

TASTE

Transformative AI-Assisted Testing in Industrial Mobile Robotics

Programm / Ausschreibung	Expedition Zukunft, Expedition Zukunft 2022, Expedition Zukunft Wissenschaft 2022	Status	laufend
Projektstart	01.10.2024	Projektende	31.12.2026
Zeitraum	2024 - 2026	Projektlaufzeit	27 Monate
Keywords	Software Testing, Robotics, Reinforcement Learning, Model-Based Learning		

Projektbeschreibung

Motivation: Das Testen hat in der Softwareentwicklung in den letzten Jahren deutlich an Bedeutung gewonnen. Mit dem Ziel, die Performanz, Sicherheit, Nutzbarkeit und Funktionalität von Software zu erhöhen, wird geprüft, ob beim Einsatz von Software Fehler oder Mängel auftreten. Da beim Testen von Software immer nur eine begrenzte Anzahl an Tests durchgeführt wird, kann man sich nie zu 100% sicher sein, dass Software fehler- und mängelfrei ist. Abhängig vom Einsatzbereich der Software können Fehler und Mängel drastische Auswirkungen haben. Aus diesem Grund wird Software in der Praxis umfassend getestet, wobei die Tests größtenteils manuell konzipiert und durchgeführt werden. Dies führt dazu, dass etwa 40-50% des Aufwands bei der Softwareentwicklung im Bereich des Testens anfällt. Bei sicherheitskritischer Software, wie sie etwa bei autonomen mobilen Robotern (AMR) zum Einsatz kommt, ist dieser Anteil sogar noch größer.

Problem: TASTE konzentriert sich auf Software zur Steuerung von AMRs, da dieser Bereich rasant wächst, eine besonders hohe Softwarequalität erfordert und aktuelle Ansätze, bei denen Tests weitgehend manuell konzipiert und durchgeführt werden, nicht zufriedenstellend sind. Es wird zwar schon seit einiger Zeit an der Automatisierung des Testens von Software gearbeitet, etwa mit modellbasiertem Automata Learning (AL) oder maschinellem Lernen auf der Grundlage von Reinforcement Learning (RL), aber durchschlagende Erfolge zeichnen sich bisher nicht ab. Die Ansätze kommen beispielsweise mit der hohen Komplexität von Software nicht zu recht (im Fall von AL) oder konvergieren wegen mangelndem Feedback zu Entscheidungen nur langsam (im Fall von RL). Darüber hinaus sind die Ergebnisse von Ansätzen, die auf maschinellem Lernen beruhen, meist nicht erklärbar. Die Konsequenz ist, dass das Testen auch bei sicherheitskritischer Software immer noch weitgehend manuell erfolgt.

Ziel: Ziel von TASTE ist es, den Aufwand für die Entwicklung und das Testen von Software zu reduzieren, ohne die Performanz, Sicherheit, Nutzbarkeit und Funktionalität der Software zu beeinträchtigen. Um dieses Ziel zu erreichen, wird eine Testmethodik entwickelt, die modellbasierte (AL) und modellfreie (RL) Ansätze miteinander kombiniert. Die Methodik nutzt die Stärken von AL, um die Herausforderungen von RL in den Bereichen Konvergenz und Erklärbarkeit zu bewältigen und die Stärken von RL, um auch bei hoher Softwarekomplexität effizient optimale Teststrategien zu entwickeln. Darüber hinaus unterstützt die entwickelte Testmethodik bei gefundenen Fehlern und Mängeln die Ursachenanalyse. Dadurch wird

nicht nur die effiziente Beseitigung von Fehlern und Mängeln erleichtert, sondern auch eine wichtige Grundlage zur Verbesserung von Programmierassistenten geschaffen, die auf generativen Verfahren der künstlichen Intelligenz (KI) beruhen und zunehmend in der Softwareentwicklung zum Einsatz kommen.

Ergebnisse: Zentrales Ergebnis von TASTE ist eine Testmethodik für Software, wobei dem Einsatz der Methodik im Kontext von AMRs besondere Aufmerksamkeit zukommt. Die Methodik wird im Idealfall in der Lage sein, effizient optimale Teststrategien zu entwickeln und abzuarbeiten, wobei (1) die Dynamik eines Systems berücksichtigt wird (z.B. ein Roboter bewegt sich durch eine sich verändernde Umwelt), (2) die hohe Dimensionalität komplexer Umgebungen kein limitierender Faktor ist und (3) tiefe Einblicke in die Ursache von Fehlern und Mängeln sowie die Auswirkungen von Maßnahmen zu ihrer Beseitigung gegeben werden. Die Anwendbarkeit und Effektivität der Testmethodik wird anhand von realen Anwendungsfällen aus dem AMR-Umfeld evaluiert. Bei der Evaluierung liegt der Fokus gleichermaßen auf technischen wie auf wirtschaftlichen und rechtlichen Aspekten. Auf Basis der Evaluierung werden die Weichen für eine umfassende wirtschaftliche und wissenschaftliche Verwertung der Ergebnisse gestellt.

Innovation: Das besondere Innovationspotenzial von TASTE ergibt sich aus der Kombination vielversprechender Methoden zur Bewältigung von zentralen Herausforderungen im Bereich des Testens von Software. Software nimmt in unserer Gesellschaft bereits heute immense Bedeutung ein. Viele sicherheitskritische Bereiche wären ohne Softwareeinsatz undenkbar. Entsprechend hoch ist der Aufwand, der in das Testen von Software fließt. Mit neuen Anwendungen wie AMR, das mit dem autonomen Fahren, dem in der Öffentlichkeit besonders viel Aufmerksamkeit zukommt, eng verwandt ist, werden grundlegende Verbesserungen im Bereich des Testens notwendig. Die angestrebten Ergebnisse versprechen Verwertungspotenzial auf unterschiedlichen Ebenen: (1) Softwareentwickler können den Aufwand für das Testen reduzieren ohne Abstriche bei der Softwarequalität in Kauf nehmen zu müssen, (2) Dienstleister können das Testen von Software im Rahmen von Testing-as-a-Service-Geschäftsmodellen anbieten und (3) Wissenschaftler sowie Anbieter von Entwicklungsumgebungen können relevante Aspekte bei der Verbesserung von KI-basierten Programmierassistenten einfließen lassen.

Abstract

Motivation: Software testing has become increasingly important in the development pipeline in recent years. With the goal of increasing the performance, security, usability and functionality of software, testing is used to determine whether errors or defects occur during the use of software. Since testing in practice is necessarily limited by a finite number of tests, one can never be 100% sure that software is completely error and defect free. And, depending on the area of application of the software, errors and defects can have drastic consequences. For this reason, software is tested as extensively as possible, with the tests largely being designed and executed manually. The consequence is that approximately 40-50% of the efforts in software development occur in the area of testing. For safety-critical software, such as that used in autonomous mobile robots (AMRs), this share is even greater.

Problem: TASTE focuses on software for controlling AMRs because it is a field growing rapidly, requires particularly high software quality, and the current approach in which tests are largely designed and executed manually are not satisfactory. Work on automating the testing of software has been ongoing for some time, for example with model-based automata learning (AL) or machine learning based on reinforcement learning (RL), but so far there have been no breakthrough successes. For example, the approaches cannot cope with the high complexity of software (in the case of AL) or converge

slowly due to a lack of feedback on decisions (the “sparse reward” in the case of RL). Moreover, the results of approaches based on machine learning are difficult to explain and interpret. The consequence is that testing is still largely manual, even for safety-critical software.

Goal: The goal of TASTE is to reduce the effort required to develop and test software without compromising the performance, security, usability, and functionality of the software. To achieve this goal, a testing methodology is developed that combines model-based (AL) and model-free (RL) approaches. The methodology exploits the strengths of AL to overcome the challenges of RL in the areas of convergence and explainability, and the strengths of RL to efficiently develop optimal testing strategies even in the presence of high software complexity. In addition, the developed testing methodology supports root cause analysis in case of found errors and defects. This not only facilitates the efficient elimination of errors and defects, but also provides an important basis for improving programming assistants that are based on generative artificial intelligence (AI) methods that are increasingly used in software development.

Results: The central result of TASTE is a testing methodology for software, concentrating on the context of AMRs. The methodology will ideally be able to efficiently develop and execute optimal test strategies while (1) taking into account the dynamics of systems (e.g., a robot moving through a changing environment), (2) not allowing the high dimensionality of complex environments to be a limiting factor, and (3) providing insight into the root cause of errors and defects and the impact of actions taken to correct them. The applicability and effectiveness of the test methodology is evaluated using real use cases from the AMR domain. The evaluation will focus equally on technical as well as on economic and legal aspects. Based on the evaluation, the course will be set for a comprehensive economic and scientific exploitation of the results.

Innovation: The innovation potential of TASTE results from the combination of promising methods for overcoming central challenges in the field of software testing. Software already plays an immense role in our society today and many safety-critical areas would be impossible without the use of software. The effort that goes into testing software is thus correspondingly high. With new applications such as AMR, which is closely related to autonomous driving, which is receiving a great deal of public attention, fundamental improvements in the area of testing are necessary. The intended results promise exploitation potential on different levels: (1) software developers can reduce the effort for testing without sacrificing software quality, (2) service providers can offer software testing as part of testing-as-a-service business models, and (3) researchers as well as providers of software development environments can incorporate relevant aspects in the improvement of AI-based programming assistants.

Projektkoordinator

- Fraunhofer Austria Research GmbH

Projektpartner

- ARTI - Autonomous Robot Technology GmbH
- KNAPP Industry Solutions GmbH
- Universität Linz Institute for Formal Models and Verification