



Article

Improving Agent Quality in Dynamic Smart Cities by Implementing an Agent Quality Management Framework

Najwa Abu Bakar ¹, Ali Selamat ^{2,3,4,*}  and Ondrej Krejcar ⁴ 

¹ Digital Cities Research Institute, Multimedia University, Persiaran Multimedia, Cyberjaya 63100, Malaysia; najwa.abakar@gmail.com

² Media and Games Center of Excellence (MagicX) Universiti Teknologi Malaysia & School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Skudai 81310, Johor, Malaysia

³ Malaysia Japan International Institute of Technology (MJIIT), Universiti Teknologi Malaysia Kuala Lumpur, Jalan Sultan Yahya Petra, Kuala Lumpur 54100, Malaysia

⁴ Center for Basic and Applied Science, Faculty of Informatics and Management, University of Hradec Kralove, Rokitanskeho 62, 50003 Hradec Kralove, Czech Republic; ondrej.krejcar@uhk.cz

* Correspondence: aselamat@utm.my

Received: 31 October 2019; Accepted: 20 November 2019; Published: 26 November 2019



Abstract: It is critical for quality requirements, such as trust, privacy, and confidentiality, to be fulfilled during the execution of smart city applications. In this study, smart city applications were modeled as agent systems composed of many agents, each with its own privacy and confidentiality properties. Violations of those properties may occur during execution due to the dynamic of agent behavior, decision-making capabilities, and social activities. In this research, a framework called Agent Quality Management was proposed and implemented to manage agent quality in agent systems. This paper demonstrates the effectiveness of the approach by applying it toward a smart city application called a crowdsourced navigation system to verify and assess agent data confidentiality. The AnyLogic Agent-Based Modeling tool was used to model and conduct the experiments. The experiments showed that the framework helped to improve the detection of agent quality violations in a dynamic smart city application. The results can be further analyzed using advanced data analytic approach to reduce future violations and improve data confidentiality in a smart city environment.

Keywords: smart city; agent quality; confidentiality; quality management; violations detection

1. Introduction

Cloud-based smart city technologies in the form of smart applications are currently used in many cities around the world. The smart city technologies generate a large amount of real-time and granular data that need to be protected. Fulfilling privacy and confidentiality requirements of smart city applications has been a challenge due to this massive amount of data collected and processed during execution. These data are collected from individual citizens via smart phones, smart cards, smart vehicles, and other Internet of Things (IoT) and wearable devices applications with built-in sensors and data capturing capabilities. These data, combined with data from other facilities in smart cities, such as smart buildings and smart transportation services, are highly interconnected with third party applications and city departments to improve the efficiency of smart city services. The correlation between the large amounts of data increases the risk for privacy and data confidentiality violations, such as the exposure of personal information, locations, and social activities [1].

Crowdsourcing and crowdsensing are effective, scalable, and cost-efficient data collection and processing systems. The capabilities of the smart devices and sensors to be interconnected in these crowdsourcing and crowdsensing systems allow for the execution of many useful applications, such as traffic monitoring and environment monitoring. Despite the benefits, many smart city applications, together with social networking, have exposed users to threats, such as inference attack, identity theft, and location tracking, that will lead to remote hijacking of the devices, as well as risking physical safety of the users [2]. To maintain sustainable cities and communities, it is vital to ensure that people in the ecosystem are happy and satisfied with the provided smart city services to enhance their quality of life [3]. Therefore, the national and international security threats and instabilities that occur in smart cities and megacities need to be continuously monitored in order to make better predictions, decisions, and necessary improvements [4].

For this study, it is believed that run-time verification of quality properties could help to detect violations that may occur during the execution of smart city applications. To perform the study, smart city applications were modeled as agent systems composed of multiple agents. The concept of agent systems, as either single-agent or multi-agent systems, has been used to actively model and implement distributed systems in recent decades. The definitions and applications of the concept also vary according to the purpose, i.e., how and where the concept is implemented. Multi-agent systems have been used to model designs and to implement solutions for various kinds of distributed problems in industries [5], social networking [6], and smart city applications [7]. Agents in smart city systems represent users (citizens) or other smart elements and devices that have their own unique profile and identity. Each agent in a smart city also has quality properties, such as privacy and confidentiality, that need to be protected. Agent quality properties are important requirements for trusted and reliable smart city applications [8]. Thus, despite the aim of the smart city applications to be effective and efficient in improving quality of life, it is also important to ensure that agents' quality properties are established.

Privacy and data confidentiality are examples of agent quality properties in a typical agent system, such as a smart city application. The violations of these quality properties occur most of the time during execution due to the dynamics of agent behavior, decision-making capabilities, and social activities [9]. Hence, an agent quality management system that contains proper run-time verification and quality assessment processes is needed to detect violations of the specified quality properties. In this research, the issues of agent quality properties violations that occur during agent execution in a smart city environment were tackled by proposing an Agent Quality Management (AQM) framework. To ensure that the execution of smart city applications fulfill the specified non-functional or quality requirements, this research worked toward improving the quality properties issues by detecting violations that may occurred during run-time. The incorporation of run-time verification and quality assessment processes at strategic system transitions and states was done to assess agent quality statuses at those states. The assessed agent quality statuses were then classified into violation or non-violation groups. The number of detected agent quality violations could be analyzed in order for necessary actions to be taken to improve the agent quality status.

The rest of the paper is structured as follows. Section 2 reviews the background and related works, while Section 3 describes the proposed AQM framework. Next, Sections 4 and 5 explain the case study and experiment procedure, respectively. Finally, Section 6 presents the results and Section 7 concludes the paper.

2. Related Research

This research belongs to the computer science domain that merges software engineering and information management research areas. In this section, existing works on quality management in agent systems, quality requirements in smart cities, and data quality management are presented.

2.1. Quality Management in Agent Systems

The solutions to managing quality or non-functional requirements, such as trust, reputation, and privacy, for agent systems were identified in a systematic literature review [10]. Trust and reputation management models, such as REGRET [11], A Fuzzy Reputation Agent System (AFRAS) [12], FIRE+ [13], Nusrat [14], and ScubAA [15], have previously been proposed. Table 1 presents the mapping of agent quality management literature with the quality properties.

Table 1. The agent quality management literature with their associated quality properties.

No.	Agent Quality Management Literature	Quality Properties				
		Trust	Reputation	Privacy	Confidentiality	Integrity
1	REGRET [11]	-	✓	-	-	-
2	AFRAS [12]	✓	✓	-	-	-
3	i*framework [16]	-	-	✓	-	-
4	FIRE+ [13]	✓	✓	-	-	-
5	Privacy Management [17]	-	-	✓	-	-
6	[14]	✓	✓	-	-	-
7	PrivaCIAS [18]	-	-	✓	-	✓
8	ScubAA [15]	✓	✓	-	-	-
9	[19]	✓	-	-	-	-
10	FTE [20]	✓	✓	-	-	-
11	TEIF [21]	✓	✓	-	-	-
12	[22]	✓	-	-	-	-
13	[23]	-	-	✓	-	-

AFRAS—A Fuzzy Reputation Agent System; FTE—Fuzzy Logic Based Trust Establishment; TEIF—Trust Establishment Model Using Indirect Feedback; PrivaCIAS—Privacy as Contextual Integrity in Decentralized Multi-Agent Systems.

For managing trust and reputation, the FIRE+ model [13] incorporated information from multiple sources to manage trust and reputation and to comprehensively assess agent performance in open systems. Nusrat and Vassileva [14] proposed multi-faceted trust values to be implemented in decentralized user modeling to manage trust and reputation. ScubAA [15] was presented as an agent trust management framework for open systems. The framework considers multiple sources of evidence to find the most trusted service agents in open agent systems. Hussein et al. [19] discussed a rule-based fuzzy ranking model to evaluate trust between a seller and buyer. Fuzzy Logic Based Trust Establishment (FTE) [20] and Trust Establishment Model Using Indirect Feedback (TEIF) [21] manages trust in the situation where direct feedback is not available and uses an agent behavior model to establish trust for agent interaction. Rishwaraj et al. [22] used temporal difference learning method to evaluate trust based on agent interaction experience. Singh and Chin [23] proposed an enhanced trust algorithm by implementing a hybrid multi-faceted computational trust model for an online social network.

The previously proposed privacy management models for privacy properties include i*framework [16], Privacy as Contextual Integrity in Decentralized Multi-Agent Systems (PrivaCIAS) [18], and Privacy Management [17]. The privacy management using i*framework defines the agent privacy requirements and reasoning of solutions via a systematic framework [16]. The PrivaCIAS privacy model considers requirements and solutions to detect and prevent privacy violations [18]. Piolle et al. [17] proposed a user-centered privacy management that deals with the privacy policy. Gharib et al. [24] presents an ontology for privacy requirements for systems that deal with personal information.

The existing works mentioned above have proposed solutions for some agent quality issues in general agent systems. Agent systems in smart city applications, however, have more complex interconnected data from multiple data collection sources and dynamic contextual information. Other quality requirements, such as confidentiality and integrity, also have not been extensively focused on.

Furthermore, the issues of agent quality properties violations that occur during run-time have not been extensively researched. This research identified the gap and complements the existing works by proposing a solution to detect agent quality properties violations by concentrating on the agent data and contextual information collected during run-time.

2.2. Quality Requirements in Smart Cities

A smart city is a concept that integrates information and communication technology in every element in an urban area to improve the efficiency and quality of life. Interactions between citizens, third parties, and city departments are getting more and more complex with the deployment of digital and online applications in various types of systems in a smart city environment [1]. Mobile devices, such as smart phones, smart vehicles, and wearable devices, are equipped with features, applications, and various types of sensors that allow for data collection and data-sharing capabilities. These devices interact, communicate, and exchange messages in a smart city environment via social networking, crowdsourcing, and crowdsensing [25].

Fulfilling non-functional or quality requirements, such as privacy and confidentiality, during the execution of smart city applications is still an open issue [26]. The needs to overcome the challenges have been discussed in the literature. The existing proposed solutions for agent quality requirements, such as trust and privacy, of smart city applications are presented in Table 2. Khan et al. [27] proposed a framework that provides features for the privacy-protected and trusted service provisioning in smart cities. An encryption scheme was proposed by Moffat et al. [28] to provide confidentiality for wearable devices that perform data collection in a smart city environment. The design and development of anonymous and accountable access control was presented by Hernández-Ramos et al. [29] to be deployed in a machine to machine enabled (M2M-enabled) IoT. It is an example of privacy-by-design where privacy-aware access control is implemented. A privacy-preserving architecture called Authorized Analytics performs differential privacy techniques to be implemented in a mobile IoT. Shehada et al. [30] proposed Secure Mobile Agent Protocol (SMAP), which provides confidentiality and integrity for a mobile agent to protect vehicular communication systems in smart cities from security attacks. A privacy-preserving rating data publishing model was introduced by Zhang et al. [31] to ensure privacy in recommender systems in smart cities. A trust strategy called data trustworthiness enhanced crowdsourcing strategy (DTCS) proposed a method for trust relationship establishment to enhance data trustworthiness in a mobile crowdsourcing system in smart cities [8].

Table 2. The mapping of the proposed solutions for smart city quality requirements with the focused quality properties.

No.	Proposed Solutions	Quality Properties			
		Trust	Privacy	Confidentiality	Integrity
1	Security and privacy framework [27]	✓	✓	-	-
2	Anonymous and Accountable Access Control Design [29]	-	✓	-	-
3	Authorized Analytics Architecture [32]	-	✓	✓	-
4	Secure Mobile Agent Protocol (SMAP) [30]	-	-	✓	✓
5	Ciphertext-Policy Attribute-based Encryption (CP-ABE) [28]	-	-	✓	-
6	Privacy-preserving collaborative filtering model [31]	-	✓	-	-
7	Data trustworthiness enhanced crowdsourcing strategy (DTCS) [8]	✓	-	-	-

The proposed solutions to establish trust, privacy, confidentiality, and integrity have been presented in the listed literature. Solutions including framework, encryption, design, architecture, protocol, model, and strategy elements have been proposed. However, despite implementing the above approaches, the dynamic agent profile status, social activities, and shared contents that keep

changing during run-time may still cause unintended violations of quality properties. Thus, a gap was identified and a solution was proposed to detect agent quality properties violations during run-time.

2.3. Data Quality Management

To perform the quality management of data or information, many researchers have been working on merging quality management with information systems [33,34]. Total Data Quality Management (TDQM) was proposed by Wang [35]. To date, TDQM is still a relevant concept and has been improved from time to time. It has been implemented to manage different types of recent data and applications, such as social media [36], manufacturing [37], financial [38], and medical [39] data. In this research, TDQM was adapted to propose the Agent Quality Management (AQM) framework for managing agent quality in smart city environment.

The TDQM cycle consists of four elements, which are define, measure, analyze, and improve. The first process is the define process where output quality parameters are identified based on characteristics of the collected input, identification of the resources to support the quality of the output, and requirements of the produced outputs and the system infrastructure. The second quality process in the cycle is the measuring process. In this phase, the scores for the defined quality parameters are calculated. To calculate the scores, data quality metrics need to be developed. The data quality requirements identified during the definition process are transformed into rules and measurable indicators (quality statuses or scores) that reflect the real condition (contextual information) of the data. In order for the parameter to receive a high-quality status or score (correct), certain conditions must exist, and if a condition does not exist, the parameter will receive a low status (error) score. Third, the analysis process investigates the data quality measurement results generated. The analysis can be simple or complex to suite the verification needs. The simplest analysis method is done by answering a list of assessment questions. Finally, the improvement process is the action taken after a thorough analysis is performed. Here, the data quality parameters may be refined, removed, or added to produce more accurate and reliable data quality statuses for the information product.

2.4. Agent Tools

There are tools that can be used to design, model, simulate, develop, and analyze agent systems. One of the tools that can be used to model and simulate agent systems is AnyLogic. The AnyLogic tool supports agent model architecture; agent synchronization; space, mobility, and spatial animation; agent connections and communication; and dynamic creation and destruction of agents [40]. For developing and executing agent systems, JADE (Java Agent Development Framework) can be used. JADE is a JAVA-based software framework for multi-agent systems that has been in development since at least 2001. The JADE platform enables the implementation of multiple agents by following Foundation for Intelligent Physical Agents (FIPA) standard architecture, agent management, and communication [41]. Another tool for constructing agent applications is the Cognitive Agent Architecture (Cougaar). Cougaar is a framework in which the architecture includes components for agent communication via message passing, naming, mobility, agent local memory, interfaces, and additional agent functionalities [42]. Finally, a tool that provides a platform for the development of agent systems is Jason. Jason infrastructure provides agent communication and agent life cycle control (creation, execution, and destruction) [43]. ACLAnalyser is a powerful tool to monitor, debug, and analyze multi-agent systems. ACLAnalyser monitors and analyses agents' interactions and saves the exchanged messages into a database to be retrieved later to identify useful information [44]. Similar to the JADE RMA (Remote Monitoring Agent) GUI (Graphical User Interface), ACLAnalyser provides agent conversation monitoring with additional advance features, such as data mining to monitor larger and more complex multi agent systems. Table 3 shows the comparison matrix that maps agent tools with its features. From the comparison matrix, AnyLogic [40] is the tool that provides design, modelling, simulation, development, and analysis features.

Table 3. Agent tools with their associated features.

No.	Agent Tools	Features				
		Design	Modelling	Simulation	Development	Analysis
1	AnyLogic [40]	√	√	√	√	√
2	Java Agent Development Framework (JADE) [41]				√	
3	Cougaar [42]				√	
4	Jason [43]				√	
5	ACLAnalyzer [44]					√

3. AQM Framework for Smart City Applications

By observing the trust, privacy, and confidentiality issues from a data quality point of view, it is suggested that quality properties in smart city applications could be checked and assessed by considering contextual data. In this research, smart city applications were modeled as agent systems. During the run-time of the modeled smart city application or agent system, the effective way to assess agent quality was by performing an analysis of agent interactions and social activities. The richness of the agent data captured during those activities has opened the opportunity for granular analysis using the data at critical points during the execution. The AQM framework is designed as an extension of the generic three stages of the existing verification process, i.e., modeling, specification, and verification [45]. The framework was proposed by adapting the TDQM model reviewed in Section 2. The TDQM principle was adapted to manage agent quality within agent systems. The framework coordinates the agent quality management processes into four stages, i.e., definition and specification, modeling and execution, verification and assessment, and analysis and improvement phases that can be implemented as a cycle. AQM checks and assesses the agent quality conditions in selected critical states and transitions. The quality checking process includes continuous monitoring during run-time in order for the quality specifications to be verified by considering contextual information, i.e., relating the agents to their individual profile status, social preferences, and content sharing activities that may affect agent quality properties.

The quality assessments for an individual agent in a smart city environment can be used to detect agent quality properties violations during run-time. The integration of TDQM in order to check, assess, and manage agent quality, and to finally detect agent quality violations, is the new contribution that complements the existing proposed solutions for quality requirements in smart cities. Figure 1 illustrates the extension of the existing generic verification process and the integration of TDQM to form the new AQM framework. Multiple processes are organized into the four elements of AQM, which are:

1. definition and specification,
2. modeling and simulation,
3. verification and assessment,
4. analysis and improvement.

In the following subsections, each of the processes implemented in the proposed AQM are explained.

3.1. Definition and Specification

The first part of the AQM is the definition and specification process, starting with the definition of agent quality goals and criteria. Agent quality criteria are the factors that constitute high-quality agents. The existing works suggested that data quality can be defined as quality assessments performed toward several defined parameters [46,47]. In this research, the focus was toward the agent quality (non-functional) requirements during run-time. The high-quality agents were the agents that fulfilled the constraints or non-functional requirements to ensure reliability and trust between agents.

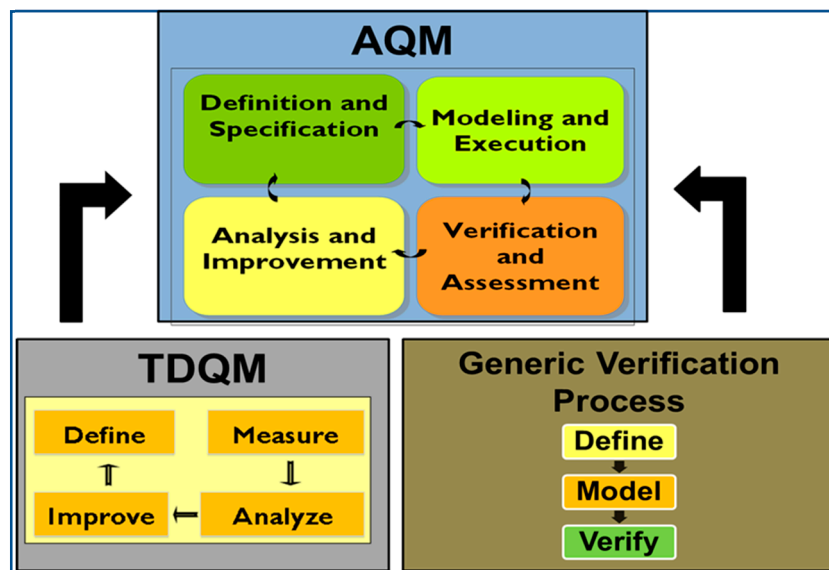


Figure 1. Extension of the generic verification process and integration of the Total Data Quality Management (TDQM) process.

Thus, the selected criteria should represent the quality of agents in various conditions and activities during execution. The selected criteria were preferably measurable in order for the introduced metrics for these criteria can be used to continuously quantify agent quality statuses during run-time. The agent parameters and critical states were defined to be included in the agent quality metric definition. An agent quality metric is a measure of agent quality properties. In order to measure the agent quality, based on the abstraction of the agent requirements definition, the agent quality metrics were constructed. Finally, at the end of the definition and specification phase, checking rules were constructed. Figure 2 shows the definition processes.



Figure 2. Definition and specification phase of the Agent Quality Management system.

The agent quality requirements, parameters, and critical states identified were transformed into the checking rules that reflect the specifications and real condition of the agent quality status. The main aim of the checking rules was to tie together the agent parameters that contribute to the checking and assessment of the agent quality statuses. The implemented information quality works were adapted [46,47], where the aggregation of multiple agent quality parameters is suggested. The rules defined in this phase are will be used in the verification and assessment phase as the checking rules

to perform run-time verification and quality assessment within the agent models. The rules for each agent quality metric range from a simple if–else statement typical in many programming algorithms to more complex formal specifications (e.g., written in temporal logic) for verification purposes. For each metric, the rules require the execution of run-time verification to detect the reachability of critical states where quality assessment is performed.

For this research, to define the checking rules mentioned above, the steps are listed below:

1. Defining the agent quality goal for an agent system: The agent quality goals during run-time are used to ensure that agents are high in quality such that the non-functional agent requirements are fulfilled. The examples of agent non-functional requirements are trust, reputation, privacy, confidentiality, and integrity.
2. Defining agent quality requirements: To define the agent quality requirements, the selected criteria should represent the quality of agents in various conditions and activities during execution. The selected criteria are preferably measurable in order for the introduced metrics so that these criteria can be used to continuously quantify agent quality statuses during run-time. For each quality criterion, the contributing factors (contextual conditions) that can cause satisfaction and possible violations of requirements are identified. The requirements that are defined are the most important non-functional requirements or constraints of the agents as an individual or group. The list is not exhaustive and it should rather be considered as the minimal agent quality requirements to be verified and assessed. The list can be extended further during the definition process in the next AQM process cycle.
3. Defining agent quality metrics: The abstraction of agent quality requirements and violations that are defined is translated into a more concrete form of representation known as agent quality metrics. The agent parameters and critical states are defined so they can be included in the agent quality metric definition. An agent quality metric is a measure of agent quality properties.
4. Defining agent quality checking rules: Rules and indicators (in the form of quality statuses) are associated with each metric based on the contextual conditions. The agent quality requirements, parameters, and critical states identified are transformed into the checking rules for each metric, which reflect the specifications and real condition of the agent quality status. The main aim of the checking rules is to tie together the agent parameters that contribute to the checking and assessment of the agent quality statuses. The rules defined in this phase will be used in the verification and assessment phase as the checking rules to perform run-time verification and quality assessment within the agent models. The rules for agent quality metrics range from simple if–else statements to more complex formal specifications (e.g., written in temporal logic) for verification purposes. For each metric, the rules require the execution of run-time verification to detect the reachability of critical states where quality assessment will be performed.
5. The agent quality metrics and checking rules for each agent quality requirements are formally specified. The specification in AQM refers to the specification of the reachability of certain states that are defined as critical. Critical states occur when the agent states that determine either agent quality requirements are violated. When these critical states are reached, the quality assessment is performed by determining the quality statuses based on the conditions that are defined as violations and non-violations for the agent. Finally, the checking rules containing the preconditions and the assigned quality statuses are specified using temporal logic and are presented as the output for this definition phase.

3.2. Modeling and Execution

The second process of the AQM is the modeling and execution phase. In this phase, the agent systems are modeled, starting from the creation of the agent population and followed by the execution of agent activities, such as creating relationship links and exchanging information between agents. Every agents' profile and default privacy preferences are also included in the model. In order to

perform continuous assessment and verification during run-time, the quality checking rules defined in the previous phase are incorporated and used to assess the quality status of agent states.

The agent systems to be verified are modeled using the AnyLogic system [40]. Each individual agent's behaviors, preferences, and activities are modeled using state charts [48]. The creation of agents and the modeled agent activities and statuses can be executed at a specified rate and time throughout the model execution. Depending on the modeled agent systems, each agent has its own parameters and attributes. The typical agent attributes include: AgentId and AgentName. Agent social activities involve the interaction between agents to request and confirm relationship links and to share information related to profiles, relationships, online statuses, and contents. The modeled and executed agent social activities are monitored and real time agent parameters and attributes (contextual information) are tracked in order to be used during quality assessment. The agent quality depends on its own preferences and the preferences of other agents that interact with the agent. Basically, for agents that interact with each other and perform social activities, there are three layers of executions that incorporate the verification and assessment process, which are the individual level, group level, and content level. Figure 3 shows the modelling and execution phase of the AQM framework.

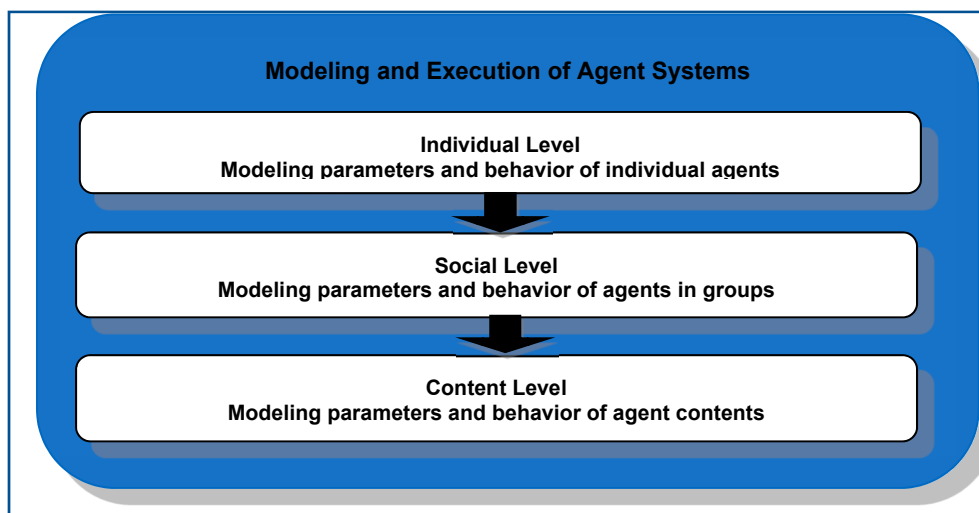


Figure 3. Modelling and execution phase of the Agent Quality Management framework.

The explanation of the sample implementation of each level is as follows:

1. Individual level: The verification and assessment of an individual agent's quality parameter includes the monitoring of agent activities by tracking the active states and verifying the reachability of critical states. Next, the assessment of the exposure level for each parameter is performed. The continuous assessment produces real-time exposure statuses that consider real time and dynamic agent contextual information that are the privacy preference settings for each agent parameter, such as the profile, the presence (online) status and the chatting availability, and the confidentiality settings for each agent content.
2. Social level: The second layer verifies the agent relationship quality parameter in which the privacy preference settings and social activities of other agents directly and indirectly affect the agent's privacy.
3. Content level. The third layer is to check the agent quality parameter as a content owner and content sharing partner. As a content owner, the agent quality depends directly on the agent preference settings, and as a content sharing partner, the agent quality depends indirectly on other agent relationship privacy preference and sharing activities.

In this modeling and execution phase, the constructed metrics and rules from the previous phase for checking and measuring the agent confidentiality are modeled and the run-time verification and assessment of AQM is implemented within the agent system model using AnyLogic [40].

3.3. Verification and Assessment

The third AQM component is the verification and assessment phase as shown in Figure 4. In this phase, a process model called Run-Time Verification and Quality Assessment (RVQA) is developed to perform the checking and assessment process. The modeled agent systems from the previous phase incorporate rules to check the reachability of critical states where violations can occur. Next, whenever the critical states are reached, quality assessments are performed to determine the quality statuses. The assessment procedure identifies the state quality conditions and assesses the quality status based on the specified checking rules. The agent quality statuses are used for classifying the agent quality satisfaction and violations. For each agent quality property level, i.e., individual, social, or content level, the total number of the detected quality violations is calculated.

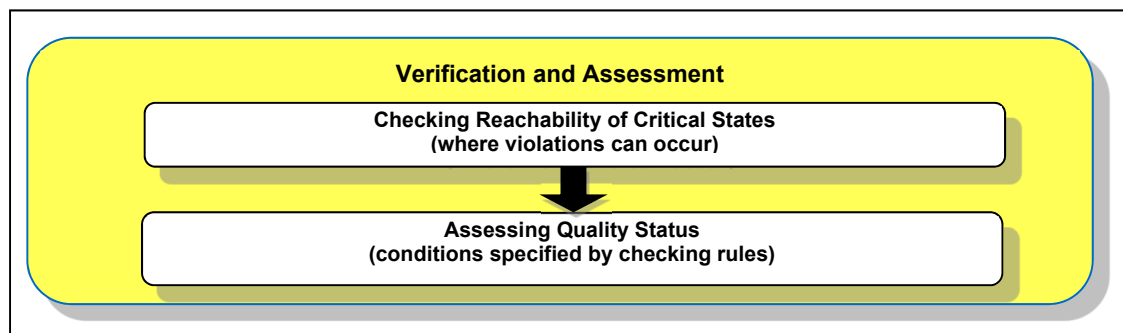


Figure 4. Verification and assessment phase of the Agent Quality Management framework.

3.4. Analysis and Improvement

Following the verification and assessment part, the final process in the AQM framework is the analysis and improvement phase as presented in Figure 5. This analysis process is performed as one of the phases in the quality management cycle. After checking whether critical states have been reached and assessing the agent quality statuses, in this phase, the statuses are classified and presented in a time series to show the level of agent run-time quality statuses. The analysis of the verification and assessment results can be used by the agent owner to minimize the quality violation levels by changing the preference settings and increasing or reducing the interaction activities. For example, by changing the privacy setting from public to private or friend only for a certain parameter, the exposure level can be reduced. The improvement procedures are also performed here to increase the effectiveness of the solution by identifying new quality metrics for the next verification cycle. From the analysis process, improvements can be suggested to add, remove, refine, or redefine agent quality requirements, metrics, and rules for checking and measuring the quality of agent systems from time to time.

3.5. Incorporating AQM within the Agent System Model

The earlier sections of this paper have presented the AQM framework as a solution for agent quality issues during run-time. Sections 3.1–3.4 explained the components of the AQM framework. This section elaborates on the incorporation of AQM components within the agent system model. Figure 6 presents the AQM framework components and the input and output taken and produced during the execution of the model.

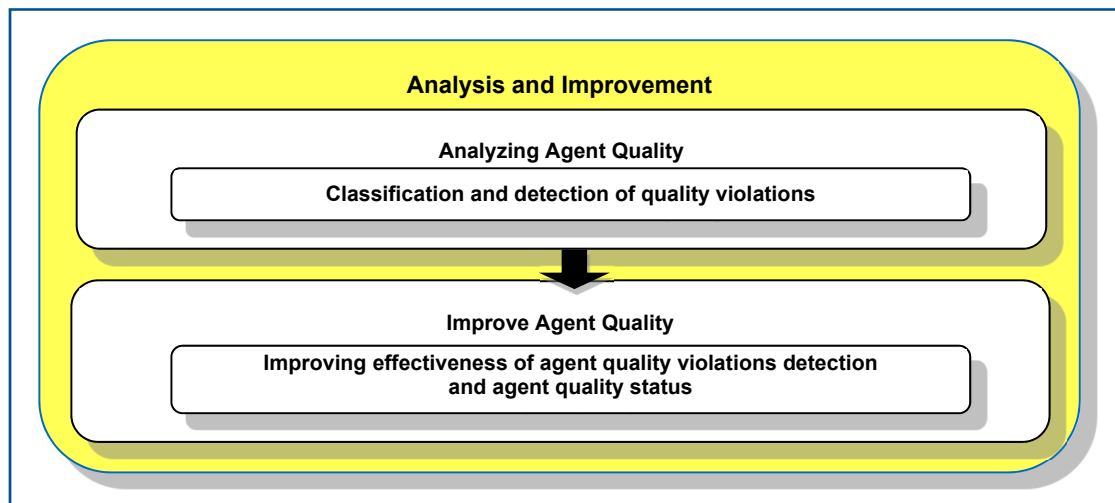


Figure 5. Analysis and improvement phase.

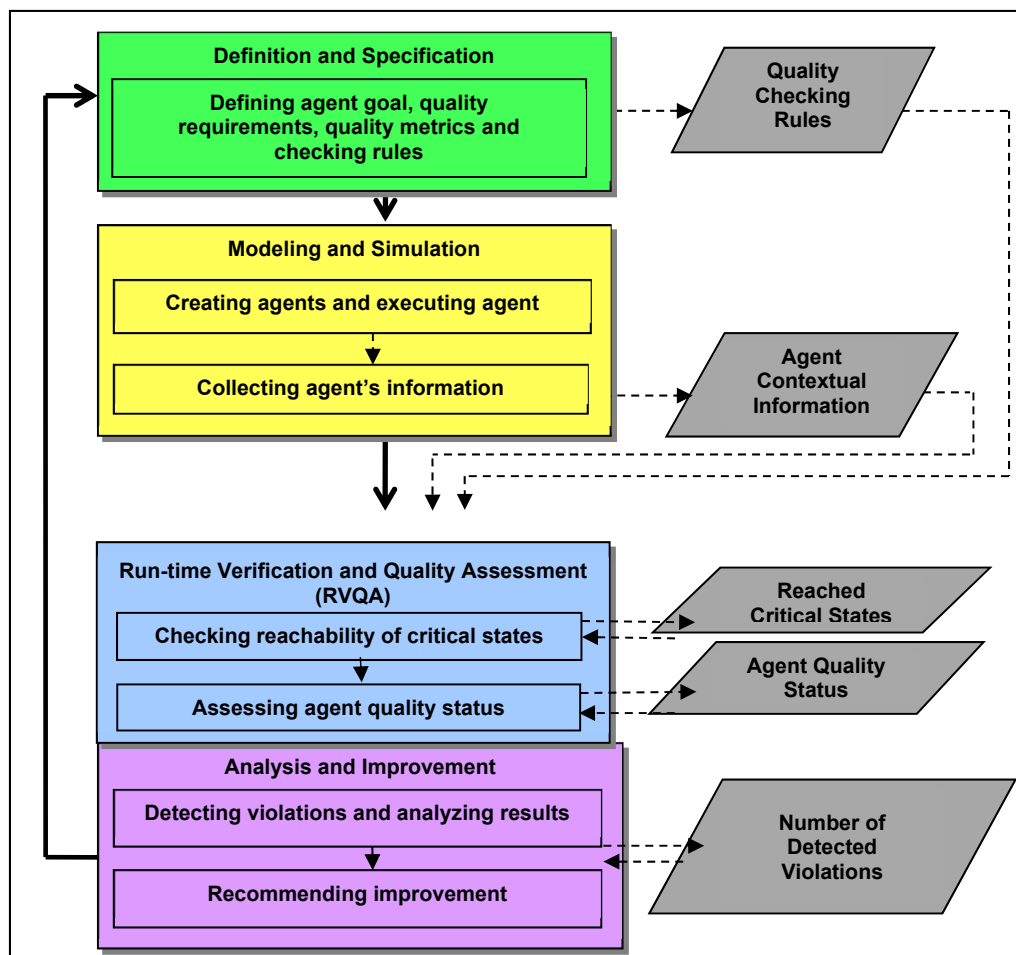


Figure 6. The components of the Agent Quality Management framework.

Depending on the type of agent systems and its features, to perform tasks and to achieve goals in agent systems, agents that represent users interact with other agents (other users). The interactions establish relationship links, involve conversations, and share information. However, these interaction activities are susceptible to quality violations that can lead to malicious activities, such as identity theft

and inference attacks, due to the possibilities of unintentional overexposing of profile information, over-sharing of contents, and revealing of physical locations [49,50]. In many situations, users are not aware that their online social activities and tasks, such as accepting relationship requests, sharing information, and buying or selling online, risk their online and offline privacy, data confidentiality, or integrity. Therefore, the AQM framework was proposed to manage agent quality and detect agent quality violations.

4. Case Study: AQM Implementation

The Crowdsourced Navigation System (CNS) [51] is a mobile crowdsourcing system that has been chosen as a case study to evaluate the effectiveness of the proposed solution. The proposed AQM was implemented for managing the data confidentiality property in CNS. The four AQM phases were incorporated in the CNS agent system. Figure 7 shows the CNS architecture that incorporates the AQM framework.

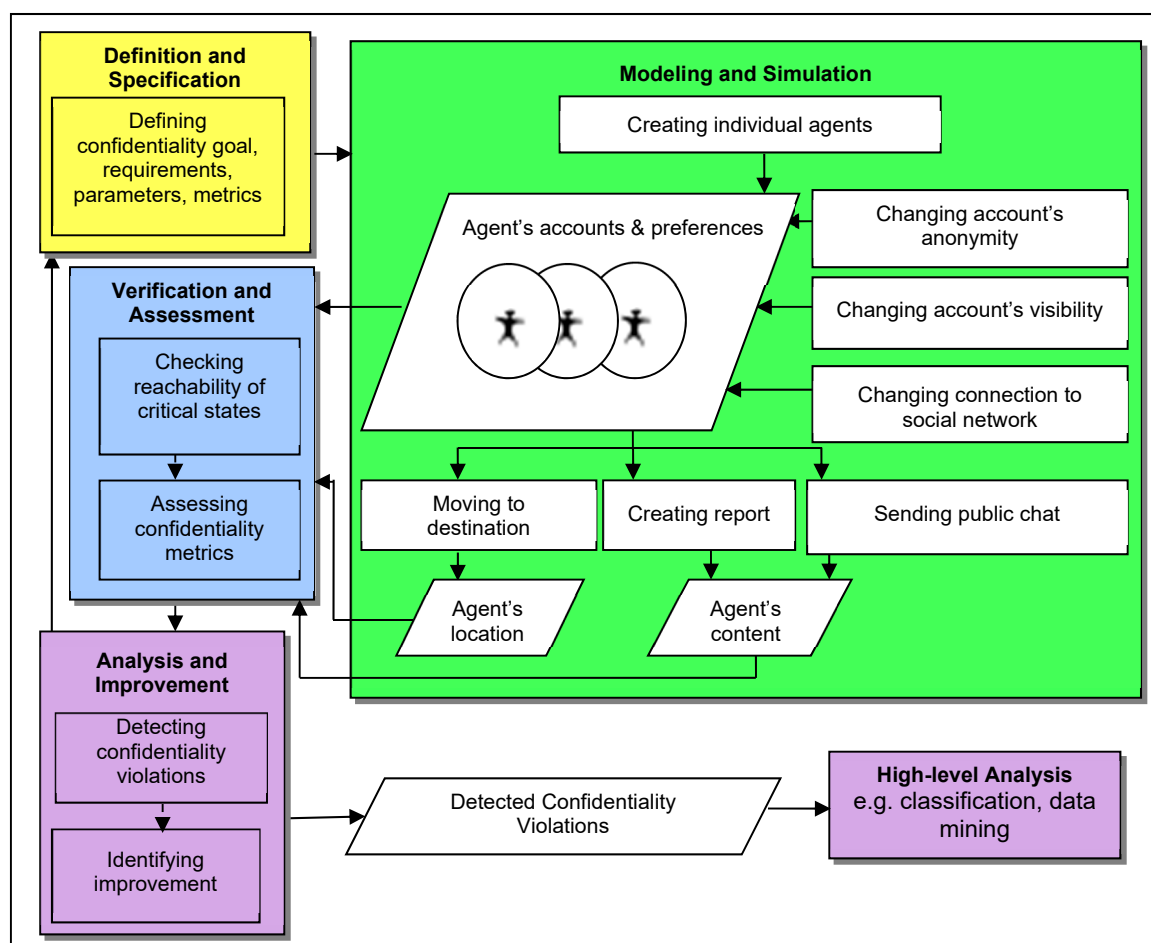


Figure 7. The Crowdsourced Navigation System (CNS) architecture that incorporates the AQM framework.

4.1. Definition and Specification for the CNS

Confidentiality is implemented as one of the agent quality goals in an open system to ensure agent data is protected [52]. Violations of data confidentiality can be in the form of data disclosures, where agent's data are unintentionally disclosed to other agents during social activities and information sharing [53]. Data confidentiality violations can put the safety of the agent's data at risk by disclosing both the agent identity and physical location at the same time. Therefore, the agent quality requirements are implemented to fulfill the need to determine the agent data confidentiality status during execution.

This can be achieved via verification and assessment by considering contextual information that provides the contributing factors or conditions for the data confidentiality violations to occur. Agent confidentiality criteria for this case study include the protection of an agent's personal identity and location, as well as social and shared content information, from potential disclosure during the execution of agent activities. The conditions include the disclosure of agent's personal identity, such as first names or company name as part of the username during account activation, the preference to allow the agent to become publicly visible, and the connection to online social network services containing personal identity together with location information.

Every active agent discloses a certain level of data depending on agent conditions at (1) the individual level, depending on the account anonymity and presence visibility; (2) the social level, depending on the social networking connection; and (3) the content level, depending on the shared content types. The definition of agent confidentiality requirements and the possible violations for confidentiality criteria, the extracted metrics with attributes, and the constructed checking rules for each of the confidentiality metrics have been specified and reported [51]. Figure 8 summarizes the identified contributing factors or conditions of data confidentiality disclosures in the CNS. The contributing factors can be divided into three levels, which are the disclosures at the individual, social and content levels.

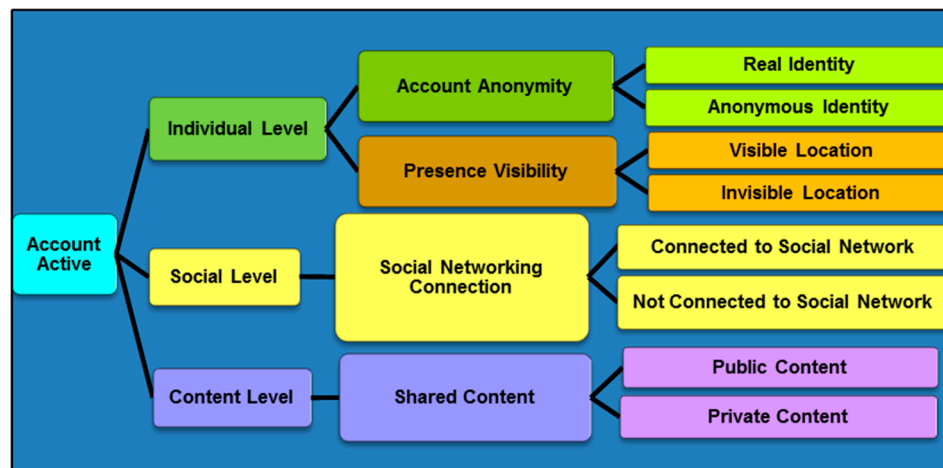


Figure 8. Contributing factors toward data confidentiality disclosure conditions in the CNS.

4.2. Modeling and Execution of the CNS

The agent-based CNS has been modeled using the AnyLogic multi-method modeling tool [40]. State chart is used to model agent activities where the runtime verification and quality assessment are implemented at strategic transitions [51]. To activate an account, a user needs to choose a username. Agents can choose whether to use a username that contains personal identity information, such as their first name or company name, or use an anonymous name. Whether the username is real or anonymous, by default, other agents can see the location of active agents. Although an agent can choose to become invisible so that its presence or online status is unknown to other agents, this choice is a constraint that limits agent functionality to report and share information with other agents. These choices and preferences are an example of conditions that determine whether the confidentiality of agent identity and location is violated.

4.3. Verification and Assessment for the CNS

The RVQA is a process model developed for the verification and assessment phase of the AQM framework to perform run-time data confidentiality checking and assessment processes. The processes are carried out within the agent-based CNS model. The RVQA process in the CNS starts when the user agents are created [51]. The run-time verification is implemented to identify when the defined critical

states are reached. The critical states are the states where the violations can occur. At the critical states, the quality assessment is performed. The assessment is performed to indicate the disclosure status of the agent identity, location, social information, and contents by considering the contextual information. The contextual information, i.e., agent activities and decisions, determine the agent data confidentiality status as disclosed or protected. Finally, the identified data confidentiality disclosures are classified and reported as detected data confidentiality violations.

The RVQA assesses the agent data confidentiality disclosure status based on the defined checking rules for each confidentiality metric [51]. As the verification and assessment processes are part of the AQM framework and the AQM phases are performed as a cycle, the RVQA can be performed at one level at a time starting from the individual level, followed by the social and content levels, to improve the effectiveness of violation detection. To perform the detection process, the confidentiality requirements and specified checking rules are used to design the algorithms to detect the data confidentiality disclosures. The detection includes the process of checking the reachability of critical states, assessing disclosure status at the critical states, and finally, reporting the number of detected confidentiality violations. The following subsections explain the RVQA process at the agent's individual, social, and content levels in detail.

4.3.1. The RVQA Process at the Agent Individual Level of the CNS

Figure 9 shows individual agent activities and the implementation of the RVQA to check an agent data confidentiality disclosure status during an agent active period. The individual agent activities include the stage when the agent account becomes active, the dynamic changes of agent decisions toward profile anonymity and presence visibility throughout the agent active period, and the stage when the agent account becomes inactive.

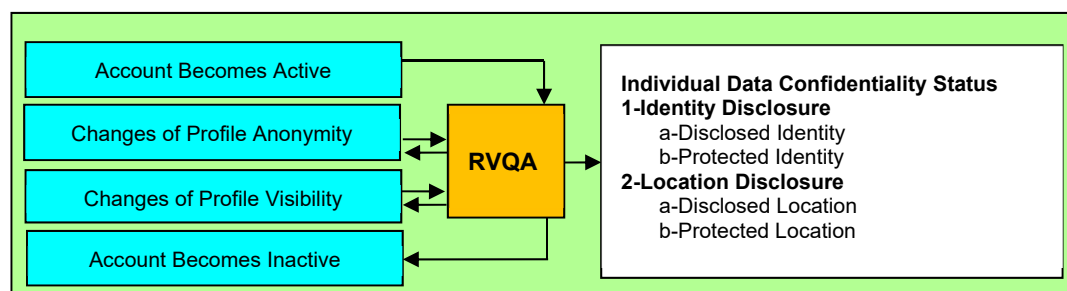


Figure 9. The RVQA implementation at the agent individual level of the CNS.

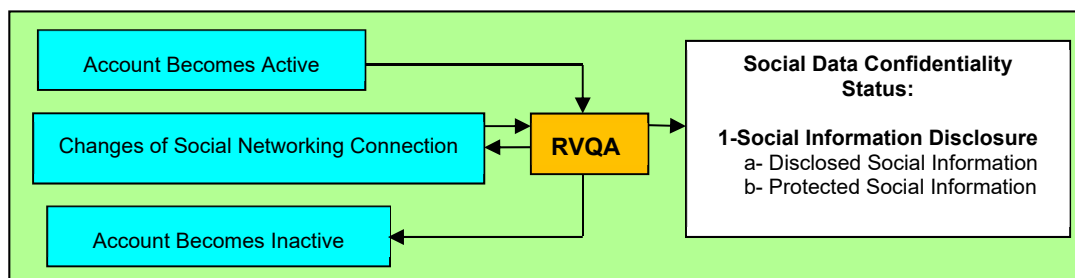
Algorithm 1 shows the RVQA confidentiality violations detection for the agent individual level of the CNS that checks agent identity and location confidentiality disclosures. As defined by the checking rules for the individual level [51], the agent account active state is a critical state, while the agent decisions for profile anonymity and presence visibility determine its identity and location confidentiality disclosures statuses. An agent dynamic profile anonymity option can be chosen to be either "real" or "anonymous" and determines whether the agent identity is "disclosed" or "protected," respectively. Similarly, the agent presence visibility option can be set to be "visible" or "invisible" and determines whether the agent location is "disclosed" or "protected," respectively.

Algorithm 1. Identity and location confidentiality disclosures detection.

Input:	Agent Index: x , Current Time: t
Output:	noDisclosedIdentity, noNotDisclosedIdentity, noDisclosedLocation, noNotDisclosedLocation, noDisclosures
1	begin
2	if AccountActive(x, t) then
3	Critical(x, t) \rightarrow true
4	if ProfileReal(x, t) then
5	IdentityDisclosed(x, t)
6	increase($x, \text{noDisclosedIdentity}$)
7	increase($x, \text{noDisclosures}$)
8	end
9	if ProfileAnonymous(x, t) then
10	IdentityNotDisclosed(x, t)
11	increase($x, \text{noNotDisclosedIdentity}$)
12	end
13	if PresenceVisible(x, t) then
14	LocationDisclosed(x, t)
15	increase($x, \text{noDisclosedLocation}$)
16	increase($x, \text{noDisclosures}$)
17	end
18	if PresenceInvisible(x, t) then
19	LocationNotDisclosed(x, t)
20	increase($x, \text{noNotDisclosedLocation}$)
21	end
22	end
23	end
24	return noDisclosedIdentity, noNotDisclosedIdentity, noDisclosedLocation, noNotDisclosedLocation, noDisclosures

4.3.2. The RVQA Process at the Agent Social Level of the CNS

Figure 10 shows the RVQA implementation to check the agent data confidentiality disclosure status at the social level. The agent social activities include the stage when the agent account becomes active, the agent establishes connection to the social network, and finally, when the agent account becomes inactive.

**Figure 10.** The RVQA implementation at agent social level of the CNS.

Algorithm 2 shows the RVQA data confidentiality violations detection for the agent social level of the CNS that checks for social information disclosures. As defined by the checking rules for the social level [51], the agent account active status determines the agent critical states, and the agent's connection to online social networking determines the agent's social disclosure status. The agent's

dynamically established or not established connections to online social networking determine whether the social information are “disclosed” or “protected,” respectively.

Algorithm 2. Identity and location confidentiality disclosures detection.

Input:	Agent Index: x , Current Time: t
Output:	noDisclosedSocialInformation, noNotDisclosedSocialInformation, noDisclosures
1	begin
2	if AccountActive(x, t) then
3	Critical(x, t) \rightarrow true
4	if SocialNetworkingConnected(x, t) then
5	SocialInformationDisclosed(x, t)
6	increase(noDisclosedSocialInformation)
7	increase(x , noDisclosures)
8	end
9	if SocialNetworkingNotConnected(x, t) then
10	SocialInformationNotDisclosed(x, t)
11	increase(noNotDisclosedSocialInformation)
12	end
13	end
14	end
15	return noDisclosedSocialInformation, noNotDisclosedSocialInformation, noDisclosures

4.3.3. The RVQA Process at the Agent Content Level of the CNS

Figure 11 shows the agent content-sharing activities and the RVQA implementation to check the agent data confidentiality disclosure status during the execution. The agent content-sharing activities include the stage when the agent’s account becomes active, contents are shared, and finally the stage when agent’s account becomes inactive.

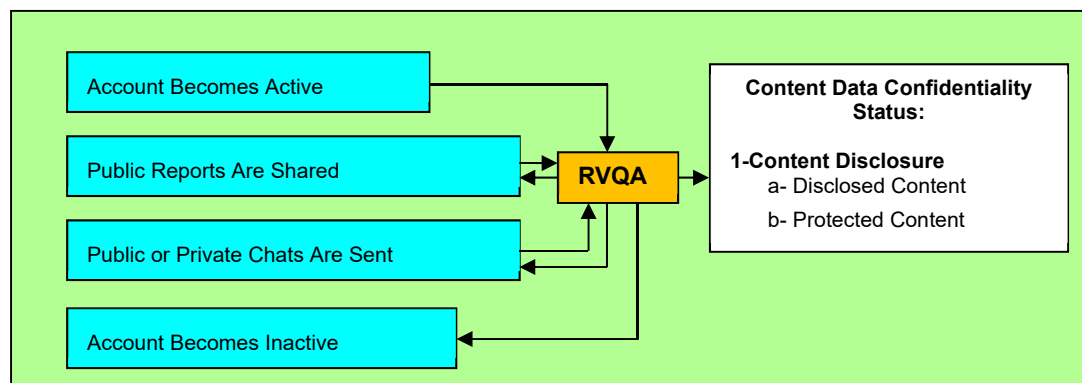


Figure 11. The RVQA implementation at the agent content level of the CNS.

Algorithm 3 shows the RVQA data confidentiality violations detection for the agent content level of the CNS. As defined by the checking rules for the content level [51], the agent account active state is an agent critical state, while the shared content types determine the agent’s content-disclosure status. Shared content types that are “public” or “private” determine whether the shared contents are “disclosed” or “protected,” respectively.

Algorithm 3. Content confidentiality disclosures detection.

Input:	Agent Index: x , Current Time: t
Output:	noDisclosedContent, noProtectedContent, noDisclosures
1	begin
2	if ContentShared(x, t) then
3	Critical(x, t) \rightarrow true
4	if ContentPublic(x, t) then
5	ContentDisclosed(x, t)
6	increase(noDisclosedContent)
7	increase(noDisclosures)
8	end
9	if ContentPrivate(x, t) then
10	ContentProtected(x, t)
11	increase(noProtectedContent)
12	end
13	end
14	end
15	return noDisclosedContent, noProtectedContent, noDisclosures

These verification and assessment processes are performed continuously for all levels whenever the defined critical states are reached. The numbers of detected violations, i.e., noDisclosedIdentity, noDisclosedLocation, noDisclosedSocialInformation and noDisclosedContent, are increased every time the data disclosures are detected. It is followed by keeping track of the total number of confidentiality disclosures detected to indicate the overall confidentiality disclosure level at that particular time. The total number of detected violations, noDisclosures, is calculated and reported. An increased number of detected disclosures means that the agent confidentiality violation level is high (low agent quality) and fewer detected disclosures means that the agent confidentiality violation level is low (high agent quality).

5. Experiment Procedure

The objective of this experiment was to detect agent data confidentiality violations in the CNS such that its effectiveness can be evaluated. The confidentiality requirements and the possible violations defined in a previous work [51] were utilized. Next, the RVQA implementation within the CNS agent model was executed in order to monitor each agent's activities. Upon execution, the monitoring process to check the reachability of critical states at the individual, social, and content levels was performed. When the critical states were reached, the confidentiality statuses were assessed based on the defined agent confidentiality metrics and checking rules. The detected data confidentiality violations were classified and the total number of detected data confidentiality violations was reported. Finally, the effectiveness of the confidentiality violation detection was evaluated using the evaluation metrics. The RVQA procedures in the CNS and CNS Searching were implemented within the executed CNS for comparison. The RVQA data confidentiality violation detection results were compared to the confidentiality violations that were found using CNS Searching. The following subsections explain the RVQA and CNS Searching procedures in detail.

5.1. RVQA Procedures for Detecting Data Confidentiality Violations

The data confidentiality violation detections were performed for each executed single agent at the individual, social, and content levels using the proposed RVQA. The RVQA procedures kept track of the number of violations detected at each level and calculated the total number of all detected violations as follows:

1. For a single agent at the individual level, the RVQA obtained the number of individual data confidentiality violations, $RVQASingleAgentIndividualDCV$, by keeping track of the number of the agent accounts that disclosed its identity and location to the public, as shown in Equation (1):

$$RVQASingleAgentIndividualDCV = noDisclosedIdentity + noDisclosedLocation \quad (1)$$

2. For a single agent at the social level, the RVQA calculated the number of social data confidentiality violations, $RVQASingleAgentSocialDCV$, by including the number of established connections to the online social network. Equation (2) was used for the calculation:

$$RVQASingleAgentSocialDCV = noDisclosedSocialInformation \quad (2)$$

3. For a single agent at the content level, the RVQA calculated the number of content data confidentiality violations, $RVQASingleAgentContentDCV$, by including the amount of contents disclosed to the public. Equation (3) was used for the calculation:

$$RVQASingleAgentContentDCV = noDisclosedContent \quad (3)$$

4. The total number of data confidentiality violations detected by the RVQA from individual, social, and content levels for all agents in the executed CNS model, $RVQATotalDetectedDCV$, was calculated using Equation (4). In this experiment, x is the agent identification and n is the number of agents executed in CNS.

$$RVQATotalDetectedDCV = \sum_{x=1}^n (RVQASingleAgentIndividualDCVx + RVQASingleAgentSocialDCVx + RVQASingleAgentContentDCVx) \quad (4)$$

5.2. CNS Searching Procedures

The data confidentiality violation detections were also performed for each executed single agent and for all agents in the CNS using CNS Searching. The CNS Searching procedures kept track of the number of data confidentiality violations detected and calculated the total number of all detected violations as follows:

1. For a single agent at the individual level, CNS Searching obtained the number of individual data confidentiality violations by keeping track of the agent identity and locations that are exposed to the public, as shown in Equation (5):

$$SearchingSingleAgentIndividualDCV = noDisclosedIdentity + noDisclosedLocation \quad (5)$$

2. For a single agent at the social level, CNS Searching did not obtain any social data confidentiality violation as it did not consider the connection to a social network as a condition for data confidentiality violations.
3. For a single agent at the content level, CNS Searching did not detect any social data confidentiality violation as it did not consider publicly published contents as a condition for data confidentiality violations.
4. The total number of data confidentiality violations detected by CNS Searching for all agents in the executed CNS model, $SearchingTotalDetectedDCV$, was calculated using Equation (6). In this experiment, x is the agent identification and n is the number of agents executed in CNS.

$$SearchingTotalDetectedDCV = \sum_{x=1}^n (SearchingSingleAgentIndividualDCVx) \quad (6)$$

6. Results and Discussion

During the experiment, each agent's active account was checked to identify whether the agent's data were disclosed or protected throughout the execution period. The disclosed identity, location, social connections, and published contents visible to the public were classified as detected agent data confidentiality violations. The detection results were presented in the form of time graphs for the first 24 h of the CNS execution.

6.1. Data Confidentiality Violations Detection Results

Figure 12 shows the number of disclosed agent identities and locations as compared to the total number of registered active accounts throughout the execution period. This time graph represents the agent data confidentiality violation detection results at the individual level. For each agent account, the agent anonymity could be either a real identity or anonymous. At the same time, the agent's presence visibility was either visible (location was disclosed) or invisible (location was protected). At the end of the 24-h execution, out of all the registered agent accounts, about 20% of the agent identities and 20% of agent locations had been disclosed to the public. According to the defined requirements, disclosing the agent identity, location, or both to the public violates agent data confidentiality. These information disclosures to the public were due to agent preferences that revealed the agent identity and made the agent's presence (location) visible to the public during the run-time.

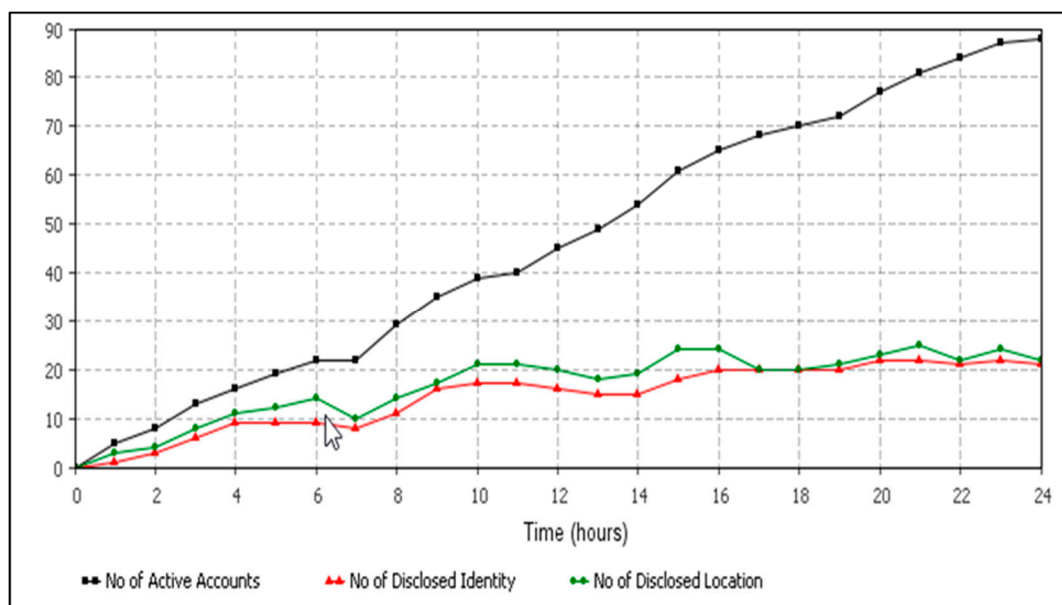


Figure 12. Total number of active accounts and the number of detected disclosed identities and disclosed locations versus time (hours).

Next, Figure 13 displays the amount of disclosed social information. This time graph represents the result of the agent data confidentiality violations detected at the social level. For this second data confidentiality level, more than 70% of agents' social information were disclosed to the public via connection to online social networking during the run-time. Disclosing social information to the public is a data confidentiality violation since it risks the disclosure of personal information, such as a real name, address, and relationship to the public [53].

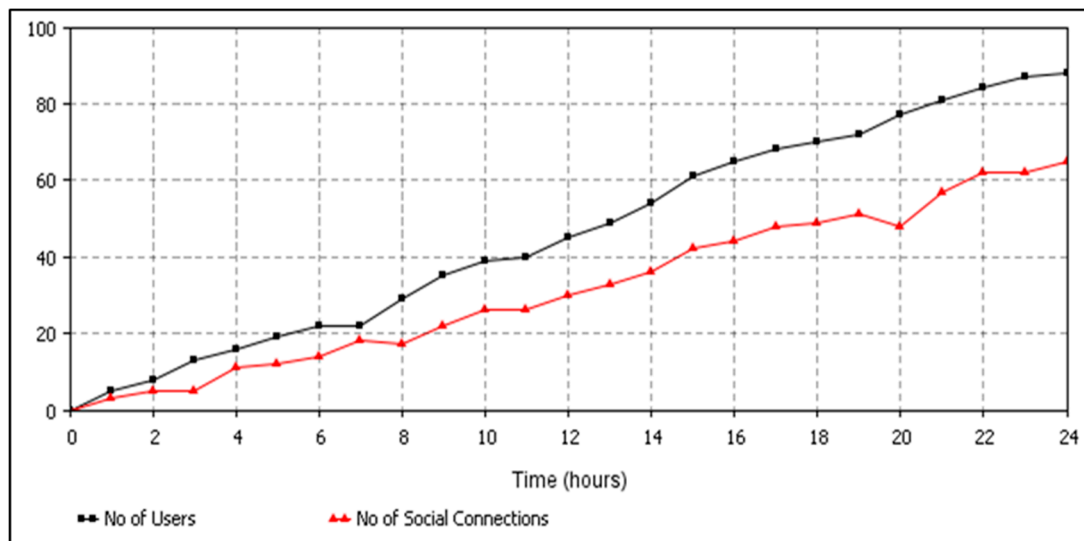


Figure 13. Total number of active accounts and the number of detected social connections versus time (hours).

Finally, Figure 14 presents the amount of disclosed contents. This time graph represents the result of agent data confidentiality violations detected at the content level. At this third data confidentiality level, about 80% of shared contents were disclosed to the public. These content disclosures occurred when agents shared public contents, such as traffic reports. These types of disclosures reveal an agent's account and risk other disclosures, such as their identity and location.

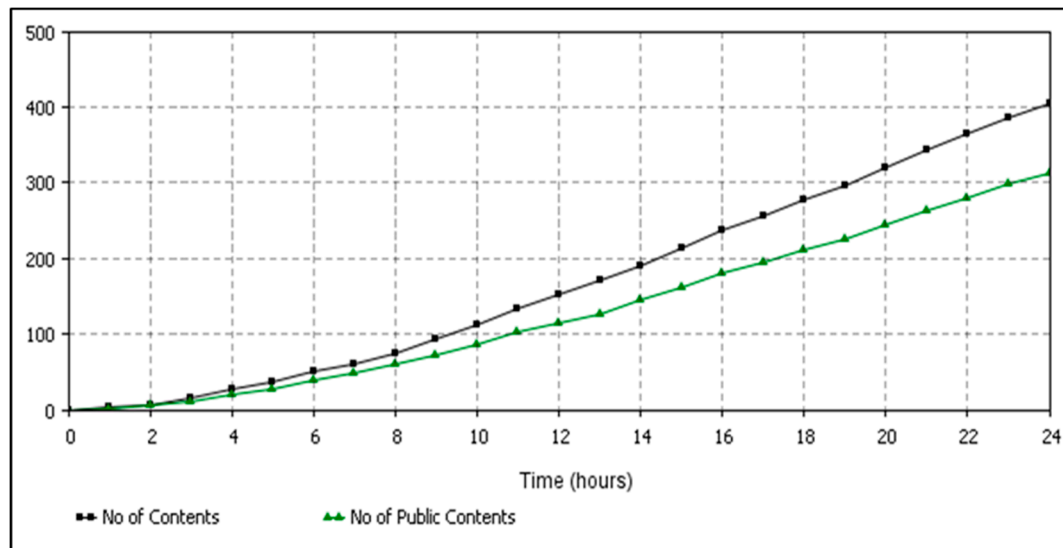


Figure 14. Total number of published contents and the number of detected disclosed public contents versus time (hours).

6.2. Comparison of Data Confidentiality Violation Detection Results

This section compares the results of the agent data confidentiality violation detection and classification using the RVQA against the detection using the CNS Searching process explained in Sections 5.1 and 5.2, respectively. The total numbers of data confidentiality violations detected by both methods throughout the CNS execution period are shown in Figure 15. At the end of the 24 h of execution, the RVQA detected a 72.45% higher number of violations as compared to the CNS Searching process. The total number of violations detected by RVQA was 137,373, while 37,851 violations

were detected by the CNS Searching process. The result shows that the implementation of the AQM framework and the RVQA process helped to detect about 70% more agent quality violations that occur during the execution of CNS as compared to the detection performed using the CNS Searching process. One of the main reasons is that more agent data confidentiality requirements and rules can be defined and implemented via the AQM framework and RVQA process. As a result, more agent data confidentiality violation conditions can be detected.

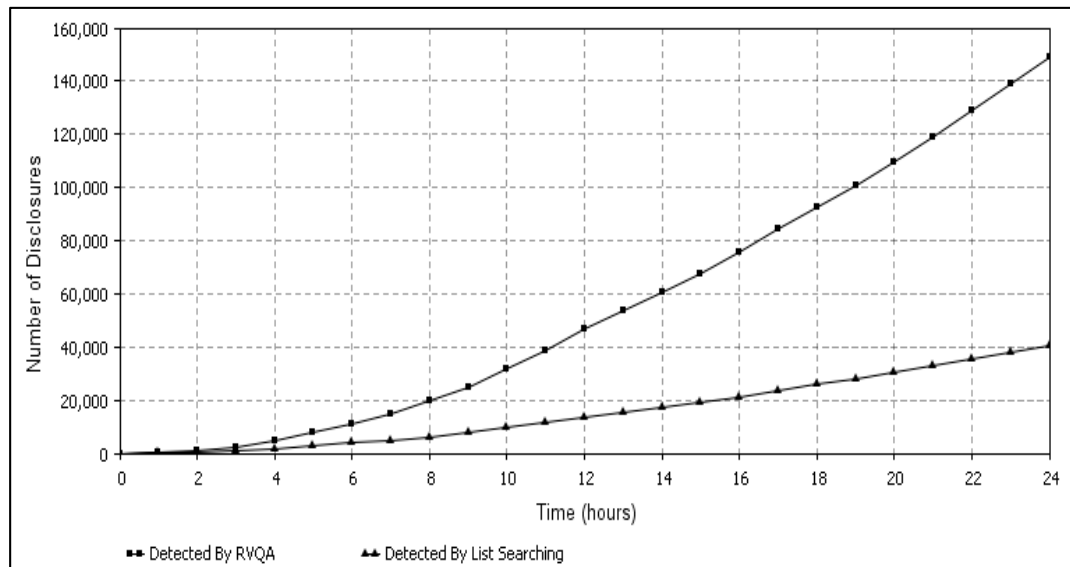


Figure 15. Number of detected data confidentiality violations in the CNS by the RVQA and CNS Searching.

6.3. Results Validation Using Evaluation Metrics

The total number of RVQA data confidentiality violation detections was evaluated using confidence and sensitivity metrics [54–56]. Confidence (precision) is the percentage of the detected violations from the overall detection result that are correct (when compared to the known data confidentiality violations). Sensitivity (recall) is the percentage of the detections from the overall known violations that are detected and correct (when compared to the known data confidentiality violations). To use the metrics, the expected total number of the detected data confidentiality violations for the experiment was calculated using Equation (7). In this experiment, x is the agent identification and n is the number of agents executed in CNS.

$$\text{ExpectedTotalDetectedDCV} = \sum_{x=1}^n (\text{noDisclosedContact}_x + \text{noDisclosedIdentity}_x + \text{noDisclosedLocationAtReal}_x + \text{noDisclosedSocialInformation}_x + \text{noDisclosedContent}_x) \quad (7)$$

Figure 16 shows the confidence of the RVQA results throughout the 24 h of the CNS execution. From the start, the graph shows a highly increased confidence percentage. The graph then gradually increases and stays constant above 95% as more than 95% of the subsequent detected data confidentiality violations were known violations that were expected to be detected. However, during the classification of violations, RVQA managed to include violations that were not in the list of expected violations. For example, RVQA classified all the disclosed locations as data confidentiality violations, while the list of expected violations only considered location disclosures as data confidentiality violations when the agent identity was also disclosed. The confidence percentage was expected to stay constant. This result, however, can be improved upon if the data confidentiality requirements and checking rules at the AQM definition phase are refined in order for all the data confidentiality detections to become the correct or expected detections (when compared to the known data confidentiality violations).

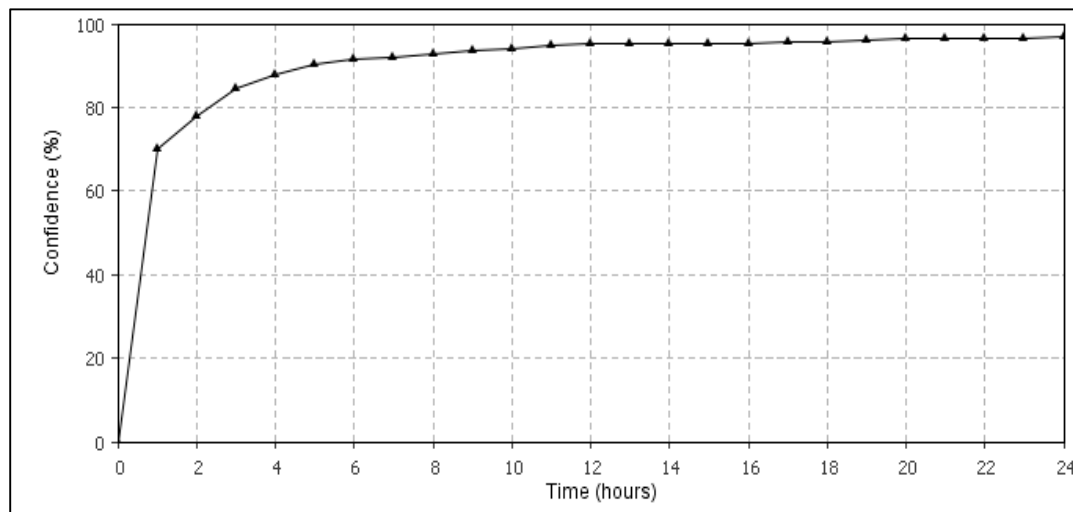


Figure 16. Confidence of the data confidentiality violations detected using RVQA.

Figure 17 shows the sensitivity of the RVQA detection results throughout the 24 h of the CNS execution. From the start, the graph shows very a high sensitivity percentage. The graph then stays constant, very close to 90%. This means that almost 90% of the violations that were expected to be detected had been correctly detected. The sensitivity or recall percentage was expected to stay constant. However, improvement may be made in the future where new data confidentiality requirements and checking rules at the AQM definition phase may be defined in order for other data confidentiality conditions, such as the disclosure of an agent contact, can be checked and the violations that may occur can be detected.

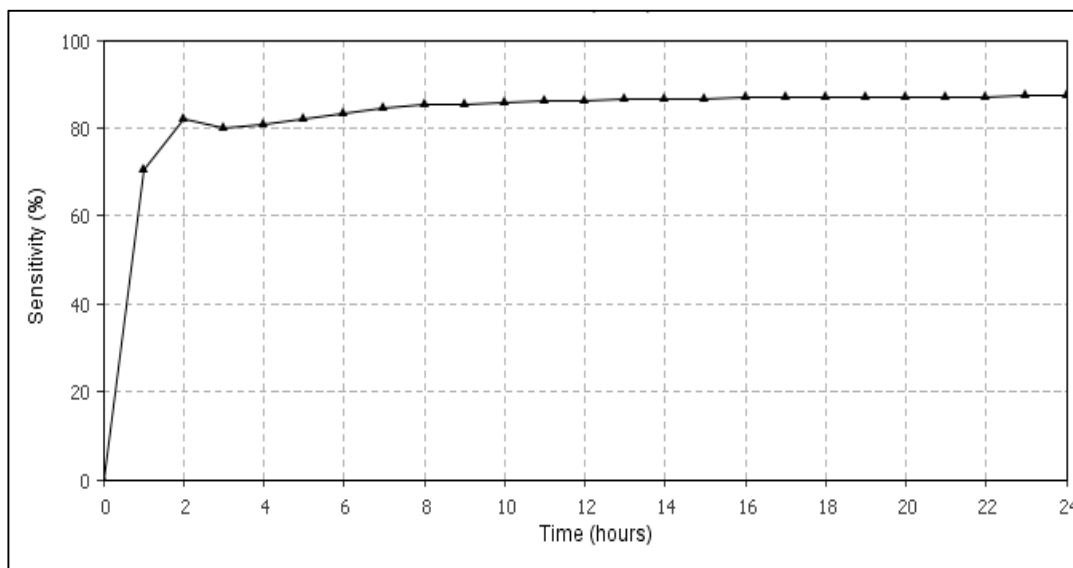


Figure 17. Sensitivity of the data confidentiality violations detected using RVQA.

The above experiment was repeated and the results at the end of the 24 h of each experiment are presented in Table 4. The table displays the results of the detected data confidentiality violations for the three repeated experiments at all levels of agent data confidentiality. The results show that the RVQA process managed to detect the expected data confidentiality violations at an average confidence between 84% to 90% and average sensitivity in the 79% to 83% range. It also shows that the RVQA process managed to correctly detect data confidentiality violations at an average confidence (precision) of 84% to 90% out of the total number of detections. In other words, the average confidence shows

that from 100 detections, 84 to 90 of the detections were correct detections (when checked with known violations that were expected to be detected). Therefore, the confidence of the RVQA was very high (close to 90%). On the other hand, the average sensitivity (recall) of the RVQA process was 79% to 83%. The average sensitivity shows that from 100 known violations that were expected to be detected, 79 to 83 detections managed to be detected. The number of detected confidentiality violations using the RVQA represents the agent confidentiality level and can be used to alert agents when the confidentiality violation level at a particular time is high.

Table 4. Total agent data confidentiality violations detection results.

Experiment	# Expected Data Confidentiality Violations	# Detected Using RVQA	Average Confidence of RVQA	Average Sensitivity of RVQA
1	152,628	137,373	89.72%	80.74%
2	180,896	161,985	88.14%	79.30%
3	135,692	122,489	84.57%	82.78%

By analyzing the agent data confidentiality violations detection results presented, improvement of the results can be performed. New data confidentiality requirements and checking rules to produce more accurate violation detections can be defined. Next, more comprehensive agent-contextual information for assessing the data confidentiality violation conditions could be gathered. The existing agent data confidentiality metrics and checking rules that do not contribute toward achieving the objective of detecting agent data confidentiality violations could be removed. Finally, the existing checking rules could be refined based on new agent data confidentiality requirements and the availability of the supporting contextual information for verifying and assessing agent data confidentiality.

The possible theoretical implications of this study include the introduction of a framework that focuses on properties assessment and has the ability for improvement in each process cycle. Smart city initiatives normally involve a significant budget, a wide range of resources, and a long-term commitment. Thus, the framework could benefit smart city developers and decision-makers analyzing their applications when trying to detect the violations of quality properties at an early stage, prior to the real development. The AQM framework would be the suitable framework to be adapted to manage various types of smart city initiatives and projects. Using the AQM framework, standardized indicators for smart city properties, i.e., smartness, sustainability, and resiliency, can be assessed from time to time.

From the analysis of previous works, verification of a system is a technique to prove the correctness of a system and to check the functional requirements of a system. This present study, however, has several findings. First, it shows that the assessment and verification stage can be used not only to verify the correctness of specified requirements, but also to detect the violations of the requirements. Second, the verification process can also be used to check not only the functional requirements, but also the non-functional requirements, i.e., quality properties of systems and applications. Another new development is the improvement component in the AQM framework that can be used to recommend a new approach and solutions to continuously enhance the systems along the design and implementation process. The limitation of the present study is that it should have used enhanced reasoning algorithms and enriched rules to increase the confidence and sensitivity levels. Efforts to improve the present study can be considered as potential future work.

7. Conclusions and Future Work

This paper studied the issues of agent quality properties in smart city applications and proposed a solution to detect violations of the properties. Based on the identified problem, the presented solution was the AQM framework that was proposed by adapting the TDQM principles to manage the identified quality properties. The framework includes four phases that are: definition and specification, modeling and execution, verification and assessment, and analysis and improvement. The main

research contribution was the use of the AQM phases to manage and to perform run-time verification and quality assessment of agent quality properties by considering dynamic contextual information in smart city applications. The outcome of the solution was the detection and classification of agent quality violations that occurred during the run-time. The AQM framework was implemented in an agent-based CNS model as a case study. The reported results show that the AQM framework with the run-time verification and quality assessment process managed to detect agent data confidentiality violations that occurred.

Overall, this work complements the existing agent quality management works and smart city quality properties solutions by focusing on management during run-time. The framework is also general enough to be used for managing any quality properties, such as privacy or confidentiality. The proposed solution presented in this paper intended to organize the management of agent quality properties and how quality requirements in smart cities are to be fulfilled. The quality assessment outputs are the number of classified agent quality statuses at certain time during an execution period. In the future, the outputs are useful for processing by other applications from different research fields, such as machine learning and big data analysis. The rules and algorithms can also be further upgraded by implementing more sophisticated techniques, such as fuzzy logic. Theoretically, by adapting these advanced techniques, the classification and violation detections' effectiveness can be improved in the future.

Author Contributions: Conceptualization, N.A.B., O.K., A.S.; methodology, A.S., and N.A.B.; software, N.A.B. and A.S.; validation, A.S., N.A.B. and O.K.; formal analysis, A.S., N.A.B.; investigation, N.A.B.; resources, O.K., and A.S.; data curation, N.A.B., O.K. and A.S.; writing—original draft preparation, N.A.B. and A.S.; writing—review and editing, N.A.B., O.K., and A.S.; visualization, N.A.B., A.S.; supervision, O.K., and A.S.; project administration, A.S., and O.K.; funding acquisition, O.K. and A.S.

Funding: This research has been funded by Universiti Teknologi Malaysia (UTM) under Research University Grant Vot-20H04, Malaysia Research University Network (MRUN) Vot 4L876, and the Fundamental Research Grant Scheme (FRGS) Vot 5F073 supported under the Ministry of Education Malaysia. The work is partially supported by the SPEV project, University of Hradec Kralove, FIM, Czech Republic (ID: 2103-2019), “Smart Solutions in Ubiquitous Computing Environments” and by project of excellence 2019/2205, Faculty of Informatics and Management, University of Hradec Kralove.

Acknowledgments: We are also grateful for the support of Sebastien Mambou and Michal Dobrovolny in consultations regarding application aspects. The APC was funded by project of excellence 2019/2205, Faculty of Informatics and Management, University of Hradec Kralove.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Eckhoff, D.; Wagner, I. Privacy in the Smart City—Applications, Technologies, Challenges, and Solutions. *IEEE Commun. Surv. Tutor.* **2017**, *20*, 489–516. [\[CrossRef\]](#)
2. Feng, W.; Yan, Z.; Zhang, H.; Zeng, K.; Xiao, Y.; Hou, Y.T. A Survey on Security, Privacy and Trust in Mobile Crowdsourcing. *IEEE Int. Things J.* **2018**, *5*, 2971–2992. [\[CrossRef\]](#)
3. Touq, A.B.; Ijeh, A. Information Security and Ecosystems in Smart Cities: The Case of Dubai. *IJISCC* **2018**, *9*, 16. [\[CrossRef\]](#)
4. Loper, M.L. Situational Awareness in Megacities. In *Technology and the Intelligence Community: Challenges and Advances for the 21st Century*; Kosal, M.E., Ed.; Springer International Publishing: Cham, Switzerland, 2018; pp. 205–235. ISBN 978-3-319-75232-7. [\[CrossRef\]](#)
5. Leitão, P.; Karnouskos, S. *Industrial Agents: Emerging Applications of Software Agents in Industry*; Industrial Agents, Morgan Kaufmann, Elsevier Inc.: Amsterdam, The Netherlands, 2015.
6. Bakar, N.A.; Selamat, A. Runtime Verification and Quality Assessment for Privacy Violations Detection in Social Networking System. In *Frontiers in Artificial Intelligence and Applications*; IOS Press: Amsterdam, The Netherlands, 2016; Volume 286, pp. 346–357.
7. García-Magariño, I.; Lacuesta, R. ABS-SmartPriority: An Agent-Based Simulator of Strategies for Managing Self-Reported Priorities in Smart Cities. *Wirel. Commun. Mob. Comput.* **2017**, *9*, 7254181. [\[CrossRef\]](#)

8. Hu, J.; Lin, H.; Guo, X.; Yang, J. DTCS: An Integrated Strategy for Enhancing Data Trustworthiness in Mobile Crowdsourcing. *IEEE Internet Things J.* **2018**, *5*, 4663–4671. [CrossRef]
9. Zhang, K.; Ni, J.; Yang, K.; Liang, X.; Ren, J.; Shen, X.S. Security and Privacy in Smart City Applications: Challenges and Solutions. *IEEE Commun. Mag.* **2017**, *55*, 122–129. [CrossRef]
10. Bakar, N.A.; Selamat, A. Agent systems verification: Systematic literature review and mapping. *Appl. Intell.* **2018**, *48*, 1251–1274. [CrossRef]
11. Sabater, J.; Sierra, C. REGRET: A reputation model for gregarious societies. In Proceedings of the 4th Int. Workshop on Deception, Fraud and Trust in Agent Societies, Catalonia, Spain, 28 May–1 June 2001; Volume 73, pp. 194–195. [CrossRef]
12. Carbo, J.; Molina, J.M.; Davila, J. Trust Management Through Fuzzy Reputation. *Int. J. Coop. Inf. Syst.* **2003**, *12*, 135–155. [CrossRef]
13. Qureshi, B.; Min, G.; Kouvatso, D. Collusion detection and prevention with FIRE+ trust and reputation model. In Proceedings of the 10th IEEE International Conference on Computer and Information Technology, Bradford, UK, 29 June–1 July 2010; pp. 2548–2555. [CrossRef]
14. Nusrat, S.; Vassileva, J. Recommending services in a trust-based decentralized user modeling system. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7138 LNCS, pp. 230–242. ISBN 9783642285080.
15. Sadra, A.; Samira, S. A trust-based service suggestion system using human plausible reasoning. *Appl. Intell.* **2014**, *41*, 55–75.
16. Yu, E.; Cysneiros, L.M. Designing for privacy in a multi-agent world. In *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2631, pp. 209–223. ISSN 0302-9743.
17. Piolle, G.; Demazeau, Y.; Caelen, J. Privacy Management in User-Centred Multi-agent Systems. In Proceedings of the 7th International Workshop on Engineering Societies in the Agents World VII (ESAW'07), Dublin, Ireland, 6–8 September 2006; Volume 4457, pp. 354–367, ISBN 9783540755241.
18. Krupa, Y. Privacy as Contextual Integrity in Decentralized Multi-Agent Systems. Ph.D. Thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, Saint-Etienne, France, 2012. Available online: <http://www.theses.fr/2012EMSE0657> (accessed on 29 October 2019).
19. Hussein, Z.M.; Zarandi, M.F.; Hussein, S.M. Trust evaluation for buyer-supplier relationship concerning fuzzy approach. In Proceedings of the 2015 Annual Conference of the North American Fuzzy Information Processings of the Society (NAFIPS) Held Jointly with 2015 5th World Conference on Soft Computing (WConSC), Redmond, WA, USA, 17–19 August 2015; pp. 1–6.
20. Aref, A.; Tran, T. FTE: A Fuzzy Logic Based Trust Establishment Model for Intelligent Agents. In Proceedings of the 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Singapore, 6–9 December 2015; Volume 2, pp. 133–138.
21. Aref, A.; Tran, T. Modeling Trust Evaluating Agents: Towards a Comprehensive Trust Management for Multi-agent Systems. In Proceedings of the Workshops at the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
22. Rishwaraj, G.; Ponnambalam, S.G.; Kiong, L.C. An efficient trust estimation model for multi-agent systems using temporal difference learning. *Neural Comput. Appl.* **2016**, *28*, 461–474. [CrossRef]
23. Singh, M.M.; Chin, T.Y. Hybrid Multi-faceted Computational Trust Model for Online Social Network (OSN). *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 1–11. [CrossRef]
24. Gharib, M.; Giorgini, P.; Mylopoulos, J. Towards an Ontology for Privacy Requirements via a Systematic Literature Review. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2017; Volume 10650 LNCS, pp. 193–208. ISBN 9783319699035. [CrossRef]
25. Maglaras, L.A.; Al-Bayatti, A.H.; He, Y.; Wagner, I.; Janicke, H. Social Internet of Vehicles for Smart Cities. *J. Sens. Actuator Netw.* **2016**, *5*, 3. [CrossRef]
26. Hassan, N.H.; Rahim, F.A. The Rise of Crowdsourcing Using Social Media Platforms: Security and Privacy Issues. *Pertanika J. Sci. Technol.* **2017**, *25*, 79–88.

27. Khan, Z.; Pervez, Z.; Ghafoor, A. Towards Cloud Based Smart Cities Data Security and Privacy Management. In Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, London, UK, 8–11 December 2014; pp. 806–811. [\[CrossRef\]](#)
28. Moffat, S.; Hammoudeh, M.; Hegarty, R. A Survey on Ciphertext-Policy Attribute-based Encryption (CP-ABE) Approaches to Data Security on Mobile Devices and Its Application to IoT. In Proceedings of the International Conference on Future Networks and Distributed Systems, ICFNDS '17, Cambridge, UK, 19–20 July 2017; ISBN 978-1-4503-4844-7. [\[CrossRef\]](#)
29. Hernández-Ramos, J.; Bernabe, J.; Moreno, M.; Skarmeta, A. Preserving Smart Objects Privacy through Anonymous and Accountable Access Control for a M2M-Enabled Internet of Things. *Sensors* **2015**, *15*, 15611–15639. [\[CrossRef\]](#) [\[PubMed\]](#)
30. Shehada, D.; Yeun, C.Y.; Zemerly, M.J.; Al-Qutayri, M.; Al Hammadi, Y. Secure Mobile Agent Protocol for Vehicular Communication Systems in Smart Cities. In *Information Innovation Technology in Smart Cities*; Springer: Singapore, 2017.
31. Zhang, F.; Lee, V.E.; Jin, R.; Garg, S.; Choo, K.K.R.; Maasberg, M.; Dong, L.; Cheng, C. Privacy-aware smart city: A case study in collaborative filtering recommender systems. *J. Parallel Distrib. Comput.* **2018**, *127*, 145–159. [\[CrossRef\]](#)
32. Joy, J.; McGoldrick, C.; Gerla, M. *Mobile Privacy-Preserving Crowdsourced Data Collection in the Smart City*; Scientific Challenges in Data and Event-driven Smart City Service and Applications (SDESS 2016); Cornell University: Irvine, CA, USA, 2016; pp. 1–6.
33. Barata, J.; Cunha, P.R. Synergies between quality management and information systems: A literature review and map for further research. *Total Qual. Manag. Bus. Excell.* **2017**, *28*, 282–295. [\[CrossRef\]](#)
34. Srma, S.; Wannapiroon, P.; Nilsook, P. Design of Total Quality Management Information System (TQMIS) for Model School on Best Practice. *Procedia Soc. Behav. Sci.* **2015**, *174*, 2160–2165. [\[CrossRef\]](#)
35. Wang, R.Y. A product perspective on total data quality management. *Commun. ACM* **1998**, *41*, 58–65. [\[CrossRef\]](#)
36. Agarwal, N.; Yiliyasi, Y. Information quality challenges in social media. In Proceedings of the 15th International Conference on Information Quality (ICIQ-2010), Little Rock, AR, USA, 12–14 November 2010; pp. 234–248.
37. Idris, N.; Ahmad, K. Managing Data Source quality for data warehouse in manufacturing services. In Proceedings of the 2011 International Conference on Electrical Engineering and Informatics, Bandung, Indonesia, 17–19 July 2011; pp. 1–6. [\[CrossRef\]](#)
38. Moges, H.T.; Dejaeger, K.; Lemahieu, W.; Baesens, B. A multidimensional analysis of data quality for credit risk management: New insights and challenges. *Inf. Manag.* **2013**, *50*, 43–58. [\[CrossRef\]](#)
39. Endler, G.; Schwab, P.K.; Wahl, A.M.; Tenschert, J.; Lenz, R. An Architecture for Continuous Data Quality Monitoring in Medical Centers. In *Studies in Health Technology and Informatics*; US National Library of Medicine National Institutes of Health: Bethesda, MD, USA, 2015; pp. 852–856. [\[CrossRef\]](#)
40. AnyLogic. AnyLogic: Multimethod Simulation Software. 2017. Available online: <http://www.anylogic.com/> (accessed on 29 October 2019).
41. Bellifemine, F.L.; Caire, G.; Greenwood, D. *Developing Multi-Agent Systems with JADE*; John Wiley & Sons, Ltd.: West Sussex, UK, 2007.
42. Cougaar Software Inc. Cougaar Agent Architecture. 2019. Available online: <http://www.cougaarsoftware.com/> (accessed on 29 October 2019).
43. Bordini, R.H.; Hübner, J.F.; Wooldridge, M. *Programming Multi-Agent Systems in Agent Speak Using Jason*; John Wiley & Sons, Ltd.: West Sussex, UK, 2017.
44. Botía, J.A.; Gómez-Sanz, J.J.; Pavón, J. Intelligent Data Analysis for the Verification of Multi-Agent Systems Interactions. In *International Conference on Intelligent Data Engineering and Automated Learning (Lecture Notes in Computer Science)*; Corchado, E., Yin, H., Botti, V., Fyfe, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1207–1214.
45. Bakar, N.A.; Selamat, A. Analyzing model checking approach for multi agent system verification. In Proceedings of the 2011 5th Malaysian Conference in Software Engineering, MySEC 2011, Johor Bahru, Malaysia, 3–14 December 2011; pp. 95–100, ISBN 9781457715310. [\[CrossRef\]](#)
46. Naumann, F. *Quality-Driven Query Answering for Integrated Information Systems*; Springer: Berlin/Heidelberg, Germany, 2002.

47. Naumann, F. *Data Profiling Revisited*; ACM SIGMOD Record: New York, NY, USA, 2014; Volume 42, pp. 40–49.
48. Borshchev, A. Simulation Modeling with AnyLogic: Agent Based. *Discrete Event Syst. Dynam. Methods* **2013**, 225–269.
49. Lindamood, J.; Heatherly, R.; Kantarcioglu, M.; Thuraisingham, B. Inferring private information using social network data. In Proceedings of the 18th International Conference on World wide web WWW 09, Madrid, Spain, 20–24 April 2009; pp. 1145–1146. [[CrossRef](#)]
50. Zheleva, E.; Getoor, L. To Join or Not to Join: The Illusion of Privacy in Social Networks with Mixed Public and Private User Profiles. Security. In Proceedings of the 18th International Conference on World Wide Web, Madrid, Spain, 20–24 April 2009; Volume 7, pp. 531–540. [[CrossRef](#)]
51. Bakar, N.A.; Selamat, A. Detection of data confidentiality violations using Runtime Verification and Quality Assessment. In Proceedings of the 2016 2nd International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR), Bangi, Malaysia, 23–24 August 2016; pp. 22–26.
52. ISO/IEC 10181-5:1996. Information Technology—Open Systems Interconnection—Security Frameworks for open Systems: Confidentiality Framework. Last reviewed and conformed in 2006. Available online: <https://www.iso.org/standard/24329.html> (accessed on 29 October 2019).
53. Waze. Waze: Privacy Issues. Available online: https://wiki.waze.com/wiki/Privacy_issues (accessed on 29 October 2019).
54. Ricardo, B.Y.; Berthier, R.N. *Modern Information Retrieval: The Concepts and Technology behind Search*; ACM Press Books; IEEE: New York, NY, USA, 2011.
55. Jacques, J.; Taillard, J.; Delerue, D.; Jourdan, L.; Dhaenens, C. MOCA-I: Discovering rules and guiding decision maker in the context of partial classification in large and imbalanced datasets. In *Learning and Intelligent Optimization*; Nicosia, G., Pardalos, P., Eds.; Springer Nature: Berlin/Heidelberg, Germany, 2013.
56. Pehcevski, J.; Piwowarski, B. *Evaluation Metrics*; Encyclopedia of Database Systems; Liu, L., Tamer Özsu, M., Eds.; Springer Nature: Berlin/Heidelberg, Germany, 2007.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).