41

Agent based Management of Distributed Systems with Variable Polling Frequency Policies

P. Dini, G. v. Bochmann
University of Montreal
CP 6128, Succursale CENTRE-VILLE, Montreal, H3C 3J7, Canada.
email: dini,bochmann@crim.ca

*T. Koch, B. Krämer**
FernUniversität Hagen
Datenverarbeitungstechnik, 58084 Hagen, Germany.
email: thomas.koch,bernd.kraemer@fernuni-hagen.de

Abstract

Management activities are based on the state of distributed system components, relations of these components, and their behaviour. Since management policies are applied across an abstraction of distributed systems, the quality of decisions is dependent on the representation fidelity of the real system state. Obviously, the data collection process updating the abstract representation of real distributed system components has a major impact on the quality of management decisions. Gathering most topical management data improves the quality of management decisions, but requires a high degree of monitoring activity. This is contradictory to the request for low impact management systems, where the amount of system resources used for management purpose should be as small as possible.

In this paper we present a twofold approach to this problem: First a high level management architecture is described where monitoring is performed by distributed agents with generic functionality for filtering and event creation. The distribution of active management agents reduces the amount of management related traffic and avoids a potential bottleneck on a centralized management station. Second an adaptive polling frequency approach is presented which enables the monitoring agents to adapt their polling frequency automatically to different behavioral parameters of managed components. The automatic adaptation reduces the performance impact of the agents significantly while at the same time a high accuracy of management relevant information about critical components is ensured. Implementation aspects of the introduced management architecture in a CORBA environment are also discussed.

Keywords

Agent based monitoring, Adaptive polling frequency, Distributed systems management

*Supported by the BMBF, Project No. INF 26

A. A. Lazar et al. (eds.), *Integrated Network Management V*

1 INTRODUCTION

Two paradigms are currently used for system management: the platform-centered management (commonly used by proprietary architectures) and the distributed management (recommended by OSI and ODP standards). Distribution is viewed either as heterogeneous management entities localized on specific sites and cooperating for a common problem solution, or homogeneous management units, distributed on many sites (so called *System Management Application Entity (SMAE)* in OSI standards). Management activities are based on the state of DS (Distributed System) components, relationships of these components, and their behavior.

Managers must accurately know the state of managed components. Within the notification approach, a DS component automatically sends notifications upon state changes to specialized agents. The agent forwards this notification to its own manager. The amount of traffic increases with the number of changes and the system performance decreases. Beside that, many times the notification data may not be relevant for the system management.

In the polling approach, supervisors (agents for the real resources, or managers for agents) send actions to collect data (eventually updated). Collecting commands could be issued periodically with a variable or fixed frequency. The process of collecting information at regular intervals is known as polling. The result of a polling operation is either new data (as information message), or no new changes (control message without reports).

The polling procedure is characterized by three important features: the *polling interval* defined as the amount of time between two consecutive polling operations within which the polled component has not transmitted new data; the *walk time* representing the amount of polling interval consumed by polling messages; and the *response time* referring to the amount of time between a command and the appropriate response. Evaluation of these features depends on several management aspects.

Gathering most topical management data usually improves the quality of management decisions, but requires a high degree of monitoring activity. This is contradictory to the request for low impact management systems, where the amount of system resources used for management purpose should be as small as possible. In this paper we present a twofold approach to this problem: In Section 2 a high level management architecture is described where monitoring is performed by distributed agents with generic functionality for filtering and event creation. Section 3 shortly defines behavioral parameters which are used in Section 4 as input for an adaptive polling frequency approach which enables the monitoring agents to adapt their polling frequency automatically with respect to different behavioral parameters of managed components. Implementation aspects of the introduced management architecture in a CORBA environment are discussed in Section 5.

2 ARCHITECTURAL ISSUES

A comprehensive management architecture requires the provision of several generic management services. Figure 1 provides an overview of the architectural concept. Different management applications share a set of generic management services. These services are focused on management issues only, distribution transparencies are provided by appropriate middleware components. The integration of different management applications into a common architecture allows simplified cooperation and coordination of activities from

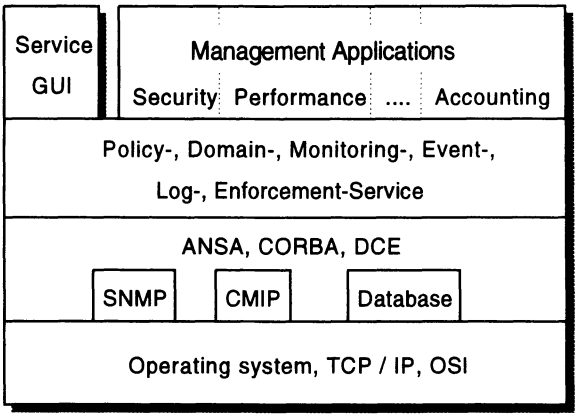

Figure 1 Overview of the management architecture

different management areas. This integrity is required to avoid unexpected side effects where activities in one management application has a major impact on the result of activities in another area. The performance management of multimedia applications, for example, depends usually on the underlying network, which is managed by a network management application. Coordination of both management activities allows us to reach an optimized result for both areas.

A common graphical user interface (GUI) provides access to the management services. This interface allows the human administrator editing of policies and domains, activation and deactivation of services and monitoring of management activity. It is important to note that the management services in this architecture are considered to be a distributed application running on an appropriate middleware platform. State of the art management protocols, like the *Simple Network Management Protocol (SNMP)* or the *Common Management Information Protocol (CMIP)*, and an object oriented database are enclosed within the middleware and therefore transparent to the management services and applications. Elementary services, like communication or data-storage, are provided by standard operating systems.

Figure 2 provides an overview of the runtime environment. Updated data is collected in two phases namely, at a physical level (from real resources to standardized records) and at a management level (from management agents to their correspondent managers). As shown in Figure 2, these data is collected from the real resources either by the source initiative as change events (physical notifications, management notifications), or by specialized software agents (physical actions, management actions).

At runtime most management activities are triggered by events created from monitoring results. Monitoring provides actual values about the state of managed objects either by requesting the values at a regular interval or by receiving notifications from the managed objects. Evaluation of monitored values may result in an event if a predefined threshold

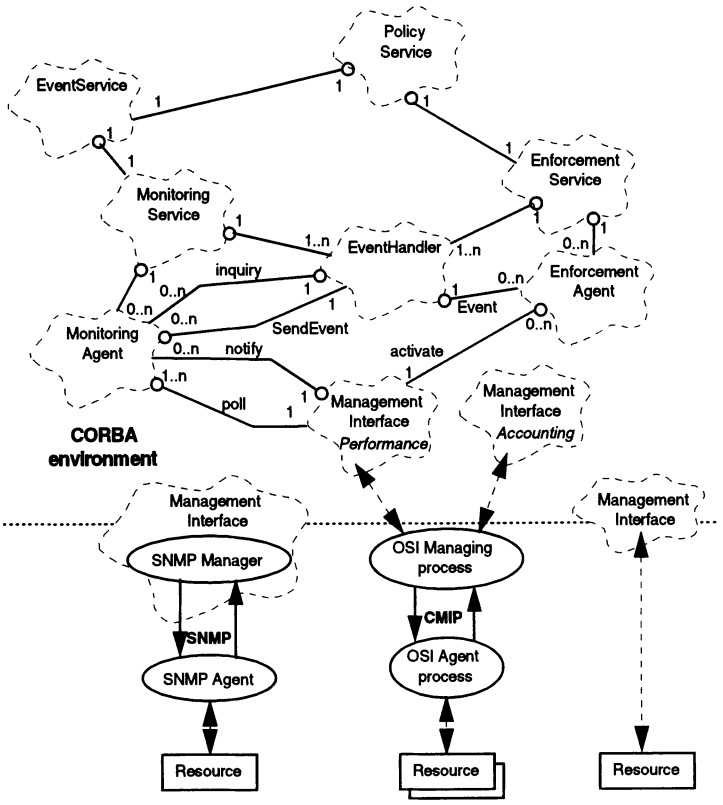

Figure 2 Agents and Interfaces in a CORBA environment

or a combination of critical values is reached. The functionality of the monitoring service comprises operations for:

- Instantiation and removal of monitoring agents based on informations provided by the event service.
- Management of agents including behavior adaption to different polling intervals, thresholds, etc.
- Provision of monitored values on request from the management service.

Monitoring is performed by generic monitoring agents (Koch and Krämer 1996, Koch 1995) which are installed by the monitoring service and located close to the managed object.

The management architecture is realized as a distributed CORBA application, therefore the interfaces are described in CORBA IDL. Figure 2 provides an overview of several

possible management interfaces. On the left side an interface to the Internet management environment is displayed. Access to the SNMP manager is performed with method invocations on a CORBA interface. Number and complexity of interface functions depend on the capability of the SNMP manager. The management interface will be implemented as CORBA server and the capabilities are fixed with the code. Therefore the functionality of the SNMP manager is not a subject of flexible policy definitions. Any change to the behavior requires recoding of the CORBA server. It is an important design decision to provide most general but still powerful capabilities. The minimum capabilities are prescribed by the functionality of the SNMP protocol. This simple approach is obviously unsufficient for comprehensive management tasks. Enhanced capabilities depend strongly on the type of resource that will be managed. The use of the CORBA interface inheritance mechanism could provide a solution. An elementary SNMP management interface to the CORBA environment provides the SNMP functionality only and serves as parent class for more specialized interfaces providing enhanced functionality for certain types of resources. This approach additionally ensures openness to new resources that may provide an enhanced SNMP MIB.

A similar problem occurs with the inclusion of the OSI management environment. A different partitioning of interface functionality is used for representation of an OSI managing process in the middle of Figure 2. Obviously the same style could be used for the representation of the SNMP manager or conversely. The problem of assigning appropriate capabilities to the interfaces could also be solved in the same way as proposed for the SNMP interface.

Figure 2 also illustrates the major advantage of a management environment that is implemented as a CORBA application. The architecture is open to any management protocol as long as an appropriate interface is defined. This allows the inclusion of applications as well as system and network components into a comprehensive management environment. The CORBA standard additionally ensures interoperability between different management service objects, running on different CORBA implementations.

3 BEHAVIORAL PARAMETERS

Unexpected operational state changes of system components imply that the service availability of components can not be guaranteed in general. State changes could either refer to the operational availability of a component or to the quality of the provided service. In the sequel, we shortly present these parameters. The interested reader can find additional details on the health behavioral parameters in the appendix, and complete information in cited papers.

3.1 Availability parameters

Several run-time measurable features for the operational availability have been proposed and defined by Dini, v.Bochmann and Boutaba (1996) as follows. *Current availability* $h(t)$ is a continuous function of time, defined as a quotient between the amount of time in which the resource has been in the enabled state, commonly called operational time, and the observation period, i.e. the period between the time t_0 of the event start and the time

t of the end of the observation period. The notation t_i represents time stamps attached to each event occurring in a system.

3.2 Dynamics parameters

Dini (1996) characterized dynamic aspects of the component behavior by several computable features. These features refer to either stability or instability of the component. The stability order k evaluates how long an operational state holds, with respect to the preceding change. There are three instability measures: the *instability order*, the *repeatability order*, and the *multiplicity order*. The instability order p gives a measure of how long in time an operational state holds. The repeatability order r defines a local instability within a vicinity of a change, whereas the multiplicity order m refers to a long time instability. Each order is dynamically computed as an integer number and the appropriate time stamp is attached to it. Time stamp lists corresponding to p, r, and m orders form the time behavioral history of a system component.

4 ADAPTIVE POLLING FREQUENCY

In this section we propose a model for an optimized monitoring agent that adapts its polling frequency automatically according to predefined criteria. The monitoring agent is initialized with a basic frequency computed for the monitored system component. We assume that the basic frequency f_0 is deduced from formulae of existing approaches presented by Stallings (1993) and Schwartz (1987). Since all these approaches ignore behavioral aspects related to the monitored components, the basic frequency f_0 represents the minimum polling frequency established with respect to global conditions. This basic frequency will be modified by use of a correction coefficient. The proposed model allows two different adaptation styles: The polling frequency is either adapted with respect to the behavior history of the monitored component, or with respect to the current value of the monitored attribute.

4.1 History based adaptation

We assume that a basic polling frequency f_0 is computed with respect to global system parameters and known by the agent. This basic frequency will be adapted by the correction factor ζ_i according to the following equation:

$$f_{0_i} = f_0 (1 + \zeta_i) \qquad \text{for } i = 1, \ldots, 4 \tag{I}$$

Consequently, the basic value of f_0 can be calculated independently according to global statistical parameters, whereas the right polling frequency is increased by the fraction $\zeta_i f_0$. This term signifies the correction we apply by taking into account the health behavioral parameters presented in Section 3. Next we will present two sets of formulae which linearly or exponentially correct the basic polling formula.

Linear adaptation

Four different formulae are proposed for the calculation of the correction factor ζ_i.

Formula 1 The *current availability* computed at the last time stamp t is represented by $h(t)$. The unavailability at t is $1 - h(t)$. Then, a first correction factor is defined with respect to these two related parameters. The quotient between the availability and unavailability is a relevant evaluation of the operational state. The first formula that we propose is to adapt the basic polling frequency by:

$$\zeta_1 = (1/h(t) - 1) \tag{1}$$

Formula 2 If the dynamics parameters are critical features for a component, we can combine them with the availability features. For the moment, we consider only the stability orders (called k and k' for the enabled and disabled operational state, respectively). Intuitively, with increasing values of k or k' the polling interval should increase, i.e. the polling frequency decreases. Therefore the polling frequency is conversely proportional to the stability orders. Since k and k' are finite integers, for $k \neq 0$ and $k' \neq 0$, we propose:

$$\zeta_2 = \zeta_1 + max(1/k, 1/k') \tag{2}$$

Formula 3 Commonly, an unavailability described by an instability order p with $(t_i - t_{i-1} = 10^{-p})$ is considered to be in the range $[p_{min}, p_{max}]$, where p_{mim} is the lowest measured value and p_{max} is the highest value accepted for a component type. Consequently, $1/(p_{max} - p_{min})$ is the ratio for each p's unity which can be added to the previous equation (2).

$$\zeta_3 = \zeta_2 + p_0/(p_{max} - p_{min}), \text{where } p_0 = max\{p_i | t < t_i\} \tag{3}$$

Formula 4 For some network components, the repeatability and multiplicity orders are relevant management features. Since $m = 1$ and $r = 1$ are the lowest limits, we assume that m_{max} and r_{max} are extreme limits accepted for an order p. Naturally, when a component behaves with $r > 1$ and/or $m > 1$, the polling frequency must be adapted in a certain manner. If we consider m_0 and r_0 similarly to p_0, we could improve the coefficient expression:

$$\zeta_4 = \zeta_3 + m_0/(m_{max} - 1) + r_0/(r_{max} - 1) \tag{4}$$

Remarks: With respect to the availability, Formula 1 multiplies the basic polling frequency by a coefficient in the range [0, 1], if we admit a minimum availability of 0.5. Each of the subsequent formula increments the maximum value of the coefficient by one, with a maximum increase of $f_{0i} = 6 \times f_0$ if Formula 4 is used.

Exponential adaptation

Another way to weight the current behavioral parameter values is to consider the exponential function, which better emphasizes new values with respect to their magnitude. Let us consider a large range of the interval $[a, b]$ described by $\exp([a, b])$. If $z \in [a, b]$, the relation $z \leq (e^z - 1)$ holds for all z. Consequently, we have a larger spectrum to distribute the polling frequency within the same range of ζ_i. Each of the previous formulae becomes $\zeta_i' = \exp(\zeta_i)$ and the maximum polling frequency is given by $f_{max} = f_0 \times 6 \times (e - 1)$, where $e = 2.73...$

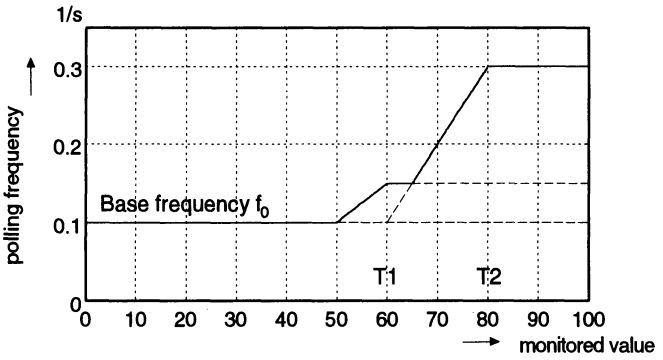

Figure 3 Threshold based frequency adaptation

4.2 Threshold based adaptation

The second adaptation style is based on the idea that in general the polling frequency should be increased if the measured value gets close to a predefined threshold. The specification of the threshold based frequency adaptation is very simple. The system administrator specifies a base frequency f_0 and up to n thresholds with individual characteristics. The concept is illustrated in Figure 3 with a base frequency $f_0 = 0.1\ ^1/_s$. Here the monitored value is in the range $[0, 100]$, which is typically for some kind of load measured in %. The monitoring agent will create a notification at the threshold values 60 (T_1) and 80 (T_2), respectively. A more detailed description of the agent behavior and the optional characteristics for each threshold is given by Koch, Krell and Krämer (1996), and in Koch and Krämer (1996). If a frequency adaptation is desired, the administrator may optionally specify three additional values for each threshold:

Factor: The factor α_j describes the desired polling frequency at the corresponding threshold T_j. The desired frequency is calculated by multiplication of the base frequency with α_j.

$$f_{T_j} = f_0 \times \alpha_j$$

The example in Figure 3 uses the factors $\alpha_1 = 1.5$ for T_1 and $\alpha_2 = 3$ for T_2.

Low: The low value β_{lj} describes the frequency adaptation for measured values *below* the corresponding threshold T_j. For simplified definition the agent assumes a linear increase from the base frequency to the increased frequency defined with α_j. The frequency on a slope is therefore described by

$$f_{lj}(v_k) = a_{lj} \times v_k + b_{lj} \qquad \text{(II)}$$

where v_k is the last monitored value. Based on this definition β_{lj} defines the value where the increased slope will meet the base frequency. The following equations are used to determine the parameters a_{lj} and b_{lj}:

$$a_{lj} = \frac{\alpha_j - 1}{T_j - \beta_{lj}} f_0 \tag{III}$$

$$b_{lj} = f_0 \left(1 - \frac{\alpha_j - 1}{T_j - \beta_{lj}} \beta_{lj}\right) \tag{IV}$$

In the example the values $\beta_{l1} = 50$ and $\beta_{l2} = 60$ are used.

High: The high value β_{hj} mirrors the functionality of β_{lj} for the degrading slope *above* the threshold T_j. The equations (II), (III) and (IV) are used with an index h instead of l for definition of the slope. In the example no values for β_{hj} are defined, therefore the agent uses by default a vertical slope for all values above the threshold.

The polling frequency will be adjusted in every measurement cycle. The agent always uses the highest value if several slopes are applicable. The resulting curve is printed as a bold line in Figure 3.

The threshold based approach could be used in combination with the history based adaptation as presented in Section 4.1. In the combined approach, the base frequency f_0 will be adapted by (I) according to the history of the managed component.

5 IMPLEMENTATION ASPECTS

As explained in Section 2 our architecture is implemented in a CORBA environment, we use the Orbix implementation (IONA 1995). The monitoring agent is therefore realized as a CORBA object consisting of several modular components. The modularity allows an easy exchange or enhancement of the internal components to create a more specialized agent (Perrow, Hong, Lutfiyya and Bauer 1995).

Figure 4 gives an overview of the generic agent. The agent provides two interfaces: The service interface is used to retrieve monitoring information from the agent, like the current value w_k or the content of the buffer. The management interface provides functionality for controlling the agents behavior.

Organizer. This component coordinates the internal activity of an agent and provides the functionality for both service and management interface. At startup time the organizer reads configuration information from an initialization file and passes the appropriate parameters to the components (indicated with dashed arrows in Figure 4). Monitoring is performed automatically with an adaptive measurement frequency. To ensure data integrity, invocations on any of the interfaces are blocked until a complete measurement and evaluation cycle is finished.

The **Filter.** transforms an input stream of values (here v_k) into an output stream (w_k) according to a number of predefined filter functions. They include identity, median with window size, and medium with window size.

The **Trigger** function performs a call to the Event Handler whenever a predefined

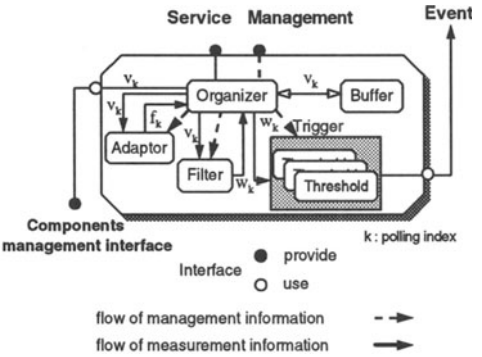

Figure 4 Monitoring agent with frequency adaptation

condition occurs on the filtered data stream. An arbitrary set of threshold values with different characteristics for each threshold can be programmed.

The **Adaptor** implements the algorithms for the calculation of an adequate polling frequency as described in Section 4. The adaptor learns about every measured value v_k and provides a new frequency f_k for the organizer.

Buffer. This component stores up to N previous measurements, where the buffer size N is adjustable through the management interface. The buffer is organized as a ring buffer, that is, as soon as the buffer is filled for the first time, it starts to override old values according to a FIFO strategy.

6 CONCLUSION

We have presented a distributed management architecture in which monitoring is performed by distributed agents with generic functionality for filtering, polling frequency adaptation and event creation. The distribution of active management agents reduces the amount of management related traffic and avoids a potential bottleneck on a centralized management station. The automatic adaptation reduces the performance impact of the agents significantly while at the same time a high accuracy of management relevant information about critical components is ensured. Several adaptation strategies allow a flexible configuration of the agent according to the individual requirements of the managed component.

REFERENCES

Dini, P. (1996) New Aspects for Run-time QoS Evaluation in Networks and Distributed Systems. In *European Simulation Multiconference*, pages 3–11, Budapest, Hungary,

June 1996. Tutorial.

Dini, P., v.Bochmann, G. and Boutaba, R. (1996) Performance Evaluation for Distributed Systems Components. In *2nd Int. IEEE Workshop on Systems Management*, pages 20–29, Toronto, Canada, June 1996. IEEE. ISBN 0-8186-7442-3.

Koch, T. and Krämer, B.J. (1996) Rules and agents for automated management of distributed systems. *Distributed Systems Engineering Journal* **3**, pages 104–114.

Koch, T., Krell, C. and Krämer, B.J. (1996) Policy Definition Language for Automated Management of Distributed Systems. In *2nd Int. IEEE Workshop on Systems Management*, pages 55–64, Toronto, Canada, June 1996. IEEE. ISBN 0-8186-7442-3.

Koch, T. (1995) Rule Based Management Architecture with Smart Agents. In M. Sloman and T. Usländer, editors, *Proc. of the Int. Workshop on Services for Managing Distributed Systems*, Karlsruhe, September 1995. Fraunhofer IITB.

IONA Technologies Ltd., Dublin, Ireland. *Orbix Programmer's Guide*, 2.0, November 1995.

Perrow, G.S., Hong, J.W., Lutfiyya, H.L. and Bauer, M.A. (1995) The Abstraction and Modelling of Management Agents. In Sethi, Raynaud and Faure-Vincent, editors, *Integrated Network Management, IV.* IFIP, Chapman & Hall, pages 466–478.

Schwartz, M. (1987) *Computer-Communications Networks Design and Analysis.* Prentice-Hall.

Stallings, W. (1993) *SNMP, SNMPv2, and CMIP: The Practical Guide to Network Management Standards.* Addison-Wesley.

APPENDIX 1 AVAILABILITY AND DYNAMICS HEALTH PARAMETERS

Availability parameters

Current availability. *Observed availability* $h(t)$ is a feature of a component representing the availability of the component's services up to a given time. Its value defines how long this component has been in operational state with respect to the observation period T.

$h : [t_0, T] \rightarrow [0,1] h(t) = t_{op}/(t - t_0)$ where t_{op} is computed within $[t - t_0]$

Within a period $[t_0, T]$, the availability $A(T)$ is considered as equal to $h(T)$. The availability of a component type is commonly defined as the average of the $h(T)$ of all components of this type across a period $[t_0, T]$. Since the availability is a global evaluation, A is calculated across many periods $T_1, T_2, T_3, \ldots$

Minimum current availability. Across an observation period, the minimum value of $h(t)$ is called *minimum observed availability* and is defined as:

$h_{min} : [t_0, T] \rightarrow [0,1]$

$h_{min}(t) = min\{h(t) | t_0 \leq t \leq T\}$

Since we have concluded that the local extreme values of $h(t)$ occur when a state change event arrives, $h_{min}(t)$ is among these peak point values.

Weighted current availability. As defined above $\underline{h}$ is a global evaluation without distinction between later or recent $h(t)$ values. Since the health is the current observed availability, it seems necessary to emphasize the recent health values more with respect to the earlier ones. We introduce the weighted average of observed availability which exponentially weights the later $h(t)$ values. Consequently, the $h(t)$ values will be taken into account by an exponential power. This power is dependent on the time and the change intervals, as follows:

$$\overline{h}(t_i) = \frac{h(t_{i-1}) + l \times h(t_i) \times \exp(t_i - t_{i-1})}{1 + \exp(t_i - t_{i-1})} \quad \text{for } (i > 0)$$

where $l = 0$ if the operational state within $[t_{i-1}, t_i]$ is disabled, and $l = 1$ if the operational state within $[t_{i-1}, t_i]$ is enabled.

Dynamics parameters

Stability order. The *stability order* k accurately qualifies a stationary state with respect to change frequencies. We say that a state has the order k of stability if for $t_i - t_{i-1} \approx k \times t_{i-1}$ no state change event occurs.

Instability order. The *instability order* p refers to state change events only (no commands) and consequently depends only on the t_i time stamps. We define the instability order of magnitude p if $\delta_{i-1,i} \approx 10^{-p} \times t_{i-1}$. Explicitly, the calculus formula is obtained by a logarithmic function $p = \lceil -\log(t_i - t_{i-1})/t_{i-1} \rceil$, where $\lceil a \rceil$ is the greatest integer less than or equal to a. The instability order is therefore an integer number.

Within a given time period t_T, a component could have different instability orders $\{p_1, p_2, \ldots p_s\}$. The instability of its operational state is first described by $[p_{min}, p_{max}]$, where $p_{min} = min\{p_i | 1 \leq i \leq s\}$ and $p_{max} = max\{p_i | 1 \leq i \leq s\}$, and second, by the complete set of order values.

Repeatability order. If the instability occurs consecutively, say r times, we call that the *instability of order* p has a *repeatability of order* r.

Thus, if $\delta_{i+j-2,i+j-1} \approx 10^{-p} \times t_{i+j-2}$, for $j = 1, 2, \ldots, r$, the managing objects must consider not only the health value, but equally, the (p, r)-*instability* of order p and *repeatability* of order r.

An instability order p_i could have, in turn, several repeatability orders $\{r_{i0}, r_{i1}, \ldots r_{iw}\}$. As for the instability order, each p_i order is characterized by $r_{i/min} = min\{r_{ij} | 0 \leq j \leq w, 1 \leq i \leq s\}$ and $r_{i/max} = max\{r_{ij} | 0 \leq j \leq w, 1 \leq i \leq s\}$. Semantically this is equivalent to the minimum, and respectively maximum number of consecutive change intervals of the same range p, within an approximation given by the error of the function $f(x) = \lceil x \rceil$. The repeatability order is also an integer number.

Multiplicity order. The *multiplicity order* m represents the number of times a real resource has been involved into an instability of order p_i (with or without a repeatability order) during a given period. Consequently, at any pair (p_i, r_{ij}) one could attach the multiplicity order $m_{(pi,rij,T)}$ which captures the number of times the instability of order p_i and repeatability r_{ij} have been raised within the t_T time period. The multiplicity order is an integer number.