10th IMEKO TC10 Conference on
Technical Diagnostics
Budapest, HUNGARY, 2005, June 09-10

WEB SERVICE RUNTIME PREDICTION AND OPERATION FAULT DIAGNOSIS

Viktor Oravec, Zoltán Balogh, Ladislav Hluchý, Baltazár Frankovič

Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia

Web service (WS) is a fast growing technology. A set of WS is one of the best applications of multi-agent system, i.e. a WS is represented by an agent. This fact results in WS encapsulation and causes difficulty to track its internal behaviour. One solution of this problem is to use a notification system, where changes in WS are represented by a set of notification messages. The set of notification messages gives only static information about WS. If time[1] between notification messages is considered, then a web service dynamics can be described.

This article proposes an approach for estimating WS runtime and prediction of WS operation faults. Prediction of WS runtimes can optimize scheduling and support efficient use of grid resources. In our approach we propose to estimate expected WS runtime based on WS invocation parameters using case-based reasoning and prediction of WS operation faults using rule-based reasoning.

This article consists of two main parts. The first one includes introduction into reasoning and its two special types, namely case-based [3] and a rule-based reasoning. In the second section, prediction of WS runtime is presented. In the third section, the diagnostics of web service operation faults is offered.

Keywords: web service, case-based reasoning, rule-based reasoning,

1. INTRODUCTION

Reasoning is a process, which works with data consisting of essential information for certain decisions [7]. The reasoning results in a set of parameters that describe the processed data for decision process. The set of parameters can be defined by a designer of the reasoning process (off-line definition), or automatically by the reasoning process (on-line definition). If the set of parameters is defined by the designer, then the reasoning evaluates the values of defined parameters only. If the set of parameters is defined by the reasoning process, then the reasoning process has to define the set of parameters and evaluates the value of each parameter.

There are two basic types of reasoning, namely case-based reasoning (CBR) and a rule-based reasoning (RBR). CBR can be viewed as a learning reasoning. CBR studies previous results and tries to apply them new cases. The RBR principle is based on the set of rules that can be set up by designer. The set of rules can be created on-line or off-line. If the set of rules is created on-line then it can be the result of learning or recognition.

1.1. Case-based reasoning

As mentioned before, CBR is reasoning about previous cases. CBR tries to use the results of previous cases to predict the result of the actual/new case [3]. An agent that supports this type of reasoning has to have access to a base of cases (e.g. database), where cases and reasoning results are stored and updated. When an agent starts reasoning about a new case, it queries the database for similar cases. If one or more similar cases are found, adaptation of results of the cases matched to the new case is performed. Prediction of results generated using CBR is provided to a requestor. After a new case is completed, a request returns the results which are then compared with the predicted values. If the results are not acceptable or no similar case is found in the database, the entity makes general reasoning about the new case and puts the results into the database.

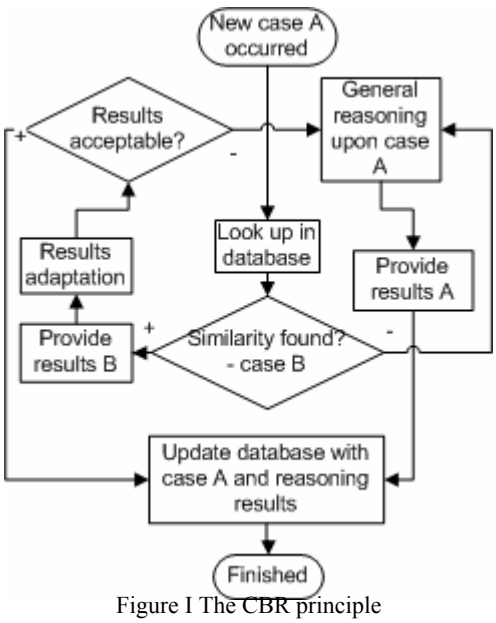

Figure I The CBR principle

[1] The time between two messages can be either the runtime of single web service operation, or the runtime of the whole web service.

If the results are acceptable a new case is added to the database. Note that if there is an old case that is the **same** as the new case, no reasoning is needed. The previously described principle is presented in the following figure (Figure I).

The CBR principle brings new problems that have to be solved; one of them is to compare two cases and test their similarity. In general, the comparison of cases is based on a set of application-relevant parameters

$$P = (p_1, p_2, ..., p_n). \tag{1}$$

The value of each parameter is stored in database for each case. For $p_i \in R$, the similarity of two cases can be determined by a distance in Euclidean space (2).

$$d(P, P') = |P - P'| = \sqrt{\sum_{i=1}^{n} |p_i - p_i'|^2}. \tag{2}$$

In the case when some parameters are more important, the distance (2) is extended by weights (3).

$$d(P, P') = |P - P'| = \sqrt{\sum_{i=1}^{n} w_i |p_i - p_i'|^2}, \tag{3}$$

where $w_i \in \langle 1, \infty)$ is a weight of the *i*th parameter, i.e. a parameter with a lower priority adds bigger distance to the final distance between cases. This type of weight can be called parameter degradation weight, i.e. the essential parameter remains and the others are suppressed. For $w_i \in (0,1\rangle$, the essential parameters have lower weights, i.e. the essential parameters are amplified and the other remain. Note that the use the Euclidean distance is not the only method how to measure affinity of two cases. Usually, a method of similarity measuring depends on a parameters' domain. Also note that if some parameters cannot be compared due to lack of information, then they cannot influence the comparison result. In this case, appropriate weights are usually replaced by zero.

1.2. Rule-based reasoning

The second type of reasoning is rule-based reasoning (RBR). RBR works with a knowledge base and a set of rules. A rule is a simple if…then… formula. Each rule has *if clause* (premise) and *then clause* (consequent). Each clause includes facts from the knowledge base. Knowledge base is a set of facts. The knowledge base describes an examined system and is defined by expert[2]. The process of definition is known as a knowledge engineering[3]. The knowledge base and the set of rules create an expert system. The expert system works on simple principle; if facts in the premise are true then facts in the consequent are also true. This situation is known as firing of rule. Firing of a rule can cause firing of the other ones. Reasoning process is finished, when no more rules can be fired, or when a goal is reached. The goal is the fact from the knowledge base and the main subject of reasoning, such as advice, diagnoses, location of fault, etc.

A sequence of fired rules is called a chain. There are two types of chaining in the expert system, namely forward chaining and backward chaining. In the case of forward chaining, reasoning starts from known facts, which initiate firing of rules. While rules are fired, the expert system knowledge is developing, i.e. other facts are becoming true. In the case of backward chaining, reasoning starts with looking for rules which have the goal as consequent. After that, reasoning looks for facts in the premise of founded rules. If facts are true, then the rule is fired. The expert system is developing backwards during reasoning. Backward chaining is efficient in case of large knowledge base. Forward chaining is efficient in case of numerous of conclusions.

2. WEB SERVICE RUNTIME PREDICTION

In this section, an example of case-based reasoning is presented. The example uses the CBR principle to predict runtime of a web service (WS) based on past cases. Each WS is launched with the invocation parameters. Some of the parameters have more significant influence on the WS runtime. For instance, consider a WS that simulates a process based on given invocation parameters. Two parameters have direct influence upon the computation time, namely upon duration of simulation and simulation step. These two parameters create vector of parameters $P = (p_1, p_2)$, where $p_k \in P_k = \{x_i \in R^+; i = 1,2,3..n_k; n_k \in N\}$ denotes duration of simulation, for $k = 1$, and a simulation step for $k = 2$. For this example, consider CBR with maximum 100 cases:
$n_1 = 10$,
$n_2 = 10$,
$p_1 \in P_1 = \{1,6,11,16,21,26,31,36,41,46\}$,
$p_2 \in P_2 = \{0.001, 0.0011, .., 0.0019, 0.0020\}$.
Note that parameters must be normalized to remove the influence of measuring units. In this example, "min-max linear" transformation (4) was used to normalize parameters.

$$p_{Nk} = \frac{p_k - \min(P_k)}{\max(P_k) - \min(P_k)}, \tag{4}$$

where

$$p_{Nk} \in P_{Nk} = \left\{ \frac{p_k - \min_k}{\max_k - \min_k}; \min_k, \max_k \in R^+; \right. \\ \left. \min_k = \min(P_k) \le p_k \in P_k \le \max_k = \max(P_k) \right\} \tag{5}$$

denotes the normalized value of the parameter p_k. Cases are stored in the table T that can be described by the relation $T : P_{N1} \times P_{N2} \to R_0^+$. The values in table T are computation times of particular cases. Computation time for

[2] An expert can be a person or a computer system that autonomously builds a knowledge base and a set of rules.
[3] The knowledge engineering is the process of codifying an expert's knowledge in such way that non-experts can understand it.

the *k*th case is denoted by $t_k = T(p_{N1k}, p_{N2k})$, where p_{N1k} and p_{N2k} are parameters values of the *k*th case in table T. If a case is not stored in database, its computation time is 0.

For each new case $P' = (p'_{N1}, p'_{N2})$, a similarity matrix $S: P_{N1} \times P_{N2} \to R_0^+$ is calculated[4], where

$$S(p_{N1}, p_{N2}) = \begin{cases} (p'_{N1} - p_{N1})^2 + (p'_{N2} - p_{N2}); \\ T(p_{N1}, p_{N2}) > 0 \\ \infty; \\ T(p_{N1}, p_{N2}) = 0 \end{cases} . \quad (6)$$

A case that is the most similar one to the new case has the lowest value in the similarity matrix. Cases in table T are ordered into a $|P_{N1}| \cdot |P_{N2}| \times 3$ dimensions matrix

$$O = \begin{pmatrix} p_{N11} & p_{N21} & T(p_{N11}, p_{N21}) \\ p_{N12} & p_{N22} & T(p_{N11}, p_{N21}) \\ & \dots & \end{pmatrix}, \quad (7)$$

where $S(p_{N1k}, p_{N2k}) \geq S(p_{N1k+1}, p_{N2k+1})$. The matrix O is used in the following computation time prediction.

Computation time of a new case is predicted from the "top *n*" cases, i.e. the first *n* rows in the matrix O (7). There are many principles how to predict some values; we use the simplest principle – the average value. Thus, the new case's computation time is the average value of "top *n*" cases' computation times.

$$t_n' = \frac{1}{n} sum\left(O_n \cdot (0 \quad 0 \quad 1)^T\right), \quad (8)$$

where O_n is a matrix that includes first *n* rows from the matrix O and function $sum(A): A \in R^n \to R$ evaluates the sum of all items in a vector A.

Let us consider only a part of the table T (TABLE I) and the new case $P' = (0.556, 0.7)$.

TABLE I A part of the table T

p_{N2} \ p_{N1}	0.333	0.444	0.556	0.667	0.778
0.4		12.531			23.125
0.5			14.797		
0.6				16.797	
0.7	7.734				
0.8		9.687			
0.9					16.406

An appropriate part of the matrix S and the new case is shown in the next table (TABLE II).

[4] Note that Euclidean distance (2) was used for evaluating the similarity of two cases.

TABLE II A part of the table S with labels

p_{N2} \ p_{N1}	0.333	0.444	0.556	0.667	0.778
0.4		0.102			0.139
0.5			0.040		
0.6				0.022	
0.7	0.050				
0.8		0.022			
0.9				0.089	

From the part of the matrix S the matrix O can be evaluated.

$$O = \begin{pmatrix} 0.667 & 0.6 & 16.797 \\ 0.444 & 0.8 & 9.687 \\ 0.556 & 0.5 & 14.797 \\ 0.333 & 0.7 & 7.734 \\ 0.778 & 0.9 & 16.406 \\ & \dots & \end{pmatrix} \quad (9)$$

and the predicted computation times are $t'_3 = 13.7603h$, $t'_4 = 12.2538h$ and $t'_5 = 13.0842h$.

A problem is to determine the value of *n*. For a flatter table T, higher n is recommended. Note that this is generally true for average prediction only.

3. WEB SERVICE FAULT DIAGNOSIS

In this section, an example of rule-based reasoning is presented. The example shows how the RBR can be used in the diagnosis of web service operations. Consider a WS that offers operations, which are executed in several stages s_i, where $i = 1,2,\dots$. Each stage s_i takes a certain time. This time is a random variable T_i, which has normal distribution with a mean t_i and a variance σ_i^2 (Figure II)[5].

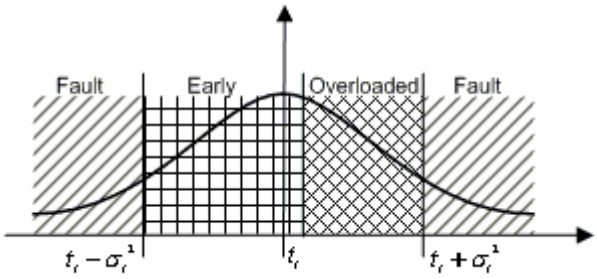

Figure II Distribution function on the random variable T_i, with completion time types

Each stage can be completed with one of the following three results, namely early completion, completion in overloaded system and faulty completion (Figure II). The *i*th

[5] Parameters t_i and σ_i^2 can be predicted from CBR in the same way as the computation time in Section 3

stage is completed early if completion time $t^{C}{}_{i}$ of the *i*th stage is between $t_i - \sigma_i^2$ and $t_i - \delta.\sigma_i^2$, where $\delta \in \langle 0,1 \rangle$ denotes the completion time delay and is set up by the RBR system designer. The *i*th stage is completed in overloaded system if its completion time $t^{C}{}_{i}$ is between $t_i - \delta.\sigma_i^2$ and $t_i + \sigma_i^2$. Completion of the *i*th stage faulty if completion time $t^{C}{}_{i}$ is greater than $t_i + \sigma_i^2$ or less than $t_i - \sigma_i^2$.

The previous consideration results in the following knowledge base and rules:

Knowledge base:

- t_i, σ_i^2 denote stage statistics.
- $K^{C}{}_{i}, K^{O}{}_{i}, K^{F}{}_{i}$ denote stage completions, namely early completion ($K^{C}{}_{i}$), completion in overloaded system ($K^{O}{}_{i}$), faulty completion ($K^{F}{}_{i}$).
- $K^{C}{}_{0}$ denotes the service start.
- $t^{C}{}_{i}$ denotes the real completion time of the *i*th stage.

Rules:

$$(K^{C}{}_{i-1} \wedge K^{O}{}_{i-1}) \vee (t_i - \sigma_i^2 < t^{C}{}_{i} < t_i + \delta\sigma_i^2) \Rightarrow K^{C}{}_{i} \quad (10)$$

$$(K^{C}{}_{i-1} \wedge K^{O}{}_{i-1}) \vee (t_i + \delta\sigma_i^2 < t^{C}{}_{i} < t_i + \sigma_i^2) \Rightarrow K^{O}{}_{i} \quad (11)$$

$$(K^{C}{}_{i-1} \wedge K^{O}{}_{i-1}) \vee (t_i + \sigma_i^2 > t^{C}{}_{i} > t_i - \sigma_i^2) \Rightarrow K^{F}{}_{i} \quad (12)$$

Note that only finite values of the completion time $t^{C}{}_{i}$ are considered, i.e. each stage must be completed in the finite time.

Having sufficient knowledge base we can use the above rules to diagnose proper web services operation.

4. CONCLUSIONS

The main aim of this paper was to present reasoning in a web service application that is representative of multi agent systems [1] [2]. The paper concentrates on two special reasoning types, namely case-based and rule-based reasoning.

The use of CBR was demonstrated to predict expected runtime of a WS. In general, CBR includes two phases, namely a searching phase and an adaptation phase. The searching phase includes searching for similar cases. This phase can be easily generalized and used in other applications. The adaptation phase processes the results from the searching phase. The adaptation phase of the example is the prediction of runtime by averaging the completion times of the most similar cases from the past. Implementation of the adaptation phase is application-dependent, and cannot be generalized.

The rule-based reasoning (RBR) is the second type of reasoning, which can be used singularly, as an expert system or in combination with CBR. If RBR is used in combination with CBR then RBR is preferred and CBR is used only if RBR gives insufficient results. RBR has many applications such as planning, configuring, diagnosis, etc. We use RBR for web service faults diagnosis.

5. ACKNOWLEDGEMENTS

The work presented in the paper was supported by the following projects:

- APVT – 51 – 011602
- VEGA 2/4148/24
- VEGA No. 2/3132/23
- K-Wf Grid - EU 6FP RTD IST project (FP6-511385)

6. REFERENCES

[1] M. Wooldridge, N. R. Jennings, “Intelligent Agents: Theory and Practice”, Knowledge Engineering Review 10(2), 1995

[2] M. Wooldridge, N. R. Jennings, “Pitfalls of Agent-Oriented Development”, Agent ’98: Proceedings of the Second International Conference on Autonomous Agents ACM Press, May 1998

[3] I. Budinská, T.T. Dang, “A Case Based Reasoning in a Multi Agents Support System”, In Proc. of the 6th International Scientific –Technical Conference, Process Control 2004, June 2004

[4] I. Budinská, T.T. Dang: Ontology Utilization in MARABu – a Support System for Modelling, Simulation and Control Design, In Proc. of the International Conference on Intelligent Engineering Systems, INES 2004, September 2004

[5] T. T Dang.: "Improving plan quality through agent coalitions", IEEE International Conference on Computational Cybernetics – ICCC’04, 6. pages, 2004

[6] Mařík V., Štěpánková O., Lažanský J., „Umělá inteligence (3)“, Academia, 2001

[7] Fogel J., “Multi-agent strategic decision making in game of incomplete information”, In Proc. of the 6th International Scientific –Technical Conference, Process Control 2004, Kouty nad Desnou, Czech Republic, June 8-11, 2004, Eds.: Stanislav Krejčí et.al, Editorial University of Pardubice, pp. 188, ISBN 80-7194-662-1; 6 pages – CD ROMJune 2004

[8] Faerman M., Su A., Wolski R., Berman F., Adaptive Performance Prediction for Distributed Data-Intensive Applications, August 9, 1999, Proceedings of the ACM/IEEE SC99 Conference on High Performance Networking and Computing, Portland

AUTHORS:
Viktor Oravec, Department of discrete processes modelling and control, Institute of Informatics, Slovak

Academy of Sciences, Dubravska cest 9, 845 07 Bratislava, upsyviki@savba.sk

Zoltán Balogh, Department of Parallel and Distributed Computing, Institute of Informatics, Slovak Academy of Sciences, Dubravska cest 9, 845 07 Bratislava, balogh.ui@savba.sk

Ladislav Hluchý, Department of Parallel and Distributed Computing, Institute of Informatics, Slovak Academy of Sciences, Dubravska cest 9, 845 07 Bratislava, hluchy.ui@savba.sk

Baltazár Frankovič, Department of discrete processes modelling and control, Institute of Informatics, Slovak Academy of Sciences, Dubravska cest 9, 845 07 Bratislava, utrrfran@savba.sk