An Inspection Technique Proposal for the Verification of Requirements Specification Documents for Multi-Agent Systems

Giovane D'Avila Mendonça[a], Gilleanes Thorwald Araujo Guedes[b] and Iderli Pereira de Souza Filho[c]

Software Engineering Post-Graduation Program, Pampa Federal University, Av. Tiaraju, 810, Alegrete, Brazil

Abstract: Requirements engineering is an important area of software engineering dedicated to eliciting, analysing, specifying, and validating software requirements to ensure the correct understanding of what needs to be developed. The requirements specification objective is to provide a detailed description of what the software must do, it involves the production of a document that can be systematically reviewed, evaluated, and approved. Problems in the requirements are appointed among the main causes of failures in software projects. Therefore, performing requirements verification and validation is extremely important to ensure the software quality. Multi-Agent systems are a type of software with particular requirements, beyond the commonly found among other systems, since they are composed by several autonomous and proactive agents that divide the problem to be solved among them. Thus, requirements engineering needs to be adapted for this kind of system and the produced documents need to be verified too. Several techniques were proposed for requirements inspection, among them there is the Perspective-Based Reading. However, as in the other approaches, this technique does not allow the inspection of particular requirements for multi-agent systems. Taking this in consideration, our work has as its objective to adapt this technique so as to allow the verification of requirements specification documents specific for this kind of system.

1 INTRODUCTION

Multi-Agent systems are characterized by being composed of several agents that interact among themselves (Wooldridge, 2009). An agent is an autonomous, proactive, reactive, and with social skills process (Vicari and Gluz, 2007), moreover, an agent is able to performing actions without the user intervention (Wooldridge and Jennings, 1995) (Pornphol and Chittayasothorn, 2006) (Sivakumar et al., 2013).

Multi-Agent systems became an alternative for the complex systems development (Labba et al., 2015). However, the development of this kind of system brought several challenges to the software engineering area, what entailed the emergence of AOSE (Agent-Oriented Software Engineering) an area that mixes characteristics of several disciplines both from Software Engineering and Artificial Intel-

ligence (Cuesta et al., 2007). AOSE has, among its objectives, to adapt software engineering practices specifically for the agent-based systems development (Guedes and Vicari, 2010), that includes to produce methodologies, processes, techniques, modeling languages, and tools for the multi-agent systems development (Genza and Mighele, 2013) (Mendonça et al., 2021). How could it be, AOSE also includes the adaptation of Requirements Engineering for multi-agent systems.

Requirements Engineering is a Software Requirements area that is concerned with requirements eliciting, analysing, specifying, and validating to ensure the correct understanding of what needs to be developed (Fuentes-Fernández et al., 2009). Requirements Engineering performs a crucial function for the development of any software, since, if the software needs were not understand correctly, the product will not satisfy those to whom its is destined (Mendonça et al., 2021).

The main failures verified in software projects are related to problems in the requirements specification,

[a] https://orcid.org/0000-0001-9445-6831
[b] https://orcid.org/0000-0001-5457-2600
[c] https://orcid.org/0000-0001-9694-0977

257

due, mainly, to difficulties in the understanding of the users needs (Koscianski and dos Santos Soares, 2007). Therefore, to perform the requirements verification and validation is essential to ensure the software quality, because errors identified in the requirements in posterior phases of the software development has a very much bigger cost to be corrected (Bilal et al., 2016). Thus, several inspection techniques were created to the requirements verification and validation. Inspections had proved to be an effective way to find failures in different software artifacts (Fogelström and Gorschek, 2007).

Taking this in consideration, we are proposing an adaptation of the Perspective-Based Reading technique (PBR) (Basili et al., 1996) (Shull et al., 2000) in such a way that it can be applied in the inspection of software requirements specification documents (SRSD) specific for multi-agent systems. Whereas Ebad in (Ebad, 2017), highlights that the Perspective Based Reading (PBR) technique is one of the most effective for requirements inspection. This technique adapted was conceived to be applied on SRSDs produced by a specific requirements engineering process for multi-agent systems we developed. The documents produced by this process use the notation of Multi-Agent Systems Requirements Modeling Language (MASRML) proposed by (Guedes et al., 2020). We use MASRML because this notation supports Belief-Desire-Intention (BDI) model, during the requirements specification, different from other notations that do not support this type of representation. Furthermore, the MASRML notation uses some concepts, as by example Internal Use Cases (IUC) and AgentRoleActors.

In this study context, we propose a verification technique, considering the definition presented in the study of (Ryan and Wheatcraft, 2017), in which the use of verification and validation terms and their definitions according to the literature was analyzed. In this study, the author defines that the verification focus is on the requirements text and structure, determining whether they are correctly written and structured according to the organization's standards, guidelines, rules, and checklists. On the other hand, requirements validation makes it possible to determine whether the requirements are defined in a clear way and whether they communicate correctly the stakeholders expectations and needs. It involves establishing whether the right thing is being built, as defined by the requirements set.

In order to assess the applicability and efficiency of our proposal to adapt the Perspective-Based Reading technique, we are applying two quasi-experiments. The first with students of Verification and Validation discipline of a Software Engineering undergraduate course. The second quasi-experiment is being applied to specialists, master's students, and final year students of an undergraduate software engineering course. Most of the subjects have professional experience in the software engineering area.

Due to the limited space in this work, the results of the quasi-experiments will be published in a future work.

2 BACKGROUND

2.1 Requirements Engineering

The objectives of Requirements Engineering are: (I) identifying software requirements; (II) analysing requirements so to classify them and to derive additional requirements, as well as to solve conflicts among them; (III) documenting the requirements; (IV) validating the documented requirements (Berenbach et al., 2009).

In SWEBOK (Bourque et al., 2014) – a reference book in the area – it is stated that the requirements engineering process covers four main subareas, (I) Requirements Elicitation; (II) Requirements Analysis; (III) Requirements Specification; and (IV) Requirements Validation.

Requirements elicitation investigates how to extract requirements and which are its origins. Requirements analysis aims to detect and to solve conflicts among the requirements and to discover the system limits. Requirements specification, by its turn, produces requirements documents that can be systematically reviewed, evaluated, and approved. Finally, requirements validation evaluates the requirements documents to ensure the requirements are understandable, correct, consistent, and complete.

2.2 Requirements Verification

Weak alignment of requirements engineering with verification can lead to problems in the deliver of software products, such as lack in the accomplishments of deadlines and poor quality (Bjarnason et al., 2014). According to (Dzida and Freitag, 1998) verification is related to the requirements correctness.

Requirements verification is the confirmation by examination that the requirements (individually or in a set) are well formed (ISO/IEC/IEEE 29148, 2011) (ISO/IEC/IEEE 29148, 2018). According to (Ryan and Wheatcraft, 2017), this means that a requirement or a set of requirements has been revised to ensure that the characteristics of good requirements are achieved.

According to (Khan et al., 2015), experts believe that review and inspection are the best strategies to remove or mitigate requirements verification and validation challenges. In this sense, (Ryan and Wheatcraft, 2017) states that requirements verification confirms, through inspection, that the requirements contain the necessary elements and have the characteristics of a well-formed requirement.

2.3 Agents and Multi-Agent Systems

An agent is a process situated in an environment designed to achieve a purpose by means of an autonomous and flexible behavior. The environment is the application domain where the agent will act (Vicari and Gluz, 2007).

A multi-agent system (MAS) consists of a set of agents that can interact among themselves. Agents are able to act in an environment and have different "influence spheres", where they can control or influence different parts of the environment (Wooldridge, 2009).

Agents use to play roles. An agent role contains a part of the social behavior of an agent and it is characterized by having a goal and/or providing a service. The goal of each role is to contribute to the accomplishment of a part of the organization (or environment) requirements in which the agent is contained (Cossentino and Seidita, 2014). The agent roles represent the functions the agents can perform inside the system. The agents can take more than one role, but generally not at the same time.

2.4 Internal Use Cases and AgentRoleActors

The specification proposed in our work uses the notation established in MASRML (Guedes et al., 2020) language, a modeling language extended from UML specifically to the multi-agent systems requirements representation.

The AgentRoleActors represent roles taken by agents. These agent roles are modeled inside the system boundary. Internal Use Cases (IUC) are use cases that are not accessed by external actors. IUCs are accessed by Agent Roles (AgentRoleActors). These IUCs can have stereotypes of the Goal, Plan, Perception, or Action kind.

2.5 Belief-Desire-Intention Model

The Belief-Desire-Intention (BDI) model is a software model developed for programming intelligent agents. It includes beliefs, desires, and intentions in the agent architecture (Bratman et al., 1987).

Beliefs represent the information state the agent has about the environment, about himself, and even about the other agents. The desires represent the goals or situations that the agent wants to achieve. Lastly, intentions represent the desires the agent believes he can achieve and acts to achieve it (Rao and Georgeff, 1995).

According to (Bordini et al., 2021) asserts that the unique strengths of BDI agent languages provide an ideal framework for integrating the wide range of AI capabilities needed to progress towards the next generation of intelligent systems.

BDI agents are among the most widely studied models of rational agents (Torre and Parlato, 2020). The Belief-Desire-Intention model is one of the most popular models for developing rational agents based on how humans act and on information derived from an environment (Ujjwal and Chodorowski, 2019).

Furthermore, according to (Xu et al., 2018), the BDI architecture, where agents are modeled based on their beliefs, desires, and intentions, provides a practical approach to developing intelligent agent systems.

2.6 Multi-Agent Systems Requirements Modeling Language

MASRML – Multi-Agent Systems Requirements Modeling Language – is a UML-based Domain-Specific Modeling Language (DSML) conceived for requirements modeling in multi-agent system projects (Guedes et al., 2020). This DSML extends the UML metamodel in order the use-case diagram can be applied in the specific domain of multiagent systems. MASRML provides mechanisms to represent the concepts of agent role, goal (desire), perception, belief, intention, plan, and action, supporting the concepts of the BDI model.

In MASRML, new concepts were created inspired by the concepts of the standard UML. The main contribution was the creation of the AgentRoleActor and InternalUseCase metaclasses to represent agent roles and the internal functionalities assigned to them, stereotyped in goals, perceptions, actions, and plans. Some associations were also created in order to associate perceptions, actions, and plans to the goals that an agent role wishes to achieve, thus establishing the conditions for a goal to become an intention and which plan will be executed in this case, as well as the possible external actions performed during the executing a plan.

In addition, MASRML internal use-cases documentation provides a clear view of the structure of

cognitive agents, allowing, for example, to determine what is needed for a given goal to become an intention or to establish the steps to be performed when executing a perception or a plan.

3 RELATED WORKS

Several methodologies were proposed supporting requirements engineering. However, the requirements validation is a subarea that has been neglected in these methodologies. Based on the study of (Mendonça et al., 2021), in which, by means of a systematic review sought to establish the state-of-art of methodologies or processes that covers requirement engineering for multi-agent systems, we observed that only three studies support the requirements validation subarea.

Although these three studies ((Cysneiros and Yu, 2002), (Haumer et al., 1999), (Bonjean et al., 2014)) contain in its phases the requirements validation subarea, none of them proposed a specific validation technique for multi-agent systems requirements. Thus, is evident the need of proposing a new approach of requirements validation for this kind of system.

Several techniques have been proposed for inspecting traditional systems, among which we can quote Fagan in (Fagan, 1976), who proposed the use of checklists for defect detection; Porter and Votta in (Porter and Votta, 1994), who presented a scenario-based technique that provides inspectors with more specific instructions than checklists, allowing them to classify defects and develop questions for each type of defect; Thelin et al. in (Thelin et al., 2004) who introduced the Usage-Based Reading (UBR), in which the inspectors use a predefined list of prioritized use cases; UBR-ir (Thelin et al., 2003), a variation of UBR that allows inspectors to classify use cases individually at the beginning of the inspection process according to the inspectors' experience and understanding. Dunsmore et al. in (Dunsmore et al., 2003), who presented Use-Case Reading (UCR) to inspect object-oriented systems considering dynamic interaction with collaborative objects.

However, these techniques were proposed to be applied in traditional systems, therefore they cannot be fully used in the requirements verification for multi-agent systems due to the specific features of this type of system.

4 PROPOSED ADAPTATION OF PERSPECTIVE-BASED READING TO INSPECT REQUIREMENTS SPECIFICATION DOCUMENTS FOR MULTI-AGENT SYSTEMS

The study presented by Ebad in (Ebad, 2017), evaluated the evidence about the efficacy of the reading techniques in the inspection of artifacts produced during the requirements, project, and coding phases. In the requirements phase (requirements engineering), Ebad states that the ad hoc technique is the simplest of the techniques, but in this technique the inspectors do not follow guidelines or orientations. In this same study, the author highlights that the Perspective-Based Reading (PBR) technique is one of the most effective for requirements inspection.

PBR provides a set of procedures that can help developers to solve software requirements inspection problems. PBR reviewers represent specific stakeholders interested in the document (as designers or testers) to verify the requirements quality (Shull et al., 2000). The combination of different perspectives result in a better analysis of the requirements specification document, making it possible that it can be completely checked (Pagliuso et al., 2002).

The PBR technique provides questions developed specifically for each step of procedure and this way it creates representations in order the reviewers can answer a series of questions about the work product (Pagliuso et al., 2002). Each inspector must define his interest in the document so that it is inspected according to the perspective of a specific involved part.

Thus, considering the evidences pointed by Ebad in (Ebad, 2017) that PBR is one of the more efficient techniques for requirements inspection and considering the statement of Shull in (Shull et al., 2000) that, depending on the environment where PBR is applied, it can be find a different set of more applicable perspectives, we decided to adapt this technique to the inspection of a requirements specification document produced based on the ISO/IEC/IEEE 29148:2018 standard (ISO/IEC/IEEE 29148, 2018) extended for the multi-agent system dominion.

That said, in this work, we are proposing the Agent Simulator Perspective to SRSD inspecting considering multi-agent systems requirements specific characteristics. This perspective aims to determine whether the requirements describe properly the functionalities to be performed by the agents. This perspective inspects Internal Use Cases (IUC), evaluating if the main, alternative, and exception scenarios

are described correctly, as well as if the pre-conditions are defined.

Moreover, basing on the Agent Simulator perspective, it is possible to verify whether the agent roles are identified and detailed, whether the initial beliefs are described, whether the perceptions are associated with the goals, and to verify the correctness of the stereotypes or associations of the internal use cases.

The Agent Simulator Perspective follows a sequence of steps. Through these steps, the inspector simulates the agents behaviors searching for errors or failures in the requirements specification document. This process can be observed in Figure 1.

As can be seen in Figure 1 the inspection process has a sequence of activities, however, due to lack of space, in this paper we will discuss only the activities performed by the inspector.

After receiving the SRSD, the inspector will perform a sequence of six steps. Each of these steps contains a verification list that must be followed during the SRSD inspection. These steps correspond to I) general inspection of the SRSD; II) general inspection of each IUC; III) inspection of the internal use cases of Goal type; IV) inspection of the internal use cases of Perception type; V) inspection of each partial use-case diagram for each AgentRoleActor; and VI) inspection of the general use-case diagram.

The activities performed by the requirements engineering team and by the inspection manager are similar to the traditional inspection process, however, the differences between the traditional process and ours are the activities performed by the inspector.

In our process, the inspector takes the agent perspective as discussed previously, so the SRSD will be inspected considering the specific characteristics of multi-agent systems, following the checklists that we are developing for each of the inspection steps.

Next we will describe every step of our inspection technique proposal.

4.1 First Step

In this step, the requirements specification document is inspected in a general way. The inspector will search for completeness, correctness, and consistency failures types. Some of them are related to the definition of agent roles, the functionalities associated with the agents, communication patterns used, and about the beliefs identified.

4.2 Second Step

In this step, the inspector must analyse each IUC searching for types of failures like incorrectness, am-

biguity, untraceability, incompleteness, and inconsistency. Some of these failures are related to the general structure of each IUC, lack of main scenarios, interpretation ambiguities, or incorrectness in the use-case scenario texts.

4.3 Third Step

In this step, the inspector must analyse each one of the internal use cases of the Goal type, searching for types of failures like correctness, consistency, and completeness. Some of these failures are related to stereotype correctness, clarity in the perceptions description that turns a goal into an intention, or the existence of perceptions associated with a goal.

4.4 Fourth Step

In this phase, the inspector will analyse each one of the internal use cases of the Perception type, searching for types of failures like correctness and completeness. Some of these failures are related to stereotype incorrectness, lack of identification or clarity of the initial beliefs, and lack of pre-conditions definition.

4.5 Fifth Step

In this step, the inspector must analyse each partial use case diagram developed for each AgentRoleActor. The inspector will search for failure types like correctness and consistency. Some of them are related to stereotypes incorrectness, incorrectness of the associations among the diagram components, and incorrectness of the associations direction.

4.6 Sixth Step

Lastly, in this step, the inspector must analyse the general use-case diagram, searching to locate failures of correctness and completeness type. Some of them are related to agent roles stereotypes incorrectness, agent roles and external users identified in incorrect places, and the lack of attribution of at least one goal to each agent role.

5 CONCLUSION AND FUTURE WORKS

The requirements verification and validation is an important phase of the requirements engineering and plays a crucial role in the successful of any project (Gupta et al., 2019). During the verification and validation phase the requirements specification must be
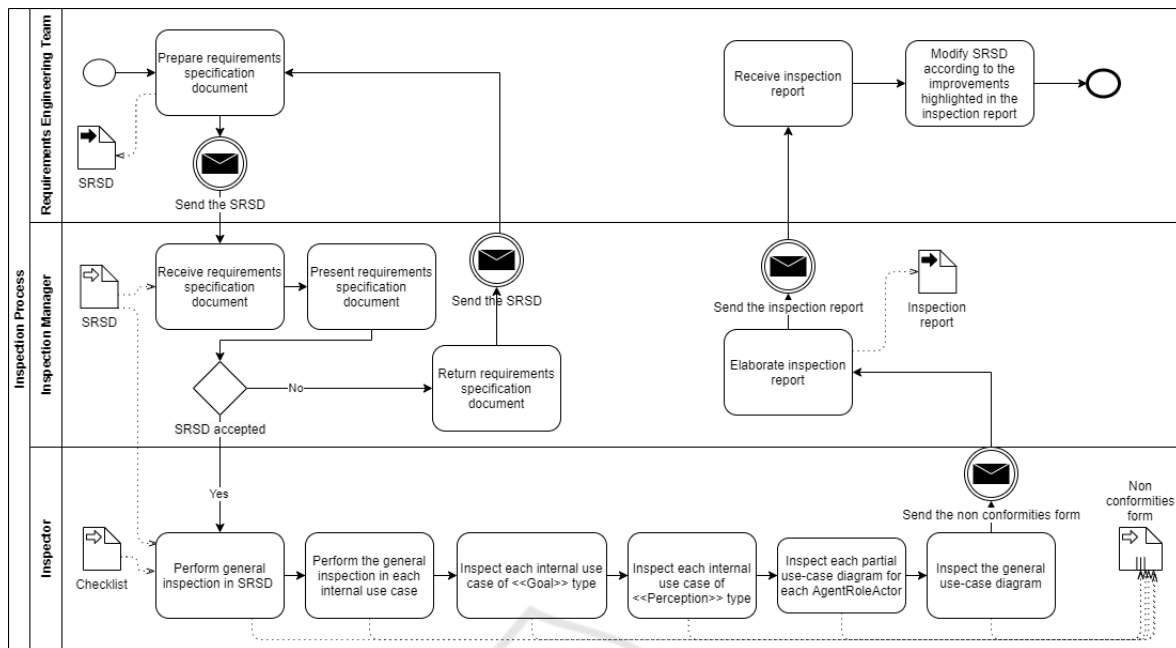
Figure 1: Inspection Process of Multi-Agent Systems Requirements.

examined to ensure that all the requirements had been declared in a non ambiguous way and free of inconsistencies, omissions, and errors (Pressman, 2011).

Several techniques were presented for requirements inspecting, as for example, the checklist-based reading, scenarios-based inspection, perspective-based reading (PBR), use-cases reading (UCR), and Use-Based Reading (UBR) (Fagan, 1976) (Porter and Votta, 1994) (Basili et al., 1996) (Dunsmore et al., 2003) (Thelin et al., 2004).

However, all these techniques were applied to inspect traditional systems, mainly object-oriented. Nevertheless, according to Padmanaban et al. in (Padmanaban et al., 2016) conventional object-oriented software tests cannot be applied to agent-oriented systems due the agents' features, such as autonomy, proactivity, social capabilities, reactivity, and mobility. Moreover, the new concepts and properties that arise with the use of the agent paradigm, implies in the construction of new verification and validation approaches, different from the conventional software engineering (Fuentes-Fernández et al., 2004).

Thus, we proposed in this work a verification technique for a kind of SRSD produced based on an extension of ISO/IEC/IEEE 29148:2018 standard and on the MASRML notation. To develop this verification technique we adapted the Perspective-Based Reading technique, creating the Agent-Simulator perspective.

PBR must be adaptable to the particular document it will be inspecting and to the notation used to produce artifacts and diagrams contained in the

document. Moreover, any technique, including PBR should be adapted to the particularities of a given environment in order to be more successful (Ciolkowski et al., 1997).

To evaluate the applicability and efficiency of our proposed inspection technique, we are applying our technique in two quasi-experiments. The inspection is being carried out from a document of requirements specification produced based on the ISO/IEC/IEEE 29148:2018 standard, that was extended to support requirements for multi-agent systems. This extended standard uses MASRML notation. The results of the quasi-experiments will be published in a future work.

As a partial result of the first quasi-experiment, we noticed that, of the 23 failures out of the 25 injected into the Software Requirements Specification Document (SRSD), about 92%, were discovered by at least one of the twelve subjects who participated in the inspection.

Therefore, we concluded that the partial results obtained in the first quasi-experiment can indicate the efficiency and applicability of our adaptation of the PBR technique for the inspection of specification requirements documents for multi-agent systems. Moreover, the quasi-experiment contributed to increase the experience in software inspections of the students participants.

We are finalizing the tabulation of the data collected during the execution of the two quasi-experiments. We intend to publish the full results in future works.

Furthermore, to complement our PBR adaptation proposal, we are developing checklists for each of the six steps presented in section 4. These checklists will have a sequence of pre-defined questions intended to guide the inspectors through the evaluation process of the requirements specification document.

This work is part of a bigger project that aims to propose a complete requirements engineering process for the multi-agent systems development. Thus, this study contributes to the proposition of a new approach for multi-agent systems requirements verification.

Our inspection proposal was developed for inspecting SRSDs produced by a specific requirements engineering process. These documents are structured according to the standard ISO/IEC/IEEE 29148:2018 extended to the multi-agent systems context and its use-case diagrams and documentation are produced by means of the notation provided by MASRML (Guedes et al., 2020). However, we believe our inspection technique proposal can be adapted to inspecting other SRSDs produced by other processes of requirements engineering for multi-agent systems.

As a future work we are developing a guide with application orientations about our inspection technique. Thus, we also intend to add this guide to the requirements engineering process currently in development.

REFERENCES

Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S., and Zelkowitz, M. V. (1996). The empirical investigation of perspective-based reading. *Empirical Software Engineering*, 1:133–164.

Berenbach, B., Paulish, D., Kazmeier, J., and Rudorfer, A. (2009). *Software & systems requirements engineering: in practice*. McGraw-Hill, Inc.

Bilal, H. A., Ilyas, M., Tariq, Q., and Hummayun, M. (2016). Requirements validation techniques: An empirical study. *International Journal of Computer Applications*, 148(14):5–10.

Bjarnason, E., Per Runeson, M. B., Unterkalmsteiner, M., Engström, E., Regnell, B., Sabaliauskaite, G., Loconsole, A., Gorschek, T., and Feldt, R. (2014). Challenges and practices in aligning requirements with verification and validation: a case study of six companies. *Empirical Software Engineering*.

Bonjean, N., Mefteh, W., Gleizes, M. P., Maurel, C., and Migeon, F. (2014). *ADELFE 2.0*, pages 19–63. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bordini, R. H., El Fallah Seghrouchni, A., Hindriks, K., Logan, B., and Ricci, A. (2021). Agent programming in the cognitive era. In *Proceedings of the 20th International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '21, page 1718–1720, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Bourque, P., Fairley, R. E., et al. (2014). *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press.

Bratman, M. et al. (1987). *Intention, plans, and practical reason*, volume 10. Harvard University Press Cambridge, MA. 1-57586-192-5.

Ciolkowski, M., Differding, C., Laitenberger, O., and Münch, J. (1997). Empirical investigation of perspective-based reading: A replicated experiment. *International Software Engineering Research Network (ISERN)*.

Cossentino, M. and Seidita, V. (2014). *PASSI: Process for agent societies specification and implementation*, pages 287–329. Handbook on Agent-Oriented Design Processes.

Cuesta, P., Gómez, A., and González, J. C. (2007). Agent oriented software engineering. *Issues in Multi-Agent Systems*.

Cysneiros, L. and Yu, E. (2002). Requirements engineering for large-scale multi-agent systems. volume 2603, pages 39–56.

Dunsmore, A., Roper, M., and Wood, M. (2003). The development and evaluation of three diverse techniques for object-oriented code inspection. *IEEE Trans. Softw. Eng.*, 29(8):677–686.

Dzida, W. and Freitag, R. (1998). Making use of scenarios for validating analysis and design. *IEEE Transactions on Software Engineering*, 24(12):1182–1196.

Ebad, S. A. (2017). Inspection reading techniques applied to software artifacts - a systematic review. *Comput. Syst. Sci. Eng.*, 32.

Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):182–211.

Fogelström, N. and Gorschek, T. (2007). Test-case driven versus checklist-based inspections of software requirements - an experimental evaluation. In *Proceedings of the 10th Workshop on Requirements Engineering*, pages 116–126.

Fuentes-Fernández, R., Gómez-Sanz, J. J., and Pavón, J. (2004). Verification and validation techniques for multi-agent systems. *Upgrade*, 5:15–19.

Fuentes-Fernández, R., Gómez-Sanz, J. J., and Pavón, J. (2009). Requirements elicitation and analysis of multiagent systems using activity theory. *IEEE*.

Genza, N. and Mighele, E. (2013). Review on multi-agent oriented software engineering implementation. *International Journal of Computer and Information Technology*, 2.

Guedes, G., Souza Filho, I., Gaedicke, L., Mendonça, G., Vicari, R., and Brusius, C. (2020). Masrml - a domain-specific modeling language for multi-agent systems requirements. *International Journal of Software Engineering & Applications (IJSEA)*, 11(5).

Guedes, G. T. A. and Vicari, R. M. (2010). A uml profile oriented to the requirements modeling in intelligent tutoring systems projects. In Bramer, M., editor, *Artificial Intelligence in Theory and Practice III*, pages

133–142, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gupta, A., Deraman, A., and Siddiqui, S. (2019). Bdi logics for bdi architectures: old problems, new perspectives. *ARPN Journal of Engineering and Applied Sciences*, 14(17):3046–3061.

Haumer, P., Heymans, P., Jarke, M., and Pohl, K. (1999). Bridging the gap between past and future in re: a scenario-based approach. In *Proceedings IEEE International Symposium on Requirements Engineering (Cat. No.PR00188)*, pages 66–73.

ISO/IEC/IEEE 29148 (2011). Iso/iec/ieee international standard - systems and software engineering – life cycle processes –requirements engineering. *ISO/IEC/IEEE 29148*, pages 1–94.

ISO/IEC/IEEE 29148 (2018). Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering. *ISO/IEC/IEEE 29148:2018(E)*, pages 1–104.

Khan, H., Asghar, I., Ghayyur, S., and Raza, M. (2015). An empirical study of software requirements verification and validation techniques along their mitigation strategies. pages 2321–5658.

Koscianski, A. and dos Santos Soares, M. (2007). *Qualidade de Software-2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. Novatec Editora.

Labba, C., Bellamine ben Saoud, N., and Dugdale, J. (2015). Towards a conceptual framework to support adaptative agent-based systems partitioning. In *2015 IEEE/ACIS 16th SNPD*, pages 1–5.

Mendonça, G., Souza, Filho, I., and Guedes, G. (2021). A systematic review about requirements engineering processes for multi-agent systems. In *13th International Conference on Agents and Artificial Intelligence (ICAART 2021)*, volume 1, pages 69–79.

Padmanaban, R., Thirumaran, M., Suganya, K., and Priya, R. (2016). Aose methodologies and comparison of object oriented and agent oriented software testing. pages 1–16.

Pagliuso, P., Tambascia, C., and Villas-Boas, A. (2002). Melhoria da inspeção de requisitos segundo a técnica de leitura baseada em perspectiva. *XI Seminário de Computação - XI SEMINCO*, 32.

Pornphol, P. and Chittayasothorn, S. (2006). An agent-based quality assurance assessment system. *5th WSEAS International Conference on E-ACTIVITIES*.

Porter, A. and Votta, L. (1994). An experiment to assess different defect detection methods for software requirements inspections. In *Proceedings of 16th International Conference on Software Engineering*, pages 103–112.

Pressman, R. (2011). *Software Engineering: A Practitioner's Approach*. Bookman.

Rao, A. S. and Georgeff, M. P. (1995). Bdi agents: From theory to practice. In *In Proceedings of the First International Conference on Multi-agent Systems (ICMAS-95*, pages 312–319.

Ryan, M. and Wheatcraft, L. (2017). On the use of the

terms verification and validation. *INCOSE International Symposium*, 27:1277–1290.

Shull, F., Rus, I., and Basili, V. (2000). How perspective-based reading can improve requirements inspections. *Computer*, 33(7):73–79.

Sivakumar, N., Vivekan, K., and Kanimozhi, M. (2013). Testing in prometheus methodology – plan oriented approach.

Thelin, T., Runeson, P., and Wohlin, C. (2003). An experimental comparison of usage-based and checklist-based reading. *IEEE Trans. Softw. Eng.*, 29(8):687–704.

Thelin, T., Runeson, P., Wohlin, C., Olsson, T., and Andersson, C. (2004). Evaluation of usage-based reading—conclusions after three experiments. 9(1–2):77–110.

Torre, S. L. and Parlato, G. (2020). A fixed-point model-checker for bdi logics over finite-state worlds. In *OVERLAY*.

Ujjwal, K. and Chodorowski, J. (2019). A case study of adding proactivity in indoor social robots using belief–desire–intention (bdi) model. *Biomimetics*, 4(74).

Vicari, R. M. and Gluz, J. C. (2007). An intelligent tutoring system (its) view on aose. *International Journal of Agent-Oriented Software Engineering*, 1(3-4):295–333.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152.

Xu, M., Bauters, K., McAreavey, K., and Liu, W. (2018). A formal approach to embedding first-principles planning in BDI agent systems. In Ciucci, D., Pasi, G., and Vantaggi, B., editors, *Scalable Uncertainty Management - 12th International Conference, SUM 2018, Milan, Italy, October 3-5, 2018, Proceedings*, volume 11142 of *Lecture Notes in Computer Science*, pages 333–347. Springer.