Enhancing Reliability in Organic Computing Using Hormone Guard

Sabikun Nahar, Utkarsh Raj, Simon Meckel, Roman Obermaisser
University of Siegen, Germany

***Abstract*—The bio-inspired concept of Organic Computing applies biological concepts to technical systems. Organic Computing utilizes an Artificial Hormone System (AHS) as a decentralized mechanism, i.e., a middleware that continuously monitors and organizes task allocations to computing nodes in distributed real-time embedded systems. By introducing different types of artificial hormones for the tasks, such as favoring and constraining hormones, appropriate task assignments are continuously realized by maintaining hormone balances through distributed closed-loop control. This process handles the increasing complexity of e.g., distributed control systems by enabling self-configuration, -adaptation, -improvement and -healing. However, faults in the artificial hormone system, for instance due to incorrect hormone values, can result in adverse and critical system behavior and even complete system failure. Therefore, this paper presents a fault recovery solution, namely the so-called hormone guard. The hormone guard plays a pivotal role within the AHS, serving as a safeguard to ensure that computation nodes do not receive incorrect hormone values. This paper presents the concept and implementation of the hormone guard and evaluates its performance and contribution to the overall system stability and reliability.**

Keywords*—*Organic Computing, Artificial Hormone System, fault, fault recovery, hormone guard.

I. Introduction

In autonomous vehicles with complex distributed systems, the reliability and availability of the system services are of utmost importance. One key aspect of the reliability is the ability to protect the system from errors or unwanted data. Given the aforementioned complexity, an adaptive system is needed that can adapt dynamically to emerging scenarios and initiate appropriate countermeasures. In recent years, Organic Computing has gained significant attention by developing such dynamic self-organizing and self-optimizing properties of biological DNA and hormone systems [14]. Previous research has developed an Artificial Hormone System (AHS) as a middleware to cope with the complexity of real-time (re-)assigning tasks of an application to distributed processing elements (PEs; also referred to as computation nodes), e.g., [9]. The AHS uses hormone-based distributed control loops for task assignments. The respective publications [3], [4] demonstrate the self-organizing properties of the AHS in terms of stability and real-time capability. Furthermore, it exhibits reliable behavior against system failures through the use of self-X properties. The concept of Organic Computing is essentially based on these self-X properties, such as self-configuration, self-optimization, and self-healing. To illustrate, in the event of a processing element (PE) crashing during system runtime, the AHS reassigns all tasks of that crashed PE, thus preventing system failure. However, a faulty PE may affect the overall task allocations by transmitting (inadvertently) manipulated hormones. Such manipulated hormones, if processed by the hormone loop of each PE, can lead to faulty system behavior, e.g., tasks being allocated too often or tasks being missing. This was demonstrated in [8] using a comprehensive fault injection framework, which simulated the occurrence of various fault cases resulting from hormone manipulations.

Given the limitations of current AHS capabilities, this paper presents a hormone guard that prevents computation nodes from processing incorrect hormone values. The concept of the hormone guard is introduced as a subroutine of *Local Diagnosis*, presented in [11], along with its implementation and the benefits to the reliability of Organic Computing-based systems.

The paper is structured as follows: After outlining the motivations of this paper in this section, Section II presents the related work. Section III presents the system model and the concept of hormone guard. In Sections IV all hormone-based faults are discussed in detail. In Section V the fault recovery strategies are presented, which are subsequently evaluated in Section VI. The paper concludes with a conclusion and future work in Section VII.

II. Related Work

Organic Computing systems aim to create adaptive and robust networks of computing devices that can autonomously adapt to changes in their environment. The concept of introducing a hormone guard draws inspiration from the widely known concept of the bus guardian. In communication systems a bus guardian is a device that aims to safeguard a communication bus by preventing any potential failures, e.g., babbling idiots, in its interconnected components. A common practice for a bus guardian is to either isolate a node from the bus or initiate a shutdown procedure for a node. These actions aim to prevent any interference caused by a faulty node and maintain the correct operation of the other nodes in the system. The paper [5] presents a guardian-based approach for detecting and overcoming babbling idiot failures, where the available bandwidth on the bus is being monopolized. By promptly activating a fail-silent mode in this case, a fair distribution of bandwidth is ensured and any detrimental effect on the overall performance of the system is prevented. Another publication [13] presents an approach using a special

device, the bus guardian, which is added to each node to protect the communication bus from babbling-idiot failures. During the design phase of the system, the bus guardian is equipped with prior knowledge of the temporal access pattern to ensure fault independence. In another research paper [7], the authors focus on self-configuration in Organic Computing systems, particularly in enhancing the assignment of important services to trustworthy nodes. The algorithm uses trust as a key factor in decision-making, aiming to achieve a well-balanced utilization of network resources. The experimental results highlight improved availability of important services with the incorporation of trust, and the scalability of the proposed self-configuration algorithm for large-scale distributed systems is emphasized. In contrast, our approach emphasizes fault recovery in the self-organizing AHS of Organic Computing architecture. The research demonstrates the effectiveness of this approach in managing diverse fault scenarios, with a specific emphasis on suppressor hormone faults impacting task allocation. Another paper [6] presents various ways to attack a self-organizing AHS, thereby impacting its operational behavior. The authors introduce detection and defense strategies aimed at countering attacks using suppressor hormones and evaluate them. Whereas in our approach, we have focused on the impact of suppressor hormone faults and introduced a specific hormone guard mechanism for fault recovery, showcasing its effectiveness in maintaining task allocation under diverse fault scenarios. In [12], a novel algorithm for detecting mutual influences in the configuration of self-adaptive systems is presented, with a particular focus on systems using local performance measurements for self-configuration. The algorithm uses established techniques, such as the maximum information coefficient, and is demonstrated in the context of smart camera surveillance systems. In contrast, our paper introduces a new approach to fault recovery in the self-organizing AHS of the Organic Computing architecture. Both papers contribute to making the complex system more reliable and stable.

In conclusion, our approach to fault recovery in the self-organizing AHS of Organic Computing architecture stands out for its unique perspective. No other research directly align with our specific focus on suppressor hormone faults and the introduction of a hormone guard mechanism.

III. System Model and Hormone Guard

The architecture of a distributed system that exhibits self-organizing and self-optimizing capabilities at runtime, based on Organic Computing, is shown in Figure 1. The architecture is explained in detail below.

A. Physical Model

The physical model describes the physical resources of the system, i.e., the distributed processing elements (PEs), which can be heterogeneous, as well as the communication links between them. The heterogeneity of PEs is accounted for by abstracting from physical components and specifying for each PE which tasks can be performed, specifically, how suitable the PE is to perform a particular task (see Section III-C). In this paper we assume that there are m PEs (with $m \in \mathbb{N}$) and that the exchange of messages can be bidirectional, as shown in Figure 1.

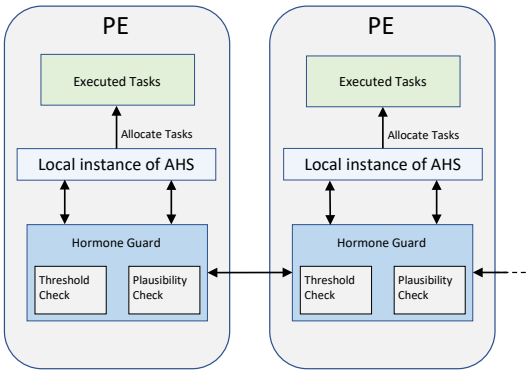

Figure 1. System architecture with hormone guard

B. Logical Model

The logical model describes the system service to be provided (i.e., the application) in terms of a directed acyclic graph, where the vertices of the graph correspond to tasks and the directed edges describe the dependencies between tasks in terms of messages. All tasks $T_1, \ldots, T_n$ (with $n \in \mathbb{N}$) that make up the system service are scheduled on the PEs of the system, where each PE, depending on its capabilities, can execute only certain tasks, i.e., a subset of the set of all tasks. This is indicated in Figure 1 by the green box.

C. Artificial Hormone System

The AHS is designed to assign the tasks of the application to the PEs of the distributed system in a decentralized self-organizing way. For this, each PE runs a local instance of AHS middleware which uses three types of artificial hormones to decide on task assignments, as introduced in [1].

- **Eager hormones** (also called eager values) express the suitability of a PE for executing a particular task, thereby abstracting from physical components (such as processor type or memory). For each task of the logical model (i.e., the application), a predefined local eager value is encoded in the PE, indicating its general suitability. To express the momentary suitability, which depends, for example, on the current task load of the PE and the current task assignments of the other PEs, the (local) eager value is converted into a so-called *modified* eager value. In general, the higher the eager value, the better the suitability of the PE for the task.
- **Suppressor hormones** lower the momentary suitability of a PE to take a certain task during the task allocation process. The main purpose of the (global task) suppressors is to indicate a successful task allocation to other PEs to prevent duplicate task allocations. There are several types of suppressors, some of which are sent (as messages) to other PEs in the system, while others

are used only locally at a PE, e.g., local monitoring suppressors to indicate a deteriorating PE condition, and load suppressors to indicate the current load on a PE from the tasks being executed.

- **Accelerator hormones** can optionally be used to increase the suitability of task execution on a PE. There are also different types of accelerators, such as organ accelerators and monitoring accelerators.

Each PE encodes PE-specific information such as maximum or minimum initial local eager values, suppressors and accelerators for each task that can potentially be executed in a configuration file. Task assignment is based on hormone flow. Each PE, more specifically the local instance of the AHS on the PE, periodically (and independently) executes a hormone-based control loop [1]. Figure 2 illustrates this hormone loop based on [10], which is a formal specification of the hormone loop of an artificial hormone system. This loop has three stages.

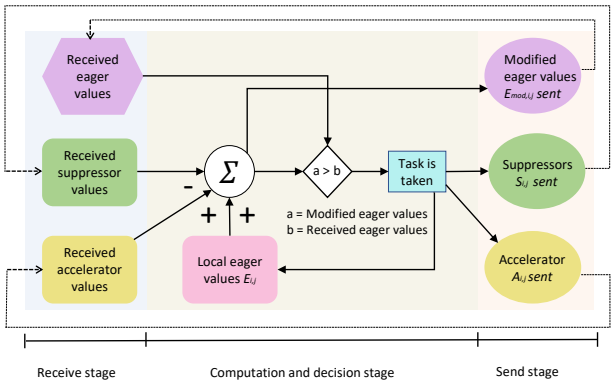

Figure 2. Hormone-based control loop based on [10]

1) **Receive stage:** In this stage, a PE receives suppressors, accelerators, and modified eager values for each task from all other PEs in the system.
2) **Compute and decision stage:** Using the received suppressors (denoted as S) and accelerators (denoted as A) from point 1, each PE computes its own modified eager value E_{mod} for each task according to Equation 1.

$$E_{\text{mod},i,j} = E_{i,j} - \sum S_{i,j} + \sum A_{i,j} \qquad (1)$$

where the subscripts i and j (with $i, j \in \mathbb{N}$) represent task indices and PE indices, respectively. The sum sign over S and A means that the various suppressors or accelerators are taken into account. The local static eager value $E_{i,j}$ indicates the predefined suitability of PE$_j$ for executing task T_i. $E_{\text{mod},i,j}$ is the modified eager value of task T_i on PE$_j$. According to the Equation 1, received suppressors lower the momentary suitability for the respective task to be taken and the accelerators increase the the momentary suitability, respectively. To decide on whether to allocate a task, the computed $E_{\text{mod},i,j}$ is compared with the received modified eager values from all other PEs ($a > b$ in Figure 2). In case of equality, another decision criterion must be used that is known to all PEs, e.g., the unique PE identifier (ID).
3) **Send stage:** The computed modified eager value $E_{\text{mod},i,j}$ of each task from point 2 is then broadcast to the other PEs in the send stage. Once a PE takes a particular task T_i, it distributes suppressors to all PEs, thus preventing other PEs from also executing T_i, i.e., according to Equation 1, a (received) suppressor lowers the suitability of a PE. The ratio of an eager value to a suppressor value for a given task determines whether the task is executed once or multiple times in the system.

The AHS approach is fully decentralized and provides several self-X properties such as self-configuration in terms of initial task allocation by exchanging hormones, self-optimization through reassignment in case of better suitability of a (new) PE during regular task offerings, and self-healing by reassigning tasks to remaining resources in case of PE failures. Upon completion of the self-configuration phase, the AHS enters a steady state phase and remains there until the need for self-optimization or self-healing arises. Due to their importance for the hormone guard, we will highlight the two phases below.

1) Self-configuration Phase: After system startup, in the so-called self-configuration phase, the AHS performs the initial task assignment and resource allocation according to the configuration parameters of each PE, e.g., possible executable tasks, load limits. Once all modified eager values become zero, meaning that all tasks are executed, the self-configuration is complete.

2) Steady Phase: In various systems and processes, the *steady phase* denotes a phase of balance or equilibrium, wherein the general system remains stable over an extended period of time. In the AHS, after the initial configuration phase, the system reaches a steady phase, which is characterized by balance and stability. During this period, assigned tasks remain in their suitable nodes until one of two specific conditions is met. Firstly, if the periodic so-called offer phase is initiated, tasks may be redistributed or reallocated based on various factors such as node availability, workload distribution or priority settings. Secondly, if a fault occurs in the system, a reevaluation of task allocation is triggered to ensure fault tolerance and efficient resource utilization. Throughout the steady phase, the AHS ensures efficient processing and management of tasks to achieve optimal performance and resilience against potential system disruptions.

D. Hormone Guard

The hormone guard is a mechanism designed to verify the correctness of hormone values against thresholds and to block any incorrect or implausible values from further use or processing. Any value outside the threshold is considered implausible and should be blocked. This mechanism is implemented in every PE. The hormone guard exhibits its capabilities through two different phases: the self-configuration

phase (i.e., start-up phase) and the steady phase (i.e., normal operation phase).

In this paper, the hormone guard is exemplarily evaluated in the presence of a higher-than-intended suppressor hormone fault case, as this particular fault exhibits different consequences in the self-configuration and steady phases. In the self-configuration phase, the occurrence of this fault could lead to the unavailability of tasks within the system. Conversely, in the steady phase, the tasks remain but are subject to an excessively high suppressor value. Both phases are thus necessary for the effective operation of the hormone guard.

IV. Hormone-Based Faults

Our publication [8] presents a hormone fault model for ADNA-based Organic Computing and highlights its impact on task distribution. Hormone faults cause imbalances in hormone levels, resulting in missing or duplicated tasks, potentially leading to catastrophic failures. The fault model primarily addresses suppressor hormone faults. Suppressors temporarily lower the suitability of task allocation on PEs (see the equation under point 2 in Section III-C), so faults in terms of a too-high or too-low suppressor value can lead to missing or duplicated tasks, potentially resulting in system service failure. Table I recalls suppressor hormone-based faults that may occur in the AHS and their impacts on the system as introduced in [8].

Table I
Suppressor hormone fault cases and their impacts

Fault cases	Impacts on the system
1. Missing suppressor	Duplicate task instantiation
2. Higher-than-intended suppressor	Missing tasks in the system
3. Lower-than-intended suppressor	Duplicate task instantiation

V. Fault Recovery Strategy

This section explains the recovery strategy that counteracts higher-than-intended suppressor hormone fault, with a particular emphasis on the hormone guard mechanism. The hormone guard serves as a crucial component in verifying the correctness of hormone values and blocking any incorrect or implausible values from further processing by performing threshold checks and plausibility analysis. By implementing the hormone guard at every PE, we enhance the system's ability to detect and mitigate faults, thereby improving overall system resilience.

Threshold checks verify that suppressor hormone levels stay within predefined ranges, as specified in a shared configuration file used by each PE. These checks apply to all hormone types, with thresholds set at application design time and remaining static during run-time. The implementation is detailed in Algorithm 1.

A. Threshold Checks

Algorithm 1 Threshold Check Algorithm

```
1: Input: suppressor hormone, threshold range
2: Output: suppressor hormone
3: function ThresholdCheck(supp_hormone, range)
4:     if supp_hormone < range.min
5:      or supp_hormone > range.max then
6:         supp_hormone.SetValue(0)
7:     end if
8:     return supp_hormone
9: end function
```

B. Plausibility Checks

Algorithm 2 Plausibility Analysis for Suppressor Hormones

```
1: Input: suppressor hormone, MAX_TASKS
2: Output: suppressor hormone
3: Initialize a map count to store hormone counts per sender
   and timestamp
4: function PlausibilityCheck(supp_hormone)
5:     senderPE ← supp_hormone.GetSender()
6:     value ← supp_hormone.GetValue()
7:     if value > 0 then
8:         count[senderPE, timestamp] += 1
9:     end if
10:    if count > MAX_TASKS then
11:        ClearHormones(senderPE, timestamp)
12:        supp_hormone.SetValue(0)
13:    end if
14:    return supp_hormone
15: end function
16: function ClearHormones(senderPE, timestamp)
17:    for all supp_hormone in receivedsuppressor do
18:        if supp_hormone.GetSender() == senderPE and
19:        supp_hormone.GetTimestamp() == timestamp then
20:            supp_hormone.SetValue(0)
21:        end if
22:    end for
23: end function
```

Faults can occur when suppressor hormone levels are within the intended range but are incorrect, for example when a PE sends suppressors for tasks it has not taken on.

The hormone guard maintains a list of positive suppressors received from all other PEs in a given hormone cycle. When this number exceeds the maximum number of tasks allowed for a PE, the hormone guard considers the sending PE to be faulty and disregards all its hormones, including those received earlier in the hormone cycle. This ensures that faulty PEs cannot hijack the entire distributed system, as non-faulty PEs learn to ignore hormones from the faulty PE, allowing all application tasks to be available.

The behavior of the hormone guard depends on the phase of the AHS, i.e., the configuration phase and the steady phase. During the configuration phase, the hormone guard

plays a critical role in ensuring the accuracy and validity of hormone information within the system. It employs a series of conditional checks. Algorithm 2 outlines such a plausibility check for suppressor hormones. When a positive suppressor is received, the hormone guard checks whether the sending PE has sent the highest eager value during the previous hormone cycle, as the AHS ensures that tasks can only be taken by PEs with the highest respective eager values. If this condition is not satisfied, the received suppressor is discarded.

Through these checks and actions, the hormone guard ensures that suppressor values are valid and consistent. This, in turn, allows the system to make accurate and informed decisions when allocating tasks.

VI. Evaluation

The evaluations use an AHS simulator, which provides a valuable means of assessing the effectiveness of the proposed fault recovery strategy by allowing different test scenarios to be examined. A detailed description of the original AHS simulator is presented in [2]. The experiments use a benchmark configuration of 8 PEs and 16 tasks. The hormone values provided in the system configuration (e.g., initial static eager values, task suppressor values) are chosen to limit the number of tasks a PE can run at the same time to prevent task concentration in a single PE. This is achieved through the use of load suppressor hormones, which reduce the PE's modified eager values according to Equation 1.

The following figures show the evaluation of the AHS in its normal, i.e., fault-free phase, the AHS when affected by hormone faults and the recovery from these faults by the hormone guard.

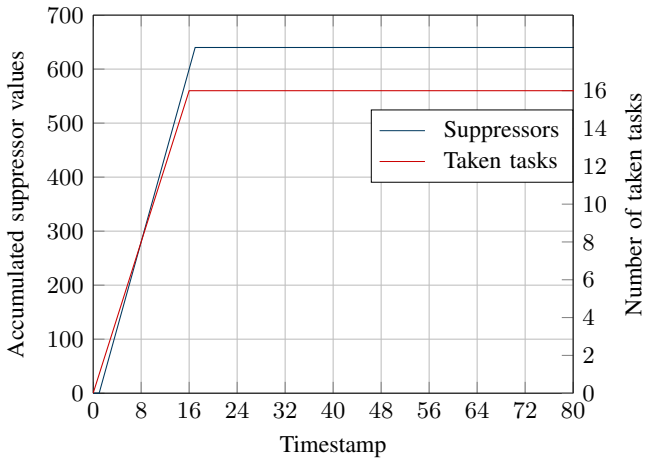

Figure 3. Accumulated suppressor hormone values without fault

Figure 3 shows an evaluation of the unaffected AHS. As in all subsequent plots, the blue line refers to the left y-axis, and the red line refers to the right y-axis. The two curves illustrate the relationship between the number of tasks assigned and the accumulated suppressor values in the system, i.e., the more tasks get assigned to processing elements, the higher the accumulated value of the suppressors until a steady state is reached at timestamp 16 after all the tasks have been assigned according to the simulation scenario.

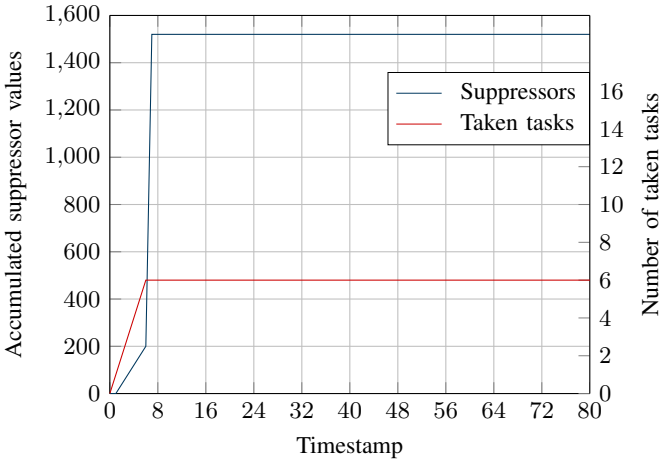

Figure 4. Accumulated suppressor hormone values during configuration phase fault injection

Figure 4 visualizes the effect of the higher-than-intended suppressor hormone fault during the configuration phase [8]. The way, the x and y axes are depicted in this plot closely resembles the arrangement in Figure 3. This figure highlights a situation where a single processing element initiates an attack by flooding all tasks across all processing elements in the system with exceptionally high suppressors. The blue line illustrates the impact of this fault on the hormone system. The accumulated suppressor level first increases identical to Figure 3 between timestamps 1 and 6. It then jumps by a factor of almost 8 due to the hormones introduced by the faulty processing element. As a result of the high suppressor values, the AHS fails to allocate any further tasks.

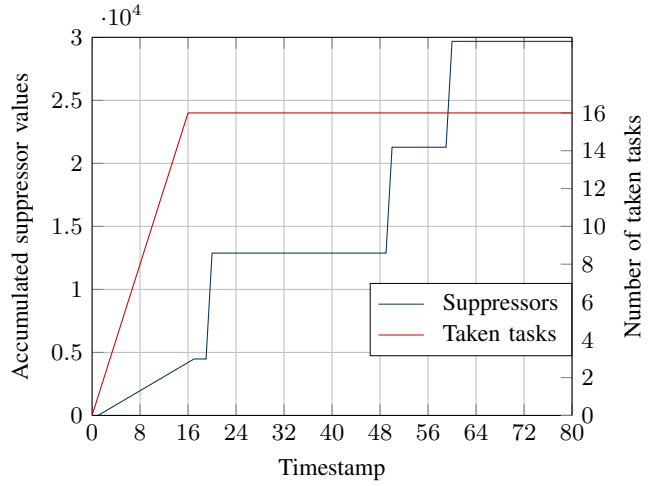

Figure 5. Accumulated suppressor hormone values during steady phase fault injection

Figure 5 illustrates the impact of a suppressor hormone fault exceeding the intended levels during a steady phase.

The scenario involves multiple processing elements initiating attacks at various timestamps, flooding all tasks in the system with exceptionally high suppressor values. Initially, during the configuration phase, the suppressor values steadily accumulated while all 16 tasks got allocated. Subsequently, faults were injected during the steady phase: the first occurred at timestamp 20, leading to a notable rise in accumulated suppressor values. Similarly, faults at timestamps 50 and 60 also result in increased suppressor values. One observes that these fault injections did not disrupt the task allocation, although they did affect the accumulated suppressor values.

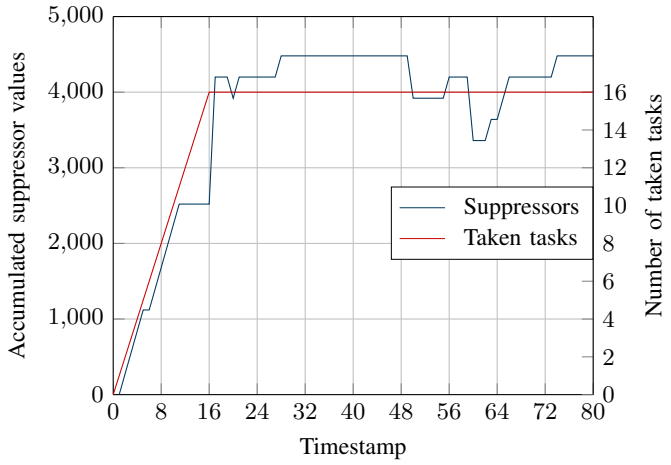

Figure 6. Accumulated suppressor hormone values with hormone guard

Figure 6 illustrates the system's recovery from a suppressor fault that exceeded the intended levels during both the configuration and steady phases, achieved through the implementation of hormone guard. The figure shows fluctuating patterns in the accumulated suppressor hormone values. The peaks signify fault injections, while the dips indicate that the hormone guard mechanism effectively mitigates the fault. Comparing with Figure 4 and Figure 5, it is demonstrated that the hormone guard is able to suppress the higher hormone values and ensures the system stability throughout the process.

VII. Conclusion and Future Work

This paper introduces a new approach for fault recovery in the self-organizing AHS of Organic Computing architecture. The research outcomes demonstrate the effectiveness of this approach in managing diverse fault scenarios within a system. The results presented in this paper provided valuable insights into the impact of suppressor hormone faults on task allocation. We observed that a single, faulty processing element could disrupt the system by flooding tasks with too-high suppressor values, rendering the system incapable of further task allocations. Furthermore, the successful implementation of the hormone guard mechanism effectively mitigates suppressor faults during both configuration and steady phases. Despite multiple faults occurring at different times, the task allocation was not disrupted, showcasing the robustness of our approach. Due to space constraints in this paper, we can only provide a small part of the recovery strategy and evaluation. We can extend these strategies to recover from other fault types outlined in the fault model.

In conclusion, our research not only introduced an innovative method for fault recovery but also demonstrated its applicability and effectiveness in different scenarios. The hormone-based approach presented in this study offers a promising avenue for enhancing the reliability and stability of complex systems, providing a foundation for further research and implementation in diverse fields.

Acknowledgment

This work was supported by the research project Self-AutoDOC funded by the Bundesministerium für Wirtschaft und Klimaschutz (BMWK) under grant number 19A21044C.

Gefördert durch:

aufgrund eines Beschlusses
des Deutschen Bundestages

References

[1] Uwe Brinkschulte and Mathias Pacher. An agressive strategy for an artificial hormone system to minimize the task allocation time. In *2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*. IEEE, 2012.

[2] Uwe Brinkschulte, Mathias Pacher, and Benjamin Betting. A simulator to validate the concept of artificial dna for self-building embedded systems. In *2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 160–169. IEEE, 2014.

[3] Uwe Brinkschulte, Mathias Pacher, and Alexander von Renteln. Towards an artificial hormone system for self-organizing real-time task allocation. In *IFIP International Workshop on Software Technologies for Embedded and Ubiquitous Systems*, pages 339–347. Springer, 2007.

[4] Uwe Brinkschulte, Mathias Pacher, and Alexander von Renteln. An artificial hormone system for self-organizing real-time task allocation in organic middleware. In *Organic Computing*, pages 261–283. Springer, 2009.

[5] Ian Broster and Alan Burns. An analysable bus-guardian for event-triggered communication. In *RTSS 2003. 24th IEEE Real-Time Systems Symposium, 2003*, pages 410–419. IEEE, 2003.

[6] Christoph Leineweber, Mathias Pacher, Benjamin Betting, Julius von Rosen, Uwe Brinkschulte, and Lars Hedrich. Detection and defense strategies against attacks on an artificial hormone system running on a mixed signal chip. In *2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, pages 135–143. IEEE, 2012.

[7] Nizar Msadek, Rolf Kiefhaber, and Theo Ungerer. A trustworthy, fault-tolerant and scalable self-configuration algorithm for organic computing systems. *Journal of Systems Architecture*, 61(10):511–519, 2015.

[8] Sabikun Nahar, Simon Meckel, and Roman Obermaisser. Fault injection framework for organic computing architecture. In *Proceedings of SAI Intelligent Systems Conference*, pages 701–717. Springer, 2023.

[9] Mathias Pacher and Uwe Brinkschulte. Implementation and evaluation of a self-organizing artificial hormone system to assign time-dependent tasks. *Concurrency and Computation: Practice and Experience*, 24(16):1879–1902, 2012.

[10] Mathias Pacher and Uwe Brinkschulte. A formal specification of the hormone loop of an artificial hormone system. In *ESOS*, 2013.

[11] Utkarsh Raj, Simon Meckel, Aleksey Koschowoj, Mathias Pacher, Roman Obermaisser, and Uwe Brinkschulte. Self-adaptive diagnosis and reconfiguration in adna-based organic computing. In *Architecture of Computing Systems - 36th International Conference, ARCS 2023, Athens, Greece, June 13-15, 2023, Proceedings*, volume 13949 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2023.

[12] Stefan Rudolph, Sven Tomforde, Bernhard Sick, and Jörg Hähner. A mutual influence detection algorithm for systems with local performance measurement. In *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 144–149. IEEE, 2015.

[13] Christopher Temple. Avoiding the babbling-idiot failure in a time-triggered communication system. In *Digest of Papers. Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing (Cat. No. 98CB36224)*, pages 218–227. IEEE, 1998.

[14] Sven Tomforde and Bernhard Sick. *Organic Computing: Doctoral Dissertation Colloquium 2018*, volume 13. kassel university press GmbH, 2019.