# A Survey on Different Mechanisms to Classify Agent Behavior in a Trust Based Organic Computing Systems

Shabhrish Reddy Uddeahal
Department of Intelligent Systems
*University of Passau*
Passau, Germany
uddeha01@gw.uni-passau.de

***Abstract*—Organic Computing (OC) systems vary from traditional software systems, as these systems are composed of a large number of highly interconnected and distributed subsystems. In systems like this, it is not possible to predict all possible system configurations and to plan an adequate system behavior entirely at design time. An open/decentralized desktop grid is one example, Trust mechanisms are applied on agents that show the following Self-X properties (Self-organization, Self-healing, Self-organization and so on). In this article, some mechanisms that could help in the classification of agents behavior at run time in trust-based organic computing systems are illustrated. In doing so, isolation of agents that reduce the overall systems performance is possible. Trust concept can be used on agents and then the agents will know if their interacting agents belong to the same trust community and how trustworthy are they. Trust is a significant concern in large-scale open distributed systems. Trust lies at the core of all interactions between the agents which operate in continuously varying environments. Current research leads in the area of trust in computing systems are evaluated and addressed. This article shows mechanisms discussed can successfully identify/classify groups of systems with undesired behavior.**

***Index Terms*—Organic Computing, Trust Communities, Trust based organic computing systems, Agents behavior.**

## I. Introduction

In recent years, Organic Computing (OC, [1]) and Autonomous Computing (AC, [2]) have developed in ways to cope with the increasing complexity in Information and communication technology systems(ICT) by means of Self-X factors(self -management, adaption, organization, healing, protection, optimization and so on). The term definition of organic computing by Sven Tomforde in [3] states that: Organic Computing System is a technical system, which is equipped with sensors (to perceive its environment) and actuators (to manipulate it). It adapts autonomously and dynamically to the current conditions of the perceived environment. This adaptation process has an impact on the systems utility, which is continuously improved by the organic system itself. To allow for such adaptive behavior [8], it employs so-called Self-X mechanisms. The term autonomic computing is emblematic of a vast and somewhat tangled hierarchy of natural self-governing systems, many of which consist of interacting, self-governing components that in turn comprise large numbers of interacting, autonomous, self-governing components at the next level down [3].

The basic idea is to transfer responsibilities from design-time to run-time and from the system engineer to the systems themselves [5]. Here agents are considered to operate on behalf of a user which self-adapts in a way to preserve a specific purpose and self-improve its behavior over time [4]. One particular challenge in the context of coupled and highly interconnected ICT is the rising trend toward open systems [27] consisting of autonomous and heterogeneous agents. Open means that agents are free to join and leave at any time, including those that are faulty or even malicious [4] [11]. Furthermore, heterogeneous refers to varying capabilities, knowledge, decision freedom, and pursue goals [13]. The question is that, can a Trust based intelligent system, consisting of interacting and self-organized agents, detect and isolate malicious agents over a period of time [10]. In this article, a few mechanisms are discussed to identify and classify agents behavior and separate faulty agents to improve the overall system performance which is the goal [6] [7].

The remainder of this article is organized as follows: After a short review of the state of the art in Section II, follows a quick understanding of Decentralized Desktop Grid as Trust Based Intelligent System in Section III, Section IV discusses the application scenario with a system and agent goal, agent types and trust graph are stated followed by different approaches to identify agents behavior in Section V. Section VI Concludes the article and provides an outlook to Future Work.

## II. Related work

The OC-Trust project aims at enhancing complexity in highly dynamic systems by trust-based algorithms and mechanisms derived from Organic Computing. Trust in a technical system has been the focus of many research communities. In the following, a selection of related work that helps to define the view of the term trust as well as present different approaches to the application of trust concepts in desktop grid systems is discussed [14]. Organic Computing method intends to design systems in a way that they are provided a definitive measure of autonomy so that they can deal with the increasing complexity of current systems. In systems that are able to adjust themselves to a dynamic change in environment or system state, trust among the subsystems can play a vital role as described in [16]. The definition of trust by Mui et al. [17] is as follows: Trust: a subjective expectation an agent has about anothers future behavior based on the history of their encounters. Jasang [18] differentiates trust and reputation, Reputation is a collective measure for the esteem of a community of agents for a single agent.

In this article, trust is defined as a local value within each agent aggregated from both reputation (i. e. the accumulated experiences of the community with an agent) and the history of its own experience with the other agent. The conventional reputation building process such as the one used in Internet-based systems such as eBay consists of a rating process carried out by the community and a behavior modification expected from the single agent in response to a (possibly inferior) reputation. In eBay, however, only the rating and reputation building is supported by the system. In this context, the idea is to automate also the behavior modification process. These adaptation abilities lead to considerable changes in the architecture of the agents involved [19]. Trust has been the focus of many multi-agent systems. S. D. Ramchurn has shown in [21], that trust concept in multi-agent systems can be distinguished into individual-level trust and system level trust. The focus here is on individual-level trust by modeling adaptive agents and thus aiming at reaching an optimal state at system-level by, e.g., autonomously isolating egoistic agents [11].

## III. TRUST BASED INTELLIGENT SYSTEMS

The system design is a distributed system without central control. The system is open since there is no central controlling entity and all communication are done peer-to-peer. An agent/node consists of both worker and submitter component [23]. Worker nodes belong to different administrative domains. Thus, good behavior cannot be assumed. Nodes participate voluntarily to submit work into the system and, thereby, increase the speedup of their jobs. However, the nodes also have to compute work units for other submitter [15]. In the decentralized or open desktop grid system, agents become submitters whenever a user application on their machine produces a grid job. The job is split into single Work Units (WUs) and distributed among available worker clients.

The workers process the jobs and return the results to the submitters which validate the results. The validation can be done either programmatically or by comparing results from different sources (depending on the application). However, these systems might be exposed to threats by clients that plan to exploit or damage the system. A worker can, for example, return a wrong result or not return a result at all. By extending each client with an agent component and modeling the relations between the agents with a trust mechanism, a counter to these threats can be expected and thus increase the robustness as well as the efficiency of the system. If for example, an agent chooses only those workers that it already had good experiences with (i.e., those with a trust level above a threshold), the expected outcome is better. By taking that approach further, Agent organizations that build bottom up because of these trust relations are analyzed. These so-called Trusted Communities (TCs) are composed of agents that mutually trust each other. This organization allows the members of a TC to omit safety overhead (like work unit replication) which makes them more efficient [9]. DGVCS is an open/decentralized desktop grid system and with a trust concept(TC) implemented for efficient communication between agents makes DGVCS a Trust based intelligent system.

## IV. APPLICATION SCENARIO

The application scenario that is commonly applicable to all the approaches discussed in Section V is a Decentralized Desktop Grid and Volunteer Computing System (DGVCS, [22], [35]) with agents acting on behalf of the users. To understand such systems, a multi-agent system is used, and nodes are modeled as agents. A multi-agent system is a system with numerous agents that are heterogeneous by nature and are present in the same system. Methods of organic computing are used to ensure robust and efficient operations. Participation of agents to cooperate among themselves to mutually gain an advantage. Every agent works for a user and periodically gets a job, which contains multiple parallelizable work units. The goal here is to get all work units processed as fast as possible by requesting other agents to work for it. Since the application scenario here is an open system, agents are autonomous and can join or leave at any time [11]. Below are some basic concepts necessary for a better understanding of DGVCS.

### A. *System Goal and Agent Goal*

The overall system aim is to allow agents to act as per the system rules and to achieve the best possible speedup. The overall system goal is measured either by the average speedup of the well-behaving agents or by considering the amount of cooperation(see Equation (3)) coupled with average submit to work ratio of agents(see Equation (4)). Herein, work($A_\mathrm{i}$,$A_\mathrm{j}$) represents the number of work units, which agent Ai successfully processed for agent Aj(see Equation (2)). Similarly, submit($A_\mathrm{i}$,$A_\mathrm{j}$) counts the work units $A_\mathrm{i}$ submitted to $A_\mathrm{j}$(see Equation (1)). In contrast, submit($A_\mathrm{i}$) is the number of work units $A_\mathrm{i}$ submitted to all other agents. Herein, work($A_\mathrm{j}$) shows the count of work units an agent processed for other agents [6].

$$submit(A_i) := \sum_{j=1, j\neq i}^{n} submit(A_i, A_j) \tag{1}$$

$$work(A_i) := \sum_{j=1, j\neq i}^{n} work(A_i, A_j) \tag{2}$$

$$cooperation := \sum_{i=1}^{n} work(A_i) \tag{3}$$

$$fairness := \sum_{i=1}^{n} \frac{min(\frac{submit(A_i)}{work(A_i)}), \frac{work(A_i)}{submit(A_i)}}{n} \quad (4)$$

//

The agent performance is measured by a speedup(see Equation (5)), $time_{\text{self}}$ is the time an agent would require computing a job containing multiple work units without any cooperation. $time_{\text{distributed}}$ represents the time to compute all work units of one job with the collaboration of other workers including all communication times( [24], [25], [26]). As a consequence, the speedup can only be determined after the results of the last work unit have been returned [6].

$$speedup := \frac{time_{self}}{time_{distributed}} \quad (5)$$

In case no cooperation agents are found, agents have to calculate work units on there own and obtain a speedup value equivalent to one. Usually, agents tend to act selfishly and only cooperate if they can get a reward. Agents need to decide to which other agents they want to work for and to what agents they have to assign work. Since the agent implementation is not controlled, agents can be even malicious. Hence, the system is vulnerable to different kinds of attacks. For instance, a Freerider can gain an advantage at the expense of cooperative agents by choosing not to work for any other agents [6].

### B. Agent Types

The following agent types are considered in this article:

- *Adaptive Agents* - Adaptive agents are cooperative and also work for other agents that have a good reputation in the system. The estimated present system load and how much the input queue of the agents is filled up are the factors on which the reputation usually depends.
- *Free-riders* - Free-riders reject all the work requests and do not work for other agents. But, they do ask other agents to work for them. This behavior impacts the utility of well-behaving agents and increases the overall system load.
- *Egoists* - Egoists pretend to work for other agents and return correct results with only a certain probability. They return faked results to agents by accepting all work requests and by blocking other agents as they evaluate the results.
- *Cunning Agents* - Cunning agents initially show good behavior but can change their behavior over time. Sometimes they even behave like Free-riders or Egoists. Such behavior is not easy to detect and may reduce overall system performance.
- *Altruistic Agents* - Altruistic agents do not reject any job requests and accept every job. Generally, such behavior is not malicious and in turn, increases the system performance. But this behavior makes it hard to isolated bad behaving agents and has an impact on system goals.

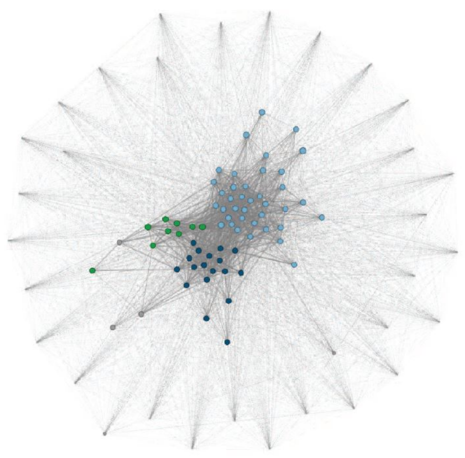

Figure 1: Simple Trust graph [6].

Adaptive agents are considered to be as well-Behaving (WB) agents, Free-riders and Egoists as Bad-behaving (BB) ones. Cunning agents seem to alter their behavior in between WB and BB, and hence cannot be classified clearly. Altruistic agents might be seen as WB agents but they also work for BB agents and thereby decrease the fairness in the system, in such contexts can be considered as BB agents.

### C. Trust graph

In a way to analyze the system state, a trust graph G = (V, E) is built where all agents are added as nodes V. Eventually, trust relationships are fetched between agents from the reputation system and added as edges E to the graph. The amount of trust between the connected agents $A_{\text{i}}$ and $A_{\text{j}}$ is represented by the weight of the edge $e_{\text{ij}}$ which connects $v_{\text{i}}$ and $v_{\text{j}}$.
Agents trust each other and tend to work together in the center of the graph. The center is called the core of the network. At the border of the graph, all isolated agents are located and are only weakly connected. The graph can be seen as a complete graph as there exists a trust relationship between every pair of agents.

## V. Different Approaches to Identity Agents Behaviour

### A. Approach 1: Trust adaptive agents

The application scenario in this approach is DGVCS as discussed in Section III. Agents are heterogeneous in terms of administrative domains, resource usage patterns and so on. How adaptive trust among agents increases the system efficiency and robustness by isolating agent types which decay the system performance is discussed here. The fitness of an agent is determined to access the success behavior parameters adopted by agents(see Equation (6)).

$$\mathit{fitness} = \alpha * \mathit{benefit} + (1 - \alpha) * (1 - \mathit{effort}) \quad (6)$$

The weight $0 \leq \alpha \leq 1$ denotes the preference of the user. The higher value of $\alpha$ means that the user wishes the agent to focus on maximizing its benefit and lower $\alpha$ means user preference is for better cooperative behavior. The fitness function is used to conclude how successful an agent's choice has been in current situations. The agent can take one of two decisions here:

- Which agent to give work units to process (submitter role).
- and whether to accept offered work units (worker role).

In the submitter role, the trust threshold for work unit distribution TT$^{sub}$, and in the worker role, the trust threshold for the acceptance of a work unit TT$^{acc}$ is determined.

*1) Adaptivity in submitter role::* The submitter role needs only the current workload in the system WL$_{total}$ from the short-term situation description as information input for the threshold decision. Based on this workload, it determines the submitter trust threshold TT$^{sub}$, which means an agent will accept all agents as workers which fulfill:

$$T_{i,j} \geq TT^{sub} \tag{7}$$

An agent that is adaptive in the submitter role is more likely to get willing workers to process its work units and result in better fitness than an agent that is unable to change its threshold at run time. Agents distinguish between personal experiences and experiences other agents had with their potential interaction partner. Essential trust and reputation values are gathered from agents.

*2) Adaptivity in worker role:* Every time an agent A$_i$ is asked to process a work unit, it analyzes the short term situation description and accordingly changes its behavior. The reputation values are scaled in between 1 and -1. An agent needs to accept every work unit if its own reputation value is negative to be able to distribute a work unit in the future. If an agent has a good reputation then it can afford to refuse work requests from other agents and hence optimize its fitness by minimizing its own effort. TT$^{acc}$ is the threshold for A$_i$ to accept a work unit from A$_j$ based on the current workload and its own reputation value:

$$T_{i,j} \geq TT^{acc} \tag{8}$$

*3) Evaluation:* The evaluation here is addressed based on how adaptivity is leading to increased efficiency and robustness in the system after the isolation of low reputation agents. The experiment is originally performed in [12], and the workload setup was such that each agent produced a job every 1500 to 4500 ticks, each job consisting an average of 31 work units with a computation size of an average 175 ticks.

- *Efficiency is increased by trust-based adaptivity:* As fitness function consists of a benefit term and an effort term, if agents can reach a high benefit with a low effort, then they are successful. But as benefit and effort are coupled by reputation, which is needed to reach a high benefit, a static strategy is not the optimal solution in every situation, especially if faulty agents threaten to exploit agents which altruistically accept work units.

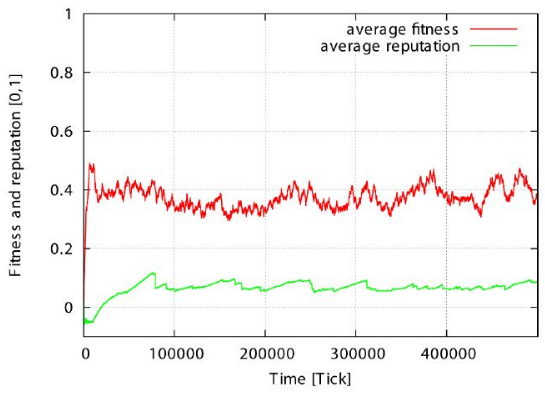

Figure 2: Fitness of adaptive agents [12].

Agent adaptivity is used to let the agent decide based on the current situation whether to optimize the benefit or the effort. Figure 2 clearly shows that adaptivity leads to oscillating reputation values resulting in high fitness. The fitness of adaptive agents shows to be much smoother because, in situations where they do not have to distribute their jobs, they improve their fitness by acting less as a worker if the $\alpha$ weight of the fitness function is changed to the weight of low effort higher than benefit agents increases whereas it decreases with Altruists [12].

- *Robustness is increased trust-based adaptivity:* First thought is that the recovery phase to be longer if the system has to cope with a higher degree of disturbance. This is because the disturbing new freeriders ask not only the well-behaving agents but also the other new freeriders to process work units for them. freeriders decline all offered work unit calculations, they get bad ratings for this behaviour and thus build up a bad reputation. As soon as the reputation has reached a certain threshold, the freeriders are isolated from the implicit Trusted Community. If there are more agents available to give bad ratings for work unit decline, this isolation process speeds up [12].

### *B. Approach 2: Graphical approach to detect attacks*

This approach concentrates on detecting possible suspicious situations by procuring a system-wide graphical representation at run time. According to a set of metrics discussed below, the graph is continuously updated and analyzed. A detailed description of this approach is illustrated in [11]. Evaluation with different metrics to identify the behavior of all agent types is monitored with the help of graphs [20]. The organization of this approach is as follows: Metrics used for continuous analysis in (1), followed by evaluation in (2).

*1) Metrics:* The metrics are selected based on the survey ( [6], [36]) and evaluation results from simulations are referenced.

- *Prestige:* In a directed graph prestige metric counts the number of incoming nodes. Every incoming edge refers to some trust from another agent. In Figure 3, the prestige for all agent types is shown. This metric helps in differentiating cooperative and non-cooperative agents. Adaptive Agents, Altruists, and Cunning Agents gain a high value. However, in the long term, Cunning Agents stay below Adaptive Agents. Altruists grow higher than Adaptive agents. Egoists and Freeriders both stay at a very low value. Egoists gain some Prestige at the beginning but lose it again.
- *Actor Centrality:* In a directed graph actor centrality metric counts the number of outgoing edges. Usually, these edges denote that this agent gave some positive rating to another agent for cooperating. Actor centrality and prestige metric behave likewise. Freerider and Egoists are at a low value, or they stay at zero. Adaptive Agents gain a high value. Cunning Agents and Altruists stay a little lower than Adaptive Agents.
- *Degree Centrality:* In an undirected graph, degree centrality counts all the edges for a node. This metric is a combination of prestige and actor centrality. In a trusted desktop grid degree centrality will count mutual trust relationships. This metric is used to identify non-cooperative agents (Freerider and Egoists). Adaptive Agents gain a very high value. Cunning Agents stay significantly below in the long term. Altruists have a value between the previous two.
- *Clustering Coefficient:* Clustering coefficient is a metric to measure the degree to which nodes tend to form a cluster in an undirected graph. It helps to find groups of connected nodes with a high density of edges. Cunning Agents and Altruists gain a very high Clustering Coefficient value. Adaptive Agents stay below. Non-cooperative agents have a value of zero again.
- *Authorities:* Authorities is a metric calculated by the HITS Algorithm [37] to rate the importance of a node in the graph. Authorities metric performs well in distinguishing Adaptive Agents and Altruists from Cunning Agents. Egoists gain a little more Authority at the beginning but start to lose it afterward. Freerider stays at a value of zero all the time.
- *Hubs:* Hubs metric aggregates the authorities of all linked nodes. Hubs have a high value if the node links to a large number of important nodes. Hubs allow us to categorize agents into a cooperative and a non-cooperative group. Similar to Actor Centrality, Egoists and Freerider only get a value higher than zero if Altruists are in the system. All cooperative agents reach a high value and stay very close.

*2) Evaluation:* The effectiveness of this approach is evaluated as a graph is shown with group size during an attack for all the agents in the group. Here several experiments for different system attack sizes are performed. The system from the below-shown images consists of 100 agents and an attack percentage of 50%, indicating 50 agents enter the system at simulation tick 100k, resulting in a system of 150 agents [11].

- *Adaptive Agents:* Mostly all agents enter the High Reputation class and stay consistent for the entire time. Some momentarily enter the Suspicious class. 50% to 75% of the agents enter the Core. No agents are in the Low Reputation class. In Figure 3, after 100k ticks, 50 additional Adaptive Agents join the system and quickly join the High Reputation class. A short time later, they also join the Core. While the size of High Reputation class increases, the Degree Centrality drops because new Adaptive Agents have a lower value. However, Core stays constant in value and slightly increases over time.

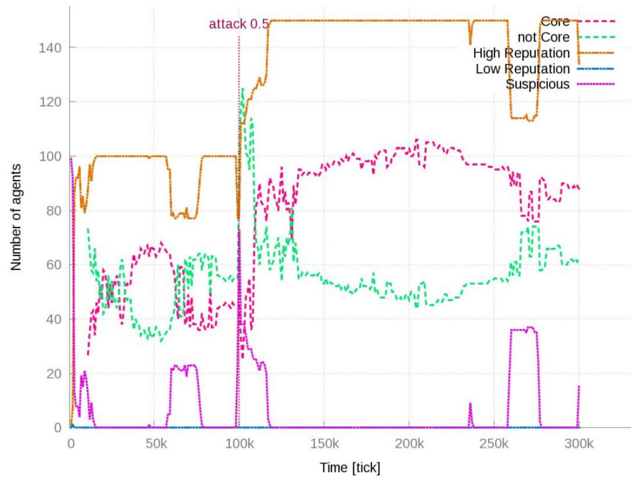

Figure 3: Adaptive agents [11].

- *Freeriders:* When Freerider enter the system they cause a Trust Breakdown Figure 4. The reputation of the Adaptive agents is lost and they enter the Suspicious class. Nevertheless, they still stay in the Core and recover back to High Reputation class. Freerider quickly moves into the Low Reputation class and stay in the Non Core. The Degree Centrality value for Low Reputation class is nearly equal to zero. Since Non core also contains some Adaptive Agents, it has a higher value, but still, it is much lower than all other classes. The Trust Breakdown is visible for a short period in the High Reputation class and significantly in Suspicious class. However, Core stays at a constant level.
- *Egoists:* The Egoist agent behaviour is similar to that of Freerider. But, there is a smaller Trust Breakdown. Adaptive Agents stay in the High Reputation class. Egoists go into Low Reputation class and stay in Non core. Compared to Freerider, the Trust Breakdown is hardly visible in High Reputation class. However, the value for Suspicious class drops for a longer period. Non Core also is constant at a medium level Figure 5.
- *Cunning Agents:* These agents behave like Adaptive Agents when they have a low reputation and like

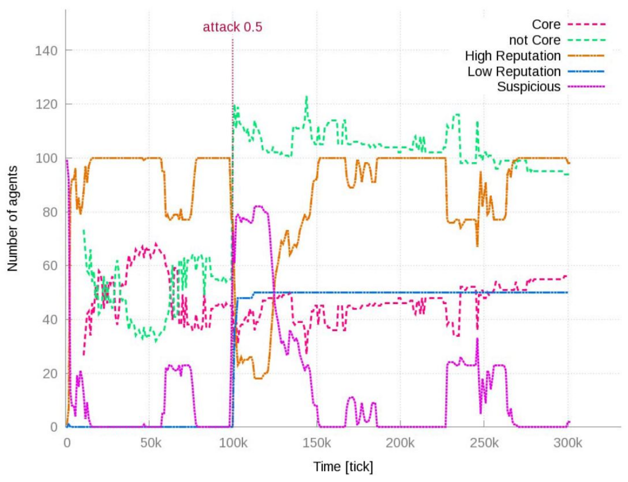

Figure 4: Freeriders [11].

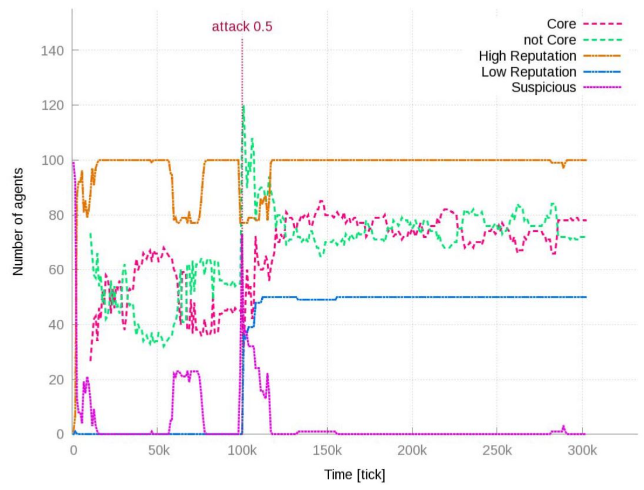

Figure 5: Egoists [11].

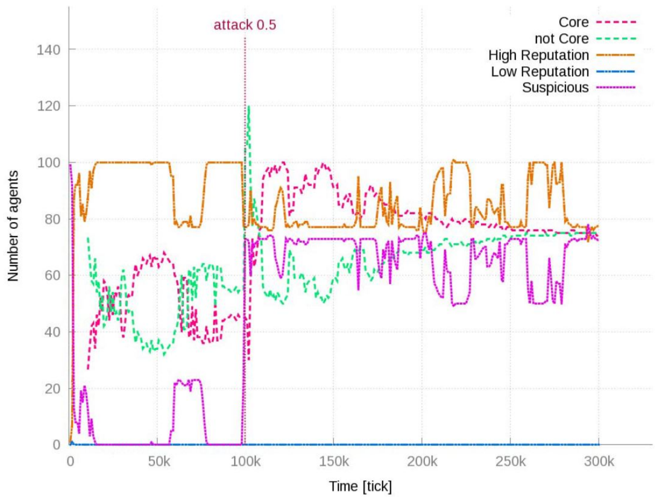

Figure 6: Cunning agents [11].

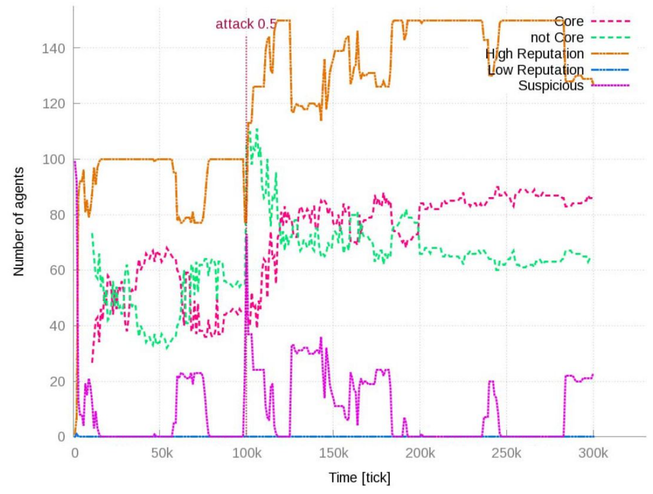

Figure 7: Altruists [11].

Freerider when they have a high reputation. As a result, they end up in the Suspicious class. Partly, they join the Core. Some Adaptive Agents also, drop to the Suspicious class. Cunning Agents can be identified by a medium value in Degree Centrality of the Suspicious class. Also, the value of Non core decreases, but less. Depending on the size of the attack, the value of the Core may get unstable for some time. However, the value of High Reputation class is nearly constant and slightly increasing Figure 6.

- *Altruists:* As these agents accept all work requests in the system, they gain a high reputation and enter the High Reputation class Figure 7. They also get included in the Core. Altruists cluster very similarly to Adaptive Agents. The value of Degree Centrality for Altruists is also very similar to the value of Adaptive Agents. Nevertheless, we can distinguish between these groups using Prestige or Clustering Coefficient.
- *All agent groups combined:* All the types above enter the system. As anticipated, Cunning Agents end up in the Suspicious class. Egoists and Freeriders end up in Low Reputation class. Adaptive Agents and Altruists can be found in High Reputation class and Core. All groups can be identified: Suspicious class contains Cunning Agents with a medium value of Degree Centrality. Low Reputation contains both Freerider and Egoists. We cannot differentiate these groups using this metric. Altruists join the high Reputation class, which slightly increases in value Figure 8.

## *C. Approach 3: Algorithmic approach*

This approach focuses on the identification and classification of malicious or faulty agents in a trust-based organic computing system is very illustrated in the article discussed by Jan Kanterta, Sven Tomforde in [4]. In order to identify group of similarly behaving agents certain metrics are applied to trust

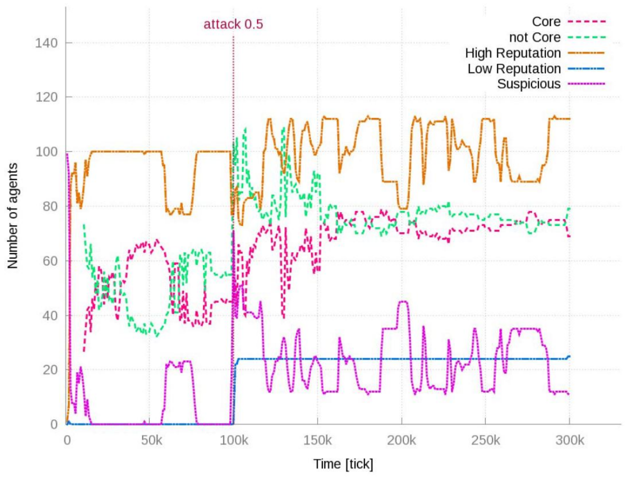

Figure 8: All agents combined [11].

graphs and then the graphs are evaluated for every node. Later, a clustering algorithm is selected to group similarly behaving agents.

*1) Algorithms:* Based on the requirements of this article the following algorithms are considered: BIRCH, DENCLUE, and Wavecluster [28]. An advantage of these algorithms is that they do not need any prior knowledge about the number of clusters or their size and are also robust against changes [4].

- *DENCLUE:* DENCLUE (DENsity-based CLUstEring) is a density-based clustering algorithm. This algorithm forms groups based on the density of points in a particular area and does handle noise effectively. DENCLUE [31] is chosen over DBSCAN (Density-based Spatial Clustering of Applications with Noise) [30] because management of noise points is more deterministic in this algorithm.
- *BIRCH:* BIRCH is an example of a hierarchy based algorithm which seeks to build a hierarchy of groups and can be achieved in two possible ways: One way is a top-down approach by starting with one group and splitting it up, and the other is a bottom-up approach which is by starting with one group per point and merging groups. Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [32] is specially designed to handle noise and outliners effectively.
- *Grid-based:* WaveCluster is an example of a grid-based algorithm and uses a multi-resolution grid to cluster points. This algorithm is based on wavelet transformations and also meets the requirements [33].

*2) Rating of Results:* The metrics used here are similar to Precision and Recall [34] used in information retrieval systems to rate the results of our clustering process. Additionally, we want to consolidate them to only one value. In Figure 9, we represented a clustering with perfect precision. No cluster with different groups exists. However, most groups are very small. In contrast, Figure 10 shows a clustering with one very large cluster for Adaptive Agents. Unfortunately, the large cluster

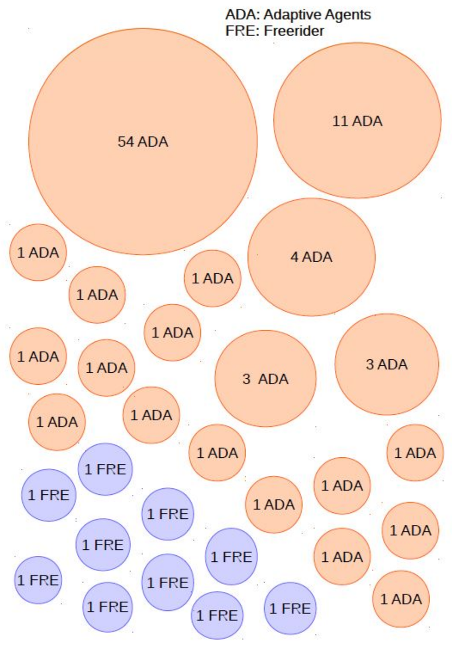

Figure 9: Clustering with perfect precision [4].

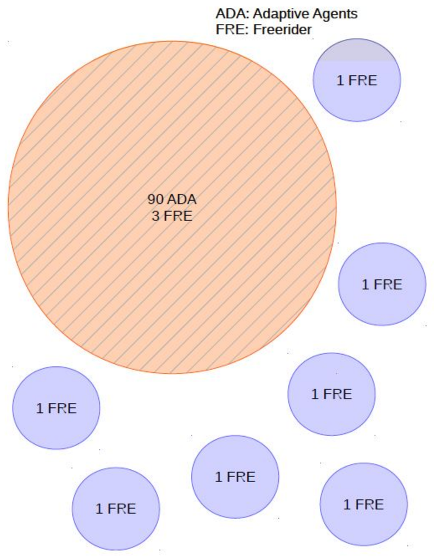

Figure 10: Clustering with less clusters [4].

also contains three Freeriders. However, from our perspective, that clustering is superior because it has fewer clusters and still a high accuracy.

*3) Evaluation:*

- *Algorithm Selection:* In a way to identify a well suitable algorithm which meets the requirements, 50 experiments with 100 agents for every algorithm is performed, and the values are averaged. In the worst case, all agent groups participate in the system, and there is a need to identify five distinct clusters. Each experiment runs

| Type | BIRCH | DENCLUE | WaveCluster |
|---|---|---|---|
| Adaptive | **51.53%** | 33.28% | 28.66% |
| Altruistic | **94.53%** | 60.30% | 61.39% |
| Egoistic | 40.05% | **45.70%** | 39.32% |
| Freerider | 40.65% | **44.36%** | 41.59% |
| Cunning | **46.65%** | 38.22% | 22.58% |

Figure 11: Average TotalShare for all selected algotithms and agents. Values range from 0 to 1 and higher is better [4].

80,000 ticks to make sure that the clustering process stabilizes. Figure 11 shows the average TotalShare for all groups per algorithm. BIRCH is the best approach for Adaptive Agents, Altruistic Agents, and Cunning Agents. For Egoistic Agents and Freeriders, the three algorithms perform similarly. A TotalShare of 50% per group may not be sufficient to find all groups. Altruistic Agents can be singled out with more than 85% share. Unfortunately, Cunning Agents behave like Adaptive Agents about half of the time. Therefore, this method cannot reliably separate them all the time, and they form a combined group with a share or more than 90% [4].

- *Independence from System Size:* To validate that the approach works autonomously from the system size, large experiments are preferred with 500, 1,000 and 1,500 agents, each with 20% of all agent types. In the Table above, we show the average TotalShare for BIRCH with increasing system size. The values are stable for all groups within the margin of error. Since BIRCH also has linear complexity, the clustering scales well even for very large systems [4].
- *Interleaved Attacks:* Until now, this approach considered groups of agents which join at the same time. The following experiment starts with Group A. After 30,000 ticks Group B joins. Both groups consist of 100 agents which are of the same types. Refer to the table above, it is shown the share per group and the combined TotalShare. We captured the TotalShare at tick 50,000 and at tick 78,000 to analyze the development over time. Most of the time, the clustering for Group A is superior because it has been longer inside the system. Also, it can be seen that the groups of Freeriders and Egoists join into one cluster. However, this only happens partially for Adaptive and Cunning Agents [4].

## VI. Conclusion and Future Work

In this article, three different approaches to identify agent behavior in a decentralized or open desktop system as an application scenario(DGVCS) as they reflect similar properties as an open trust based intelligent organic computing system. Organic Computing systems typically con-sist of diverse interacting agents. Because of the open nature, uncertain or even malicious agents are free to join - which will decay the system's performance. This openness is combined with the missing possibilities to intervene from the outside agents are autonomous and act selfishly. Consequently, this work investigates possibilities to monitor the system status from the outside by observing interaction and trust relationships. In this article, different approaches to identify agents and categorize them based on their behavior in a trust-based organic computing system is illustrated. After explaining the application scenario in detail, the taxonomy of approaches to identify agent behavior is discussed. Evaluations shown above have been focused on factors like the agent behavior, fitness, and reputation of our trust adaptive agents. The approach estimates the average system efficiency as well as the robustness of the system concerning disturbance by faulty agents.

In future work, trust-based adaptivity algorithms will improve the short term situation description by considering additional information benefits the agents to make better decisions. Further work regarding robustness will focus on varieties of disturbance sizes and there impacts on the recovery phase. Dependency can be quantified and can be used to validate trust-based mechanisms against varying disturbance sizes. And there could be an approach with questions on how the outside and hierarchically designed theories can be extended by open grid systems solutions. For example, a completely self-organized detection of faulty agents and their colluding agents can be blended with the present techniques to obtain a faster, robust and more reliable clustering information for recovery technique.

## References

[1] C. Mller-Schloer, Organic Computing: On the Feasibility of Controlled Emergence, in: CODES and ISSS 2004 Proceedings, September 8-10, 2004, ACM Press., 2004, pp. 25.

[2] J. O. Kephart, D. M. Chess, The Vision of Autonomic Computing, IEEE Computer 36 (1) (2003) 4150.

[3] Sven Tomforde and Bernhard Sick and Christian Muller Schloer, Organic Computing in the Spotlight, (03) (2017) 54-60.

[4] Jan Kantert, Sven Tomforde, Richard Scharrer, Susanne Weber, Sarah Edenhofer, and Christian Mller-Schloer. 2017. Identification and classification of agent behaviour at runtime in open, trust-based organic computing systems, J. Syst. Archit. 75, C (April 2017), 68-78. DOI: https://doi.org/10.1016/j.sysarc.2017.02.003.

[5] S. Tomforde, H. Prothmann, J. Branke, J. Hhner, M. Mnif, C. Mller-Schloer, U. Richter, H. Schmeck, Observation and Control of Organic Systems, in: Organic Computing - A Paradigm Shift for Complex Systems, Birkhuser Verlag, 2011, pp. 325 - 338.

[6] Jan Kantert and Sarah Edenhofer and Sven Tomforde and Jorg Hahner and Christian Muller-Schloer, Robust Self-Monitoring in Trusted Desktop Grids for Self-Configuration at Runtime, Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2014, London, United Kingdom, September 8-12, 2014, pp. 178 - 185.

[7] Jan Kantert and Sebastian Bodelt and Sarah Edenhofer and Sven Tomforde and Jorg Hahner and Christian Muller-Schloer, Interactive Simulation of an Open Trusted Desktop Grid System with Visualisation in 3D, Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2014, London, United Kingdom, September 8-12, 2014, pp. 191 - 192.

[8] S. Tomforde, I. Zgeras, J. Hhner, C. Mller-Schloer, Adaptive control of Wireless Sensor Networks, in: Proceedings of the 7th International Conference on Autonomic and Trusted Computing (ATC10), held in Xian, China (October 26-29, 2010), 2010, pp. 77 91.

[9] Lukas Klejnowski, Yvonne Bernard, Jrg Hhner and Christian Mller-Schloer, An Architecture for Trust-Adaptive Agents, 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop, Sept. 2010, pp. 51-58.

[10] Yvonne Bernard, Lukas Klejnowski, Jrg Hhner and Christian Mller-Schloer, "Towards Trust in Desktop Grid Systems" 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGrid 2010, 17-20 May 2010, Melbourne, Victoria, Australia,637–642.

[11] Jan Kantert, Hannes Scharf, Sarah Edenhofer, Sven Tomforde, Jrg Hhner, and Christian Mller-Schloer, 2014,"A Graph Analysis Approach to Detect Attacks in Multi-agent Systems at Runtime". In Proceedings of the 2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems (SASO '14). IEEE Computer Society, Washington, DC, USA, 80-89.

[12] Bernard, Yvonne and Klejnowski, Lukas and Cakar, Emre and Hahner, Jorg and Mller-Schloer, Christian.(2011). "Efficiency and Robustness Using Trusted Communities in a Trusted Desktop Grid". Proceedings - 2011 5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2011.10.1109/SASOW.2011.28.

[13] S. Tomforde, J. Hhner, B. Sick, Interwoven Systems, InformatikSpektrum 37 (5) (2014) 483487, Aktuelles Schlagwort.

[14] G. Brancovici and C. Mller-Schloer, A generic and modular system architecture for trustworthy, autonomous applications, in International Conference on Autonomic and Trusted Computing (ATC), Hong-Kong, China, 2007, pp. 169178.

[15] L. Mui, M. Mohtashemi, and A. Halberstadt, A computational model of trust and reputation, in Proc. 35th Annual Hawaii International Conference on HICSS System Sciences, Jan. 710, 2002, pp. 24312439.

[16] S. Tomforde, J. Hhner, H. Seebach, W. Reif, B. Sick, A. Wacker, I. Scholtes, Engineering and Mastering Interwoven Systems, in: ARCS 2014 - 27th International Conference on Architecture of Computing Systems, Workshop Proceedings, February 25-28, 2014, Luebeck, Germany, University of Luebeck, Institute of Computer Engineering, 2014, pp. 18.

[17] J. Kantert, S. Tomforde, M. Kauder, R. Scharrer, S. Edenhofer, J. Hhner, C. Mller-Schloer, Controlling Negative Emergent Behaviour by Graph Analysis at Runtime, ACM Transactions on Autonomous and Adaptive Systems (TAAS) (2016) accepted for publication.

[18] A. Jsang, R. Ismail, and C. Boyd, A survey of trust and reputation systems for online service provision, Decision Support Systems, vol. 43, no. 2, pp. 618644, March 2007.

[19] Resnick, Zeckhauser, Swanson, and Lockwood, The value of reputation on ebay: a controlled experiment.[Online] Available:http://www.si.umich.edu/presnick/papers/postcards/

[20] T. Schler and C. Mller-Schloer, An observer/controller architecture for adaptive reconfigurable stacks, in Proceedings ARCS 05. Springer, 2005, pp. pp. 139153.

[21] S. D. Ramchurn, D. Huynh, and N. R. Jennings, Trust in multi-agent systems, Knowl. Eng. Rev., vol. 19, no. 1, pp. 125, 2004

[22] S. Choi, H. Kim, E. Byun, M. Baik, S. Kim, C. Park, and C. Hwang, Characterizing and classifying desktop grid, in Cluster Computing and the Grid, 2007. CCGRID 2007. 7th IEEE International Symposium on, 2007, pp. 743 748.

[23] L. Klejnowski, Trusted Community: A Novel Multiagent Organisation for Open Distributed Systems, Ph.D. thesis, Leibniz Universitt Hannover (2014).URL-http://edok01.tib.uni-hannover.de/edoks/e01dh11/668667427.pdf

[24] R. Jain, G. Babic, B. Nagendra, C.-C. Lam, Fairness, Call Establishment Latency and Other Performance Metrics, ATM-Forum 96 (1173) (1996) 16.

[25] A. Demers, S. Keshav, S. Shenker, Analysis and Simulation of a Fair Queueing Algorithm, in: Symposium Proceedings on Communications Architectures and Protocols, SIGCOMM 89, ACM, New York, NY, USA, 1989, pp. 112.

[26] J. C. Bennett, H. Zhang, WF2Q: Worst-case Fair Weighted Fair Queueing, in: INFOCOM 96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, Vol. 1, IEEE, San Francisco, CA, USA, 1996, pp. 120128.

[27] Y. Wang, J. Vassileva, Trust-Based Community Formation in Peer-to-Peer File Sharing Networks, in: Proc. on Web Intelligence, IEEE, Beijing, China, 2004, pp. 341348.

[28] R. Xu, D. Wunsch, Survey of clustering algorithms, Neural Networks, IEEE Transactions on 16 (3) (2005) 645678.

[29] S. Lloyd, Least squares quantization in pcm, Information Theory, IEEE Transactions on 28 (2) (1982) 129137.

[30] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise., in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Vol. 96, The AAAI Press, Menlo Park, California, 1996, pp. 226231.

[31] A. Hinneburg, H.-H. Gabriel, Denclue 2.0: Fast clustering based on kernel density estimation, in: Advances in Intelligent Data Analysis VII, Springer, Berlin Heidelberg, DE, 2007, pp. 7080.

[32] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, in: ACM SIGMOD Record, Vol. 25, ACM, Montreal, Canada, 1996, pp. 103114.

[33] G. Sheikholeslami, S. Chatterjee, A. Zhang, Wavecluster: a wavelet-based clustering approach for spatial data in very large databases, The VLDB Journal 8 (3-4) (2000) 289304.

[34] D. L. Olson, D. Delen, Advanced data mining techniques, Springer Science and Business Media, 2008.

[35] S. Choi, H. Kim, E. Byun, M. Baik, S. Kim, C. Park, and C. Hwang, Characterizing and classifying desktop grid, in Cluster Computing and the Grid, 2007. CCGRID 2007. 7th IEEE International Symposium on, 2007, pp. 743 748.

[36] S. Wasserman, Social network analysis: Methods and applications. Cambridge university press, 1994, vol. 8.

[37] J. M. Kleinberg, Authoritative sources in a hyperlinked environment, Journal of the ACM (JACM), vol. 46, no. 5, pp. 604632, 1999.