社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＩＣＳ－133 (29)
2003／9／16

Performance Improvement of Collaboration Engineering Systems by the application of Qualitative Reasoning

Tad GONSALVES,　Kiyoshi ITOH　and　Ryo KAWABATA

Information Systems Laboratory, Faculty of Science & Technology,　7-1 Kioicho, Chiyoda-ku, Tokyo, 102-8554 Japan
E-mail:　t-gonsal@sophia.ac.jp,　itohkiyo@sophia.ac.jp,　r-kawaba@sophia.ac.jp

Abstract　This paper discusses the design and implementation of a novel system performance improvement Expert System (ES) with a Qualitative inference engine. The motive for using Qualitative Reasoning is to overcome the computational complexity posed by the triple-input-triple-output contexts interactions in the Multi-Context Map (MCM) queuing network which models the system. The ES analyses the GPSS simulation data of system performance, consults the MCM knowledge base of the system, and with its inference engine driven by qualitative rules draws the parameter-tuning plan to resolve bottlenecks. The ES has been successfully applied in improving a typical benchmarking system in Collaboration Engineering.

Keywords　Collaboration Engineering, bottlenecks, performance evaluation, performance improvement, qualitative reasoning, expert system.

定性推論による協調エンジニアリングシステムの性能改善

タッド　ゴンサルベス、伊藤　潔、川端　亮
上智大学理工学部機械工学科情報システム研究室　〒102-8554 東京都千代田区紀尾井町 7-1
E-mail:　t-gonsal@sophia.ac.jp,　itohkiyo@sophia.ac.jp,　r-kawaba@sophia.ac.jp

あらまし
本論では、協調エンジニアリングシステムの性能評価・性能改善を目的とした，定性推論を含有するエキスパート・システム（ES）の設計や実施方法を提案する．ESの推論エンジンに定性推論を適応する動機は、システムのモデルであるMulti Context Map(MCM)待ち行列ネットワークにある三重の入出力コンテキストの相互作用の改善にかかわる計算量・複雑性を回避するためである．ESは、GPSSシミュレーション・データを分析して，ボトルネックを検出し，システムのMCM知識ベースを参考し，定性的規則を利用してシステム性能改善のためのパラメータ・チューニングプランを作る．このESは，協調エンジニアリングにおけるベンチマーク・システムの評価・改善に十分に達成した．

キーワード　協調エンジニアリング，ボトルネック，性能評価，性能改善，定性推論，エキスパート・システム

1. Introduction

MCM is a descriptive model of a collaboration engineering system, wherein people with differing perspectives collaborate to offer service to customers [1]. The collaborators working in different contexts interact with each other through the exchange of Token, Material and Information (TMI). Service at a context begins when the TMI from the preceding contexts assemble at that context and after the end of service TMI proceeds to the next junction or context in the MCM. MCM queuing 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 a shortage (underflow) to an abundance (overflow). At times service at a context is delayed due to lack of synchronization, i.e., one or two of the entities arrive at the context, but has to patiently wait for the arrival of the third entity before servicing can begin at the context. These and similar host of phenomena give rise to, what is known in network analysis parlance as bottlenecks.

The conventional quantitative approach to resolve the bottlenecks would be to write a couple of equations and solve them. 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. Randall Davies, for instance, discusses an interesting application of QR in tracing faults in electronic circuits [2]. We use the techniques of QP in establishing the qualitative rules that drive the inference engine of the bottleneck resolving ES.

In this short paper we present a scheme of defining and classifying bottlenecks that may generally occur at the contexts and junctions of MCM. Then we discuss the algorithm, qualitative inference used by the ES to resolve the bottlenecks. 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 in the downstream of the improved bottleneck. Towards the end of the paper we consider a collaboration system as a benchmark. The ES analyses the data output from GPSS simulation, refers to the MCM knowledge base of the system and in interaction with the user produces the parameter-tuning plan.

2. 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 two different locations.

2.1 Definition of Context bottlenecks

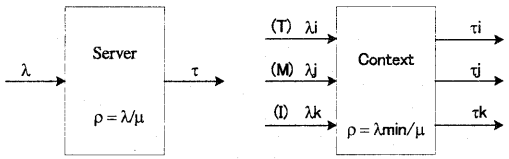


Fig.1a: Single M/M/1 server

Fig.1b : 3 I/P & 3 O/P context in MCM

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 the 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

In case of the context, no service can begin without the arrival of all three of λ_i, λ_j, λ_k. Thus, for synchronization at the junction,

$$\lambda = \min(\lambda_i, \lambda_j, \lambda_k)$$

With this, the utilization rate of the context will be :

$$\rho = \lambda_{min} / \mu$$

This implies that with the increase in λ_{min} 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; $q > 1.0$, is another landmark indicating the possibility of bottleneck, from failure-to-safety aspect [3]. On the other hand, $\rho < 0.3$ 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. This imbalance, in addition to overflow and underflow is another aspect

of 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 q_i, q_j, $q_k > 1.0$)
4. q is long ($q > 1.0$)

2.2 Definition of Junction bottlenecks

MCM is 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 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.

2.3 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; combination of three characteristics results in a *complex* bottlenecks, while combination of all the four characteristics is not possible. Further, Sy junction bottleneck is always a simple bottleneck due to imbalance.

Cause	q-low	q-high	imbalance	q-high & imbalance
ρ low	simple	XXX	simple	XXX
ρnormal	XXX	simple	simple	compound
ρ high	simple	compound	compound	complex

Table 1: Simple, compound and complex bottlenecks

3. 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 draws 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.

3.1 Qualitative Rules

The rules which control the functioning of the inference engine of the ES are qualitative in nature. 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.

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. The table below summarizes this qualitative way of thinking.

3.1.1 Rules for contexts in isolation

Rules 1-10: Qualitative rules for a context

Depending on the state of q and ρ, increase or decrease 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.

case	symptom				state of	solution				
	q_i	q_j	q_k	!	λ_i, λ_j, λ_k	τ_i	τ_j	τ_k	Combination	μ
1	1	1	1	H	overflow				AND/OR	9
2	0	1	1	H	overflow/imbalance	;	;	;	AND/OR	9
3	0	1	1	M	imbalance	○			-	○
4	0	1	1	L	underflow/imbalance	9	○	○	AND/OR	;
5	0	0	1	H	overflow/imbalance	;	;	;	AND/OR	9
6	0	0	1	M	imbalance	○	○		-	○
7	0	0	1	L	underflow/imbalance	9	9	○	AND/OR	;
8	0	0	0	H	overflow				AND/OR	9
9	0	0	0	M	optimum flow	○	○	○	-	○
10	0	0	0	L	underflow	9	9	9	AND/OR	;

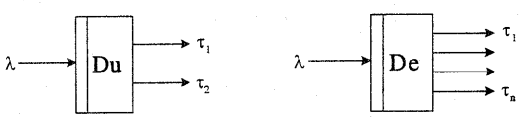
q_i q_j q_k : queue-length of TMI; λ_i, λ_j, λ_k : inter-arrival time of TMI; H : High; M : Medium; L : Low

: decrease the parameter; : increase the parameter; ○: do not change the parameter.

Table 2 : Qualitative rules for a context in isolation

3.1.2 Rules for junctions

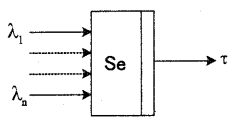
Rules 11-12: Du, De junctions



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

These two junctions are passive, allowing the input to appear as output without any modification. Increase the input to increase the output; decrease the input to decrease the output.

Rule 13: Se junction



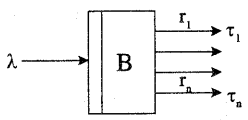
$\tau = \text{sum}(\lambda_1, \lambda_2, \ldots\ldots \lambda_n)$

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.

Rule 14: Br junction



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

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

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

Decrease r_1 to decrease τ_1; Increase r_1 to increase τ_1

Decrease r_2 to decrease τ_2; Increase r_2 to increase τ_2

.....

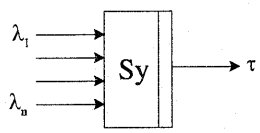
Decrease r_n to decrease τ_n; Increase r_n to increase τ_n

b. If $r_1, r_2 \ldots r_n$ are not modifiable then,

Decrease λ_1 to decrease $\tau_1 \ldots . \tau_n$;

Increase λ_1 to increase $\tau_1 \ldots . \tau_n$

Rule 15: Sy junction



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

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

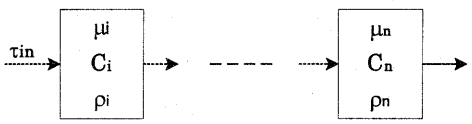
To resolve the queues:

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

3.1.3 Rules for contexts and mini-structures

ES also has the knowledge of each mini-structure obtained from the MCM dafter. Listed below are the rules arranged according to the mini-structures[4].

Rule 16: Contexts in tandem

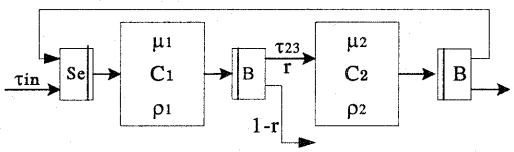


Incase of directly linked contexts, there are no junctions in between.

Increase τ_{in} to increase ρ_n

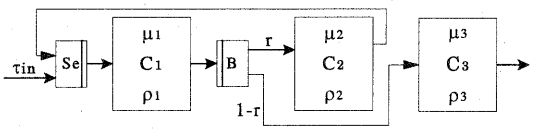
Decrease τ_{in} to decrease ρ_n

Rule 16: Bottleneck context within a loop



do not decrease τ_{23}; do not increase τ_{23}; dec./inc. τ_{in}

Rule 17: Bottleneck context immediately after the loop



decrease tin to decrease ρ_2; increase tin to increase ρ_2

3.2 Guidelines for applying 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 it finalizes the tuning plan. (ES has the entire structure of the system in its knowledge base). The section below discusses the feasibility and the advisability conditions.

3.2.1 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 C_{oi} is the bottleneck to be resolved then the upstream substructure of C_{oi} consists of all the contexts and junctions the throughput of which directly or indirectly affects C_{oi}. The ES collects a set of parameters in the upstream by changing the values of which the bottleneck at C_{oi} 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.

3.2.2 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 C_{oi} and, in addition, all the contexts and junctions that are affected by the changes made in the upstream of C_{oi} to resolve the bottleneck at C_{oi}. If the changes made in C_{oi} 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. 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, C_{oi} is improved to the degree in which the parameters in the above set are increased or decreased.

4. Collaboration Eng. benchmarking system

To test the operation of our QR-based ES, we consider as a benchmark, a general system in Collaboration Engineering. There are in all fifteen different service offering contexts that constitute the system. The contexts interact through the exchange of TMI. The system presents a complex queuing network with TMI

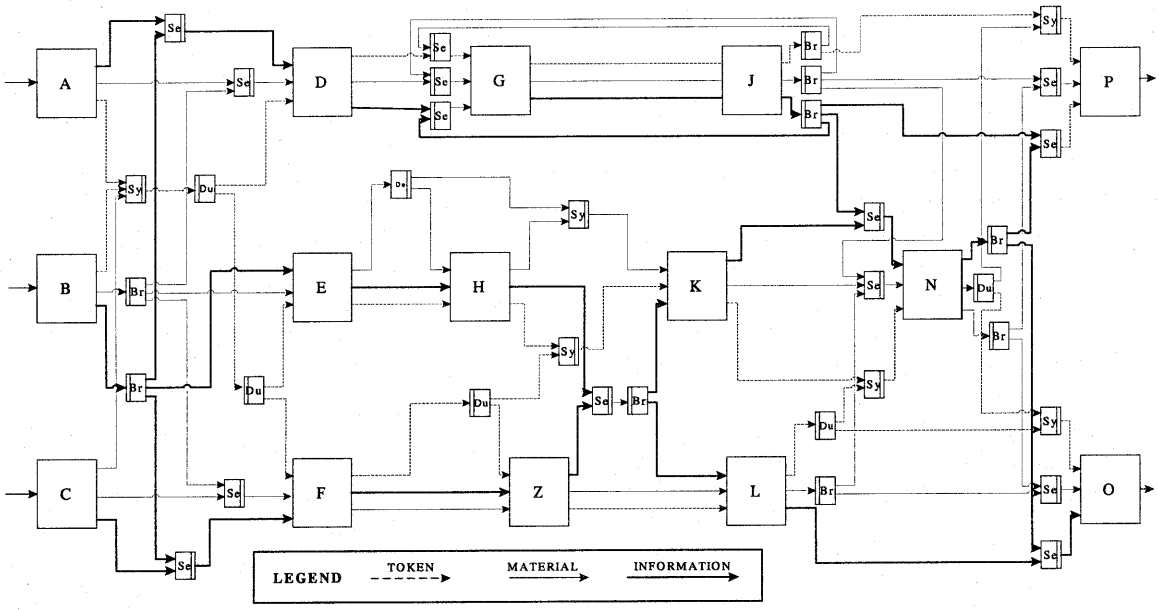


Fig. 3. General System in Collaboration Engineering

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 parameter-tuning plan interacting with the user. Below we present a sample ES-user interaction.

ES : MCM performance evaluation shows the following bottlenecks.
Compound bottlenecks: Contexts A,D,G,P,E,H,K,C,O
Simple bottlenecks: Contexts L, O
Choose bottleneck to resolve.
User Choice : Context N.
ES : Characteristics of bottleneck at context N are:
$\rho = 0.215$; $T_q = 0.000$; $M_q = 19.764$; $I_q = 87.685$
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 N context
$\rho = 0.532$ $(0.3 < \rho < 0.7)$
$T_q = 0.106$; $M_q = .696$; $I_q = 0.443$ $(T_q, M_q, I_q < 1.0)$
In addition, bottlenecks at contexts E, K & K are resolved.

5. Conclusion

The three-input three-out contexts that constitute the MCM accurately represent the workflow in Collaboration Engineering 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 Collaboration Engineering and succeeds as performance evaluation & performance improvement tool.

6. References

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

[2] Randall Davies, Qualitative Reasoning about Physical Systems, D.G. Bobrow, ed., The MIT Press, Cambridge, Massachusetts, 1985.

[3] 伊藤潔，本位田真一，沢村淳，誌田圭介，“定性推論と定量推論を導入した待ち行列ネットワーク診断と改善法，”人工知能学会誌，Vol. 5, No. 1, pp.92-105, Jan. 1990.

[4] 沢村淳，誌田圭介，本位田真一，伊藤潔，“知識工学的手法による待ち行列ネットワークのボトルネック診断，”情報処理学会論文誌，Vol. 30, No. 8, pp.990-1002, Aug. 1989.