbalance is created $\longrightarrow$ simple bottleneck is converted into compound bottleneck.
2. Imbalance is worsened$\longrightarrow$ compound bottleneck remains the same.
3. Imbalance & r are improved $\longrightarrow$ resolves a compound bottleneck.

Case 3: $\mathbf{C_{dw1}}$ is an overflow bottleneck (Fig. 16)

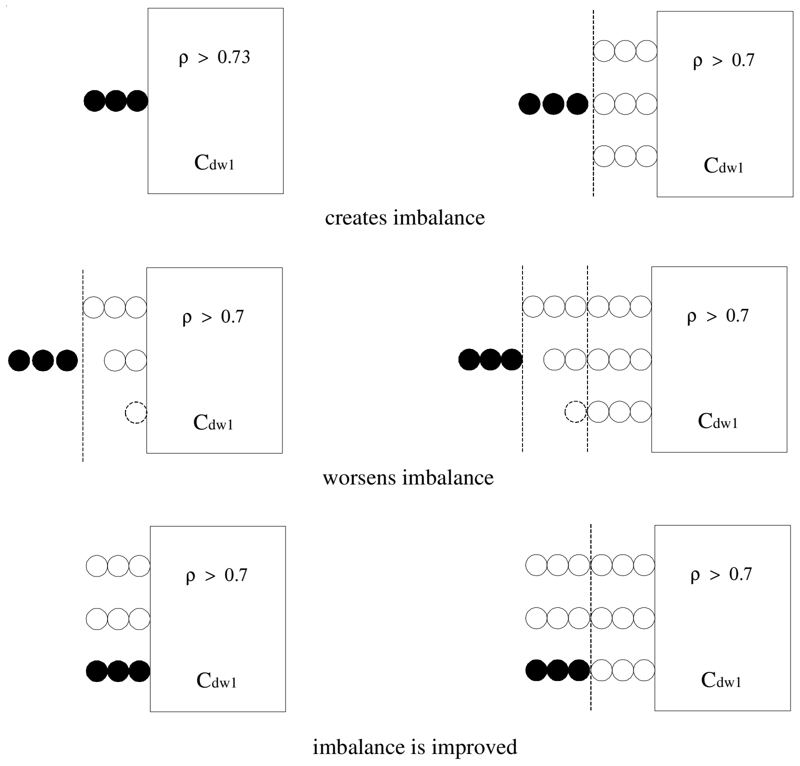


Fig. 16 Effect of resolving a bottleneck on an overflow bottleneck context in the downstream.

Creates an imbalance $\longrightarrow$ simple is converted into compound and compound into complex.
Imbalance is worsened $\longrightarrow$ compound and complex bottlenecks remain the same.
Imbalance is improved $\longrightarrow$ compound bottleneck is converted into simple bottleneck and complex into compound.

6 Collaborative Engineering benchmarking system

To test the operation of our QR-based ES, we consider as a benchmark, a general system in Collaborative Engineering (Fig. 17). There are in all fifteen different service offering contexts that constitute the system. The contexts interact through

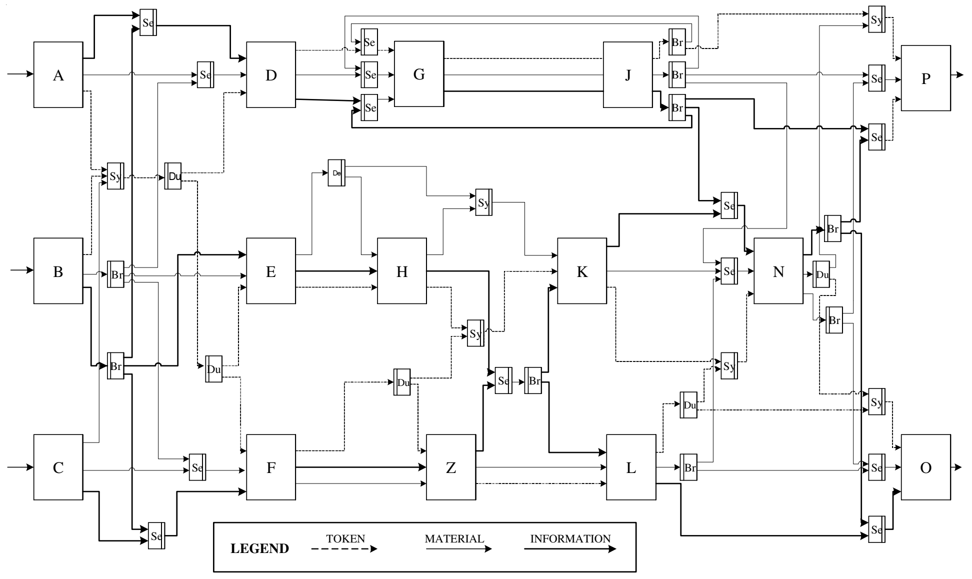


Fig. 17 MCM of a benchmarking system in Collaborative Engineering.

the exchange of TMI. The system presents a complex queueing network with TMI branching, duplicating and synchronizing among the contexts. Diversity in the input to the system and in the flow of TMI gives rise to overflow, underflow and imbalanced bottlenecks, resolution of which is an in-depth challenge to the ES. The ES devises a parameter-tuning plan interacting with the user. Below we present a sample ES-user interaction.

Scenario 1

ES : MCM performance evaluation shows the following bottlenecks.

Compound bottlenecks: Contexts A,D,G,P,E,H,K,C.

Simple bottlenecks: Contexts L, O.

Choose bottleneck to resolve.

User Choice : Context P.

ES : Characteristics of bottleneck at context P are:

$\rho = 0.215$; Tq = 0.000; Mq = 19.764; Iq = 87.685

ES: Can you increase context service time? (feasibility)

User : No.

ES : To increase flow to context P, increase input to context B (mini-structure knowledge, rule 16). In addition this will also resolve bottlenecks at contexts E,H,K due to underflow (advisability). To resolve the TMI imbalance queues, can you

increase the token flow from context N to context P? (feasibility).
User : Yes.
ES : Here is the improvement plan – Increase M & I flow from context B to E; Increase T flow from context N to P.
Improved performance : Characteristics of context P after improvement:
$\rho = 0.532$ $(0.3 < \rho < 0.7)$.
Tq = 0.106; Mq = .696; Iq = 0.443 (Tq,Mq,Iq < 1.0).
In addition, bottlenecks at contexts E, K & K are resolved.

Scenario 2
ES : MCM performance evaluation shows the following bottlenecks.
Simple bottlenecks: Contexts J,B,Z,D. Compound bottleneck: Context E.
Complex bottleneck: Context N.Choose bottleneck to resolve.

User Choice : Context N.
ES : Characteristics of bottleneck at context N are:
$\rho = 0.719$; (TMI)q = 1.576;
Tq = 0.000, Mq = 13.266, Iq = 1.814
ES: Can you increase context service time? (feasibility)
User : No.
ES : Incidentally, do you want to resolve the bottleneck at context J, just in front of context N?
User : Yes.
ES: Is it feasible to change the M & I flows at br junctions in between contexts B & E ?
User : Yes.
ES : This might lead to a slight overflow at context Z. OK?
User : Yes.
ES•@: To resolve the TMI imbalance at context N you have to reduce I flow from context J to context N. This will not have an adverse effect on context P (advisability). OK ?
User : Yes.
ES: Here is the improvement plan.
Decrease the M flow to contexts D & E.
Increase the M & I flow to contexts F.
Decrease the I flow from context J to N.
Decrease the M flow from context L to N.

Improved performance : Characteristics of context N after improvement:
$\rho = 0.589$ $(0.3 < \rho < 0.7)$; (TMI)q= 0.313 (< 1.0).

$Tq = 0.000$; $Mq = 0.714$; $Iq = 0.671$ ($Tq, Mq, Iq < 1.0$)
Characteristics of context J after improvement: $\rho = 0.594$ ($0.3 < \rho < 0.7$).

7 Conclusion

The three-input three-out contexts that constitute the MCM accurately represent the workflow in a Collaborative System, but pose a formidable task in bottleneck resolution. We have adopted an ingenious method of overcoming the above problem by the application of Qualitative Reasoning. As our example has shown, this method works for any general system whose MCM representation may contain any number of contexts. Although it is semi-automatic, it satisfactorily responds to the benchmark system in Collaborative Engineering and succeeds as performance evaluation & performance improvement tool.

8 Acknowledgement

This research has been supported by the funds from the Open Research Center Project, MEXT of the Japanese Government (2002-2006).

9 References

[1] Cooling, J. 2003. *Software Engineering for Real-time Systems*. Addison- Wesley.

[2] Apte, C. 1986. Using qualitative reasoning to understand financial arithmetic. In *Proc. AAAI'86*. 942-949.

[3] Davies, R. 1985. Qualitative Reasoning about Physical Systems. In Bobrow, D.G., editor, 1985, *Reasoning about Physical Systems.* The MIT Press, Cambridge, Massachusetts.

[4] de Kleer, J. 1985. In Bobrow, D.G., editor 1985, *Reasoning about Physical Systems.* The MIT Press, Cambridge, Massachusetts.

[5] Shelly, G. B., Cashman, T. J., and Rosenblatt, H. J. 2003. *System Analysis & Design* (5th ed.). Thomson Course Technology, Boston.

[6] Hoffer, J. A., George, J. F. and Valacich, J. S. 2002. *Modern System Analysis & Design* (3rd ed.). Prentice-Hall International, Inc.

[7] Itoh K., Hirota T., Okabe, M., and Kawabata R. 2003. *Foundation of Information System's Technology* (Japanese). Kyoritsu, Tokyo.

[8] Hasegawa, A., Kumagai, S., and Itoh K. 2000. Collaboration Task Analysis by Identifying Multi-Context and Collaborative Linkage, *CERA*, Vol. 8, No. 1, 61-71.

[9] Klienrock, L. 1975. *Queueing systems*. John Wiley & Sons, Inc.

[10] Itoh K., Honiden, S., Sawamura, J., and Shida, K. 1990. A Method for Diagnosis and Improvement on Bottleneck of Queueing Network by Qualitative and Quantitative Reasoning. *Journal of Artificial Intelligence* (Japanese), Vol. 5, No. 1. 92-105. (Jan.).

[11] Rajagopalan, R. 1984. Qualitative modeling in the turbojet engine domain. In *Proc. AAAI'84*.

[12] Hellerstein, J. 1992. Obtaining Quantitative Estimates from Monotone Relationships. In B. Faltings & P. Struss editors 1992, *Recent Advances in Qualitative Physics*. The MIT Press, Cambridge, Massachusetts.

[13] Sawamura, J., Shida, K., Honiden, S., and Itoh, K. 1989. Bottleneck Diagnosis for Queueing Network using Knowledge Engineering. *Journal of Information Processing* (Japanese) Vol. 30, No. 8. 990-1002. (Aug.).