Industrial Engineering & Management

Alaya, Ind Eng Manage 2016, 5:4
DOI: 10.4172/2169-0316.1000199

Research Article Open Access

EE-(m,k)-firm: Operations Management Approach In Enterprise Environment

Bechir Alaya*

College of Business and Economics, Qassim University, Buraydah 52571, Saudi Arabia

Abstract

Task management is of a paramount importance because of the daily execution of more or less urgent and important tasks. In this paper, we propose a policy to indicate the necessary tasks and calculate their ranking, using a compromise between the available resources and the quality of service (QoS) granularity in the same task type. We applied a guaranteed technique in order to achieve an intelligent loss of tasks according to the importance of each task. A dynamicity of constraints was then used to attain an increase of availability, performance, reliability and system dependability. The results obtained from the proposed policy reveal that this type of policy can be extremely valuable for companies that wish to focus their efforts and resources to guarantee a satisfactory QoS for their clients.

Keywords: Enterprise environment; Tasks management; Quality of Service guarantee; (m,k)-firm model

Introduction

The enterprise environment is constantly subject to more or less significant disturbances which greatly influence the enterprise performance and its quality of service (QoS). These disturbances are specified by completely uncontrollable variables by the enterprise, because they are numerous and have a very different nature.

In addition these variables include, of course, the immediate competition from its main competitors within the enterprises. These variables considerably show the current operation of the system and its performances. Particularly, the performance of company compared to its immediate competitors or compared to the optimal performance required by managers.

In this paper, we founded our study on the above comparison of performance, as the system performances are fixed by the enterprise's managers. Enterprise modeling remains always a challenge, despite the significant advances in modeling technology. Such a modeling must cover all aspects of the system studied: functional, physical, informational and organizational. It is necessary to make the favorable choice of model of data and processes and to develop an integration platform to exploit the partial models.

The modeling for different points of the company is necessary. Such a modeling is part of the answer to the need for integrating the production functions and specially the maintenance and QoS guarantee. The policy that we propose can be generalized and therefore applied to all the enterprise functions.

Companies deal with large volumes of data that require real-time processing, i.e., they must be completed before given deadlines.

The goal is to guarantee an acceptable QoS in the services presented to the clients. The adapted systems to the management of these kinds of data with QoS guarantees are real-time database systems (RT-DBSs) [1,2].

Because of some similarities existing between the data management in RTDBSs and in companies [3], we propose to adapt some results obtained on the QoS management in RTDBSs to manage companies' performances.

Our main objectives were to design an efficient model which responds to performances requested by clients and managers and provide the QoS guarantees and a certain robustness when clients' demands grow up quickly or when the company resources become congested [4].

We present a model based on (m,k)-firm model [5-8] to take into account the congestion of systems workloads in companies.

The remainder of the paper is organized as follows: Section 4 describes some key related works to QoS management in enterprise environment; Section 5 represents the QoS management in enterprise environment using the (m,k)-frim model; Section 6 explains how the proposed EE-(m,k)-firm policy provides policy-driven QoS management services and mechanisms for tasks management in enterprises; Section 7 defines a method to guarantee an intelligent loss of tasks according to their importance in the system. Section 8 describes a method to automate the treatment of EE-(m,k)-firm constraints, according to the task type. Section 9 analyzes the results of the experiments we conducted to evaluate the effectiveness of EE-(m,k)-firm policy in providing QoS management in constrained and contended environments in enterprises and Section 10 presents the conclusion and some remarks.

Related Work

The QoS management in enterprise environment [9-11] is a topical problem. Recently, several studies have been based on this topic. In [12], the authors proposed that to reach a superior performance, it is necessary to suggest (i) the adoption of appropriate strategic behaviors to client, to competitor and to technology and (ii) the targeting of the appropriate market segments, notably early adopters, innovators, early majority, laggards and late majority. In their proposition, the strategic behavior of corporate performance relationship is subject to the

***Corresponding author:** Bechir Alaya, College of Business and Economics, Qassim University, Buraydah 52571, Saudi Arabia, Tel: 96663800050; Fax: 96663800565; E-mail: alabechir@yahoo.fr

Received October 11, 2016; **Accepted** November 09, 2016; **Published** November 11, 2016

Citation: Alaya B (2016) EE-(m,k)-firm: Operations Management Approach In Enterprise Environment. Ind Eng Manage 5: 199. doi:10.4172/2169-0316.1000199

company's strategy by examining this relationship on high technology markets and considering further contribution of the appropriate target market selection. This approach provides useful orientation to business managers to the steps that they should take to augment their performances.

In [13], the authors proposed a technique to improve the risk analysis in Enterprise Resource Planning (ERP) [14]. They aimed to obtain a more structured systematic model of the different relationships between the risk factors/effects associated with ERP projects and attain a better understanding. The major objectives of their work were to (i) allow a collaborative approach to risk analysis, (ii) help the administrators in treating and controlling project risk and (iii) help the administrators to comprise the links between the development of a relevant risk analysis strategy and the evaluation of a global risk index for each factor used.

The authors of [3] accentuated the effect of interpersonal factors on company's performance through the relationship quality and the intervening roles of intercompany trust. The authors justified that trust plays an instrumental role in enhancing the components of the inter-firm relationship quality. They showed that inter firm relationship quality is positively related to superior financial performance, and most of the associations between each of the interpersonal factors and inter firm trust were moderated by the importer's size and foreign supplier's origin as well as the length of the relationship and which party initiated the relationship.

Demand Response Management (DRM) which is a key component of the future smart grid Demand Response Management (DRM) was the subject of [15]. In fact, the author studies DRM with different public service companies. First, the links between residential users and utility companies are structured as a two-level part. Notably, interaction between the residential clients is expressed as an evolutionary game, while the competition between the public service company is formulated as a non-cooperative game. Second, the authors proved that the proposed approaches are capable to converge to their own equilibrium.

An approach was suggested in order to provide the network and security controls with transparency that many enterprise clients would like [16]. In this paper, the authors demonstrated that cloud computing has a large potential to change how enterprises function and treat their information systems [17]. They proposed a virtual cloud pool (VCP) abstraction in order to process enterprise data center resources, logically unify cloud and present the vision behind CloudNet. It equally permits the pooling of resources through data centers to give the enterprises the ability of having cloud resources that are adaptive and dynamic to their requirements.

In [18], the authors proposed that the likelihood and amount of earnings management do not differ from working conditions, proposing that firms using less accrual earnings management tend to completely offset by increasing real approaches. Consequently, effectively reporting environments do not perforce decrease the entire earnings management. In [19], the authors treated the quality of earnings management when the capacity to manage accruals is constrained by high quality auditors. For firms that meet or just beat earnings benchmarks and firms that issue seasoned equities, the authors find that city-level auditor industry expertise and audit fees are associated with higher levels of real earnings management. These studies suggest an unintended consequence of high quality auditors constraining accrual earnings management, namely firms resorting to potentially even more costly real earnings management.

Depending on the requirement of enterprise, several types of information systems have been improved for various goals [20]. A study in [21] attempted to demonstrate the role of each type of information systems in firms' organizations. As Figure 1 shows, according to O'Brien and Marakas [22] the applications of information systems that are implemented in today's business world can be classified in several different ways. In enterprises world, there are varieties of information systems such as, Office Automation Systems (OAS), Expert System (ES), Transaction Processing Systems (TPS), Management Information Systems (MIS), Executive Information Systems (EIS), Decision Support System (DSS), etc. Each type of information system has a specific objective in management operations and in organizational hierarchy [23].

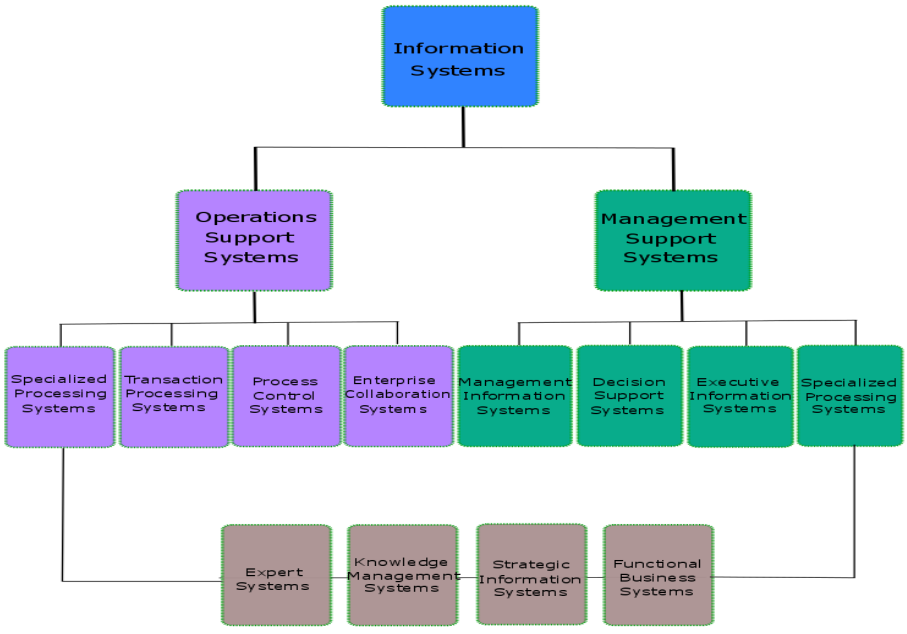

Figure 1: Applications of information systems.

Other researches were proposed in [24,25] to present customized views of enterprise systems to various stakeholders according to their competencies and requirements. For a better QoS, they were interested in developing and improving the services and languages offered by such tools on a continuous basis. They discuss the weaknesses and strengths of different approaches [26] interested in language development and proposed a modeling framework more able to support the main extension scenarios currently found in practice (Figure 1).

The authors of [27] presented a quadratic integer programming model for the problem of scheduling tasks. They based their studies on a telecommunication service company facing seasonal demand and time varying capacity. Their purpose was to meet the promised lead-time with the accusative of balancing the workload. They proposed two variants of a variable neighborhood search approach.

QoS Management in Enterprises Environment

(m,k)-firm method

The recurrence of tasks in real-time systems allow to ignore some invocations (or jobs) using (m,k)-firm constraints. These constraints specify that in a window of k invocations, at least m tasks ($0 \leq m \leq k$) must respect these deadlines [28]. Otherwise, for k tasks, m tasks are required and (k-m) tasks are optional. It may be noted that traditional firm tasks, i.e. strict and not critical, are a special case of (m,k)-firm tasks such that m=k. Indeed, in this case, all invocations of the same task must respect their deadline. Bernat showed through an example why it is best to use two parameters to define this type of constraints [29]. This explains why the ratio of success does not give enough information on executions of tasks: a 90% success ratio is equivalent to 1 instance lost from 10 or 100 instances from 1000. In the first case, the tasks that miss these deadlines are consecutive; consequently, the second case can lead to very inaccurate or erroneous results.

The (m,k)-firm method is characterized by two parameters: the first shows the number of task instances that must respect their deadlines and the second limits the frequency at which the instances can fail. In this case, a task having (9,10)-firm constraints is more critical than a task having (900.1000)-firm constraints. Once (k-m+1) task instances cannot respects their deadline, the task goes to a dynamic failure state. The goal of a system subject to (m,k) -firm constraints is then to limit the number of tasks that go into a dynamic failure state.

Furthermore, it has been shown that the concept of (m,k)-firm constraints is appropriate for specification (management) of QoS of real-time application [30]. Indeed, a system subject to (m,k)-firm constraints provides different QoS levels corresponding to different values between m and k ($0 \leq m \leq k$). In other words, the QoS increases depending on the number of actual optional invocations. On the other hand, it is apparent that a system under (m,k)-firm constraints requires less resources than a classic system because some invocations may be disregarded. To effectively manage the tasks under (m,k)-firm constraints, new scheduling algorithms have been proposed [31]. They are divided into two main groups.

- **Dynamic algorithms:** the task priority is determined based on system state. For example, the Distance Based Priority (DBP) protocol calculates the probability of a task falling in a failed state from the task history.
- **Static algorithms:** priority is determined in an offline manner using a fixed parameter, such as the success ratio m/k. An exemplary protocol is the static Enhanced Rate Monotonic (ERM) which determines the mandatory and optional task instances in advance. Briefly, the static algorithms provide a deterministic vision of the system, while the dynamic algorithms rather provide a probabilistic vision. The dynamic algorithms take into account any system modification.

(m,k)–firm constraints application in enterprises environment

We proposed to exploit the (m,k)-firm constraints in enterprise environment in order to manage the various tasks criticality levels in different manufacturing processes in industrial enterprises. The production management in industrial enterprises aims to meet the client requests by integrating many constraints: the costs, products quality, deadlines, customer demand, necessary personnel, infrastructure, supply of raw materials, etc.

The focus of this paper is on the products quality constraints and procurement constraints. Procurement is the problem of the availability of raw materials and supplies according to production needs to answer clients' request or to satisfy the requested QoS. The production management must take into account four main types of constraints:

- Commerce system demand and receipts order with quantities, deadlines, quality and price;
- Procurement upstream with suppliers in order to avoid the stock-outs and market losses;
- The management of human resources, available material resources and product QoS;
- The essential problem is that of the production optimization under constraints of QoS guarantee [32].

The main objective in enterprise environment is both quantitative and qualitative: (i) the understaffing is reflected in hiring or recourse to additional time or to interim external staff, and (ii) the qualifications and the competences acquired through experience represent the qualitative aspect which is more delicate to approach and essential to be competitive.

The QoS degradation implies the degradation of system performance in such a way that the system continues to function but with a disequilibrated level of QoS. In an overload situation, the production and QoS degradation is inevitable since clients' demands will always be dropped or delayed although many clients' demands can tolerate some delay if they arrive with a permitted mode. Moreover, the effect on QoS in industrial production depends on how and when the degradation is present.

In, the authors demonstrated that, with the concept of (m,k)-firm model, the ability of real-time tasks to miss some deadlines without drastically degrading the QoS. Our approach consists in adapting the (m,k)-firm model to industrial environments. The proposed method can be described as follows: a task in process of industrial production is constrained by (m,k)-firm requirements if at least m task instances within a range of k consecutive tasks respect their intended deadlines. If more than (k-m) deadline of tasks fail in k consecutive tasks at that moment, we can mention that the tasks will fall in a dynamic failure state. Consequently, the QoS constraints will not be satisfactory for the customers. For each branch of industrial production, the values of m and k vary according to the criticality of tasks and system load.

In practice, the values indicated by the industrial systems are not all of the same importance. For example, in industrial production of aircraft, the importance of the motors temperature is greater than that of the cabin. In this case, it is tolerable to lose some instances of tasks in the cabin, in order to use them in the motors.

Whereas, we can assign (10,100) -firm constraints for motor temperature and (9,10)-firm constraints for the temperature of the cabin. The (m,k)-firm constraints are fixed by the enterprise manager for each branch of enterprise and the values change according to the criticality of these tasks.

Quality of Service - Adaptation and Management

The tasks of an industrial production system are decomposed into several classes according to their tolerance to tasks loss.

We consider three classes of tasks in the industrial environment: critical task, hard non-critical task and optional task. With this technique, which we called (m,k)-firm in Enterprise Environment (EE-(m,k)-firm), we can realize a compromise between the available resources and the QoS granularity in the same type of task.

In this work, we focused on the adaptation of the number of tasks to the system load state. We assumed that measures of the system capacity were available on the one hand and that we had a significant number of client demands on the other hand.

We also assumed a system situation in a production enterprise, whose actual performance is N, was overloaded. We supposed that Optimal QoS (Opt-QoS) was the quality of the client demand necessitating M tasks. In order to be coherent with the system performance, it was requisite to throw (M−N) tasks. Consequently, we had to reduce the quality of the client demand and if necessary, we could remove some tasks. However, the removal without applying a control method would be arbitrary.

The removed tasks are lost from the system, causing QoS degradation, notably if some critical tasks [33] are removed. In this work, we adapted the EE-(m,k)-firm constraints, that serve to discard some tasks but intelligently.

The three classes of tasks were proposed to adjust the QoS requested by the clients based on real system capacity. We proposed that constraints for each task category were fixed as follows: EE-(m_c,k_c)-firm for critical tasks, EE-(m_h,k_h)-firm for hard non-critical tasks and EE-(m_o,k_o)-firm for optional tasks. Notably, m_c tasks must be executed among k_c tasks.

The system capacity was calculated using the formula: $m_c + m_h + m_o$, where m_c and k_c present the constraints of critical tasks .The constraints of different task classes are organized as follows: $m_c > m_h > m_o$. In the industrial environment, during production times we usually propose that $m_c = k_c$, given that these types of tasks are critical and that it is not recommended to lose them.

With the application of our EE-(m,k)-firm policy, we suppose that the required capacity necessary to respond to an enterprise transaction is M. With

(1) $M = k_c + k_h + k_o$,

(2) $N = m_c + m_h + m_o$.

Based on the previous suppositions, we broached the problem of equitable sharing of resources between the different clients asking for a service. We proposed how to equilibrate the QoS at the tasks level in a production enterprise according to the available system capacity. With the QoS degradation, we required to equitably share the resources between all clients that wish to get services with an acceptable QoS.

We began by calculating the required capacity by all the current clients. Then, we calculated the rate that presents the ratio between the available system capacity (N) and the required capacity.

$$Rate = \frac{N}{\sum_{i=1}^{k} DR_i}$$

Given that:

- k present the number of available tasks in the system.
- DR_i present the demanded resource by task i.

Taking an example of a production enterprise that received 5 clients' demands. Each client demand necessitating an optimal number of tasks to attain the best QoS requested.

The numbers of necessary tasks for each client to gain an optimal QoS were 65, 20, 120, 35 and 75. The total capacity needed by the system to meet these demands should be: 65 + 20 + 120 + 35 + 75=315 (tasks).

In case where the system had a capacity to respond to only 290 tasks, then it would not be able to answer all the necessary tasks for an optimal QoS for any client. We must then calculate the Rate ratio as indicated previously, i.e. Rate=(290/315) x 100=92.06%. Then we applied this rate to all three capabilities required 65 × 60=92.06%, 92.06%=20 × 18, 120 × 111=92.06%, 92.06% × 35=32 and 75 × 92.06%=69.

We noticed that the sum of the number of new patches required for each customer was 290 tasks that corresponded exactly to the system capacity in the enterprise.

Some advantages of the fair sharing of actual resources are present such as (i) a bad client does not affect the QoS guaranteed to other clients, (ii) guaranty an optimal performance for all clients and (iii) fair sharing of system resources between all clients in order to control the system congestion.

A Guarantee Method to Intelligent Tasks Loss in Enterprises Environment

In order to guarantee an intelligent loss of tasks according to their importance, we defined a method which describes how a client demand is composed of k tasks. These tasks follow a well-defined organization when the system resources are not available to respond to all necessary tasks to have the requested QoS.

Given the differentiation between tasks, we already proposed that the critical tasks are two categories. In one part, the mandatory and critical tasks (C_1) without which the client demand will not be realizable. In the other part, the critical tasks (C_2) without which, the client demand can be realizable, but the QoS will be extremely poor.

Similarly to the hard and non-critical tasks, we proposed a classification into two categories, namely H_1 and H_2. Finally, for optional tasks that usually reflect the QoS degree we have two classes (O_1, O_2) according their importance for QoS level.

Consequently, The k tasks of a client demand based on EE-(m,k)-firm constraints are represented by a succession of k elements from $\{C_1, C_2, H_1, H_2, O_1, O_2\}$ as described previously.

In enterprise environment, it is difficult to have an approach that guarantees an optimal QoS for all clients arriving to the system. Using this intelligent loss specification, each client can show these EE-(m,k)-firm constraint according to requested QoS and the available system resources. A minimum QoS is guaranteed if at least all critical and mandatory tasks are executed. Notably, if some optional tasks are missed by the system, the degradation will be only to EE-(m_o, k_o)-firm constraints, but not to EE-(m_c, k_c)-firm constraints.

These constraints are extremely appropriate in order to extract all requirement of a client demand. In all cases, a client demand is represented by a succession of critical tasks, hard tasks and optional tasks.

The loss of tasks in a type of critical tasks and/or in a type of hard tasks necessary for a client demand will cause some degradation in the following tasks until a new demand occurs, although the optional tasks loss has no effect. Hence, the tasks necessary to have an optimal QoS to a client demand has the following structure: {$C_1C_2O_2O_2C_1O_1H_2O_1$ - - - - $H_1O_1O_2O_2$}, where all O tasks are optional, H tasks are hard and C are critical.

Consequently, scheduling tasks method must pay more attention to C tasks since they are mandatory, they must also take more care of H tasks since they are hard (Figure 2).

In Figure 2, shows the example of a client demand requiring 33 tasks to get the best QoS. We apply our method EE-(m,k)-firm method

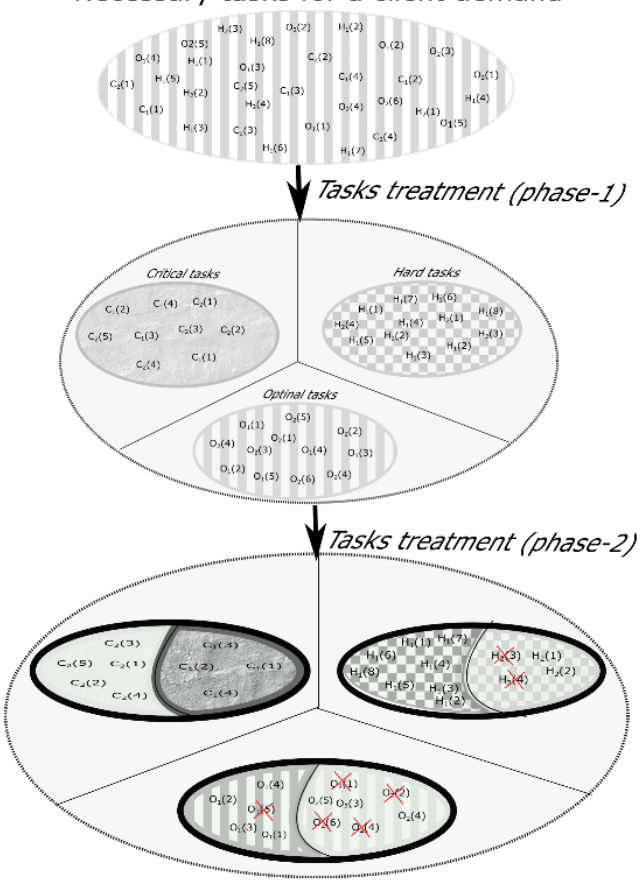

Figure 2: EE-(m,k)-firm application.

to this client demand. We propose that the enterprise system can respond to just 27 tasks, according to system load availability. Without an intelligent approach for the selection of tasks to exclude, the QoS degradation will be probably severe. This figure shows the first and second treatment of tasks, according to EE-(m_C,k_C)-firm, EE-(m_H-k_H)-firm and EE-(m_O,k_O)-firm constraints. We can notice from the same figure two categories of each task type, notably, (O_1,O_2) for optional tasks, (H_1,H_2) for hard tasks and (C_1,C_2) for critical tasks.

We chose this treatment in order to give more precision to EE-(m,k)-firm constraints. Indeed, in some cases, the necessary resources for a hard task can respond at an important number of optional tasks. Similar to critical tasks and hard tasks, we can apply the same treatment method. Each enterprise manager adjusts the EE-(m,k)-firm constraints according to system load and the effect of the loss of each task type. Notably, it can give three categories of each task type, if is it necessary in the system.

With the first treatment, the EE-(m,k)-firm constraints are presented as follow:

- EE-(9-9)-firm for critical tasks
- EE-(10-12)-firm for hard tasks
- EE-(7-12)-firm for optional tasks

The tasks that will be lost are 2 hard tasks and 5 optional tasks. For more prevision to tasks loss, we use the second treatment that gives:

- EE-(4-4)-firm for first type of critical tasks
- EE-(5-5)-firm for second type of critical tasks
- EE-(8-8)-firm for first type of hard tasks
- EE-(2-4)-firm for second type of hard tasks
- EE-(4-5)-firm for first type of optional tasks
- EE-(3-7)-firm for second type of optional tasks

Diagnostic of Task Constraints

Dynamicity of EE-(m,k)–firm constraints

To improve performance, availability, reliability, and system dependability, we applied a method of dynamic treatment of tasks. The objective was to automate the treatment of EE-(m,k)-firm constraints.

The need of responding to critical and hard tasks is most crucial when it comes to sensitive systems where an error can be humanly costly other than financially. This is the case, for example, for nuclear reactors, chemical factory, aircraft systems and many others. Following to importance of client demand and the importance of meeting their QoS requirements, we associated a dynamic analysis modules to EE-(m,k)-firm constraints, to optimize the gap between the rate of received QoS and the rate of desired QoS. In some situations, the system stops the operation of any other tasks (critical, hard or optional) to respond to tasks that are humanly more critical.

A deviation detecting module between the provided QoS by the system and the desired QoS by the client has become necessary. Then, according to task type, a consultation of gap impact must be carried out. Finally, the system decides the necessary values of EE-(m,k)-firm constraints.

Detection and localization gap

The detection procedure aims to determine the occurrence of a gap between the values m and k of each task type that has a specific EE-(m,k)-firm constraints fixed by the system. Indeed, because the properties of different tasks according to their types, the difference

between M_C and K_C is more important than the difference between M_H and K_H and also between M_O and K_O. However, this detection procedure will be applied to all possible types of constraints. Generally, for proper operation of an enterprise, these differences are usually of zero mean, which represents an optimum QoS to clients.

A means to auto-observe the gap between different EE-(m,k)-firm constraints is to estimate the needed values for each constraints type (K_C, K_H and K_O). The estimated values of M_C, M_H and M_O are then respectively subtracted from maximum constraints K_C, K_H and K_O to form the gaps E(C), E(H) and E(O) as follows:

$$\begin{cases} E(C) = K_C - M_C \\ E(H) = K_H - M_H \\ E(O) = K_O - M_O \end{cases}$$

Given that $K_C > M_C$, $K_H > M_H$ and $K_O > M_O$ (Figure 3).

At production times in an enterprise, the gap E (·) will significantly deviate according to the increase of system load, it will be equal to zero except when the system operates normally. In real applications, the differences are not exactly a zero value for the systems absence that accurately reflects the actual state of resources. Besides, the assigned measurements that aim to reflect the available resources are often marked by measurement noises.

The optimal QoS for clients varies according to values of measurement noises. With the proposed treatment of EE-(m,k)-firm constraints (Figure 3), and depending on the criticality of supplied products (chemical, nuclear ...), the values of KC and KH must be accurately measured. The gaps will then be written as:

$$\begin{cases} E(C) = K_{mC} - M_C \\ E(H) = K_{mH} - M_H \\ E(O) = K_{mO} - M_O \end{cases}$$

Where Km (·) is the value measured by the system and which is composed by the real value K(·) and the various types of noises relating to the calculation uncertainties .

To guarantee the application of EE-(m,k)-firm constraints, we propose a comparison method of each gap E (.) at an optimal predefined threshold for each type of task: Threshold ε for critical tasks, ε' for hard tasks and ε" for optional tasks, respectively. At every crossing of threshold, an alert is sent to the system for a new QoS management, we will then have:

$$\begin{cases} E(C) \le \varepsilon \leftrightarrow Alert = 0 \\ E(C) > \varepsilon \leftrightarrow Alert = 1 \end{cases}$$

$$\begin{cases} E(H) \le \varepsilon' \leftrightarrow Alert = 0 \\ E(H) > \varepsilon' \leftrightarrow Alert = 1 \end{cases}$$

$$\begin{cases} E(O) \le \varepsilon'' \leftrightarrow Alert = 0 \\ E(O) > \varepsilon'' \leftrightarrow Alert = 1 \end{cases}$$

After detecting the presence of a gap between M and K, it is necessary to locate the task type affected by this gap. This is nominated by the gap localization.

At the realization, we proceed at a structuring of all generated gaps during the system function. Generally, we constructed a first set of gaps E_i (·) that depend on the tasks types. From these basic gaps, we form two types of gaps: hard gap and soft gap.

In case of hard gap, after receiving an alert, the system immediately acts even by an intelligent violation of allocated resources to other clients' demands. This gives a dynamicity of resources allocation and EE-(m,k)-firm constraints. However, in case of soft gap, the system does not immediately act, but waits for the availability of resources to respond to this task type. During system function, the EE-(m,k)-firm constraints dynamically vary according to priority of client demands, system load and gap type.

We applied the proposed method described in this section to the example given in the previous section. The results are presented in Figure 4. Because of the resources release, all EE-(m,k)-firm constraints are changed. This change affects the tasks types, notably, the gaps between all tasks will automatically change. Indeed, the regular prevision and dynamic calculation of necessary constraints and of different gaps considerably support the QoS guarantee to clients. A

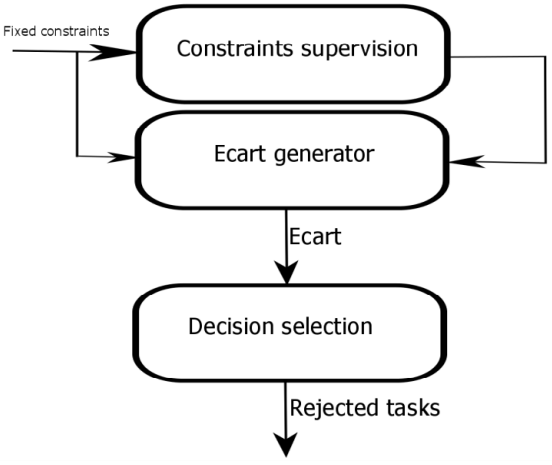

Figure 3: Treatment structure.

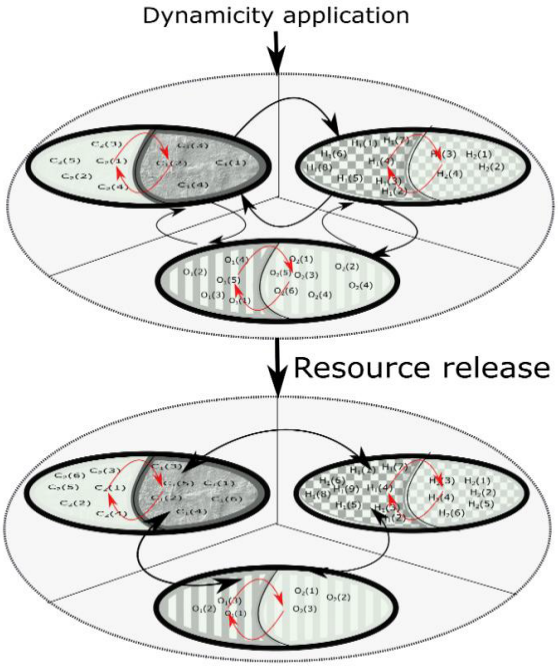

Figure 4: Dynamicity of EE-(m,k)-firm constraints application.

change of tasks types occurs, but the number of tasks does not vary (Figure 4).

We will have a decrease in optional tasks number and an increase in critical tasks number. For hard tasks, the number varies depending on the decrease and increase of critical and optional tasks.

Simulations and Results

We now detail the implementation of the EE-(m,k)-firm policy. Four types of decisions should be taken by our policy. We first describe the necessary data structures, and then we consider each of these decisions separately.

Description of data structures

Table 1 shows the data structure for each client demand. In a table noted table of demands in which each line contains the tasks number of a demand, and the class of popularity (EE-(m,k)-firm constraints), indeed, three classes are present.

The first refers to the C tasks (Critical) which are the most requested tasks by the system. The second regroups tasks of average importance H (Hard). The third contains optional L tasks (Low), i.e. least required by the system.

Tasks table (Table 1) records various information about the demanded tasks. Note that, the demands may not have the same number of tasks. Equally, EE-(m,k)-firm constraints do not have a fixed number.

Each entry in the demands table conserves track of each task, the number of tasks currently execute for each demand, and the classes on which tasks reside (Table 1).

The data structure for each unit is shown in table.1. Each entry in the units table corresponds to a unit and maintains several counters that keep track of free and served resources.

First, for load balancing on the units, the choice of the tasks will be on a lightly loaded unit that is selected for execution (Table 2).

This is achieved by traversing the entrance of the tasks table to find all the units that contain the type of requested task (including tasks in progress) and then looking into the set of the corresponding units to find the least loaded unit. Whereas unit U_i has completed execution of a task T_j, the data structures must be updated, to indicate a resource liberation on U_i. This is done by resetting the counter in the entry of tasks table (Table 3).

Demand-id	Requested tasks	EE-(m,k)-firm constraints		
		C	H	L
Demand_1	14	4	4	6
Demand_2	7	3	2	2
Demand_3	9	4	3	2
Demand_4	4	3	1	0

Table 1: Demands table.

Task-Num	Number of task on execution	Number of unit tasks	Unit-id			
1	60	4	U1	U3	U2	U4
2	70	2	U2		U3	
3	50	3	U3		U1	U4
4	20	4	U2	U3	U1	U4

Table 2: Tasks table.

Id-Unit	Total resources	Free resources	Served resources	Resources in free admission
U1	1000	100	900	10
U2	3500	700	2800	5
U3	2000	300	1700	7
U4	1500	200	1300	20

Table 3: Units table.

Unit and task-type selection

After having taken the decision to get the process of responding to a client demand, the EE-(m,k) -firm policy must select the tasks using the different constraints.

The execution begins when the system completes the selection of different tasks types of a demand. Note that the EE-(m,k)-firm policy does not change the task type simply because a resource is released by this task or because it has caused a unit overload.

This avoids the problem of changing the task type which slightly affects the QoS requested by the client. The estimated profit of Pt_i to execute a task i of a demand is a measure of future load that can be reduced from the current unit. This is calculated as follow:

$$Pt_i = \left(\frac{1}{t_i} - \frac{1}{t_i + 1}\right)\sum_{j=0}^{i-1} n_j \, w^{i-j-1}$$

Where W represents the weighting factor. The motivation for using this formula is to change the task type where the advantage in terms of load is expected to be higher in the future.

The load that can be changed in the near future (execution task time) is given by the load on the previous task. However, the load on the previous tasks represents the load that can be changed gradually in the future. To further improve the performance of the immediate load transfer, the profit to execute a task was calculated by weighting exponentially.

The load on the precedent tasks can be found from entries matching tasks in the tasks table.

Also, if there are t_i tasks of the current demand i, creating a modification of task type result $(1/(r_i - 1)/(r_i + 1))$ of profit in terms of load movement. The current number of tasks r_i will be also available in the current task entry.

Algorithm 1

T_i=number of tasks of i type

Pt_i=Execution profit of task i

S_i=unit number that can execute the task i

S_{th}=threshold of tasks number

P_i=Popularity of i task

P_{min}=min (P_i)

P_{max}=max (P_i)

P_{moy}=(Σi=1..N Pi)/N

Class L=[P_{min} (P_{min} + P_{moy})/2]

Class H=[(P_{min} + P_{moy})/2, (P_{max} + P_{moy})/2]

Class C=[(P_{max} + P_{moy})/2, P_{max}]

V (V_1 ... ..V_j ..... .V_N)

For (j=1 to N)

If ($V_i \in$ H Class)

For (i=2 to tasks number)

If ($S_i > S_{th}$)

R=round ((S_i-s_{th})/quota)

If (Pt_i superior to all benefit of another tasks) then

Execute the task i in the first selected unit

Used model

In the simulations, our system was composed of a demands manager and 10 homogeneous units. In a 90 minute peak period, requests arrivals were generated by a Poisson process with arrival rate λ [34]. Since, the capacity in unit output was 5000 tasks, and the maximum of λ rate was 25 demands per minute. Therefore, each unit can hold 25 tasks for the execution. The simulation model was evaluated several times. The results presented below are an average of 100 simulations. The distribution of demands and tasks popularity follows a Zipf-like [35] distribution, with parameter ϴ=0.75.

We integrated an admission controller in our simulations to reject a request if the available resources cannot meet more than 25% of requested tasks number of C type.

Simulation Results

First, we discuss the impact of the tasks number on the system response time (access delay) with *"fixed EE-(m,k)-firm constraints"* and *"dynamic EE-(m,k)-firm constraints"*. Figure 5 describes the representative results for different values of demands, such as 10, 20, 30, 40 and 50.

Figure 5 shows that the response time for all curves decreases with the increase of tasks number. The time considerably decreases between a low tasks number and an important tasks number. Indeed, the load balancing between different tasks types is significantly reduced. This is due to dynamics of tasks treatment that affects several factors. In particular, the access delay, in case of a unit overload improved with a dynamic EE-(m,k)-firm constraints, since it depends on tasks criticality that will be dynamically treated by the system. Thus, the system, which will be able to answer, has several tasks with these dynamic treatments, leading to improve management of the storage space and the QoS. This means that the EE-(m,k)-firm policy brings several benefits other than reducing the access delay (Figure 5).

At the tasks execution, the system begins the tasks dispersion between the necessary units. The second application of EE-(m,k)-firm policy results in the correct application of dynamicity of the policy showing that mC and kC have the highest priority. The graph in Figure 5 shows the behavior we expected. We can equally notice from the same figure that EE-(m,k)-firm policy, with fixed or dynamic constraints in all loads requirement, gives a shorter response time. But, we note that when increasing the tasks number, the difference between results decreases. Consequently, we can predict that if the number of tasks attains a certain threshold, there will be no difference between the different algorithms.

Figure 6 shows that our policy with a dynamic treatment of EE-(m,k)-firm constraints significantly reduces the rejection rate. The difference between the curves, using fixed and dynamic constraints, shows the improvement of tasks acceptance rate. The gap between sub-curves of EE-(m,k)-firm policy with a dynamic constraints m and k on different numbers of demands, shows the effectiveness of this dynamicity on the rejection rate. The ratio between the decrease of rejection rate with the increase of demands number shows that when the tasks number increases, the curves of our policy will be confused. We can conclude from these comparisons that our proposed policy achieved the desired results, even with a large tasks number.

The served tasks rate present the ratio among the number of received and executed tasks and all requested tasks (Figure 6).

We describe the case of little workload arriving to the case of the best workload. With dynamic constraints of EE-(m,k)-firm policy and in case of weighty system workload, our policy substantially affects the rate of served tasks. Consequently, at different workloads, EE-(m,k)-firm policy is powerful to overcome the system congestion problems (Figure 7). From this study, we notice that the EE-(m,k)-firm policy provides satisfactory results.

Outstandingly, EE-(m,k)-firm policy leads to an important number of served tasks in the case of high workload, up to 98% with dynamic constraints and about 49% with fixed constraints (Figure 7).

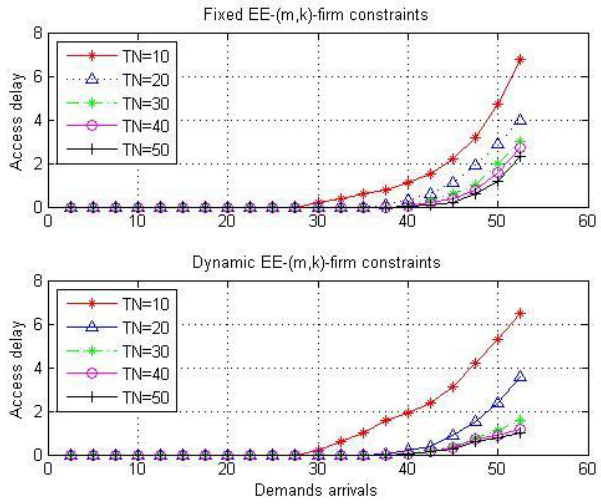

Figure 5: Access delay.

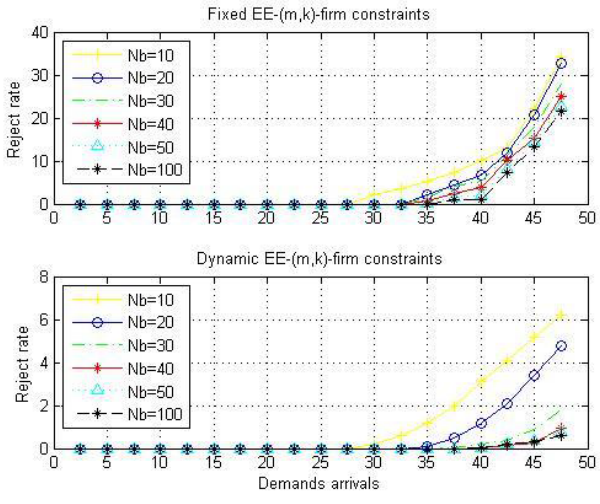

Figure 6: Reject rate.

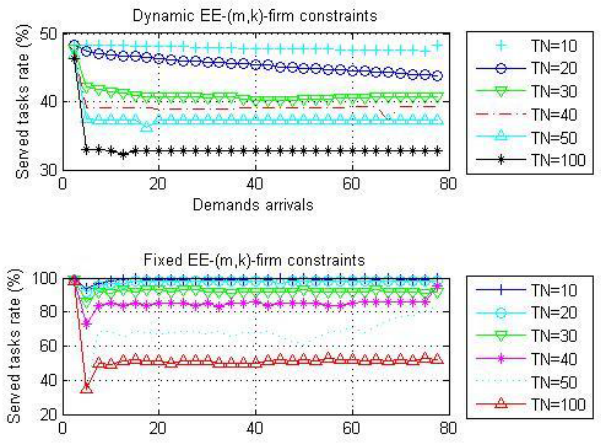

Figure 7: Served tasks rate.

Conclusion

The main purpose of this study was to present a novel policy of a specific treatment technique of tasks in enterprises environment. We proposed the EE-(m,k)-firm policy to indicate the necessary tasks and calculate their ranking, using a compromise between the available resources and the QoS granularity in the same task type. Based on an in-depth review of the relevant literature, three categories of tasks are possible in enterprise environment, namely critical tasks, hard tasks and optional tasks.

Afterwards, a guaranteed technique was applied to losses tasks intelligently according to importance of each task. A dynamicity of EE-(m,k)-firm constraints is then used to attain an increase of availability, performance, reliability and system dependability.

The results obtained from the proposed policy reveal that a "lack of awareness regarding the benefits of dynamic treatment of tasks in an intelligent and dynamic manner in enterprise environment is the most important reason behind the implementation of EE-(m,k)-firm policy. This type of policy can be extremely valuable for companies that wish to focus their efforts and resources to guarantee a satisfactory QoS for end-users and challenges toward the successful implementation of tasks management.

References

1. Hamdaoui M, Ramanathan P (1995) A Dynamic Priority Assignement Technique for Streams with (m,k)-firm Deadlines. IEEE Trans Computers 44: 1443-1451.
2. Decker A (2014) Factors Influencing The Quality of Accounting Information System And Its Implications on The Quality of Accounting Information. Research Journal of Finance and Accounting Vol 5.
3. Barnes BR, Leonidou LC, Siu NYM, Leonidou CN (2015) Interpersonal Factors as Drivers of Quality and Performance in Western-Hong Kong Inter organizational Business Relationships. Journal of International Marketing 23: 23-49.
4. Shuai D, Shuai Q, Dong Y (2007) Particle model to optimize resource allocation and task assignment. Information Systems journal 32: 987-995.
5. Wang Z, Chen J, Sun Y (2004) An integrated DBP for streams with (m,k)-firm real-time guarantee. J Zhejiang Univ Sci 5: 816-826.
6. Cho H, Chung Y, Park D (2010) Guaranteed dynamic priority assignment schemes for real-time tasks with (m,k)-firm deadlines. ETRI Journal 32, No: 3, pp: 422-429.
7. Goossens J (2008) (m k)-firm constraints and DBP scheduling: impact of the initial k-sequence and exact schedulability test. In16th International Conference on Real-Time and Network Systems. Rennes, pp: 61-65.
8. Li J, Song Y, Simonot-Lion F (2006) Providing real-time applications with graceful degradation of QoS and fault tolerance according to (m,k)-firm model. IEEE Transactions on Industrial Informatics 2: 112-119.
9. Partha P, Atighetchi M, Soule N, Ishakian V, Loyall J, et al. (2014) Secure and QoS-Managed Information Exchange Between Enterprise and Constrained Environments. 17th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), pp: 141-149.
10. Arboleda H, Paz A, Jiménez M, G Tamura M (2016) Development and Instrumentation of a Framework for the Generation and Management of Self-Adaptive Enterprise Applications. Ingeniería y Universidad Vol: 20: 303-334.
11. Arboleda H, Paz A, Jiménez M, Tamura G (2015) A framework for the generation and management of self-adaptive enterprise applications. 10th Computing Colombian Conference (10CCC).
12. Slater F, Tomas GHM, Olson EM (2007) On the importance of matching strategic behavior and target market selection to business strategy in high-tech markets Stanley. Journal Academy of Marketing Science 35: 5-17.
13. Davide A, Riccardo D, Valeria M (2012) Risk assessment in ERP projects. Information Systems 37: 183-199.
14. Mehrjerdi YZ (2010) Enterprise resource planning: risk and benefit analysis. Business Strategy Series 11: 308-324.
15. Chai B, Chen J, Zhang Y (2014) Demand Response Management With Multiple Utility Companies: A Two-Level Game Approach. Smart Grid, IEEE Transactions on 5: 722-731.
16. Wood T, Ramakrishnan KK, Shenoy PJ, van der Merwe JE (2012) Enterprise-Ready Virtual Cloud Pools: Vision, Opportunities and Challenges. Computer Journal 55: 995-1004.
17. Gunadi E, Plumbaum T, Albayrak S (2015) Applied Distributed Information Retrieval in Enterprise Search. SCST@ECIR, CEUR Workshop Proceedings, CEUR-WS.org Vol: 1338.
18. Ewert R, Wagenhofer A (2005) Economic Effects of Tightening Accounting Standards to Restrict Earnings Management. The Accounting Review 80: 1101-1124.
19. Chi W, Lisic LL, Pevzner M (2011) Is Enhanced Audit Quality Associated with Greater Real Earnings Management. Accounting Horizons 25: 315-335.
20. Krell K, Matook S, Rohde F (2016) The impact of legitimacy-based motives on IS adoption success: An institutional theory perspective. Information and Management 53: 683-697.
21. Al-Mamary YHS, Shamsuddin A, Aziati AHN (2014) The Role of Different Types of Information Systems In Business Organizations : A Review. International Journal of Research (IJR) 1(7).
22. Brien JA, Marakas GM (2010) Management information systems", (10thedn) by McGraw-Hill/Irwin, a business unit of The McGraw-Hill Companies.
23. Alam KA, Ahmad R, Akhunzada A, Nasir MHN, Khan SU (2015) Impact analysis and change propagation in service-oriented enterprises: A systematic review. Information Systems journal 54: 43-73.
24. Kadiri SE, Grabot B, Thoben K, Hribernik K, Emmanouilidis C et al. (2016) Current trends on ICT technologies for enterprise information systems. Computers in Industry 79: 14-33.
25. Atkinson C, Gerbig R, Fritzsche M (2015) A multi-level approach to modeling language extension in the Enterprise Systems Domain. Information. Systems 54: 289-307.
26. Nikolow D, Slota R, Polak S, Mitera D, Pogoda M, et al. (2013) Model of QoS Management in a Distributed Data Sharing and Archiving System. Procedia Computer Science 18: 100-109.
27. Nguyen TH, Wright M (2014) Variable neighborhood search for the workload balancing problem in service enterprises. Journal of Computers & Operations Research 52: 282-290.
28. West R, Zhang Y (2004) Dynamic window-constrained scheduling of real-time streams in media servers. IEEE Trans Comput 53: 744-759.
29. Bernat G (1998) Specification and Analysis of Weakly Hard Real-Time Systems. PhD thesis, Université de les Illes Balears.

30. Wang Z, Song Y, Poggi EM, Sun Y (2002) Survey of Weakly-Hard Real-Time Schedule Theory and its Application. Distributed Computing and Applications to Business, Engineering and Science (DCABES) pp: 429-437.

31. Ramanathan P (1999) Overload Management in Real-Time Control Applications Using (m,k)- Firm Guarantee. IEEE Trans. Parallel Distrib Syst 10: 549-559.

32. Dixon M, Verma R (2013) Sequence effects in service bundles: Implications for service design and scheduling. Journal of Operations Management 31: 138-152.

33. Goossens J (2008) (m k)-firm constraints and DBP scheduling: impact of the initial k-sequence and exact schedulability test. In16th International Conference on Real-Time and Network Systems. Rennes, pp: 61-65.

34. Dakshayini M, Nair TRG (2010) An Optimal Prefix Replication Strategy for VoD Services. Journal of comuting, Vol: 2.

35. Breslau L, Cao P, Fan L, Phillips G, Shenker S (1999) Web Caching and Zipf-like Distributions: Evidence and Implications. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies 1: 126-134.