Agent-Based Software Testing: A Definition and Systematic Mapping Study

Pavithra Perumal Kumaresen[1], Mirgita Frasheri[1], Eduard Paul Enoiu[1]
[1]Mälardalen University, Västerås, Sweden

Abstract—The emergence of new technologies in software testing has increased the automation and flexibility of the testing process. In this context, the adoption of agents in software testing remains an active research area in which various agent methodologies, architectures, and tools are employed to improve different test problems. Even though research that investigates agents in software testing has been growing, these agent-based techniques should be considered in a broader perspective. In order to provide a comprehensive overview of this research area, which we define as agent-based software testing (ABST), a systematic mapping study has been conducted. This mapping study aims to identify the topics studied within ABST, as well as examine the adopted research methodologies, identify the gaps in the current research and point to directions for future ABST research. Our results suggest that there is an interest in ABST after 1999 that resulted in the development of solutions using reactive, BDI, deliberative and cooperate agent architectures for software testing. In addition, most of the ABST approaches are designed using the JADE framework, have targeted the Java programming language, and are used at system-level testing for functional, non-functional and white-box testing. In regards to regression testing, our results indicate a research gap that could be addressed in future studies.

I. INTRODUCTION

Nowadays, continuous development and integration processes are subject to large and frequent changes. Consequently, software development organizations need to deliver reliable and high-quality software products while having to consider more stringent time constraints. A side effect of such constraints is the limitation in the amount of development and testing that can be performed before delivering the software [1]. Intelligent and automated techniques can be used to tackle this problem. For example, agents have already been used to automate different aspects of testing and improve test efficiency and effectiveness [2], [3].

Agents are software systems that operate in an environment which they can perceive and act upon, while also being able of autonomous actions [4]. Depending on the flexibility of such actions, agents might be able to take initiatives and select their own goals, and interact with others when deemed fit. What is missing from the state of the art is a comprehensive approach for defining agent-based software testing and its applications.

To tackle this gap, first we propose a definition of the research area as Agent-Based Software Testing (ABST). Secondly, we present the results from a systematic mapping study, in which we identify the areas of application together with the tools, techniques, and methods used in the development of agent-based systems for software testing, while also examining the research methodologies adopted in these different works.

Based on the results obtained from this study, we identify research trends and gaps which can be useful for both researchers and practitioners.

II. A DEFINITION OF AGENT-BASED SOFTWARE TESTING

Agents are software systems – which could be embodied into physical entities – that operate in an environment which they can perceive and act upon, and are able of performing autonomous actions [5]. Agents are used in many domains, can take different physical forms [6], and have different properties such as: (i) operating on their own without human interventions (autonomy), (ii) interacting with other agents (social ability), (iii) perceiving their environment and responding in a timely fashion to changes that occur in it (reactivity), and (iv) taking initiative to exhibit a goal-oriented behavior (proactiveness). An agent can be capable of learning, by acquiring new knowledge and skills, which can be used to take better decisions in the future.

Agents have already been used to automate different aspects of software development [4], [7], [8]. Several researchers have proposed different approaches for using agents specifically in software testing [9]–[11], by considering different aspects that relate to test management, test design, execution and evaluation. Jeff Offutt [12] outlined in a keynote from 2016 that there is a need in test automation for more intelligent tests exhibiting self-determination and self-awareness. However, the relation and overlap between the use of (intelligent) agents in software testing and what part of the testing process these methods optimize is unclear. Hence, we propose that *Agent-Based Software Testing (ABST) should be defined as the application of agents (e.g., software agents, intelligent agents, autonomous agents, multi-agent systems) to software testing problems by tackling and automating complex testing tasks*.

The aim of ABST research is to address software testing problems through agent-based paradigms, by additionally using the variety of techniques from artificial intelligence and software engineering. We realize that the process of defining the ABST research area is evolutionary and iterative. The ABST definition needs to be further discussed as the evidence and knowledge in this area grows and refines.

III. RESEARCH METHOD

A systematic mapping involves several steps like identification of papers, analysis, and classification of selected papers in the area of interest (i.e., agent-based software testing). The systematic mapping study is performed in accordance with the
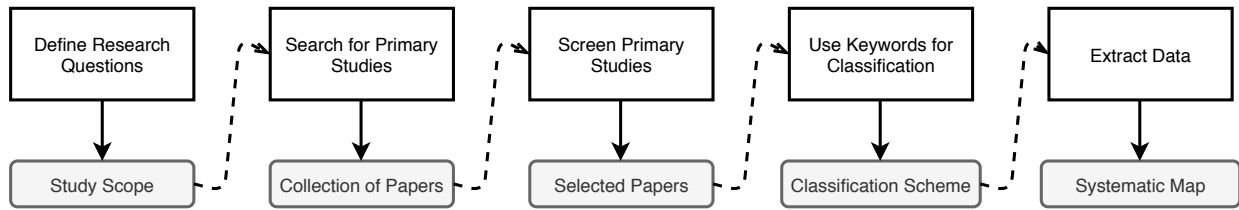
Fig. 1: The Mapping Study Steps Performed in this Study.

steps shown in Figure 1 based on the guidelines of Petersen *et al.* [13]. There are five steps involved in performing this mapping study starting with the definition of the research questions which gives a scope for the search of studies. Based on the scope of the research, a set of search strings are derived and applied in the selected databases to identify studies within this research area. From the identified studies, the most relevant ones are screened by applying certain inclusion and exclusion criteria. We extracted the abstract descriptions, introduction, and conclusion sections of the filtered papers and we studied these to identify the keywords used during the classification stage for answering our research questions. Finally, these papers are studied in-depth to extract the data for each category dimensions and then the obtained results are discussed to show the final systematic map.

A. Definition of the research questions

The main objective of this section is to devise a set of research questions (RQs) based on the definition of ABST and better frame the scope of this research. To this end, we formulate three research questions. The main objective of the first question is to identify the research interest and types of contributions in this domain:

> RQ1: What is the current state of agent-based software testing research?

As RQ1 is a broad research question, thus four sub-questions (RQ1.1– RQ1.4) have been identified:

- RQ1.1: What number of academic studies on agent-based software testing has been published?
- RQ1.2: What are the publication channels used to publish studies on agent-based software testing?
- RQ1.3: What kinds of contributions are provided by studies on agent-based software testing?
- RQ1.4: What research methods have been used in empirical studies on agent-based software testing?

The main objective of the second research question is to identify the method-related characteristics of the agent-based approaches:

> RQ2: What are the characteristics of these agent-based systems used in software testing?

This question is divided into the following three sub-questions focusing on agent-based systems' architecture and implementation:

- RQ2.1: What are the agent architectures used in agent-based software testing?
- RQ2.2: What are the development frameworks used for implementing agent-based software testing?
- RQ2.3: What are the programming languages adopted by agent-based software testing?

The main objective of the third question concerns test-related characteristics of agent-based systems used for software testing:

> RQ3: What are the testing characteristics provided by the agent-based software testing approaches?

This question was divided into three sub questions as follows:

- RQ3.1: What are the testing levels targeted by the agent-based software testing approaches?
- RQ3.2: What are the testing areas targeted by agent-based software testing?
- RQ3.3: What are the types of applications targeted by these agent-based software testing approaches?

B. Search process

Searching for papers is a critical phase in a systematic mapping study as it ensures the comprehensive coverage of the research topic under study. We devised a set of search strings for conducting the search in multiple digital libraries in the ABST research area. Based on the ABST definition we used the following search string: "Agent AND Software AND Testing". The search format varies with different databases as follows:

- IEEE: (("Abstract": software) AND ("Abstract": testing) AND ("Abstract": agent))
- ACM: [Abstract: software] AND [Abstract: testing] AND [Abstract: agent]
- SCOPUS: TITLE-ABS-KEY ("Software" and "testing" and "agent")

The following databases were selected to perform the search: IEEE Xplore digital library, ACM digital library and the SCOPUS scientific database. The selected databases are pertinent to this study as these return the most manually collected publications on ABST. The search string used across these databases retrieved 2663 papers (i.e., 667 papers for IEEE, 860 papers for ACM, and 1136 for Scopus).

We mention here that there are other methods that can be used to improve the coverage of such a mapping study. One

Databases	Initial Search	Screening	Detailed Study
IEEE	667	26	22
ACM	860	9	7
SCOPUS	1136	13	12

TABLE I: Paper Screening.

example of such a method is the snowballing (backward and forward) search [14]. The selection of a start set of papers used to perform snowballing search is one of the main challenges in this procedure. Snowballing is not necessarily an alternative to the database search process. Due to the nature of this study, we used a different approach to check the reliability of the relevant literature and ensure the best possible coverage of the literature. We used the comprehensive related work collected in a paper outlying the area of test agents by Enoiu and Frasheri [15]. We selected these papers and checked their inclusion against the results obtained from our search process. All papers included in this paper have been found using our search strategy and databases. However, a detailed snowballing process could potentially extend the confidence in our search process.

C. Paper Screening

The process of screening was performed to refine the search results by eliminating the duplicated entries and the non-related ABST papers based on certain screening criteria. A set of inclusion and exclusion criteria were applied to the title, abstract, and keywords sections of the papers to identify the relevant ones. If the data from these sections was insufficient to obtain a decision, we used the introduction and conclusion sections to apply the criteria (as shown in Table I). Out of the 2663 papers obtained from the search results a set of 48 papers were selected. These 48 papers were studied in detail to perform the data extraction. Due to lack of relevant data with respect to the research questions seven papers were excluded. Finally a set of 41 papers was obtained. The final set of primary studies is available also as a data set [16].

In the inclusion criteria we focused on the relevant aspects by including: (1) papers that are relevant to using agent-based systems in software testing and (2) papers that include the development and maintenance of agent-based systems used in software testing. In addition we used the following exclusion criteria for: (1) papers that are not relevant to agent-based systems, (2) papers on software testing using search-based techniques using machine learning, (3) papers that are not available in full text, (4) papers on the usage of agents to maintain and increase the system performance or other criteria rather than testing the software systems, and (5) papers on testing agent-based systems rather than the usage of these systems in software testing.

D. Classification

The classification scheme is composed of the following facets based on our research questions: publication and research trends, agent characteristics and testing characteristics. The results obtained in this study are based on manual data

extraction and analysis which is done iteratively and revised continuously to mitigate the risk of missing relevant data. The classification scheme used for data extraction is developed based on the revised keywords and adaptations of the commonly accepted and used categorizations. These categories are well established in both the agent and testing areas of research.

1) Publication and Research Trends: We used the following categories for extracting data from each paper in relation to RQ1:

- (RQ1.1) Publication rate. It identifies the number of publications in ABST area within a specified period of time. The period of time is defined from the publication dates of the included publications.
- (RQ1.2) Publication type. It identifies the channels of publications in ABST area (e.g., journal papers, conference papers, workshop papers).
- (RQ 1.3) Research Contributions. An existing classification of research approaches by Wieringa et al. [13], [17] was used in this mapping study: validation research, evaluation research, solution proposal, philosophical papers, opinion papers and experience papers.
- (RQ 1.4) Empirical Research Methods. For the empirical studies concerned with ABST we identified three different study types: experiment, case study or comparative study.

2) Agent characteristics: This category covers the characteristics of the agent-based systems (RQ2) used for software testing. It consists of the following sub-facets corresponding to each sub-research question:

- (RQ 2.1) Agent Architecture. To have an overview of the system structure, it is important to understand its architecture, showing how the parts of an agent system interact [18] with each other. The agent architecture is considered as the functional controller of an agent involved in making decisions and reasoning to solve problems and achieving goals [19]. Over the years, several surveys and reviews [18], [19] have been conducted on agent architectures. The agent architecture classification used in this study was based on the classification scheme proposed by Friedenberg and Silverman [20]:
  - Simple Reactive. Agents in this kind of system are based on response and reasoning facilities similar to that of logical sequences. Nevertheless, in unpredictable and dynamic circumstances they lack decision making capabilities and rely on other agents.
  - Reactive/Subsumptive. In this architecture, agents are viewed as a collection of simple behavior modules that have a hierarchical organization.
  - Belief-Desire-Intention (BDI). In this architecture, agents can store the state of their system and environment (beliefs), maintain goals (desires), and contain means to convert the belief and desires into actions (intentions).
  - Deliberative. Agents have the potential to solve complex problems, including provision for planning,

and can perform a sequence of actions to achieve a certain goal. These agents take advantage of contemporary AI technologies (e.g., neural nets, Fuzzy Logic).

– Blackboard/Co-Operative. These agents act as a team, where they obtain the state knowledge of the system through a central common agent (blackboard).

- (RQ 2.2) Development Frameworks and Tools. This category identifies the development tools used for each ABST approach. An agent-based system can be designed and built with a help of a toolkit or a framework. Several reviews have been conducted to identify the tools that are used for developing intelligent agent technologies [21], [22]. JADE, JADEX, JACK and MADKIT are examples of agent development frameworks. This category aims to identify the development frameworks used for implementing ABST approaches.

- (RQ 2.3) Programming Language. This category identifies the commonly used programming languages targeted by agent-based systems [1].

3) Testing Characteristics: This category covers the following test-related characteristics of the agent-based systems (RQ3) used for software testing:

- (RQ 3.1) Testing Level. This category classified the papers based on the level of abstraction at which ABST is performed. For our purpose we used the following testing levels by adapting some existing classifications [1], [23]:

– Acceptance Testing. It assesses the software with respect to its customer requirements.

– System Testing. It assesses the software with respect to its architectural design.

– Integration Testing. It checks the proper integration of lower units and the correct operation.

– Unit/Component Testing. It assesses software with respect to its implementation.

- (RQ 3.2) Testing Areas. The following classification of testing areas is based on the categorization proposed by Spillner et al. [23]:

– Functional Testing. Test cases are created based on the functional requirements of the system.

– Non-Functional Testing. Test cases are designed based on attributes describing the system as a whole and some of its non-functional characteristics such as reliability, usability and performance.

– White-Box/Structural Testing. It uses the internal structure of the system for creating test cases.

– Regression Testing. The purpose of regression testing is to check whether changes to existing software have introduced errors to functionality that performed correctly in the software's previous version.

---

[1]Some of the popular programming languages are Java, Python, C, Ruby, and C#. Data according to L. Kim, "10 Most Popular Programming Languages Today, Available: http://www.inc.com/larry-kim/10-Most-Popular-Programming-Languages-Today.html. [Accessed 11 April 2020].
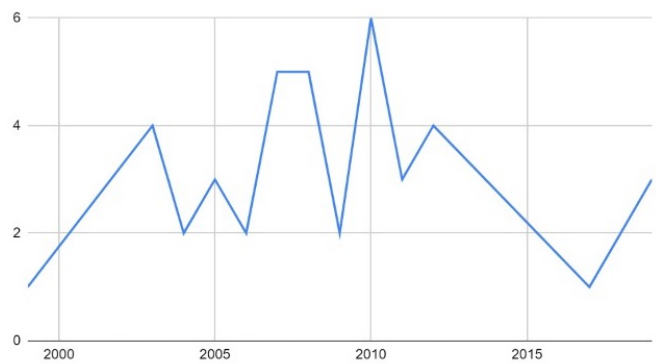

Fig. 2: ABST publications by year.

- (RQ 3.3) Types of Applications Under Test. This category identifies the type of applications that are tested using agent-based systems. Agents are used in a wide range of application domains. We extracted data from each paper related to the application under test. The identified types are: web based applications, network applications, industrial applications, distributed software applications.

Thus, a simple classification scheme has been formed based on the research questions, which allows for a direct extraction of data items to each facet and reduces the complications of further deeper analysis for the comparison of the results.

E. Data extraction and Mapping

A final set of papers was obtained by applying the inclusion and exclusion criteria. These papers were studied in detail to extract the data under each category of the classification scheme using an excel sheet. Thus, for every paper that was studied, the respective column details were entered for each facet. Once the data items were identified, a systematic map was developed.

IV. DATA ANALYSIS AND RESULTS

The results of this mapping study were analyzed both quantitatively and qualitatively. A quantitative analysis gives the quantitative results under each category whereas the qualitative analysis is used to extract information based on interpretation.

A. Publication and Research Trends

The aim of this research question is to establish the annual number of academic studies on ABST as well as the main channels where ABST approaches are disseminated.

1) RQ 1.1: Figure 2 lists the number of publications by year of the primary studies from 1999 until 2020. This categorisation is valuable as it indicates that although the number of academic studies on ABST remains rather low, there is a continuous interest over the last decade (peaking in 2010).

2) RQ 1.2: The aim of this research question is to identify the main channels where ABST studies are disseminated. The results suggest that 78% of the primary papers were published in peer-reviewed international conferences, 17% in journals and just 5% were published in workshops.

Architecture	Primary Studies
Reactive	[24], [25], [26], [27], [28], [29], [9], [10], [30], [31], [11], [32], [33], [34], [35], [36]
BDI	[37], [38], [39], [40], [41], [42], [43], [44]
Co-Operative	[45], [46], [47], [48], [49], [15], [50], [51]
Deliberative	[52], [53], [54], [55], [3], [56], [57], [58]

TABLE II: Agent Architectures used in ABST research.

3) RQ 1.3: The classification results based on the research contributions show that 50% (21 papers) of the studies proposed an ABST solution, while nearly 40% (16 papers) have contributed by validating ABST approaches and only 10% (4 papers) have evaluated their proposed ABST solutions. These results highlight a need for ABST research to provide more significant practical contributions, as well as widening the proposed academic solutions towards practical evaluations.

4) RQ 1.4: The analysis of the results related to the empirical research methods used in ABST research shows that 23 studies have conducted experiments in this research area, 4 studies used the case study method while 2 papers perform a comparative study. On the other hand, 32% of the studies did not use any empirical research methods.

> *Answer RQ1: There is a continuous interest in ABST after 1999 with the majority of studies being disseminated in conferences and proposing both solutions and performing empirical studies.*

B. Agent Characteristics

The analysis under this category gives an overall understanding of the agent-based system characteristics used in ABST.

1) RQ 2.1: The aim of this research question is to categorise studies on ABST based on key agent architectures emerging from the papers being studied. We show the overall results in Table II. 39% of the primary studies are categorized as using the reactive architecture, 60% (20% for each of the other categories) of the studies used BDI, deliberative and co-operative architectures. For one study [59] we could not categorize the ABST approach based on the information provided on the used agent architecture.

2) RQ 2.2: The aim of this research question is to identify the reported frameworks used for developing ABST solutions. In Table III we show seven agent-development frameworks used in ABST. In 61% of the studies, the ABST solutions are proposed as methods and not implemented in practice. Our results suggest that the most common platform used for ABST development is JADE (10 papers). Some other platforms used are ADK, IBM Aglet, JADE LEAP and JADEX (extensions of JADE), FIPA OS and JACK Intelligent Agent. The scarcity of ABST solutions being implemented would indicate that this research is currently rather restricted and has not yet scaled to practical and industrial use cases.

3) RQ 2.3: The aim of this research question is to categorise studies on ABST based on programming languages targeted by these agent-based systems emerging from the papers

being studied. Our results suggest that in more than half of the papers (51%), ABST targets the Java programming language. In almost 37% of the papers there is no explicit mention of any programming language used. Other programming languages like C, C++, Python, and Perl are rarely targeted.

> *Answer RQ2: Research in ABST has used reactive, BDI, deliberative and cooperative agent architectures. The majority of these solutions are targeting the Java programming language and are using JADE framework.*

C. Testing Characteristics

1) RQ 3.1: The aim of this research question is to identify the reported testing level at which ABST approaches are used. The overall results are shown in Figure 3. Our results suggest that most of the ABST studies (i.e., 28 papers) are targeting system level testing. In addition, 14 of the studies are performing ABST at unit level while just two studies are on integration level testing [2].

2) RQ 3.2: The aim of this research question is to categorise studies on ABST based on the testing areas emerging from the papers being studied. As shown in Figure 3, most of the ABST approaches are targeting functional testing (18 studies), white-box testing (19 studies) and non-functional testing (13 studies). The scarcity of ABST research in regression testing (4 studies) indicates that this area of research has not scaled to regression test selection and continuous integration practices.

3) RQ 3.3: The aim of this research question is to identify the reported types of applications targeted by the ABST approaches. In Figure 3 we show the overall results of this classification. The results suggest that ABST is mostly used in software testing of distributed software applications (46% of the studies). In addition, ABST targets other application domains such as web-based applications (19.5% of the studies), network applications (almost 15% of the studies), and industrial automation systems (19.5% of the studies).

> *Answer RQ3: Most of the ABST approaches are used to perform functional and white-box testing at system level and are targeting web-based and distributed software applications.*

V. RELATED WORK

Several mapping studies have been conducted in the software testing research area [60]–[63] to identify the level of growth, usage, and effects of different types of testing in various application domains. The usage of various technologies for efficient testing has also been studied and mapped.

Cruz et al. [64] found that software testing is one of the areas with the most secondary studies and replications in software engineering. Additionally, a mapping study performed by Engström *et al.* [60] identified the potential gap

---

[2]We note here that some ABST approaches focused on multiple testing abstractions levels.

Frameworks	Description	Primary Studies
ADK	ADK (Agent Development Kit) is a commercial agent platform that mainly emphasizes on the mobility and security aspects.	[37]
AGLET	A java mobile agent platform.	[26]
FIPA OS	An open-source FIPA compliant software framework using a simple task-based approach as an internal agent structure.	[24]
JACK	A framework for multi-agent system development for BDI software.	[57]
JADE	JADE (Java Agent DEvelopment Framework) is a framework used to build multi-agent systems implemented in Java programming language and complies with the FIPA specifications.	[48], [39], [40], [28], [55], [3], [10], [32], [36], [42]
JADEX	A JADE extension developed for rational agents, the BDI model and focusing on web services.	[41]
JADE LEAP	JADE-LEAP is a modified version of the JADE platform that can be used in any devices which are java enabled.	[37], [52]

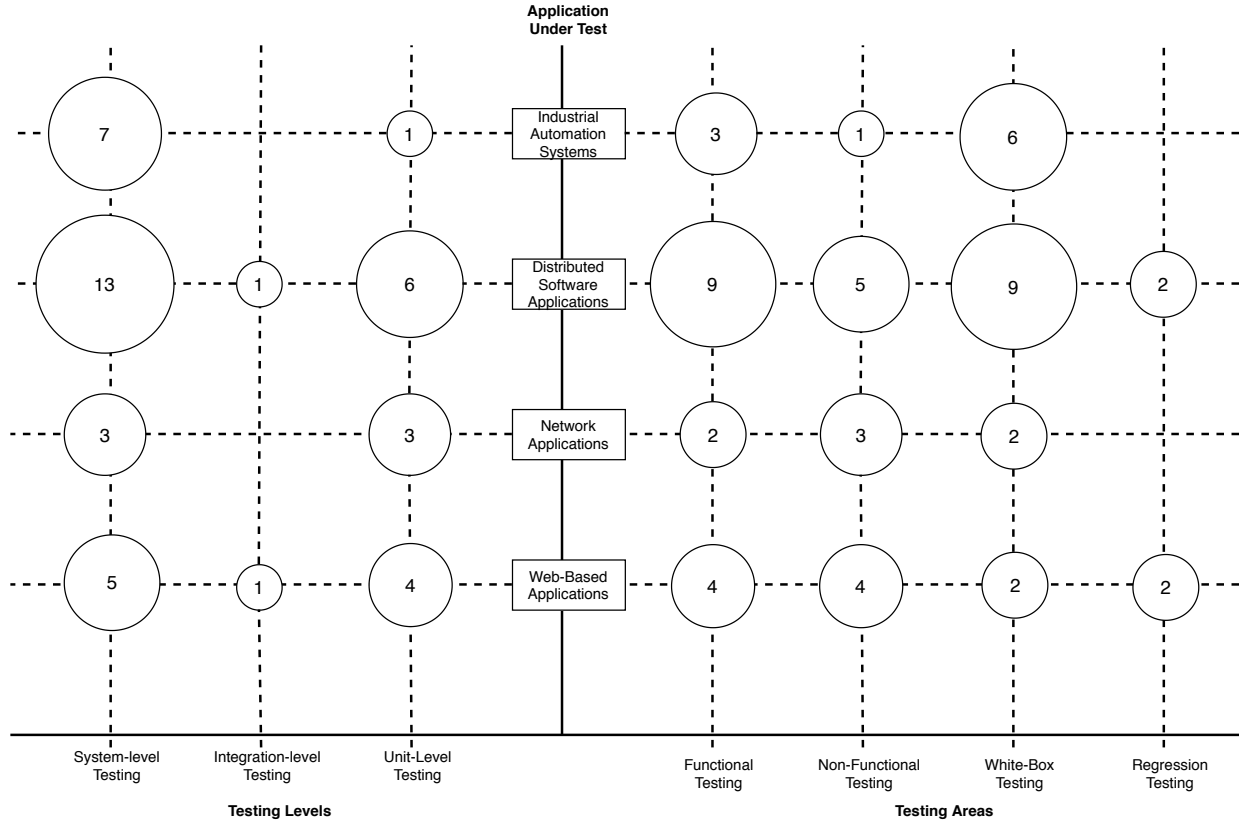TABLE III: Agent Development Frameworks and Tools.

Fig. 3: Distribution of ABST research over a range of testing characteristics, testing levels and applications.

between research and practice in software testing. Several mapping studies have been conducted to study the testing trends in various application domains for testing such as cloud testing [65], web testing [61], combinatorial testing [63] and agile testing [62]. These studies aim at identifying the gaps in current testing research, as well as trends and directions for future testing research. Recently, test automation and intelligent testing using search-based approaches have been used to improve test efficiency and effectiveness. One of these directions relates to the use of agents in software testing. Norouzi *et al.* [66] conducted a survey to identify the different behaviors, forms and other trends in the use of intelligent agents across various domains like education and virtual assistance. To summarize, to the best of our knowledge, this is the first mapping study on agent-based software testing and more studies are needed to improve the study of ABST and its wider application to software development problems.

VI. CONCLUSION AND FUTURE WORK

The use of agent-based systems in software testing and test automation is a growing area of research that should be recognized and considered. Therefore, in this study we present a definition of the agent-based software testing (ABST) research area and conduct a systematic mapping study based

on several identified concepts and dimensions related to agent-based systems and software testing.

The definition highlights that ABST is the application of agents (e.g., software agents, intelligent agents, autonomous agents, multi-agent systems) to software testing problems by automating complex testing tasks. We found a total of 41 related papers in ABST research area. Our results suggest that there is an interest in this area after 1999 that resulted in the development of solutions using reactive, BDI, deliberative and cooperate agent architectures for software testing. The majority of the implemented ABST approaches are designed using the JADE framework and are using the Java programming language. In addition, our results suggest that the majority of the ABST approaches are used at system-level testing for functional, non-functional and white-box testing. The results of our systematic map also indicate that the current body of knowledge concerning ABST does report only four studies on regression testing. Future work could include the extension of this systematic mapping study into a systematic literature review.

REFERENCES

[1] P. Ammann and J. Offutt, *Introduction to software testing*. Cambridge University Press, 2016.

[2] K. S. Decker and K. Sycara, "Intelligent adaptive information agents," *Journal of Intelligent Information Systems*, vol. 9, no. 3, pp. 239–260, 1997.

[3] C. Malz, N. Jazdi, and P. Gohner, "Prioritization of test cases using software agents and fuzzy logic," in *International Conference on Software Testing, Verification and Validation*. IEEE, 2012, pp. 483–486.

[4] M. Wooldridge, "Agent-based software engineering," *IEE Proceedings-software*, vol. 144, no. 1, pp. 26–37, 1997.

[5] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The knowledge engineering review*, vol. 10, no. 2, pp. 115–152, 1995.

[6] P. Balaji and D. Srinivasan, "An introduction to multi-agent systems," in *Innovations in multi-agent systems and applications-1*. Springer, 2010, pp. 1–27.

[7] M. Winikoff, "Future directions for agent-based software engineering." *IJAOSE*, vol. 3, no. 4, pp. 402–410, 2009.

[8] L. Erlenhov, F. G. de Oliveira Neto, R. Scandariato, and P. Leitner, "Current and future bots in software development," in *International Workshop on Bots in Software Engineering (BotSE)*. IEEE, 2019, pp. 7–11.

[9] J. Tang, "Towards automation in software test life cycle based on multi-agent," in *International Conference on Computational Intelligence and Software Engineering*. IEEE, 2010, pp. 1–4.

[10] T. U. Salima, A. Askarunisha, and N. Ramaraj, "Enhancing the efficiency of regression testing through intelligent agents," in *International Conference on Computational Intelligence and Multimedia Applications*, vol. 1. IEEE, 2007, pp. 103–108.

[11] P. Dhavachelvan and G. Uma, "Multi-agent-based integrated framework for intra-class testing of object-oriented software," *Applied Soft Computing*, vol. 5, no. 2, pp. 205–222, 2005.

[12] F. Wotawa, M. Nica, and N. Kushik, *Testing Software and Systems: 28th IFIP WG 6.1 International Conference, ICTSS 2016, Graz, Austria, October 17-19, 2016, Proceedings*. Springer, 2016, vol. 9976.

[13] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th International Conference on Evaluation and Assessment in Software Engineering*, 2008, pp. 1–10.

[14] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

[15] E. Enoiu and M. Frasheri, "Test agents: The next generation of test cases," in *International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2019, pp. 305–308.

[16] P. P. Kumaresen, M. Frasheri, and E. P. Enoiu, "Data Set Primary Studies for Agent-Based Software Testing: A Systematic Mapping Study," Jun. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3874307

[17] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," *Requirements engineering*, vol. 11, no. 1, pp. 102–107, 2006.

[18] R. Girardi and A. Leite, "A survey on software agent architectures." *IEEE Intelligent Informatics Bulletin*, vol. 14, no. 1, pp. 8–20, 2013.

[19] K. O. Chin, K. S. Gan, R. Alfred, P. Anthony, and D. Lukose, "Agent architecture: An overviews," *Transactions on science and technology*, vol. 1, no. 1, pp. 18–35, 2014.

[20] J. Friedenberg and G. Silverman, *Cognitive science: An introduction to the study of mind*. Sage, 2011.

[21] T. Rendón Sallard and M. Sànchez-Marrè, "A review on multi-agent platforms and environmental decision support systems simulation tools," 2006.

[22] F. Leon, M. Paprzycki, and M. Ganzha, "A review of agent platforms," *Multi-Paradigm Modelling for Cyber-Physical Systems (MPM4CPS), ICT COST Action IC1404*, pp. 1–15, 2015.

[23] A. Spillner, T. Linz, and H. Schaefer, *Software testing foundations: a study guide for the certified tester exam*. Rocky Nook, Inc., 2014.

[24] J. Lanslots and A. Vecchio, "Using intelligent agents for distributed measurements," in *IEEE International Symposium on Intelligent Signal Processing*. IEEE, 2003, pp. 49–54.

[25] H. F. El Yamany, M. A. Capretz, and L. F. Capretz, "A multi-agent framework for testing distributed systems," in *30th Annual International Computer Software and Applications Conference*, vol. 2. IEEE, 2006, pp. 151–156.

[26] J. Zhang and D. Xu, "A mobile agent-supported web services testing platform," in *International Conference on Embedded and Ubiquitous Computing*, vol. 2. IEEE, 2008, pp. 637–644.

[27] N. Friess, H. Crawford, and J. Aycock, "A multi-agent approach to testing anti-spam software," in *International Workshop on Database and Expert Systems Application*. IEEE, 2009, pp. 38–42.

[28] U. Manzoor, J. Irfan, and S. Nefti, "Autonomous agents for testing and verification of software after deployment over network," in *World Congress on Internet Security*. IEEE, 2011, pp. 36–41.

[29] G. Chen, S. Ma, and M. Xia, "A kind of software fault diagnosing framework based on multi-agent," in *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering*. IEEE, 2012, pp. 760–762.

[30] J. Grundy, G. Ding, and J. Hosking, "Deployed software component testing using dynamic validation agents," *Journal of Systems and Software*, vol. 74, no. 1, pp. 5–14, 2005.

[31] P. Dhavachelvan, G. Uma, and V. Venkatachalapathy, "A new approach in development of distributed framework for automated software testing using agents," *Knowledge-Based Systems*, vol. 19, no. 4, pp. 235–247, 2006.

[32] M. G. Devasena and M. Valarmathi, "Multi agent based framework for structural and model based test case generation," *Procedia engineering*, vol. 38, pp. 3840–3845, 2012.

[33] H. Zhu, "Cooperative agent approach to quality assurance and testing web software," in *International Computer Software and Applications Conference*, vol. 2. IEEE, 2004, pp. 110–113.

[34] N. C. Narendra, "Large scale testing of pervasive computing systems using multi-agent simulation," in *International Workshop on Intelligent Solutions in Embedded Systems*. IEEE, 2005, pp. 27–38.

[35] S. K. Gardikiotis, V. S. Lazarou, and N. Malevris, "Employing agents towards database applications testing," in *International Conference on Tools with Artificial Intelligence*, vol. 1. IEEE, 2007, pp. 173–180.

[36] A. D. da Costa, C. Nunes, V. T. da Silva, B. Fonseca, and C. J. de Lucena, "Jaaf+ t: a framework to implement self-adaptive agents that apply self-test," in *Proceedings of the Symposium on Applied Computing*, 2010, pp. 928–935.

[37] L. Miclea, E. Szilard, G. Toderean, A. Benso, and P. Prinetto, "Agent based dbist/dbisr and its web/wireless management," in *International Test Conference*, vol. 2. IEEE, 2003, pp. 131–139.

[38] D. Kung, "An agent-based framework for testing web applications," in *Proceedings of the 28th Annual International Computer Software and Applications Conference*, vol. 2. IEEE, 2004, pp. 174–177.

[39] B. Ma, B. Chen, X. Bai, and J. Huang, "Design of bdi agent for adaptive performance testing of web services," in *International Conference on Quality Software*. IEEE, 2010, pp. 435–440.

[40] C. Zhao, G. Ai, X. Yu, and X. Wang, "Research on automated testing framework based on ontology and multi-agent," in *International Symposium on Knowledge Acquisition and Modeling*. IEEE, 2010, pp. 206–209.

[41] C. D. Nguyen, A. Perini, and P. Tonella, "ecat: a tool for automating test cases generation and execution in testing multi-agent systems," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: demo papers*. Citeseer, 2008, pp. 1669–1670.

[42] X. Bai, B. Chen, B. Ma, and Y. Gong, "Design of intelligent agents for collaborative testing of service-based systems," in *Proceedings of the 6th International Workshop on Automation of Software Test*, 2011, pp. 22–28.

[43] M. Winikoff, "Bdi agent testability revisited," *Autonomous Agents and Multi-Agent Systems*, vol. 31, no. 5, pp. 1094–1132, 2017.

[44] S. Enyedi, L. Miclea, and I. Stefan, "Agent-based testing and repair of heterogeneous distributed systems," in *International Conference on Automation, Quality and Testing, Robotics*, vol. 1. IEEE, 2008, pp. 104–108.

[45] Q. Huo, H. Zhu, and S. Greenwood, "A multi-agent software engineering environment for testing web-based applications," in *International Computer Software and Applications Conference*. IEEE, 2003, pp. 210–215.

[46] L. Liu, X.-S. Zhou, J.-H. Gu, and Z.-Y. Yang, "Agent-based automated compatibility software test for nlsf," in *International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693)*, vol. 4. IEEE, 2003, pp. 1986–1989.

[47] Y. Chengqing, W. Yinglong, and W. Jizhi, "A ipv6 network performance test system using multi-agent," in *International Conference on Electronic Measurement and Instruments*. IEEE, 2007, pp. 2–113.

[48] J. Gao and Y. Lan, "Automatic test task allocation in agent-based distributed automated testing framework," in *2009 International Conference on Computational Intelligence and Software Engineering*. IEEE, 2009, pp. 1–5.

[49] C. V. Jordan, F. Mäurer, S. Löwenberg, and J. Provost, "Framework for flexible, adaptive support of test management by means of software agents," *Robotics and Automation Letters*, vol. 4, no. 3, pp. 2754–2761, 2019.

[50] R. Maamri and Z. Sahnoun, "Maest: multi-agent environment for software testing," *Journal of Computer Science*, vol. 3, no. 4, pp. 249–258, 2007.

[51] G. Farias, A. Dantas, R. Lopes, and D. Guerrero, "Distributed test agents: a pattern for the development of automatic system tests for distributed applications," in *Proceedings of the 9th Latin-American Conference on Pattern Languages of Programming*, 2012, pp. 1–11.

[52] D. J. Mala and V. Mohan, "Intelligentester-software test sequence optimization using graph based intelligent search agent," in *International Conference on Computational Intelligence and Multimedia Applications*, vol. 1. IEEE, 2007, pp. 22–27.

[53] C. Malz and N. Jazdi, "Agent-based test management for software system test," in *International Conference on Automation, Quality and Testing, Robotics (AQTR)*, vol. 2. IEEE, 2010, pp. 1–6.

[54] M. A. B. Júnior, F. B. de Lima Neto, and J. C. S. Fort, "Improving black box testing by using neuro-fuzzy classifiers and multi-agent systems," in *International Conference on Hybrid Intelligent Systems*. IEEE, 2010, pp. 25–30.

[55] C. Malz and P. Göhner, "Agent-based test case prioritization," in *International Conference on Software Testing, Verification and Validation Workshops*. IEEE, 2011, pp. 149–152.

[56] D. A. Ostrowski and R. G. Reynolds, "Knowledge-based software testing agent using evolutionary learning with cultural algorithms," in *Congress on Evolutionary Computation-CEC*, vol. 3. IEEE, 1999, pp. 1657–1663.

[57] D. J. Mala and V. Mohan, "Intelligentester-test sequence optimization framework using multi-agents." *JCP*, vol. 3, no. 6, pp. 39–46, 2008.

[58] S. Karlsson, "Exploratory test agents for stateful software systems," in *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 1164–1167.

[59] A. V. Mirgorodskiy and B. P. Miller, "Diagnosing distributed systems with self-propelled instrumentation," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2008, pp. 82–103.

[60] E. Engström and K. Petersen, "Mapping software testing practice with software testing research—serp-test taxonomy," in *International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2015, pp. 1–4.

[61] S. Akbulut, Y. T. Gebreyesus, A. Mishra, and A. Yazici, "Systematic mapping on quality in web application testing," in *International Informatics and Software Engineering Conference (UBMYK)*. IEEE, 2019, pp. 1–5.

[62] T. D. Hellmann, A. Chokshi, Z. S. H. Abad, S. Pratte, and F. Maurer, "Agile testing: a systematic mapping across three conferences: understanding agile testing in the xp/agile universe, agile, and xp conferences," in *Agile Conference*. IEEE, 2013, pp. 32–41.

[63] R. E. Lopez-Herrejon, S. Fischer, R. Ramler, and A. Egyed, "A first systematic mapping study on combinatorial interaction testing for software product lines," in *Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2015, pp. 1–10.

[64] M. Cruz, B. Bernárdez, A. Durán, J. A. Galindo, and A. Ruiz-Cortés, "Replication of studies in empirical software engineering: A systematic mapping study, from 2013 to 2018," *IEEE Access*, vol. 8, pp. 26 773–26 791, 2019.

[65] A. A.-S. Ahmad, P. Brereton, and P. Andras, "A systematic mapping study of empirical studies on software cloud testing methods," in *International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2017, pp. 555–562.

[66] N. Norouzi, K. Kim, J. Hochreiter, M. Lee, S. Daher, G. Bruder, and G. Welch, "A systematic survey of 15 years of user studies published in the intelligent virtual agents conference," in *Proceedings of the 18th international conference on intelligent virtual agents*, 2018, pp. 17–22.