Artificial intelligence in the service of system administrators

Article *in* Journal of Physics Conference Series · December 2012

DOI: 10.1088/1742-6596/396/5/052038

4 authors, including:

Vincent Barra
Université Clermont Auvergne
104 PUBLICATIONS **1,175** CITATIONS

SEE PROFILE

Artificial intelligence in the service of system administrators

C HAEN, V BARRA, E BONACCORSI and N NEUFELD

[1] Université Blaise Pascal, 34, avenue Carnot - BP 185, 63006 CLERMONT-FERRAND cedex, FR

[2] European Organization for Nuclear Research, CERN CH-1211, Genève 23, CH

E-mail: christophe.haen@cern.ch

Abstract. The LHCb online system relies on a large and heterogeneous IT infrastructure made from thousands of servers on which many different applications are running. They run a great variety of tasks : critical ones such as data taking and secondary ones like web servers. The administration of such a system and making sure it is working properly represents a very important workload for the small expert-operator team. Research has been performed to try to automatize (some) system administration tasks, starting in 2001 when IBM defined the so-called "self objectives" supposed to lead to "autonomic computing". In this context, we present a framework that makes use of artificial intelligence and machine learning to monitor and diagnose at a low level and in a non intrusive way Linux-based systems and their interaction with software. Moreover, the multi agent approach we use, coupled with an "object oriented paradigm" architecture should increase our learning speed a lot and highlight relations between problems.

1. Introduction

LHCb [1] is one of the four large experiments at the Large Hadron Collider at CERN. The infrastructure of networks and servers deployed in order to manage the data produced by LHCb and control the experiment are critical for its success. The LHCb online system [2] relies on a large and heterogeneous IT infrastructure, which consists of thousands of servers, representing a multitude of different hardware configurations. The team responsible for the whole infrastructure is composed by only three full time workers and part time helpers, mainly students. This results in a huge workload per person, including night on-call duty and a potential loss of knowledge every time a student leaves the team.

To address this problem, we are looking for a solution based on artificial intelligence (A.I.), which would improve with experience and also act as a knowledge and problem history database. The final goal is to ease the work of our system administrator team.

2. State of the art

The first attempts that were done were based on the so called 'rule based expert system' [3], which consists of a knowledge base and an inference engine. The knowledge base, defined as a set of rules, is very modular and all the conclusions given by the engine can be explained to the user. Nevertheless, it is very difficult and demanding to keep the base up to date. Moreover, the diagnostics of the engine do not improve with experience.

In 2001, IBM introduced the concept of 'autonomic computing' in a manifest [4] regarding the coming problems of IT infrastructures. The major issue pointed out was the increasing complexity of the installations. The solution IBM offers [5] [6] is to give 'self-managing' capabilities to IT systems. It relies on two fundamental concepts, which are still nowadays the bases of all autonomic systems [7] :

- The self-X : in order to be 'self-managing', a system must verify four properties :
  - self-configuration : auto configuration, following high level directives
  - self-optimization : system should improve its performances
  - self-healing : system detects, diagnoses, repairs
  - self-protection : against attacks, or cascade failures
- MAPE-K loop :
  - monitor
  - analyze
  - plan
  - execute
  - knowledge

  This control loop is the reference model. It introduces the different working steps which lead to autonomic computing.

In principle, every element of an autonomic system (e.g. web services, DB cluster) should implement its own MAPE-K loop. This naturally leads to a multi-agent system : the autonomic systems are supposed to dialogue and negotiate in order to keep the whole system running. Many implementations of this loop have been done. Some aim at being generic and are distributed as frameworks [8], [9], [10], [11], while other approaches are much more specific [12],[13]. In any case, there are two situations : either we have access to the source code of the software, in which case it is possible and better to adapt it to give it autonomic capabilities. Or the source code is completely hermetic.
Most of the papers concern the first situation, which covers various cases :

- Writing new software following some principles [14], [15]
- Adapting the code by following principles of aspect oriented programming [16], [17]

Some papers offer a method that they call 'non intrusive', which only works for Java programs, since their solution consists in injecting code at runtime via the Java Virtual Machine [16].
Papers about the second case are very rare because the only grip we can have on software are those offered either on purpose via API, or via security issues. It is thus very hard to have a generic approach.
It is worth to mention one other approach, which consists of giving the users a new language in order to describe their system [9],[18].

Another common point of the different approaches is that they are 'model based' or 'architecture based'. This means that the autonomic mechanism has a simplified representation of the system, which allows you to see whether the system behaves like expected (a monitoring system) and eventually to test changes on the model before applying them. This methodology is also used in systems based on the control theory and attempts have been made to mix control theory and autonomic computing [14],[19],[20].

In conclusion, one can find several common points :

(i) In one way or another, there is a rule based system which makes the link between measurements and actions

(ii) Need to describe what to monitor

(iii) Need to describe how to monitor

(iv) Need to specify the rules to trigger an action

(v) Need to specify what to do in case of problems

3. Architecture

3.1. Method

The good results obtained by the MAPE-K loop suggest to follow this principle. The last four points quoted above can be improved and this is what we aim at.

First of all, we want to work in the most general context possible, by working with the OS (Linux only) system calls in order to get status information about processes or files. Those calls exist whatever the software is. Also their behaviors and the data possible to retrieve are precisely defined and formated by standards. It is possible in Linux to see any application as a set of elementary entities (e.g. processes, files) with well defined attributes. The flip-side is that we have to work at low level and not use high level directives.

The second point concerns the MAPE-K loop itself. The existing solutions are using one loop per system and multi agents theory to make them communicate. We believe that the MAPE-K loop can be unique for all the systems and that the monitoring can be independent of the internal behavior of a system and thus be taken out of the MAPE-K loop. The monitoring can then be looked at in terms of working services rather than in terms of working components of a service. It also offers the advantage that we can transparently use third-party software for monitoring (like Icinga [21]). Regarding monitoring, 'Model based' approaches to detect problems are very efficient, but painful to put in place, especially in case of large and heterogeneous infrastructure like in LHCb. Writing simulation models and describe all the interactions between the components is an almost insurmountable task. In order to solve that problem, we offer to use artificial intelligence and learning algorithms : systems like neural networks, once trained, can perfectly substitute to a simulation model. One can thus associate an A.I. system to each system one wants to diagnose.

The basic idea of our approach is that if at the low level, all the elementary entities of an application are in an appropriate state, the application should also behave properly. Nevertheless, for big applications, the amount of entities makes it difficult to check all of them. We thus need to optimize the diagnostic and reinforcement learning seems to be an appropriate way. This kind of technique will allow us to minimize the amount of checks before finding the faulty entity.

Unfortunately, the reinforcement techniques require a big amount of training and replications to be efficient and it is not possible in an environment like LHCb. Experience Sharing between the various systems becomes a key point [22] - a basic web server works the same way, no matter the content it serves. Our proposition is to copy the object model: defining an application as a composition/aggregation of subsystems, using concepts like inheritance and instance in order to share the experience and optimize the convergence speed when solving a problem. It thus becomes possible to describe a general website only once and then, for every actual website, simply refine parameters (like path to the configuration files), reducing the workload for the users considerably.

On top of that, using a very generic but well defined interface (interaction point between OS and software) allows us to use the principle of 'convention over configuration', which encourages the default behavior.

To summarize :

- 'Abstract profiles' reduce the 'description' workload a lot and they increase the learning speed. Current work is ongoing to automatically generate actual profiles by observing servers and matching the observations with abstract profiles.
- Having a more service oriented approach for monitoring and decoupling it from the diagnostic allows to use existing or already in place solutions. Making use of artificial intelligence techniques let you benefit from the advantages of the model based approach and reduces the disadvantages.
- Following the principle of 'Convention over Configuration' reduces the work and this leads not only to diagnostics, but also to recovery solutions.
- A unique MAPE-K loop will help to point out relations between services that might have been forgotten by the user in the profiles, highlight monitoring failures and warn of coming problems.

3.2. Analysis of a problem

As each of the elementary entities is defined at the system level, we can develop a so called 'Agent' for each of them. They are in charge of checking the proper behavior of the entity and offering a default solution in case of problem. At a higher level, the analysis of a system is based on its description (called 'Profile') as a composition of other systems and Agents. Each profile is managed by a so called 'Coordinator', which contains reinforcement learning algorithms : it essentially chooses the order in which the sub-Profiles and Agents should be checked to find the responsible of a problem and optimize its choices with experience.

The general algorithm to analyze a problem list is described in Fig. 1

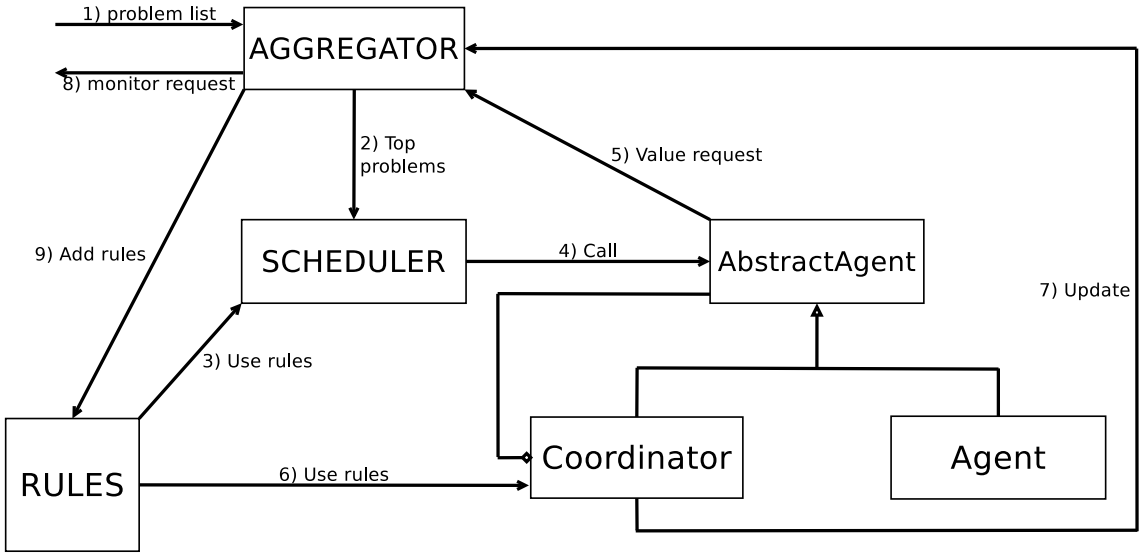

Figure 1. Principle of the Analyzer Core

(i) Ongoing problems are given as a list of Profile names to the Aggregator, which will keep the exhaustive list of alarms during the diagnostic. It then performs a sort in the alarms to keep only the lowest in the profile tree.

(ii) All the 'top' problems are given to the Scheduler, which chooses in what order to solve them.

(iii) For this, it uses priority rules (e.g. the web site needs the database to work) either described in the Profiles, or learned with the experience.

(iv) The scheduler then calls, in order, the Coordinators responsible for the given profiles.

(v) Coordinators can eventually ask Aggregator information about alarms. This functionality is not yet used.

(vi) To establish a priority order, Coordinators also use the rules database, or use classical reinforcement learning techniques based on probability.

(vii) Once a problem is solved by a Coordinator, it returns to the Aggregator

(viii) The Aggregator will update the alarms list.

(ix) The results of the previous operation will allow to find out new dependencies and add rules in the database that need to be acknowledged by the user in order to become effective.

Values measured by Agents can be kept periodically and regularly. Basic techniques of data mining (polynomial regression, Student tests on means) will then permit to warn the user about potential problems to come (e.g. lack of disk space). More complex techniques can also be applied in order to find dependencies more elaborate than direct rules.

4. Results

4.1. Monitoring

A light monitoring software has been developed. Its purpose is not to replace more complete software like Icinga, but rather to use their results to feed A.I. algorithms. The monitoring software relies on external dynamic libraries for the A.I. algorithms. Thus, any algorithm can be added, as long as it respects the interface.

4.2. Analyze

At the time of this writing, a first version of the Analyzer Core has been developed. All the efforts have been put into the diagnostic algorithms. This means that currently, the software is able to diagnose a problem, but cannot offer a solution to it. This will be easy to implement, because of the well defined behavior of the atomic entity. The data mining part is also planned on a longer term road-map.

4.2.1. Test bench The aim of the test bench is to represent a situation that we actually have in production and to see how the analyzer reacts to problems. The test bench is composed of the following :

- One NFS server
- One MySql server
- One website (Site 1) getting its content from Mysql
- One website (Site 2) getting its content from NFS and Mysql

The information entered in the database is the following :

- NFS
  - nfs daemon
  - files it serves
  - exports file
- Mysql
  - mysql daemon

- Site 1
  - http daemon
  - files it serves
- Site 2
  - http daemon
  - files it serves
  - fstab entries

The tests aim at showing the advantage of Experience Sharing between similar entities (Site 1 and Site 2), of having priority rules and the ability of the software to discover those rules. Indeed, for our tests the dependencies are not given in the start configuration. The test protocol consists in creating up to four simultaneous random problems on the various systems and measure the actions of the software under various circumstances :

- No rule used, no Experience Sharing.
- Rules effective after appearing 200 times. No Experience Sharing.
- Rules effective as soon as appearing. No Experience Sharing.
- Repeat previous three situations but with Experience Sharing between Site 1 and Site 2.

The first value of interest is the percentage of diagnosed problems in the various tests. Fig. 2 makes clear that the situation improves a lot: from 90.8% to 99.8%. An analysis of the simulations shows that the non spotted errors are actually found, but considered as false positives. This situation appears when distinct problems happen at the same time and have similar consequences on the alarms sent by the monitoring. For example, if the nfs or the http daemons are down, Site 2 becomes unreachable for the monitoring. If both daemons are down at the same time, the nfs daemon needs to be fixed first. If the software decides to fix httpd first - because there is no priority rule - it will get the false positive situation.

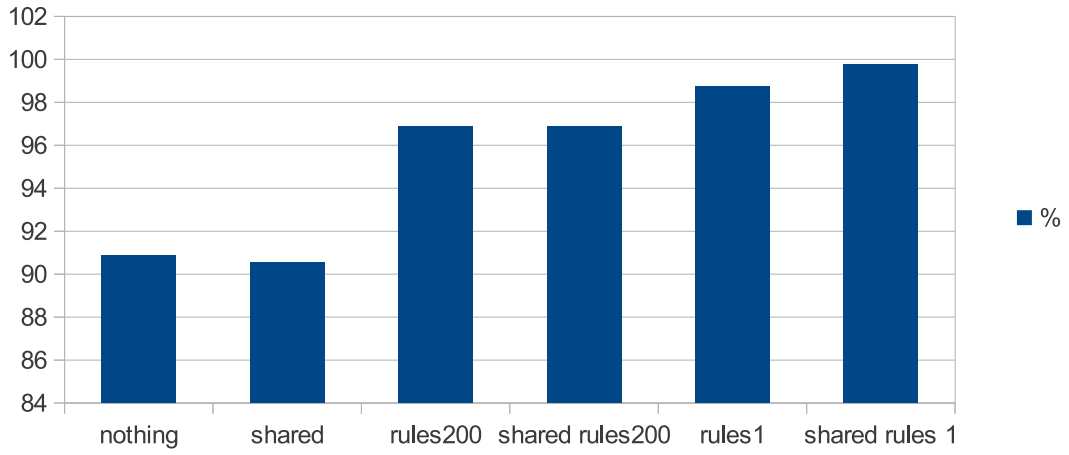

Figure 2. Percentage of problems diagnosed

The second value of interest is how many time a users or a monitoring tool needs to interfere with the software in order to solve a problem. In order to update probabilities and deduce priority rules, the aggregator needs to update its alarm list after each attempt to solve one of the problems. In case of n concurrent problems, at least n interactions are needed. Fig. 3 shows

the percentage of extra actions (i.e. more than the optimal) in different situations. The priority rules clearly help to improve the situation. The 20% of the cases that required additional user intervention, when no rules were used, are managed in an optimal way if we use the rules as soon as they are deduced.

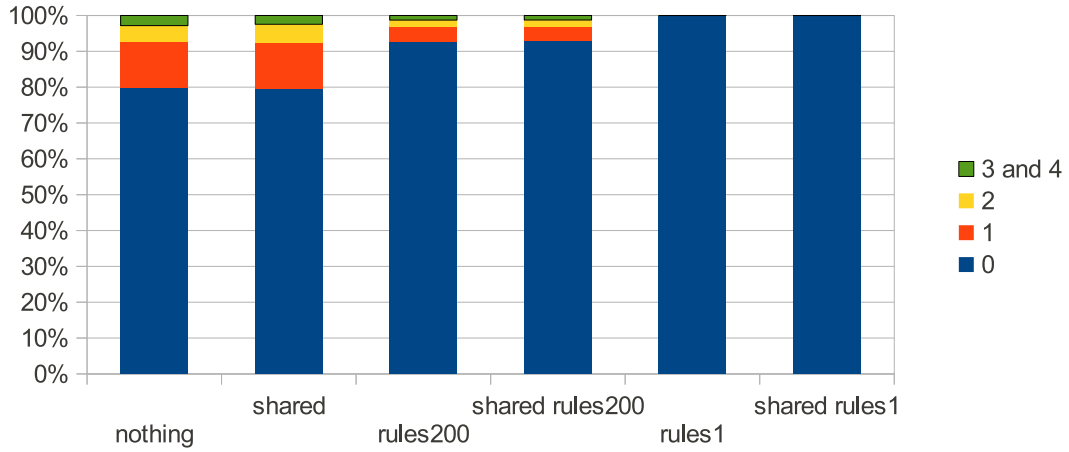

Figure 3. Additional user actions with respect to the optimal solution

To finish, it is interesting to compare the amount of involved agents when solving problems. Less involved agents, means faster solving. Fig. 4, similar to Fig. 3, shows how many additional agents are visited. In that case as well, the priority rules improve the percentage of optimal resolutions from 60% to 89.6%. The percentage of situations requiring more than one additional check decreases from 32% to 1%.

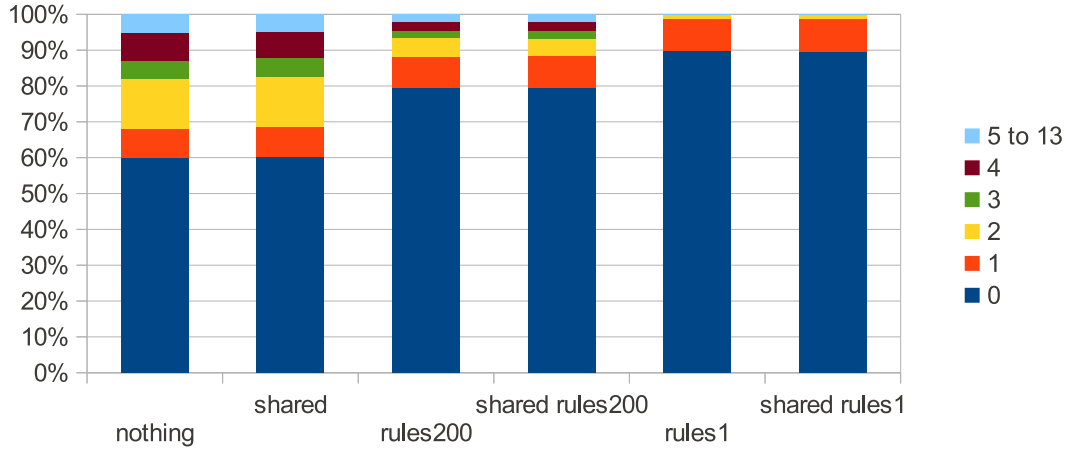

Figure 4. Additional visited agents with respect to the optimal solution

Finally, in all situations, the dependency rules were found by the software. The added value of Experience Sharing is not really obvious on those plots. The biggest gain is about half a percent compared to the replication without the Experience Sharing. This can be explained by the amount of replications, which is in the order of thousands. The sharing is supposed to minimize the consequences of the lack of replication. Another reason is the small amount of children of the coordinator in our example. With only two children per site, we have a 50% chance to take the faulty child.
In order to illustrate the benefits of Experience Sharing, we ran a trivial test : we declared 10 profiles, all composed of 10 files and we simulated problems on those files. In 50% of all cases a selected file had a problem. The other 50% of failures are equally shared between the remaining 9 files. Only 20 problems were simulated, once with the Experience Sharing between all the profiles, once without. Fig. 5 shows that the number of optimal resolutions goes from 16.25% to 25.75% while there are 13.75% less of cases requiring more than five extra checks. Figs. 6 illustrates the learning speeds by comparing the number of optimal resolutions : while the shared curve reaches the optimal value ($50\% \pm 5\%$) after about 30 replications, the not shared curve needs about 175 replications.

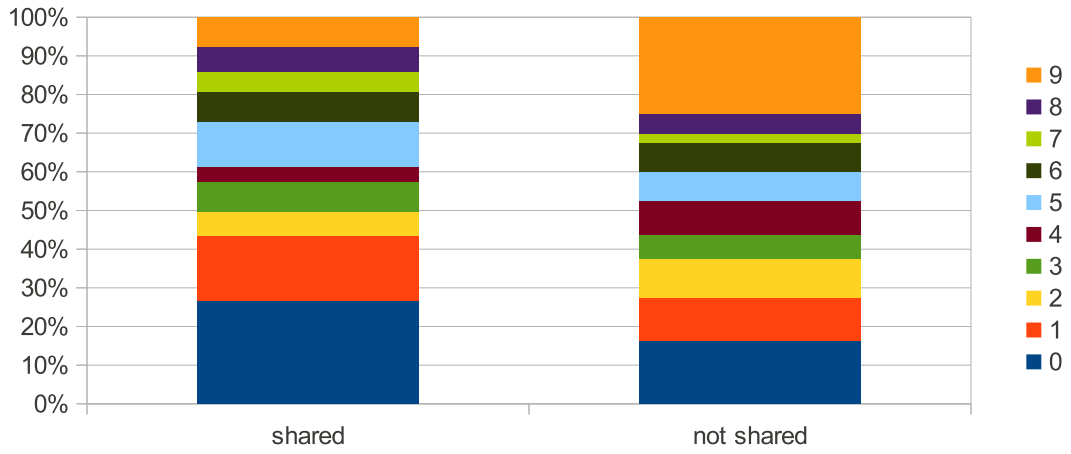

Figure 5. Additional visited agents on the second simulation

4.2.2. Real test The first version of the software has now been used for a few weeks against our real web services. This setup is very similar to our test bench, including nfs file server and database for the contents. Nevertheless, in that case, we refined the dependencies. At the time of writing, all the causes of service interruption could be properly diagnosed by the software.

5. Outlook

In the coming weeks, the plans are to improve the code of the agents to cover also more exotic cases and to add suggestions to fix the problems. Further improvements concern the data mining functions. We also plan a non interactive mode of running where the software does not try to improve its rules.
In parallel of this, efforts are put into tools in order to manipulate the software (CLI and GUI interfaces) and the profiles (definition of advanced configuration, code generator, automatic generation of profiles from meta-profiles).

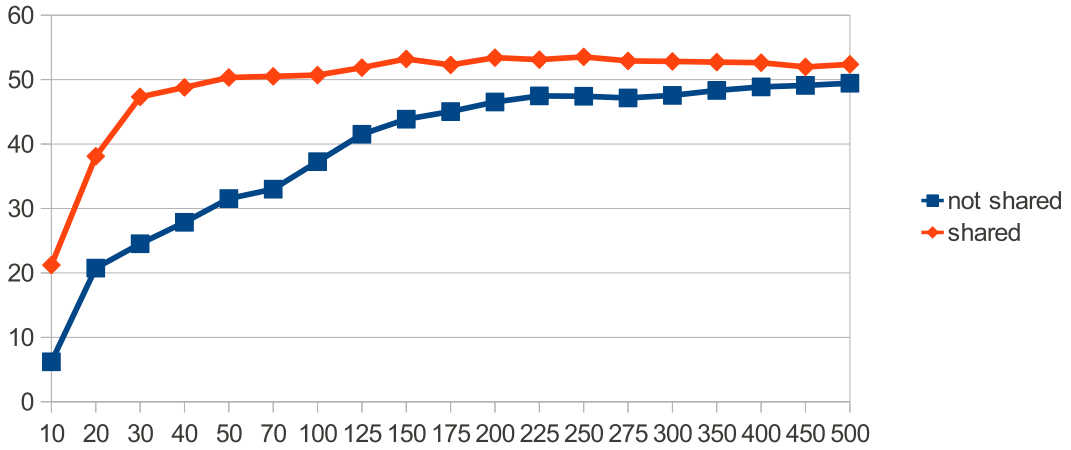

Figure 6. Optimal solving (in %) vs. replications

References

[1] Alves A A *et al.* (LHCb) 2008 *JINST* **3** S08005

[2] Neufeld N (LHCb) 2003 *Nucl. Phys. Proc. Suppl.* **120** 105–108

[3] Ginsberg M 1993 *Essentials of artificial intelligence* (Morgan Kaufmann)

[4] Horn P 2001 Ibm's perspective on the state of information technology URL `http://www.research.ibm.com/autonomic/manifesto/`

[5] IBM An architectural blueprint for autonomic computing

[6] Kephart J and Chess D 2003 *Computer* **36** 41 – 50 ISSN 0018-9162

[7] Huebscher M C and McCann J A 2008 *ACM Comput. Surv.* **40**(3) 7:1–7:28 ISSN 0360-0300 URL `http://doi.acm.org/10.1145/1380584.1380585`

[8] Bigus J P; Schlosnagle D A P J R M I W N D Y 2002 *IBM Systems Journal* **41**(3) 350–371

[9] Cheng S W, Garlan D and Schmerl B 2006 *Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems* SEAMS '06 (New York, NY, USA: ACM) pp 2–8 ISBN 1-59593-403-0 URL `http://doi.acm.org/10.1145/1137677.1137679`

[10] Kaiser G, Parekh J, Gross P and Valetto G 2003 *Autonomic Computing Workshop. 2003. Proceedings of the* pp 22 – 30

[11] Li T, Peng W, Perng C, Ma S and Wang H 2010 *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* **40** 90 –99 ISSN 1083-4427

[12] Martin P, Powley W, Wilson K, Tian W, Xu T and Zebedee J 2007 *Proceedings of the 2007 International Workshop on Software Engineering for Adaptive and Self-Managing Systems* SEAMS '07 (Washington, DC, USA: IEEE Computer Society) pp 9– ISBN 0-7695-2973-9 URL `http://dx.doi.org/10.1109/SEAMS.2007.19`

[13] Solomon B, Ionescu D, Litoiu M and Iszlai G 2010 *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* SEAMS '10 (New York, NY, USA: ACM) pp 94–103 ISBN 978-1-60558-971-8 URL `http://doi.acm.org/10.1145/1808984.1808995`

[14] Karsai G, Ledeczi A, Sztipanovits J, Peceli G, Simon G and Kovacshazy T 2001 *In 2 nd International Workshop in Self-adaptive software, (IWSAS-01), Robert Laddaga, Howard Shrobe, and Paul*

[15] Poirot P E, Nogiec J and Ren S 2008 *Nuclear Science, IEEE Transactions on* **55** 284 –289 ISSN 0018-9499

[16] Chan H and Chieu T C 2003 *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* OOPSLA '03 (New York, NY, USA: ACM) pp 312–313 ISBN 1-58113-751-6 URL `http://doi.acm.org/10.1145/949344.949428`

[17] Sadjadi S M, McKinley P K and Cheng B H C 2005 *SIGSOFT Softw. Eng. Notes* **30**(4) 1–7 ISSN 0163-5948 URL `http://doi.acm.org/10.1145/1082983.1083086`

[18] Trencansky I, Cervenka R and Greenwood D 2006 *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications* OOPSLA '06 (New York, NY, USA: ACM) pp 521–529 ISBN 1-59593-491-X URL `http://doi.acm.org/10.1145/1176617.1176626`

[19] Diao Y, Hellerstein J, Parekh S, Griffith R, Kaiser G and Phung D 2005 *Engineering of Computer-Based Systems, 2005. ECBS '05. 12th IEEE International Conference and Workshops on the* pp 441 – 448
[20] Stoilov T and Stoilova K 2010 *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies* CompSysTech '10 (New York, NY, USA: ACM) pp 305–310 ISBN 978-1-4503-0243-2 URL http://doi.acm.org/10.1145/1839379.1839433
[21] icinga team T Monitoring with icinga URL https://www.icinga.org/
[22] Fisch D, Janicke M, Kalkowski E and Sick B 2009 *Intelligent Agents, 2009. IA '09. IEEE Symposium on* pp 31 –38