Analytical model of object request broker based on Corba standard

V P Mochalov, N Y Bratchenko and S V Yakovlev
North-Caucasian Federal University, 1 Pushkin Street, Stavropol, 355000, Russia
E-mail: mochalov.valery2015@yandex.ru

Abstract. This paper introduces formal approaches to an integration of the program components of a distributed application using an Object Request Broker (ORB) of the Common Object Request Broker Architecture (CORBA) standard for distributed computing. The authors develop a system of functional models in order to analyze the interaction algorithms of distributed system elements.

1. Introduction

In modern industrial automation systems, telecommunications, e-commerce, and Internet-of-Things, the separate elements of applications are distributed over a network with a huge amount of Active-X and JavaBean components. Thus, it is necessary to develop technologies of distributed applications, in terms of design and implementation.

The program components of a distributed application that supports an associated business process are interacting with each other, creating a uniform program code. As for hardware and software platforms, such an interaction is based on the Common Object Request Architecture (CORBA) standard [1].

This standard implements an interaction of the distributed program components in a network, and its core consists in an Object Request Broker (ORB), a logical bus.

The ORB generates different mechanisms to execute and implement the requests of interacting program components, namely, program components selection for a developed application, data exchange, resources integration, distributed system design, and distributed business application development.

A client process initiates an operation and calls a message through an exchange subsystem. Using broadcast messaging, the ORB finds required program components, activates them, sends the request and parameters, calls a requested method, and transfers the results to a called client through its logical bus [1].

2. Models and methods of program components integration system

The interaction model of program components can be formally represented by a closed stochastic network with a central service node [2].

The ORB is root node $M$ while the other $(M-1)$ nodes are the program components of a physically distributed application that supports an associated business process. In this system, the program components are bound with the rules defined by the corresponding business processes and circulate...
between the network nodes, using as a mediator of the central ORB node. For the requests of different classes from $R = (1, r)$, the probabilities of transition can be described by the matrix [2]

$$
P_r \left( n_j^i \right) = \begin{bmatrix} P_{r,1,1} & P_{r,1,2} & \cdots & P_{r,1,M} \\
P_{r,2,1} & P_{r,2,2} & \cdots & P_{r,2,M} \\
\vdots & \vdots & \ddots & \vdots \\
P_{r,M,1} & P_{r,M,2} & \cdots & P_{r,M,M} \end{bmatrix},
$$

(1)

where $P_r \left( n_j^i \right)$ denote the probabilities of transition and $n_j^i$ is the number of requests passing from node $i$ to node $j$.

Let us assume that the service time of each node $i$ in this model obeys the exponential distribution with parameter $\mu_i$, while the interaction of program components is described by a multidimensional random process:

$$
N(t) = \left\{ n_1(t), n_2(t), \ldots, n_R(t) \right\}.
$$

In addition, let the requests be processed in accordance with the First Come, First Served (FCFS) principle and let the request rate be independent of $R$. The ORB network nodes have independent processors, and hence $k$ requests of the $r$th class are served $P_k \left( k_i \right)$ with the probability:

$$
P(k) = P_1 \left( k_1 \right) \cdot P_2 \left( k_2 \right) \cdots P_n \left( k_n \right),
$$

where $k = (k_1, k_2, \ldots, k_n)$ and $k_i = (k_{i1}, k_{i2}, \ldots, k_{ir})$.

The distribution of requests in the processor queues is given by the product of probabilities $P_r \left( k_i \right)$ and the executed requests form a Poisson flow. So this ORB network satisfies the requirements of a Jackson network and can be written in the multiplicative form [2]:

$$
P(n) = G^{-1} \left( N_1, N_2, \ldots, N_R \right) \prod_{i=1}^{M} Z_i \left( n_i \right),
$$

(2)

where $P(n)$ denotes the probability that the network is in state $n$,

$$
n = (n_1, n_2, \ldots, n_M); \quad G \left( N_R \right) = \sum_{k} \prod_{i=2}^{R} \left( \frac{\mu_i}{\mu_i} \right)^{k_i};
$$

(3)

$$
Z_i \left( n_i \right) = \frac{n_i!}{\prod_{R=1}^{R} \mu_i \left( n_i \right)} \prod_{r=1}^{n_i} \frac{1}{l_{ir}^{n_i}},
$$

(4)

$$
l_{ir} = \sum_{j=1}^{M} l_{ij} P_r \left( r \right), i = 1, M, r = 1, R.
$$

(5)

Using the results established in [3], the following takes place:

– the ORB capacity for a request of the $r$th class is:

$$
\lambda_r \left( N_R \right) = \sum_{n=1}^{N} P_r \left( n_r, N_j \right) \frac{n_r}{n_i} \mu_r \left( n_i \right);
$$

(6)

– the number of requests of the $r$th class in the ORB is:

$$
L_r \left( N_R \right) = \sum_{n=1}^{N} P_r \left( n_r, N_j \right) n_i;
$$

(7)

– the average waiting time for a request of the $r$th class is:

$$
T_r \left( N_R \right) = \frac{[1 + L_r \left( N_R \right) - 1]}{M_r}.
$$

(8)
In practice, the values of these indicators depend on system load at request arrival times. Obviously, in some state the ORB may block a successive call. In this case, the call is in repeated $N$ times till being served or rejected. The probability of a successful call makes up [3]:

$$P_{\text{suc}} (N) = P_{\text{blk}}^{N-1} (1 - P_{\text{rej}} (1)) (1 - P_{\text{rej}} (2)) ... (1 - P_{\text{rej}} (N-1)) (1 - P_{\text{blk}}).$$

The probability of a rejected call has the formula:

$$P_{\text{rej}} (N) = P_{\text{blk}}^{N} (1 - P_{\text{rej}} (1)) (1 - P_{\text{rej}} (2)) (1 - P_{\text{rej}} (N-1)).$$

After the transformations, it is possible to write:

$$P_{\text{suc}} (N) = (1 - P_{\text{blk}}) \left[ 1 + \sum_{i=1}^{N-1} P_{\text{blk}}^{i} \prod_{j=i+1}^{N} (1 - P_{\text{REJ}} (j)) \right].$$

$$P_{\text{REJ}} (N) = P_{\text{blk}} \left[ P_{\text{REJ}} (1) + \sum_{i=1}^{N-1} P_{\text{blk}} P_{\text{REJ}} (i+1) \prod_{j=i+2}^{N} (1 - P_{\text{REJ}} (j)) \right].$$

These indicators can be improved by reducing system cost connected with the interaction processes of program components and the ORB [3]. Let us assume that the distributed system includes $d$ processor modules while a business process is supported using $n > d$ program components. Evidently, for an efficient implementation of a distributed application, it is necessary to solve three main problems, namely [4],

- to partition $n$ program components into $d$ groups so that each group contains the program components with the largest frequencies of interaction;
- to design request rules that eliminate blocking and deadlocks;
- to reduce the system cost of integration for distributed program components.

For solving the first problem, let us assume that frequency $P(i,j)$ of mutual requests is known for any interacting program components $f_i$ and $f_j$. Let us split all $n$ program components into $d$ groups:

$$\Phi = \{F_1, F_2, ..., F_d\} \text{ so that } \bigcup_{i=1}^{n} f_i = \bigcup_{i=1}^{d} F_i, \quad F_i \cap F_l = \emptyset, \quad k \neq l.$$  

A conflict occurs in case of a simultaneous call of different program components placed on the same processor module. Frequency $C_k$ of conflicts on the $k$th processor module is:

$$C_k = \sum_{i,j} P(i,j),$$

while the total frequency of all conflicts in the system makes:

$$C = \sum_{k=1}^{d} C_k = \sum_{k=1}^{d} \sum_{i,j} P(i,j).$$

Value $C$ can be decreased with cyclical transfers of the $i$th program component ($PC_i$) between the $k$th and $l$th groups, where $k, l \in d$. The corresponding variation of $C$ is calculated by

$$\Delta C = \sum_{i \notin PC_k} \left[ P(i,j) + P(j,i) \right] - \sum_{j \notin PC_l} \left[ P(i,j) + P(j,i) \right].$$

(9)

A sequential application of the same steps (9) stops when any transfer of $PC_i$ between the $k$th and $l$th groups does not affect $\Delta C$.

Variation $\Delta C$ can be calculated recursively using a convenient method. Let us introduce a system of operators $R = \{R_{n,i}, i=1,n, t=1,d\}$, also called partitions, for the transfer of $PC_i$ in the $i$th class, where $t \in d$.

Then $\Delta_n (\Phi) = C(R_n \Phi) - C(\Phi)$, where $\Delta_n (\Phi)$ is the corresponding variation for the frequency of conflicts.
Let $\Delta^q_{\varphi}(\varphi)$ be the increment of $\Delta_q(\varphi)$ under partition $R_q(\varphi)\in R$.

Then $\Delta^q_{\varphi}(\varphi) = \Delta_q\left(R_q(\varphi)\right) - \Delta_q(\varphi), i = \overline{1,n}; q = \overline{1,d}; \Delta_q\left(R_q(\varphi)\right) = \Delta_q(\varphi) + \Delta^q_{\varphi}(\varphi)$.

This recursive method yields a rational distribution of $n$ program components over $d$ processor modules, reducing the nonproductive cost of communication service requests.

The second problem related to request scheduling is to find an execution sequence of requests that minimizes the total job time. Clearly, the blocking of requests can be avoided under the condition:

$$t_{p+1}^i - t_p^i > Z_k^i - Z_k^j, \quad L = \overline{1,k},$$

where $t_p^i$ is the $p$th demand of the $i$th type; $t_{p+1}^i$ is the $(p+1)$th demand of the $j$th type; $Z_k^i$ and $Z_k^j$ specify the service times of these demands by the $L$th processor; finally, $l_j = \max\left\{Z_k^i - Z_k^j\right\}$.

This problem statement leads to the following. Let us consider directed symmetric graph $(X,U)$, where $X$ and $U$ denote the sets of request types and arcs, respectively. Then it is necessary to construct a loop connecting the required vertices just once with a minimum sum of the arc lengths, i.e.:

$$\sum_{i,j=0}^n l_{ij} x_{ij} \rightarrow \min, \quad \sum_{i=0}^n \sum_{j=0}^n x_{ij} = n_i, i = \overline{0,r}, j = \overline{0,r}.$$

The third problem of an integration system design for distributed program components is to reduce the frequency of data exchange between the program components and the ORB. This problem is solved by partitioning the set of program components into independent fragments and organizing a proper interaction between these fragments and the ORB.

The desired fragmentation is described by the matrices:

$$V = \left[\begin{array}{c}
V_1 \\
\vdots \\
V_m
\end{array}\right]_{k=1}^m \quad \text{and} \quad L = \left[\begin{array}{c}
L_1 \\
\vdots \\
L_n\end{array}\right]_{j=1}^n,$$

where

$$V_{ik} = \begin{cases} 1 & \text{if } PC_i \in \varphi_k, \\ 0 & \text{otherwise;} \end{cases} \quad L_j = \begin{cases} 1 & \text{if } PC_j \in \varphi_k, \\ 0 & \text{otherwise.} \end{cases}$$

Let the $i$th program component be executed $m_i$ times and the $j$th program component be requested with probability $P_{ij}$. Then the number of requests from $PC_i$ to $PC_j$ is:

$$\sum_{k=1}^n \sum_{j=1}^n m_i P_{ij} V_{ik},$$

where $V_{ik}$ denotes the indicator function of the program components belonging to the $k$th fragment.

If each program component belongs to a single fragment, then the average number of inter-fragment requests can be written as:

$$C = \sum_{k=1}^n \sum_{i=1}^n \sum_{j=1}^n m_i P_{ij} V_{ik} \left(1 - V_{ik}\right) \rightarrow \min.$$

This problem is solved using standard tools of Mathematica.

3. Conclusions

In this paper, let us have introduced formal approaches to an integration of the program components of a distributed application using an ORB of the CORBA standard for distributed computing and also developed a system of functional models to analyze the interaction algorithms of distributed system elements. They can be used to design efficient distributed systems.

The integration problem of program components has been reduced to an implementation of three main algorithms, namely, a rational distribution of program components over system processor modules, minimal length scheduling, and nonproductive cost minimization for data exchange processes. The authors have suggested a complex of analytical models to study an integration system.
of distributed applications, which can be used for assessing the interaction algorithms of its elements and the impact of interaction processes on system characteristics.

References

[1] Vinoski S 1997 CORBA: Integrating diverse applications within distributed heterogeneous environments *IEEE Communications Magazine* 35(2) 46-55 doi:10.1109/35.565655

[2] Suri R 1983 Robustness of queuing network formulas *Journal of the ACM (JACM)* 30(3) 564-594 doi:10.1145/2402.2995

[3] Whitt W 1983 Performance of the queueing network analyzer *Bell System Technical Journal* 62(9) 2817-2843 doi:10.1002/j.1538-7305.1983.tb03205.x

[4] Mochalov V P, Bratchenko N Y and Yakovlev S V 2014 Development of a methodology for building a distributed control system for telecommunications networks based on CORBA technologies *Vestnik NCSTU* 6(45) 50-57

[5] Mochalov V P, Bratchenko N Y, Yakovlev S V and Yambulatov E I 2015 Development of fault-tolerant distributed control systems for telecommunication networks and services (Stavropol: North-Caucasian State Technical University)