# An Autonomic Context Management Model based on Machine Learning

Ionut Anghel, Tudor Cioara, Ioan Salomie and Mihaela Dinsoreanu
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
{Ionut.Anghel, Tudor.Cioara, Ioan.Salomie, Mihaela.Dinsoreanu}@cs.utcluj.ro

***Abstract*—In this paper we approach the context management problem by defining a self-healing algorithm that uses a policy-driven reinforcement learning mechanism to take run-time decisions. The situation calculus and information system theories are used to define and formalize self-healing concepts such as context situation entropy and equivalent context situations. The self-healing property is enforced by monitoring the system's execution environment to evaluate the degree of fulfilling the context policies for a context situation, and to determine the actions to be executed in order to keep the system in a consistent healthy state.**

***Keywords—self-healing; context aware; reinforcement learning.***

## I. Introduction and Related Work

Context awareness is a key challenge for pervasive computing and an important step towards delivering functionalities in a way that best matches user requests, digital resources availability and environment conditions [1]. The complex nature of today's context aware systems execution environment makes the management and adaptation processes extremely difficult tasks. The self-management capabilities have become a mandatory requirement for creating systems that automatically change their observable behavior and structure according to the internal or external conditions of their environment. Self-management capabilities can be achieved by using the self-healing autonomic computing paradigm for the development and integration of self-* enhanced components into context aware systems [2]. A self-healing component is capable of detecting the run-time execution errors and deciding / taking necessary actions to bring itself back to a healthy state [3].

Self-healing related research focuses on developing models and techniques that would allow the system components to discover / recover from failures. A policy-based self-healing model that can be used to manage complex systems in an autonomic manner is described in [4]. The proposed model constructs an on-line graph by mapping the managed system states onto graph nodes and the actions onto graph edges. Self-healing techniques that use reinforcement learning in dealing with crashes and denial of service have been studied in [5]. The authors of [6] present a bio-inspired self-healing system based on the human cells healing mechanism. The system creates redundant components that may substitute the failed ones. In [7], a failure diagnosis solution based on decision trees is described. The system prioritizes fault sources that have a major impact by ignoring the ones that generate a small amount of errors.

The main disadvantage of the above presented solutions is the lack of generality which makes them hard to adapt and apply for the context management problem. In this paper we propose a generic self-healing algorithm that can be used to automatically detect, diagnose, tolerate and / or repair the failures which may appear during the context adaptation processes. By mapping the situation calculus theory onto context aware systems we define the self-healing property as the fulfilling degree of the predefined policy set which describe the systems correct behavior and drives the system execution, accordingly. Using set and information system theories, we define and formalize the concepts of context situation entropy and equivalent context situations. Based on these concepts, the self-healing property of a context aware system is enforced by monitoring the system execution environment to: (i) evaluate the degree of fulfilling the context policies for the current context situation (context situation entropy), (ii) obtain the equivalent context situations and (iii) select the actions that should be executed in order to keep the system in healthy states. The action selection phase has four main steps: (1) select the context policy with the highest contribution to context entropy, (2) for the selected policy, determine the set of affected context resources, (3) organize these resources into inter-independent resource groups (IIRGs) and (4) use reinforcement learning (if necessary) to generate the best action sequence to be taken in order to bring the IIRG's context resources as close as possible to a policy compliant state. By exploring all possible system's states using a reinforcement learning technique, our algorithm determines the action plan that brings the system as close as possible to a healthy state.

## II. The Self-Healing Algorithm

In order to define and formalize the self-healing property of a context aware system we use the *situation calculus theory* [8]. We have identified three main situation calculus concepts that can be used to define the self-healing property: (1) the *situation* concept, representing the complete state of the context aware system execution environment in a moment of time, (2) a *set of conditions* that guide and control the system execution and (3) the *action plans* executed by the context aware system in order to enforce the set of conditions for a specific situation. A context situation represents the system internal state together with a snapshot of its execution

978-0-7695-4324-6/10 $26.00 © 2010 IEEE
DOI 10.1109/SYNASC.2010.37

environment taken in a specific moment of time. Goal based context policies are used to represent the set of conditions used to control the context aware system execution [9]. Using the above detailed concepts, we define the self-healing property of a context aware system as a function, which associates to each context situation an action plan that should be executed in order to enforce the set of context policies. The self-healing function is non-injective implying that for a given policy, there are distinct context situations for which the same action plan should be executed to enforce the policy. As a result, the context situations are grouped in clusters discriminated by the action plan.

Fig. 1 presents the proposed self-healing algorithm. The algorithm has three main phases: (1) evaluating the context entropy for the current context situation (line 9), (2) obtaining the equivalent context situations (lines 13-14) and (3) determining the action plans that should be executed in order to keep the system in healthy functional states (lines 19-21). Next sections detail each phase of the self-healing algorithm.

```
1  input: P - the set of context policies
2         s - the current context situation
3         S^A - the set of existing equvalence classes
4         T_E - the entropy threashold
5  output: a healthy context aware system state
6  procedure selfHealingAlgorithm(P, s, S^A, T_E)
7  begin
8    // evaluate the context entropy
9    E_S = ∑ pw_i ∑ ri_ij * v_ij
10   if (E_S < T_E) then
11     return   // do nothing
12   else
13     s^a = determineEquivClass(P, s, S^A)
14     if (s^a in S^A)
15       // action plan already defined
16       executeAction(a)
17     else
18       // generate and execute action plan
19       a' = determineActionsSequence(P, E_S, T_E)
20       s^a = createNewEquivClass(P, s, a')
21       S^A = S^A + s^a
22       executeAction(a')
23     end if
24   end if
25 end
```

Fig. 1. The self-healing algorithm.

## III. THE CONTEXT ENTROPY EVALUATION

The system context situation entropy measures the level of system's self and execution environment disorder by evaluating the degree of fulfilling the context policies. If the evaluated system context situation entropy is below a predefined threshold $T_E$, then all the policies are fulfilled and adaptation is not required. Otherwise the system must execute adaptation actions to control its self and execution environment and to keep the system entropy below $T_E$.

The entropy for a context situation is computed with the following formula:

$$E_S = \sum pw_i \sum rw_{ij} * v_{ij} \quad (1)$$

where:

- $pw_i$ is the weight of the policy i, and represents the importance of the policy in the context.
- $rw_{ij}$ is the weight of the system resource i in the policy j; the resource weight reflects the system resource importance for the policy; if a policy imposes no restrictions to a resource, then the weight of the resource for that policy is zero.
- $v_{ij}$ is the deviation between the system resource j recorded value and the policy i accepted value.

We also define the entropy contribution ($E_i$) of a policy as:

$$E_i = pw_i \sum ri_{ij} * v_{ij} \quad (2)$$

Taking into account the system toleration to changes, we have defined two types of entropy thresholds: *restrictive threshold* and *relaxed threshold*. In the first case, we define the threshold $T_E = 0$ as lowest entropy value. This means that, whenever a policy-imposed restriction is broken, the self-healing algorithm is triggered. In the second case, for each policy we define an accepted entropy contribution value $E_i$ and compute current threshold $T_E$ as in (3). For the relaxed threshold, broken policies are tolerated if their entropy contribution is lower than the accepted value.

$$T_E = \sum pw_i * v_{ij} * (100 + E_i \% 100) \quad (3)$$

## IV. THE EQUIVALENT CONTEXT SITUATIONS

According to the *set theory* [10], an equivalence class is a subset of elements with the same collection of properties. The collection of properties that must be fulfilled by all elements of an equivalence class defines an equivalence relation. For a context aware system, we define the equivalence relation over the context situations set using the property of fulfilling and failing context policies. Two context situations are equivalent if both fulfill and fail the same context policies.

To clasify the context situations in equivalence classes we use the *information system theory* [11]. An information system is defined as an association (W, Atr) where: W is a non-empty finite set of objects and Atr is a non-empty finite set of attributes. For this association a function $a_f$ that assigns to every object in W, a value for each attribute in Atr, can be defined. For a context aware system, the non-empty set of finite objects W is mapped to the set of context situations S while the set of attributes Atr is mapped to the set of context policies P. We associate the $a_f$ function to the $Fs_k$ function that assigns to a context situation $s_k$, a list of values returned by evaluating all context policies. Two context situations $s_1$ and $s_2$ belong to the same equivalence class if and only if $Fs_1$ and $Fs_2$ generate the same lists of policy evaluation values (Fig. 2).

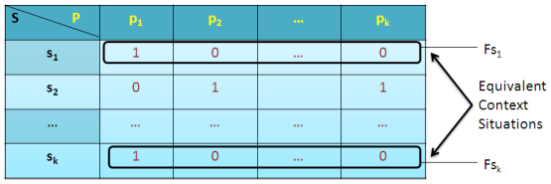


Fig. 2. Determining the context situation equivalence classes.

As a result, the equivalence class is defined as the set of context situations for which the same context policies are broken and the same action plan is executed in order to keep the system in a healthy state.

## V. The Action Selection Phase

The action selection phase of the self-healing algorithm considers the broken context policies one by one, ordered by their contribution to the overall context entropy (see (1)).

The action selection phase can be divided in four main steps (see Fig. 3): (1) select the context policy with the highest contribution to context entropy (line 9), (2) for the selected context policy, determine the set of affected context resources and organize these resources into IIRGs (line 10) and (3) use reinforcement learning (if necessary) to generate the best action sequence to be taken in order to bring the IIRG's context resources as close as possible to a policy compliant state (lines 11-15).

```
1  input: P - the set of context policies
2         ES – the entropy value for the curent context situation
3         TE - the entropy threashold
4  output: actionsToBeTaken - the sequence of actions to be
           executed
5  procedure determineActionsSequence(P, ES, TE)
6  begin
7   ActionsList actionsToBeTaken;
8   while ES > TE do
9    policy = determineLargestContributionToEntropyPolicy ()
10   IIRGs = IIRGExtraction(policy)
11   while ( faultyResourcesNotTaken(policy) ) do
12    faultyResource = selectFaultyResource(policy)
13    IIRG = selectInterIndependentResourcesGroup(IIRGs)
14    seqActions = reinforcementLearning(faultyResource, ES,
                                         rootNode, rootNode)
15    add(actionsToBeTaken, seqActions)
16   end while
17  end while
18  return actionsToBeTaken
19 end
```

Fig. 3. The actions selection phase algorithm.

The IIRGs are defined and used into the reinforcement learning algorithm to avoid the situation when healing a context policy may cause failing another one because their associated context resources are depended. For example, a temperature sensor and a humidity sensor are dependent taking into consideration a room temperature policy and a human thermal comfort factor policy. In this situation, an action executed to heal a specific context policy (e.g. room temperature policy), changes the policy associated context resources values and triggers a variation of its depended context resources values with the result of failing another context policy (e.g. human thermal comfort factor policy).

If all IIRG's context resources comply to the context policy rules, no action is executed. In the next sub-sections we present the IIRG extraction technique and the reinforcement learning algorithm used to determine the action plan needed to be executed.

### A. Inter-Independent Resources Groups Extraction

A resource is dependent on another resource when its recorded value changes with the variation of the other resource value. We represent the dependency between a set of context resources as a directed weighted graph having the context resources as nodes, the dependency relation as edges and the dependency degree as edge weights. The dependency degree between two context resources is computed using a dependency function (Df) with the following output: (i) 0 if there is no path between i and j and (ii) 1/k where k is the length of the shortest path from i to j.

An IIRG is a set of inter-independent resources (i.e. $\forall R_i, R_j \in$ IIRG, $R_i$ and $R_j$ are inter-independent). To determine the *inter-independent resources* the dependency graph is represented as a matrix whose elements $d_{i,j}$ represent the dependency degree between $R_i$ and $R_j$. Two context resources $R_i$ and $R_j$ are *inter-independent* if the distance matrix contains zero values in positions (i,j) and (j,i). The IIRG extraction algorithm is presented in Fig. 4.

```
1  input: currentPolicy - context policy with highest
          contribution to entropy
2  output: IIRG list for the currentPolicy
3  procedure IIRGExtraction(currentPolicy)
4  begin
5   resourcesPriorityQueue =
                 buildQueueOnResourcesWeights(currentPolicy)
6   groupNumber = 0
7   while ( notEmpty(resourcesPriorityQueue) ) do
8    groupNumber = groupNumber + 1
9    push( iirg[groupNumber], pop (resourcesProprityQueue) )
10   interindependent = TRUE
11   while ( notEmpty (resourcesPriorityQueue) AND
                                    (interindependent) ) do
12    resource = popResource(resourcesPriorityQueue)
13    if ( IndependentOfResources(resource, iirg[groupNumber]) ) then
14     push( iirg[groupNumber], resource)
15    else
16     interindependent = FALSE
17    end if
18   end while
19  end while
20  return IIRGlist
21 end
```

Fig. 4. The IIRG extraction algorithm.

Fig. 5 shows the IIRG extraction algorithm tracing for a dependency graph with three context resources. Initially a priority queue according to the resources weight is build: ($R_2$, $R_3$, $R_1$).

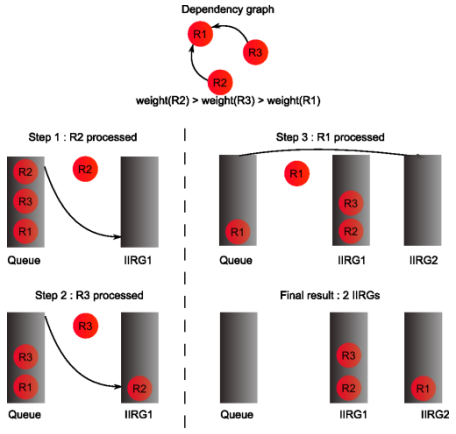


Fig. 5. The IIRG extraction algorithm tracing.

In the *first step*, $R_2$ is de-queued and placed in the first IIRG (IIRG1). In the *second step*, $R_3$ is de-queued. According to the dependency graph, $R_2$ is independent of $R_3$ and $R_3$ is independent of $R_2$ and therefore $R_3$ is added to IIRG1. In the *third step*, $R_1$ is de-queued. Because it is not inter-independent

with $R_2$ and $R_3$ (the resources from IIRG1), it is placed in a new IIRG (IIRG2).

### B. *Determining the Best Action Sequence using Reinforcement Learning*

The sequence of actions that need to be executed to bring the system in a healthy state is determined for each IIRG using a reinforcement learning algorithm (see Fig. 6). The learning process considers all possible context situations and builds a decision tree by simulating the execution of all available actions for a specific context situation through a reward / penalty approach. The sequence of actions that the system must take is represented by the minimum entropy path in the reinforcement learning decision tree.

```
1  input: faultyResource - faulty resource
2         rootNode - reinforcement learning state tree root node
3         currentNode - reinforcement learning state tree current node
4  output: Decision tree minimum path wich represents
           the sequence of actions
5  procedure reinforcementLearning(faultyResource, currentEntropy,
                                   rootNode, currentNode)
6  begin
7    Actions = getActions(resource)
8    for each ( action ∈ Actions) do
9      newEntropy = simulateActionExecution(faultyResource, action)
10     newNode = addChild(currentNode, newEntropy, resource)
11     if ( allCycles() = FALSE ) then
12       if (newEntropy < T_E ) then
13         return pathFromRootActionsList(currentNode)
14       else
15         if (newEntropy < currentEntropy) then
16           reinforcementLearning(faultyResource, newEntropy, root, newNode)
17         end if
18       end if
19     else
20       return minimumPathActionsList()
21     end if
22   end for
23 end
```

Fig. 6. The reinforcement learning algorithm.

The learning process may terminate by generating different results as shown in Fig. 7 and discussed below.

*Case 1*: The algorithm finds only a state (*SAxRx*) that has its specific entropy lower than the predefined threshold. In this case the sequence of actions (*AxRx*) that lead to this state is selected and the search is stopped.

*Case 2*: The current entropy is higher than the threshold, but smaller than the minimum entropy determined so far. The minimum entropy is replaced with the new entropy and the search process is continued.

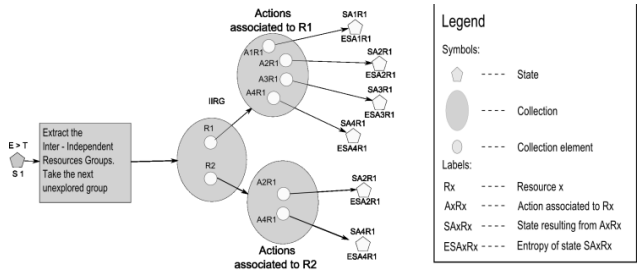


Fig. 7. The reinforcement learning process tracing.

*Case 3*: When the current entropy is higher than both the threshold and the minimum entropy, the reinforcement learning algorithm continues the search process. If all exercised paths of the decision tree are cycles, the algorithm stops and chooses the path leading to a state with the minimum entropy.

## VI. Conclusions

In this paper we propose a generic self-healing model and algorithm that can be used to automatically detect, diagnose, tolerate and / or repair the failures which may appear during the context adaptation processes. For further development, we intent to assess the self-healing algorithm reinforcement learning performance taking into consideration the execution time and the memory / processor load.

## Acknowledgment

The work has been done in the context of the EU FP7 GAMES project [12].

## References

[1] D. Fournier, S. Ben Mokhtar and N. Georgantas, "Towards Ad hoc Contextual Services for Pervasive Computing", *Proceedings of the 1st workshop on Middleware for Service Oriented Computing*, ISBN:1-59593-425-1, pp. 36 – 41, 2006.

[2] D. A. Menascé and J. O. Kephart, "Autonomic Computing", *IEEE Internet Computing*, vol. 11, no. 1, pp. 18-21, 2007.

[3] P. Koopman, "Elements of the Self-Healing System Problem Space", *Proceedings of ICSE WADS*, 2003.

[4] R. M. Bahati and M. A. Bauer, "Adapting to Run-time Changes in Policies Driving Autonomic Management", *Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems*, Pages 88-93, ISBN:978-0-7695-3093-2, 2008.

[5] M. Amoui, M. Salehie, S. Mirarab and L. Tahvildari, "Adaptive Action Selection in Autonomic Software Using Reinforcement Learning", *Proc. of the Fourth International Conference on Autonomic and Autonomous Systems*, pp. 175-181, 2008.

[6] T. Kemppainen, "Biology-inspired self-healing system design", University of Helsinki Department of Computer Science, Seminar on self-healing information systems, 2007.

[7] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan and E. Brewer, "Failure Diagnosis Using Decision Trees", *First Int. Conference on Autonomic Computing*, ISBN:0-7695-2114-2, 2004.

[8] F. Pirri and R. Reiter, "Some contributions to the metatheory of the Situation Calculus", *Journal of the ACM*, pp. 325–361, 1999.

[9] T. Cioara, I. Anghel, I. Salomie and M. Dinsoreanu, "A Policy-based Context Aware Self-Management Model". *Proc. of 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Romania, September 26-29, 2009.

[10] J. Thomas, "Set Theory: Third Millennium Edition", *Springer Monographs in Mathematics*, Springer-Verlag, ISBN 978-3-540-44085-7, p. 642, 2003.

[11] R.D. Galliers, M. Markus and S. Newell, "Exploring Information Systems Research Approaches". NY: Routledge, ISBN-10 041577196X , 2007.

[12] GAMES Research Project, http://www.green-datacenters.eu/