SEMIoTICA - Security Scenarios Modeling for IoT-based Agriculture Solutions

Julio Ariel Hurtado[1], Leandro Antonelli[2], Santiago López[1], Adriana Gómez[3], Juliana Delle Ville[2], Frey Giovanny Zambrano[1], Andrés Solis[1,4], Marta Cecilia Camacho[5], Miguel Solinas[6], Gladys Kaplan[7], and Freddy Muñoz[8]

[1] IDIS, Universidad del Cauca, Colombia
`{ahurtado, santiagolopez94, freyzambrano, afsolis}@unicauca.edu.co.edu.co`
[2] Lifia - Facultad de Informática, Universidad Nacional de La Plata, Argentina
`{lanto,jdelleville}@lifia.info.unlp.edu.ar`
[3] Universidad Tecnologica de Pereira, Colombia
`adrianagomezr@utp.edu.co`
[4] Corporacion Universitaria Comfacauca, Colombia
[5] Institución Universitaria Colegio Mayor del Cauca, Colombia
`cecamacho@unimayor.edu.co`
[6] LARYC, FCEFyN, Universidad Nacional de Cordoba, Argentina
`miguel.solinas@unc.edu.ar`
[7] Universidad Nacional de La Matanza, Argentina
`gladyskaplan@gmail.com`
[8] Fundacion Universitaria de Popayan, Colombia
`lfreddyms@gmail.com`

Abstract. Agriculture is a vital human activity contributing to sustainable development. A few decades ago, the agricultural sector introduced the Internet of Things (IoT), playing a relevant role in precision and smart farming. IoT developments in farms require a lot of connected devices working cooperatively. It increases the vulnerability of IoT devices mostly because it lacks the necessary built-in security due to their constrained context and computational capacity. Additionally, storage and data processing connecting with edge or cloud servers are the reason for many security threats. To ensure that IoT-based solutions meet functional and non-functional requirements, particularly security, software companies should adopt a security-focused approach to their specification. This paper proposes a method for specifying security scenarios integrating requirements and architecture viewpoints in the context of the IoT in agriculture solutions. The method comprises four activities. First, the description of scenarios for the intended software. After that, scenarios with incorrect system use should be described. Then, these are translated into security scenarios using a set of rules. Finally, the security scenarios are improved. Additionally, this paper describes a preliminary validation of the approach, which software engineers in Argentina and Colombia performed. The results show that the approach proposed allows software engineers to define and analyze security scenarios in the IoT and agriculture contexts with good results.

Keywords: IoT · Quality Scenario · IoT Requirements · Smart Farming · Industry 4.0

1 Introduction

The International Telecommunication Union (ITU) defines IoT (Internet of the Things) as a "global infrastructure for the information society that provides advanced services through the connection of objects (physical and virtual) relying on the interoperability of current and future knowledge and communication technologies" [31]. According to [17] as an interconnected network, IoT contributes to making decisions based on the information collected, and its interaction does not need human intervention. The definition includes the concept of a Cyber-Physical system, which is a complex abstraction that requires a conceptual map [5] rather than a simple definition to state its concept.

Regarding software development, requirements analysis is a critical activity for defining software functionalities, attributes, and quality properties. This process is particularly different for software construction by using emergent technologies like IoT. Traditional software development practices must adapt to these new technologies and business contexts [17]. Requirements engineering involves collaboration between clients and development teams for incorporating the right features into the finished product[27]. Inconsistencies between initial requirements and the final product can lead to re-engineering processes, increasing the project scope and cost[26]. Requirements engineering works with both types of knowledge, explicit and tacit[1]. Tacit knowledge is difficult to communicate because experts and development teams often have different backgrounds and use distinct terminologies [20], making it challenging to elicit information from stakeholders.

Software products are defined by a set of functional and non-functional requirements. The latter is responsible for the software product's quality and is most frequently considered when developing an IoT system according to its specific application domain [17]. A way to specify software requirements is to describe use scenarios through storytelling techniques. This approach is effective because it is a way to incorporate details that are essential to provide a rich consolidation of knowledge. Scenarios employ natural language, allowing experts to use them without complex formalisms. This makes them highly effective in promoting communication and collaboration among diverse groups of experts [3].

The main challenges associated with these requirements in developing these products are limited processing and storage capacity, performance reliability, availability, accessibility, interoperability, security, privacy, scalability flexibility, and context awareness [17, 19]. In this line of thought, security is an aspect that is highly relevant in IoT-based software because it protects resources such as modules, code, and others from unauthorized access [19]. With scenarios, experts can describe various situations and work together to improve them, learning from one another in the process. This can be especially valuable when dealing with complex problems that require inputs from multiple perspectives. Overall, scenarios can be a powerful tool for fostering cooperation and achieving better outcomes in a wide range of domains.

The software architect must consider designing the whole system when stakeholders identify security concerns rather than adding security technologies in an ad-hoc manner [15]. As Bruce Schneier points out[25], security is a process and a chain that is only as strong as its weakest link. Therefore, software providers must adopt a security-centric approach to designing and developing IoT-based solutions that conform to functional and non-functional requirements like security [14].

The agricultural sector now requires data collection and advanced technologies to improve production while using limited resources. Sustainable agriculture can help preserve nature without compromising the needs of future enerations[6, 13]. The Food and Agriculture Organization (FAO) has identified population growth, resource scarcity, and degradation as key challenges. There is a need to increase efficiency, productivity, and quality in agrifood systems while protecting the environment[24]. To achieve this, new developments and technologies must be introduced to automate traditional farming methods and make farm labor more efficient. The Internet of Things (IoT) appears as technology to transform conventional processes [6, 13].

This paper proposes a scenario-based method for specifying the architecture's security aspects. The method is composed of four essential activities. The first activity consists in describing scenarios of the intended software application. The second activity consists in describing scenarios related to the previous ones but referring to incorrect usage of the application. The third activity consists in applying a set of rules to map attributes from the previous scenarios to the architecture scenarios. Finally, the four activity consists in describing the architectural scenarios in more detail. Moreover, this document describes a preliminary evaluation of the proposed approach. Given the security challenges facing IoT agriculture, the research question in his paper is: how adequately elicit security requirements in IoT- based smart agriculture solutions?

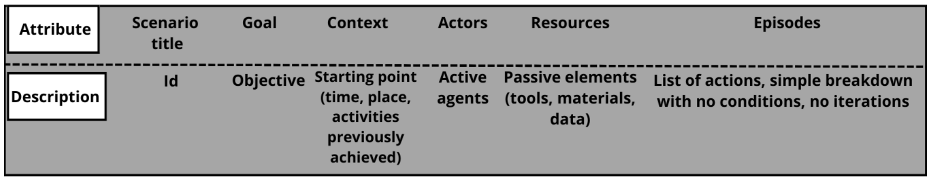
The paper is organized in the following way. Section 2 describes some background about the scenarios. Then, section 3 describes some related work. Section 4 details our contribution, which is the proposed approach. Section 5 describes the tool to support the proposed method. Section 6 presents the preliminary evaluation. Finally, Section 7 discusses some conclusions.

2 Background

This section describes two types of scenarios. First, it describes scenarios that focus on the functionality of a software application. Afterward, the section describes scenarios that focus on architectural security concerns.

2.1 Scenarios for describing functionality

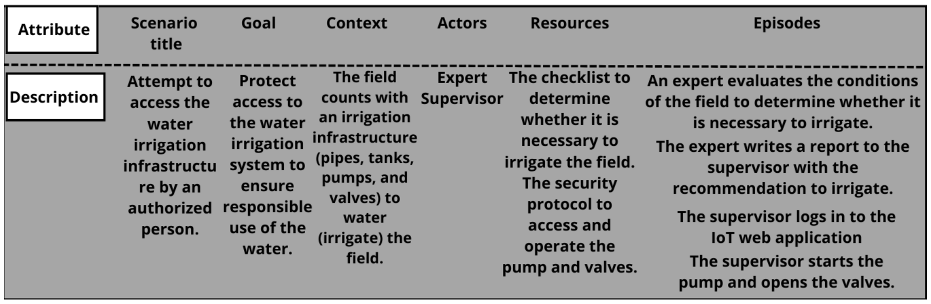
A scenario [3] is an artifact that describes situations (in the application or the software domain) using natural language. It describes a specific situation that arises in a certain context to achieve some goal. There is a set of steps (the episodes) to reach that goal. In the episodes, active agents as actors use materials, tools, and data as resources to perform some specific action. Although there are many templates to describe scenarios, this paper will use the scenarios proposed by Leite et al. [21]. Figure 1 summarizes the template.



Attribute	Scenario title	Goal	Context	Actors	Resources	Episodes
Description	Id	Objective	Starting point (time, place, activities previously achieved)	Active agents	Passive elements (tools, materials, data)	List of actions, simple breakdown with no conditions, no iterations

Fig. 1. Template for describing scenarios that focus on functionality.

Let's consider the following example that describes a scenario about how the irrigation system is activated. This task can be done in different ways regarding the technological infrastructure that the farm has. For example, an operator can manually start the irrigation by physically accessing the machine room with the pumps. In this situation, there is no IoT software application. This paper is going to focus on another scenario where a software application performs the activation of the pumps. For example, an agriculture expert evaluates the field conditions to determine whether irrigating is necessary and provides the information to the farm supervisor. Then, the supervisor activates the irrigation pipe through an IoT-based web application. Table 2 summarizes the situation.



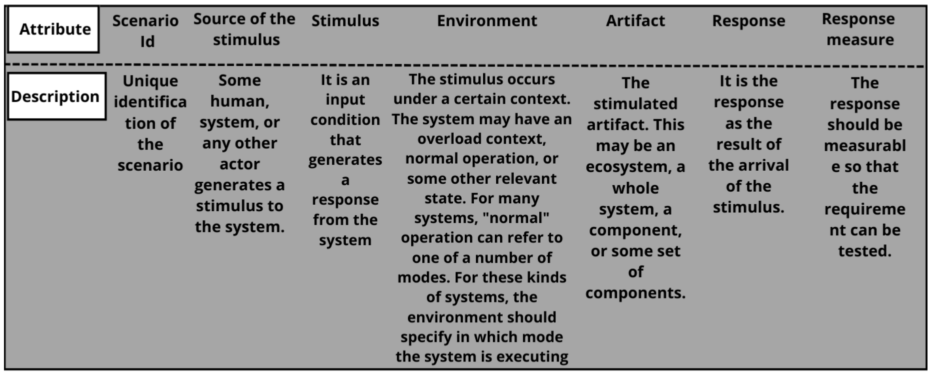
Attribute	Scenario title	Goal	Context	Actors	Resources	Episodes
Description	Attempt to access the water irrigation infrastructure by an authorized person.	Protect access to the water irrigation system to ensure responsible use of the water.	The field counts with an irrigation infrastructure (pipes, tanks, pumps, and valves) to water (irrigate) the field.	Expert Supervisor	The checklist to determine whether it is necessary to irrigate the field. The security protocol to access and operate the pump and valves.	An expert evaluates the conditions of the field to determine whether it is necessary to irrigate. The expert writes a report to the supervisor with the recommendation to irrigate. The supervisor logs in to the IoT web application The supervisor starts the pump and opens the valves.

Fig. 2. Authorized attempt to start the irrigation system.

The previous scenario describes an authorized person's legitimate use of the software application to activate the irrigation system. This scenario could be similar to Use Cases or User Stories [20]. Nevertheless, the software system can be vulnerable to hack attacks, where a malicious user desires to break into the web software application to start the irrigation system just for fun or to destroy the crop. This incorrect and harmful description of the software application is related to misuse cases [12].

2.2 Scenarios for describing architectural security concerns

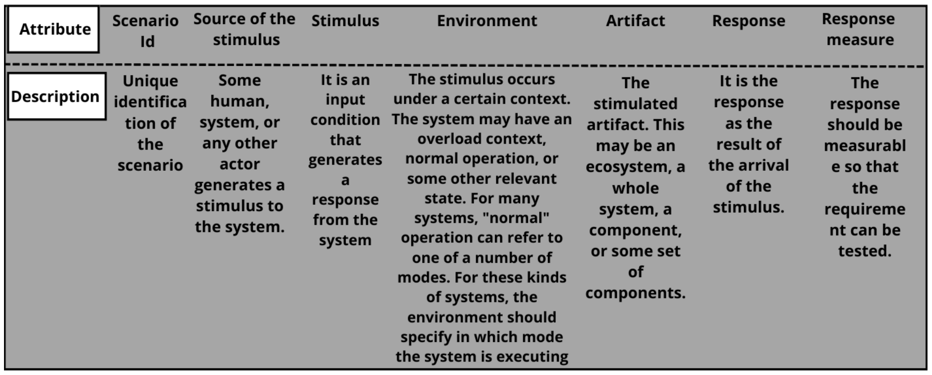
Software architecture is the designing process of the system's fundamental structure and organization to achieve specific quality attributes, which are the critical non-functional characteristics determining the system's overall effectiveness. Quality attributes are specified through quality scenarios, which define how the system should behave under various conditions. A Quality Attribute (QA) Scenario is a specific, testable scenario that demonstrates how a quality attribute requirement is satisfied. A QA scenario is typically structured with an id, a stimulus that triggers the interaction with the software application, the environment where the interaction occurs, the artifact affected, the response, and some quantitative description of the response. Table 3 summarizes the template.



Attribute	Scenario Id	Source of the stimulus	Stimulus	Environment	Artifact	Response	Response measure
Description	Unique identification of the scenario	Some human, system, or any other actor generates a stimulus to the system.	It is an input condition that generates a response from the system	The stimulus occurs under a certain context. The system may have an overload context, normal operation, or some other relevant state. For many systems, "normal" operation can refer to one of a number of modes. For these kinds of systems, the environment should specify in which mode the system is executing	The stimulated artifact. This may be an ecosystem, a whole system, a component, or some set of components.	It is the response as the result of the arrival of the stimulus.	The response should be measurable so that the requirement can be tested.

Fig. 3. Security scenario template.

Security refers to the system's capability to defend against danger, ensure its safety, and protect system data from unauthorized disclosure, modification, or destruction. Security involves protecting computer systems themselves through technical and administrative safeguards. Additionally, security can refer to the degree to which a particular security policy is enforced with some level of assurance. The three fundamental types of security concerns are confidentiality, integrity, and availability. Confidentiality refers to the protection of data and processes from unauthorized disclosure or access by individuals or entities that are not authorized to access it. Integrity refers to protecting data and processes from unauthorized modification, intentional or accidental. It includes ensuring that data is not tampered with or corrupted during storage, processing, or transmission. And availability refers to the protection of data and processes from denial of service attacks or other forms of disruption that can prevent authorized users from accessing or using them. This includes ensuring that systems are available and responsive when needed and that they can handle high levels of traffic or activity without becoming overloaded or crashing. Figure 4 describes an example of a security scenario that refers to the same situation of the requirement scenario described in Figure 2.



Attribute	Scenario Id	Source of the stimulus	Stimulus	Environment	Artifact	Response	Response measure
Description	Unique identification of the scenario	Some human, system, or any other actor generates a stimulus to the system.	It is an input condition that generates a response from the system	The stimulus occurs under a certain context. The system may have an overload context, normal operation, or some other relevant state. For many systems, "normal" operation can refer to one of a number of modes. For these kinds of systems, the environment should specify in which mode the system is executing	The stimulated artifact. This may be an ecosystem, a whole system, a component, or some set of components.	It is the response as the result of the arrival of the stimulus.	The response should be measurable so that the requirement can be tested.

Fig. 4. Security scenario example.

3 Related works

The complexity of IoT software applications is a concern identified several researchers. Thus, there are some proposals to deal with this complexity. Nguyen et al. [16] propose FRASAD, a model-driven software development framework to manage the complexity of Internet of Things (IoT) applications. Karaduman et al. [11] is another proposal to deal with the complexity. Their approach includes activities such as requirements development, domain-specific design, verification, simulation, analysis, calibration, deployment, code generation, and execution. Nevertheless, these proposals do not consider security, which is our main concern.

Some other approaches consider the security issue, but it is considered in terms of implementation, while our proposal considers the security in terms of the specification of requirements. Cardenas et al. [2] propose a process and a tool to apply formal methods in Internet of Things (IoT) applications using the Unified Modeling Language (UML). They have developed a plug-in tool to validate UML software models regarding the design of a secure software application. Slovenec et al. [28] present a taxonomy of security requirements to consider them when designing and implementing the software application. El-Gendy et al. [7] propose a security architecture to provide security enabled IoT services, and provide a baseline for security deployment. Sotoudeh et al. [29] proposes a model as a reference architecture to meet all security requirements.

There are some approaches that consider security in requirements, but they do not emphasize the way to specify security requirements precisely. Iqbal et al. [9] present a literature review about an In-Depth Analysis of IoT Security Requirements, but the work they present does not refer about how to specify them. Özkaya et al. [18] proposal deal with different non-functional requirements: security, scalability, and performance. And they try to balance the different requirements or decide which one to satisfy when there is a conflict. Carvalho et al, [4] also deals with conflicts, but their approach deals with non-functional Requirements.

Finally, Kammuller et al. [10] present an approach to specify security requirements for IoT applications. They combine a framework for requirement elicitation with automated reasoning to provide secure IoT for vulnerable users in healthcare scenarios. They map technical system requirements using high-level logical modeling. Then they perform an attack tree analysis. And finally, a security protocol analysis. Their work pays more attention to the tree analysis to identify the situation, while our approach pays attention to how to describe security requirements precisely.

4 Our approach

This section is organized in the following manner: firstly, we provide a description of the general approach, followed by a detailed explanation of each step.

4.1 Our approach in a nutshell

Our proposed approach consists of several steps. Firstly, we describe scenarios that outline the intended usage of the software. Next, we create scenarios that describe incorrect usage of the application in an attempt to exploit any vulnerabilities. We then establish rules for converting these scenarios into security scenarios. Lastly, we refine and improve the security scenarios. Figure 5 provides a summary of our approach.

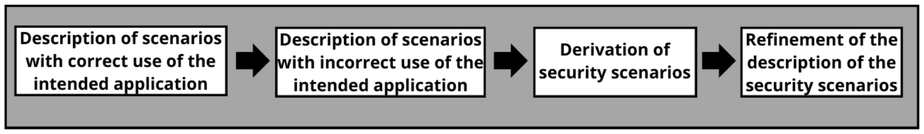

Description of scenarios with correct use of the intended application
Description of scenarios with incorrect use of the intended application
Derivation of security scenarios
Refinement of the description of the security scenarios

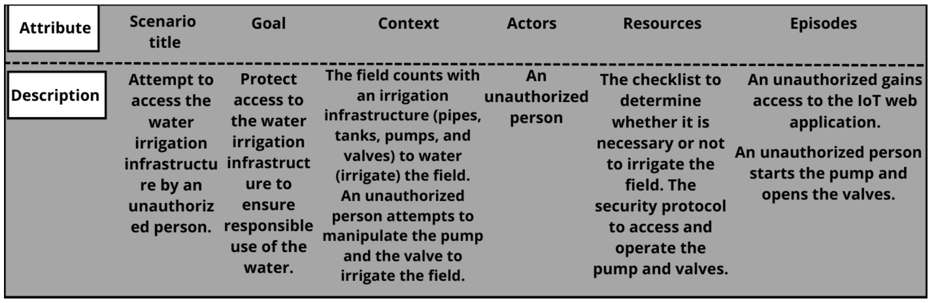
Fig. 5. Our approach in a nutshell.

4.2 Description of the scenarios with the correct use of the indented software application

This step describes the scenarios that focus on the correct use of the software application regarding security concerns. This step should be executed by a requirements engineer or analyst (or a group of them) that must interact with the experts of the domain (clients, users, and stakeholders in general) to capture the software application's requirements and specify scenarios. Those should describe the functionality of the intended software, and they should also consider security concerns. That is why the analyst eliciting and defining scenarios should have some background in security non-functional requirements to consider this concern in the specification. The result of this step is a set of scenarios that describes the functionality like the one described in Figure 2.

4.3 Description of the scenarios with the incorrect use of the indented software application

This step consists in analyzing the scenarios described in the previous step to find security issues and describing the ones that explode the problems and compromise the security of the software application. Ideally, this step should be done by the same requirement engineer (or group of them) that participated in the previous tasks. They should analyze every scenario in detail, and considering guides like the ones proposed by Gupta et al. [8] and Yazdinejad [32], they must describe scenarios of incorrect use of the software application. Basically, they should describe scenarios that explode possible vulnerabilities. For example, considering the scenario that describes the correct use of the software application to activate the irrigation system (Figure 2), the requirements engineer can determine that the access to the system (and therefore the access to the activation of the pumps) determine a security breach. Hence, the analyst describes a scenario where an unauthorized person gets access to the software application and, consequently to the irrigation infrastructure). Figure 6 describes the complete scenario.

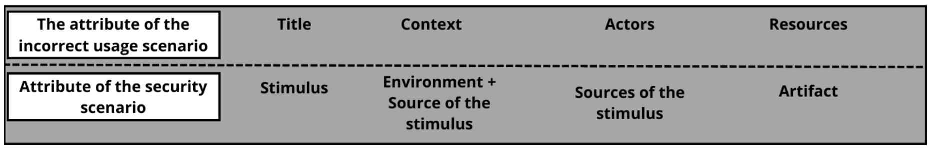


Attribute	Scenario title	Goal	Context	Actors	Resources	Episodes
Description	Attempt to access the water irrigation infrastructure by an unauthorized person.	Protect access to the water irrigation infrastructure to ensure responsible use of the water.	The field counts with an irrigation infrastructure (pipes, tanks, pumps, and valves) to water (irrigate) the field. An unauthorized person attempts to manipulate the pump and the valve to irrigate the field.	An unauthorized person	The checklist to determine whether it is necessary or not to irrigate the field. The security protocol to access and operate the pump and valves.	An unauthorized gains access to the IoT web application. An unauthorized person starts the pump and opens the valves.

Fig. 6. Unauthorized attempt to start the irrigation system.

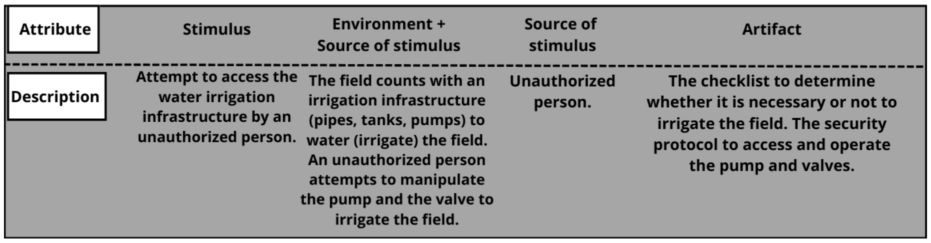
4.4 Derivation of security scenarios

This step describes a set of rules to map the information contained in a scenario that describes the incorrect use of the intended system to obtain a first draft of a scenario to describe security concerns. It is essential to mention that the scenario with incorrect usage will not provide complete information about the security scenario. The rules proposed will use only four attributes (title, context, actors, and resources), and this information will be used to fill four attributes to the security scenario (stimulus, environment, source of the stimulus, and artifact). Therefore, with this information, the following step can refine the security scenario. Figure 7 summarizes the mapping between attributes of both types of scenarios. Following the example of the scenario that describes the incorrect use of the software application described in Figure 6, the scenario obtained applying the proposed rules is the one described in Figure 8. It is important to mention that this scenario (the resulting from the mapping rules) needs to be refined in the following step, which is why this simple mapped security scenario is still far from the security scenario (like the one described in Figure 4).



The attribute of the incorrect usage scenario	Title	Context	Actors	Resources
Attribute of the security scenario	Stimulus	Environment + Source of the stimulus	Sources of the stimulus	Artifact

Fig. 7. Mapping rules between attributes of the incorrect and security scenarios.



Attribute	Stimulus	Environment + Source of stimulus	Source of stimulus	Artifact
Description	Attempt to access the water irrigation infrastructure by an unauthorized person.	The field counts with an irrigation infrastructure (pipes, tanks, pumps) to water (irrigate) the field. An unauthorized person attempts to manipulate the pump and the valve to irrigate the field.	Unauthorized person.	The checklist to determine whether it is necessary or not to irrigate the field. The security protocol to access and operate the pump and valves.

Fig. 8. Mapping rules between attributes of the incorrect and security scenarios.

4.5 Refinement of the security scenarios

Some adjustments and improvements should be made to the scenarios derived from the mapping rules in the previous step. Some new information should be added, and some information should be rephrased. Consequently, the requirements engineering should use his experience and knowledge to provide further information and paraphrase some others based on the elicitation meeting and his expertise in the field. For example, the identification of the security scenario should be provided. But this is a minor issue since the indication must be an id to identify the scenario inside the software development process, and it is more related to documentation definitions. Afterward, the attributes of stimulus, environment, source of stimulus, and artifact should be rephrased considering the information obtained in the previous step. The attributes environment and source of stimulus mainly captured data from the same attribute in the incorrect usage scenario, so in the security scenario, the information should be split and divided into two attributes. Finally, the attributes response and response measure should be completed. Although the mapping rules do not provide information to meet these attributes, the information provided in the rest of the scenario provides the context necessary so the requirement engineers can describe these two attributes. It is important to mention that the response measure attribute, in particular, should be described with quantitative measures. Therefore, engineering requirements should pay attention to that, although the tool described in the following section provides some support. Figure 9 summarizes the refinements. Then, the scenario described at the beginning of this paper in Figure 4 is an example of the scenario that this approach pursues to obtain.

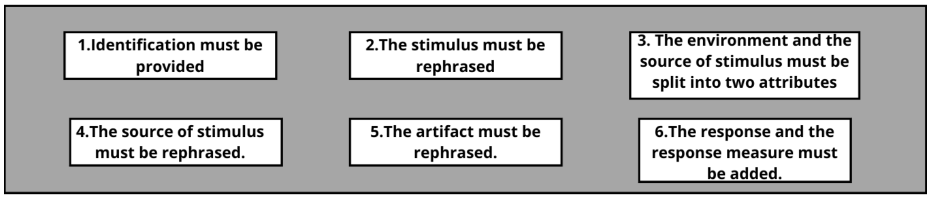

1.Identification must be provided
2.The stimulus must be rephrased
3. The environment and the source of stimulus must be split into two attributes
4.The source of stimulus must be rephrased.
5.The artifact must be rephrased.
6.The response and the response measure must be added.

Fig. 9. Refinement to the security scenarios.

5 Assessment of the approach

5.1 Assesment Desing

Our aim is to assess the acceptance of our approach by security experts in the context of IoT-based smart agriculture, using the Technology Acceptance Model (TAM) to guide our evaluation. Specifically, we are interested in understanding how much our approach is accepted by this target audience. To evaluate the usefulness and ease of use of our approach, we have adopted the well-known and widely used metrics of Perceived Usefulness and Perceived Ease of Use as defined in Fred D. Davis's work. To this end, we have designed and administered a survey to a group of expert security professionals who are representative of our target audience and possess experience in eliciting security requirements. By administering the survey to a group of expert security professionals who are representative of our target audience and possess experience in eliciting security requirements, we will be able to gather valuable feedback and insights. These insights will help us identify areas of strength and weakness in our approach, and ultimately guide improvements that enhance its overall acceptance.

5.2 Survey Application and Data Collection

We conducted a survey with a group of five experts in the software and networking security field. Prior to the survey, we presented our methodology to the group and spent approximately 40 minutes discussing and addressing any questions they had. Once we presented our approach, we administered a survey that included 17 closed-ended questions and three open-ended questions. The survey aimed to gather insights from the experts on the easy to use and usefulness perception of our method. Most of the experts found the proposed security scenario method to be a useful tool for specifying the requirements of agricultural IoT solutions. Half of the experts surveyed agreed that the proposed method simplifies the process of specifying security requirements, resulting in better quality and control of the specification. The experts noted that the proposed method is well-defined, easy to understand, and flexible, making it ideal for defining scenarios. Additionally, the evaluation revealed that the majority (over 60 percent) found it to be clear, well-structured, and interactive in its development.

5.3 Results and Analysis

While the method was generally perceived as useful and easy to use for developing security scenarios, it was suggested that it needs to be more specific to determine its usefulness in practice. The experts suggested that the method can be enhanced to include specific aspects of cybersecurity, as well as development and implementation elements that are essential to ensuring the security of agricultural IoT systems. This would enable a complete specification of the security requirements of these systems. Furthermore, it was noted that users need to interact with the method to remember its steps. During the evaluation, experts identified some areas for improvement such as incorporating vulnerabilities and risks commonly found in IoT systems, considering different types of users and adversaries, and taking into account various attack vectors.

By doing so, the proposed method can be further refined to better meet the needs of users and enhance the security of agricultural IoT systems, particularly adding this information in the lexicon associated.

6 Prototype of the tool support

A software tool was prototyped in order to provide support to the proposed approach. The tool was implemented in Python [22] using libraries such as Spacy [30], an nlp processing library and textblob [23]. This tool consists of a web application, that can be used on desktop computers as well as mobile phones. The application manages different projects and different kinds of artifacts using natural language. Scenarios are one kind of artifact, but the application can be extended easily to manage other artifacts (User Stories, Use Cases, etc.). The prototype provides support for the different activities of the approach. The prototype provides some edition form to allow users to write scenarios of correct and incorrect use of the software application.

The prototype also provides a form to list all the scenarios describing the functionality of the intended software, and by selecting one or more scenarios, the prototype performs the derivation of security scenarios by applying the mapping rules proposed. Then, the security scenarios can be edited in order to improve their description.

The prototype includes some natural language processing tools that make it possible to provide support to assist the requirements engineering to describe the security scenarios. For example, the prototype can verify the use of terms that belong to a glossary. Thus, using Lemmatization and Stemming techniques, the prototype can verify whether certain expressions are used. This is very important in the attributes of response to ensure the use of the correct technique to cope with the issue the scenario is describing. Another feature is the identification of quantitative descriptions. Natural language processing tools make it possible to assess whether this type of expression is present (for example, in the response measure attribute) to be sure that the scenario is correctly written.

7 Conclusion

This paper proposed an approach to describing security scenarios to design a robust software architecture considering IoT technology in the agricultural domain. Developers of IoT applications should be concerned about security (and some other non-functional requirements) since the risk of exposing physical artifacts to intruders is considerable. Moreover, it is difficult to identify the threat and design a countermeasure. Generally, these issues are identified when it is too late when some intruder explodes the vulnerability. Therefore, this paper presents a lightweight approach that begins with a description of the functional requirements. The misuse of the application is identified in order to design countermeasures to deal with it. The paper also described a prototype tool to help apply the proposed approach. Finally, a preliminary assessment was also provided.

The survey applied to five security experts found that the proposed security scenario method is generally useful for specifying agricultural IoT solutions, but needs improvement in certain areas. Experts suggested incorporating specific cybersecurity aspects, vulnerabilities and risks commonly found in IoT systems, and different types of users and adversaries. They also noted the method needs to be more specific and interactive for users to remember its steps. The results provide valuable insights for refining and improving the method to meet user needs and enhance security.

Currently, the most widely used development process is agile development, but we propose a complementary and lightweight technique specifically for IoT applications the smart farm field. As future work, we aim to enrich the proposal with additional guidelines for writing scenarios for each stage. Additionally, further experimentation is necessary before we make the approach more complex. However, we firmly believe that the approach should be strengthened and made more robust.

Acknowledgements This paper is partially supported by funding provided by the STIC AmSud program, Project 22STIC-01.

References

1. Ahmed, U.: A review on knowledge management in requirement engineering. IEEE **5** (2018)
2. Cardenas, H., Zimmerman, R., Viesca, A.R., Al Lail, M., Perez, A.J.: Formal uml-based modeling and analysis for securing location-based iot applications. In: 2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS). pp. 722–723 (2022). https://doi.org/10.1109/MASS56207.2022.00109
3. Carrol, J.: Five reasons for scenario-based design. In: Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers. vol. Track3, pp. 11 pp.– (1999). https://doi.org/10.1109/HICSS.1999.772890

4. Carvalho, R.M.: Dealing with conflicts between non-functional requirements of ubicomp and iot applications. In: 2017 IEEE 25th International Requirements Engineering Conference (RE). pp. 544–549 (2017). https://doi.org/10.1109/RE.2017.51
5. CPS2023: Cyber-physical systems. https://ptolemy.berkeley.edu/projects/cps/ (2023)
6. Dhanaraju, M., Chenniappan, P., Ramalingam, K., Pazhanivelan, S., Kaliaperumal, R.: Smart farming: Internet of things (iot)-based sustainable agriculture. Agriculture **12**(10), 1745 (2022)
7. El-Gendy, S., Azer, M.A.: Security framework for internet of things (iot). In: 2020 15th International Conference on Computer Engineering and Systems (ICCES). pp. 1–6 (2020). https://doi.org/10.1109/ICCES51560.2020.9334589
8. Gupta, M., Abdelsalam, M., Khorsandroo, S., Mittal, S.: Security and privacy in smart farming: Challenges and opportunities. IEEE Access **8**, 34564–34584 (2020). https://doi.org/10.1109/ACCESS.2020.2975142
9. Iqbal, W., Abbas, H., Daneshmand, M., Rauf, B., Bangash, Y.A.: An in-depth analysis of iot security requirements, challenges, and their countermeasures via software-defined security. IEEE Internet of Things Journal **7**(10), 10250–10276 (2020). https://doi.org/10.1109/JIOT.2020.2997651
10. Kammüller, F., Augusto, J.C., Jones, S.: Security and privacy requirements engineering for human centric iot systems using efriend and isabelle. In: 2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA). pp. 401–406 (2017). https://doi.org/10.1109/SERA.2017.7965758
11. Karaduman, B., Mustafiz, S., Challenger, M.: Ftg+pm for the model-driven development of wireless sensor network based iot systems. In: 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). pp. 306–316 (2021). https://doi.org/10.1109/MODELS-C53483.2021.00052
12. Khamaiseh, S., Xu, D.: Software security testing via misuse case modeling. In: 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech). pp. 534–541 (2017). https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2017.98
13. Khan, N., Ray, R.L., Sargani, G.R., Ihtisham, M., Khayyam, M., Ismail, S.: Current progress and future prospects of agriculture technology: Gateway to sustainable agriculture. Sustainability **13**(9), 4883 (2021)
14. Martin, T., Geneiatakis, D., Kounelis, I., Kerckhof, S., Nai Fovino, I.: Towards a formal iot security model. Symmetry **12**(8) (2020). https://doi.org/10.3390/sym12081305, https://www.mdpi.com/2073-8994/12/8/1305
15. Myagmar, S., Lee, A.J., Yurcik, W.: Threat modeling as a basis for security requirements. In: Symposium on Requirements Engineering for Information Security (SREIS). University of Pittsburgh (August 2005), http://d-scholarship.pitt.edu/16516/
16. Nguyen, X.T., Tran, H.T., Baraki, H., Geihs, K.: Frasad: A framework for model-driven iot application development. In: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). pp. 387–392 (2015). https://doi.org/10.1109/WF-IoT.2015.7389085

17. Ojo-Gonzalez, K., Bonilla-Morales, B.: Requerimientos no funcionales para sistemas basados en el internet de las cosas (iot): Una revisión. I+ D Tecnológico **17**(2), 30–40 (2021)
18. Ozkaya, O., Ors, B.: Model based node design methodology for secure iot applications. In: 2018 26th Signal Processing and Communications Applications Conference (SIU). pp. 1–4 (2018). https://doi.org/10.1109/SIU.2018.8404490
19. Pal, S., Hitchens, M., Rabehaja, T., Mukhopadhyay, S.: Security requirements for the internet of things: A systematic approach. Sensors **20**(20), 5897 (2020)
20. Potts, C.: Using schematic scenarios to understand user needs. In: Proceedings of the 1st Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques. p. 247–256. DIS '95, Association for Computing Machinery, New York, NY, USA (1995). https://doi.org/10.1145/225434.225462, https://doi.org/10.1145/225434.225462
21. do Prado Leite, J.C.S., Hadad, G.D.S., Doorn, J.H., Kaplan, G.N.: A scenario construction process. Requirements Engineering **5**, 38–61 (2000)
22. Python: Python framework. https://www.python.org/ (2023), accessed: 2023-03-11
23. Python: Textblob library. https://pypi.org/project/textblob/ (2023), accessed: 2023-03-11
24. Rose, D.C., Wheeler, R., Winter, M., Lobley, M., Chivers, C.A.: Agriculture 4.0: Making it work for people, production, and the planet. Land use policy **100**, 104933 (2021)
25. Schneier, B.: Cryptography is harder than it looks. IEEE Security & Privacy **14**(1), 87–88 (2016). https://doi.org/10.1109/MSP.2016.7
26. Serna M, E., Serna A, A.: Proceso y progreso de la formalización de requisitos en ingeniería del software. Ingeniare. Revista chilena de ingeniería **28**(3), 411–423 (2020)
27. Shankar, P., Morkos, B., Yadav, D., Summers, J.D.: Towards the formalization of non-functional requirements in conceptual design. Research in engineering design **31**, 449–469 (2020)
28. Slovenec, K., Vuković, M., Salopek, D., Mikuc, M.: Securing iot services based on security requirement categories. In: 2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM). pp. 1–6 (2022). https://doi.org/10.23919/SoftCOM55329.2022.9911319
29. Sotoudeh, S., Hashemi, S., Garakani, H.G.: Security framework of iot-based smart home. In: 2020 10th International Symposium onTelecommunications (IST). pp. 251–256 (2020). https://doi.org/10.1109/IST50524.2020.9345886
30. Spacy: Spacy library. https://spacy.io/ (2023), accessed: 2023-03-11
31. Y.2060, I.T.: Itu-t y.2060. https://handle.itu.int/11.1002/1000/11559 (2012)
32. Yazdinejad, A., Zolfaghari, B., Azmoodeh, A., Dehghantanha, A., Karimipour, H., Fraser, E.D.G., Green, A.G., Russell, C., Duncan, E.: A review on security of smart farming and precision agriculture: Security aspects, attacks, threats and countermeasures. Applied Sciences (2021)