



Contents lists available at ScienceDirect

## Science of Computer Programming

journal homepage: [www.elsevier.com/locate/scico](http://www.elsevier.com/locate/scico)

Original software publication

## PX-MBT: A framework for model-based player experience testing

Saba Gholizadeh Ansari<sup>a,\*</sup>, I.S.W.B. Prasetya<sup>a</sup>, Mehdi Dastani<sup>a</sup>, Gabriele Keller<sup>a</sup>,  
Davide Prandi<sup>b</sup>, Fitsum Meshesha Kifetew<sup>b</sup>, Frank Dignum<sup>c</sup><sup>a</sup> Utrecht University, Utrecht, the Netherlands<sup>b</sup> Fondazione Bruno Kessler, Trento, Italy<sup>c</sup> Umeå University, Umeå, Sweden

## ARTICLE INFO

## Keywords:

PX testing

Emotion modeling

Model-based testing

Agent-based testing

## ABSTRACT

As video games become more complex and widespread, player experience (PX) testing becomes crucial in the game industry. Attracting and retaining players are key elements to guarantee the success of a game in the highly competitive market. Although a number of techniques have been introduced to measure the emotional aspect of the experience, automated testing of player experience still needs to be explored. This paper presents PX-MBT, a framework for automated player experience testing with emotion pattern verification. PX-MBT (1) utilizes a model-based testing approach for test suite generation, (2) employs a computational model of emotions developed based on a psychological theory of emotions to model players' emotions during game-plays with an intelligent agent, and (3) verifies emotion patterns given by game designers on executed test suites to identify PX-issues. We explain PX-MBT architecture and provide an example along with its result in emotion pattern verification, which asserts the evolution of emotions over time, and heat-maps to showcase the spatial distribution of emotions on the game map.

## Code metadata

Code metadata description	
Current code version	2.0.2
Permanent link to code/repository used for this code version	<a href="https://github.com/ScienceofComputerProgramming/SCICO-D-23-00335">https://github.com/ScienceofComputerProgramming/SCICO-D-23-00335</a>
Permanent link to Reproducible Capsule	<a href="https://zenodo.org/records/7506758">https://zenodo.org/records/7506758</a>
Legal Code License	Apache License 2.0
Code versioning system used	Git
Software code languages, tools, and services used	Java, Python
Compilation requirements, operating environments and dependencies	Java 11 or higher, Maven 3, Windows/ Mac/Linux
If available, link to developer documentation/manual	<a href="https://github.com/iv4xr-project/eplaytesting-pipeline">https://github.com/iv4xr-project/eplaytesting-pipeline</a>
Support email for questions	<a href="mailto:s.gholizadehansari@uu.nl">s.gholizadehansari@uu.nl</a>

\* Corresponding author.

E-mail address: [s.gholizadehansari@uu.nl](mailto:s.gholizadehansari@uu.nl) (S. Gholizadeh Ansari).<https://doi.org/10.1016/j.scico.2024.103108>

Received 31 October 2023; Received in revised form 2 February 2024; Accepted 18 March 2024

Available online 20 March 2024

0167-6423/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

## 1. Motivation and significance

*Player experience* (PX) testing has become an increasingly critical aspect of game development to assist game designers in anticipating players' experience in terms of enjoyment [1], flow [2] and engagement [3]. While functional testing is intended to test the functionality of the game [4], PX testing verifies whether the emotions and psychology of players shaped during the interaction with the game are close to the design intention. This helps designers in early development stages to identify design issues that may lead to game abandon, improve the general experience of players, and even invoke certain experiences during the game-play [5–7]. However, validating a game design relies either on trained PX testers or acquiring information directly from players with methods such as interviews, questionnaires and physiological measurements [8,9], which are labour-intensive, costly and not necessarily representing all users profiles and their emotions. We address this problem by automated testing using human cognition theories that have been used for decades in cognitive psychology. Theories such as Ortony, Clore, and Collins (OCC) theory [10] provide a coherent outlook of the cognitive process as they have identified common patterns among humans on the emergence of the same emotion. The OCC theory is widely used in modeling emotional agents [11–15] especially when access to sufficient data is not possible. Despite the influence of emotions on forming the experience of players [16,17], such theories have not been employed in PX testing before [18].

To automate PX testing, we present PX-MBT, an agent-based player experience testing framework that allows designers to express emotional requirements as *patterns* and verify them on executed test suites generated by a *model-based testing* (MBT) approach. It allows the designers to gain insight into emotions the game induces over time and space, which ultimately helps them to adjust their design to produce a well-received game product. The framework uses a computational model of emotions based on the OCC theory [19] to generate the emotional experience of agent players on a game under the test (GUT). While [19] focuses on introducing a computational model of emotion named JOCC and its formalization, PX-MBT contributes to providing a PX testing software (see 2.1 for further explanation). This software enables the expression of emotion requirements and the generation of covering test suites to provide diversity in the game behavior and, as a result, in the evoked emotions for verifying the requirements. This is where the model-based testing approach with its fast test suites generation contributes. We employ an *extended finite state machine* (EFSM) model [20] that captures all possible game-play behaviors serving as a subset of human behaviors, at some level of abstraction. We utilize *search based algorithms* (SB), such as MOSA [21] and *model checking* (MC) [22,23] as two model-based test generation strategies, along with a selection algorithm to maximize the test suites diversity. Findings from our 3D game case study [24] support that SB and MC, due to their different search strategies, produce distinctive test cases that can identify different variations of emotions over space and time that cannot be identified by just one of the test suites. The introduction of this software impacts game testing by automating the verification of players' emotions and their experiences. The use of a model-based testing approach, coupled with an emotional test agent, presents the potential for a cost-effective testing alternative to traditional methods. It also opens up a spectrum for potential scientific discoveries in the automation of PX testing.

PX-MBT is an open source framework which is a part of the iv4XR project.<sup>1</sup> The framework is accessible on GitHub, and also includes resources such as examples, models, and documentation. To use the framework on a given GUT, the user needs to provide an EFSM model that at least subsume the scenarios in the GUT they would like to test. Subsequently, they decide about the test suite generation method (e.g. MC or SB). At this point, the user needs to design an agent character to resemble a certain type of player by specifying e.g. the player's goals and their priorities as well as a selection of game events that have impacts on the player's emotions. Generated test cases, then, could be run against the GUT and the computational model of emotions would generate emotions upon perceiving a game event. A trace file of emotions with their intensity is produced for every test case execution. Emotion patterns can then be verified over the produced traces. This paper focuses on the contribution of PX-MBT as software. The methodology underlying the PX-MBT was presented in our prior publication [24].

## 2. Software description

### 2.1. Software architecture

Fig. 1 shows the architecture of the framework with its four main components: The *Basic (test) Agent* component, provided by aplib<sup>2</sup> [25], is responsible for controlling the in-game player-character. The *Model-Based Testing* (MBT) component generates tests using EvMBT<sup>3</sup> [20], incorporating search-based test generation algorithms. A complementary test generation approach by exploiting Linear Temporal Logic model checking [22] (the LTL-MC sub-component) is also provided e.g. for improving coverage and the potential emergence of diverse emotional behaviors. LTL-MC is implemented as a part of aplib in a package called `eu.iv4xr.framework.extensions.ltl`. This was a strategic decision to ease integration with future tools that want to reuse the model checker but not necessarily the PX functionality. The *Model of Emotions* component implements a computational model of emotions. It is provided by jocc.<sup>4</sup> The composition of this model of emotion and aplib basic test agent creates an *Emotional Test Agent*.

<sup>1</sup> <https://iv4xr-project.eu>.

<sup>2</sup> <https://github.com/iv4xr-project/aplib>.

<sup>3</sup> <https://github.com/iv4xr-project/iv4xr-mbt>.

<sup>4</sup> <https://github.com/iv4xr-project/jocc>.

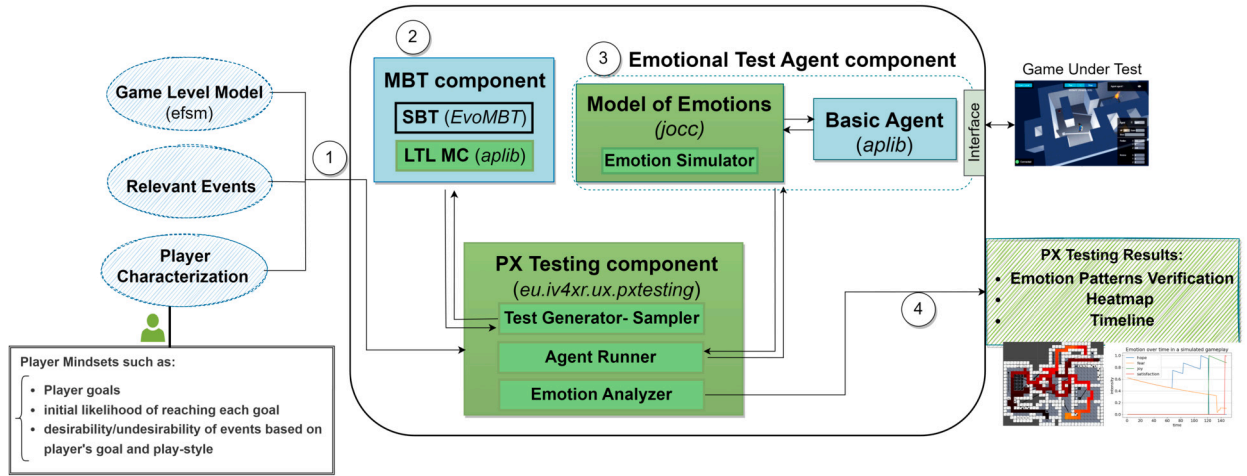


Fig. 1. Architecture of PX-MBT, highlighting original software components in green. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

The *PX Testing* component, located in a package called `eu.iv4xr.ux.pxtesting`<sup>5</sup>, serves as the main interface for game designers to do PX testing on a game through its three key sub-components: the *Generator-Sampler* in `eu.iv4xr.ux.pxtesting` to generate test cases for the GUT using algorithms in the *MBT* component and subsequently apply sampling over the generated test suite. The *Agent Runner* is responsible for configuring and managing the execution of the test agent, while the *Emotion Analyzer* verifies emotion patterns and generates emotion heatmaps and timeline graphs.

The *Interface* component in Fig. 1 allows the GUT to be controlled programmatically. In `aplib`, this interface is referred to as an “environment” which is established by extending a class named `Environment`. The main functionalities to implement are a method to send a command to the GUT and a method to observe its state, subsequently returning observations. Details about building this interface are provided in PX-MBT’s documentations.<sup>6</sup> To use PX-MBT, game designers need to have these inputs; ① in Fig. 1:

- An EFSM that abstractly models the functional behavior of the game.
- A selection of game events that have impacts on the player’s emotions (e.g. defeating an enemy, acquiring gold).
- A so-called *player characterization*, specifying characteristics the designer wants to address in the agent to resemble a certain type of player, such as: a player’s goals and their priorities, the player’s initial mood and beliefs before playing the game, and the desirability of incoming events for the player. E.g. a player might experience positive emotions on defeating an enemy, while another, who avoids conflicts, may find acquiring a gold bar more desirable. For more details, see [24].

Some of these inputs require technical proficiency, such as writing the EFSM, which needs assistance from the technical team (software engineers). However, analyzing graphical results and testing patterns does not require technical expertise, as long as they adhere to the provided instructions.

**Software originality and novelty.** Highlighted parts of the architecture in green in Fig. 1, *Model of Emotions* (`jocc`), LTL-MC sub-component for additional test suite generation approach, and *PX testing* component (`eu.iv4xr.ux.pxtesting`) are original software components. Notably, the *Model of Emotions* and *PX Testing* are novel components contributed to this paper. The software is designed with a modular architecture and housed in separate repositories to facilitate integration and the application of key components in other projects.

## 2.2. Software functionalities

PX-MBT offers the following main functionalities:

1. **Model-based test suite generation.** Given the EFSM model, the *MBT* component, ② in Fig. 1, aids in generating a test suite consisting of abstract test cases to be executed on the GUT. This process utilizes either `EvoMBT`, which employs various search algorithms for test generation, or `LTL-MC` within `aplib` which employs model checking for test generation. The explanation of how these two methods work in the framework can be found in [24].
2. **Abstract test cases translation and execution.** Due to the EFSM model’s abstraction, the translation of abstract test cases into actual instructions is required for obtaining emotion traces during execution. This execution involves the use of a test agent on the GUT. The basic test agent created by `aplib` does the executions.

<sup>5</sup> <https://github.com/iv4xr-project/eplaytesting-pipeline/tree/main/src/main/java/eu/iv4xr/ux/pxtesting>.

<sup>6</sup> <https://github.com/iv4xr-project/eplaytesting-pipeline/blob/main/docs/interface.md>.



Fig. 2. Scenes of the Lab Recruits game.

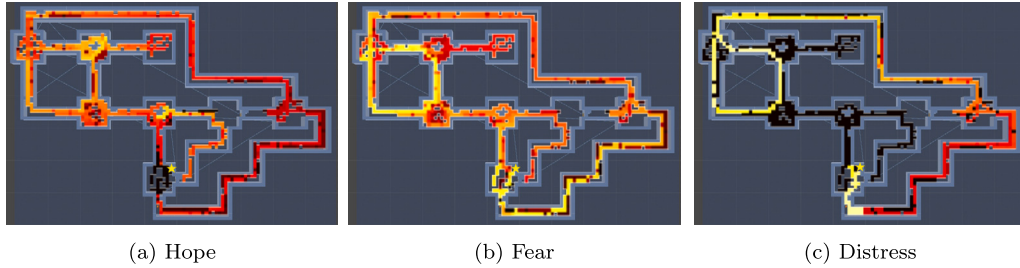


Fig. 3. Aggregated heat-map visualization of the level under test.

3. **Emotion trace generation** Attaching the *Model of Emotions* to the *basic test agent* creates an *emotional test agent*, (3) in Fig. 1, which is able to simulate emotions based on incoming events. Via an interface, the emotional test agent is connected to the GUT. Each test case of the test suite is then given to the agent for execution. The agent computes its emotional state upon observing events and records it in a *trace* file.
4. **Emotion pattern verification.** Finally, when the whole test suite is executed on the GUT, the *PX Testing* component analyzes the traces to verify given emotional requirements and provide heat-maps and timeline graphs of emotions for the given game level ((4) in Fig. 1).
5. **Data Export.** PX-MBT exports a range of artifacts. Test cases are exported as text, graphically through dot syntax, in serialized binary format, and in custom CSV formats. Emotion trace files are exported in CSV, while heat-maps and timeline graphs are saved in PNG.

### 2.3. Building, importing, and using PX-MBT

Instructions on how to build PX-MBT and how to import it into one's own project are provided in its Github location. Its main Readme also links to various documentation on e.g. how to build an interface to connect it to a new game. The documentation also includes few demos.

## 3. Illustrative example

PX-MBT is used to test emotional experience on Lab recruits,<sup>7</sup> a configurable 3D game, designed to enable researchers to define a multi-floor game level, along with its logic to challenge an intelligent agent, e.g. to benchmark if the agent can win the level. An example level of the game (Fig. 2) presents a maze environment with rooms, doors, and hazards in which the agent or human player has to disclose the connection between buttons and doors to reach an end-goal flag, where capturing it represents the objective of a particular level. The EFSM model of the level consists of 35 states and 159 transitions. Here, we use the model checking technique to generate test suites that satisfy the transition coverage with the ending criterion in the end-goal flag, leading to the generation of 74 test cases, within 60 seconds time budget.

Executing the test cases produces emotion traces, from which visualization graphs such as heat-maps can be generated. Fig. 3 shows the aggregated heat-maps of hope, fear, and distress emotions on the example level. The brighter color (e.g. yellow) shows the higher intensity of an emotion. Such visualization can be beneficial. For example, when the designers evaluate their level on the presence of emotions, e.g. fear, and their distribution through the level such as fear through the level, they might aim for the presence of fear in a specific scenario as an intended player experience, whereas its absence in certain places might actually be undesirable.

PX-MBT also performs emotion requirement verification. Table 1 shows a number of emotion patterns expressible in PX-MBT. Such a pattern specifies the order in which different emotions are stimulated, or avoided. For instance,  $Sat(J \neg S)$  is a requirement

<sup>7</sup> <https://github.com/iv4xr-project/labrecruits>.

**Table 1**

Emotion pattern check.  $H$  = hope,  $F$  = fear,  $J$  = joy,  $D$  = distress,  $S$  = satisfaction,  $P$  = disappointment and  $\neg X$  = absence of emotion  $X$ .

Emotion patterns	$TS_{MC}$
$Sat(\neg DS)$	✓
$UnSat(\neg FS)$	✓
$Sat(J \neg S)$	✗
$UnSat(JD)$	✓
$Sat(JFS)$	✓
$Sat(DHP)$	✓
$Sat(DHS)$	✓
$Sat(DH \neg DS)$	✗
$Sat(FDHFJ)$	✗
$Sat(HFDDDHFFJ)$	✗
$Sat(FDDHFP)$	✗

in which the designer expects at least one execution path where joy ( $J$ ) is stimulated at least once through the execution, but the agent never reaches the end-flag goal with satisfaction ( $S$ ). The table shows this requirement has failed, which indicates that the designer needs to put some challenging objects like fire or enemies in the vicinity of the end-goal flag. A video demonstration about the use of the software during this example is available for better comprehension.<sup>8</sup> As explained in Section 2, PX-MBT requires the interface to connect to games, allowing a test agent to control the game. Interfaces built for a small game named MiniDungeon<sup>9</sup> and a commercial game named Space Engineers<sup>10</sup> can serve as examples.

#### 4. Impact

PX-MBT provides a working setup for practitioners to facilitate further research in automated PX testing. Integration with two aforementioned games are included in the project: Lab Recruits and MiniDungeon. These games are highly configurable to create different setups to facilitate experiments with them. Examples of other studies that also make use of these games and exploit their configurability are in [26,19]. Components such as an interface with the GUT, EFSM models, and player characterizations will have to be developed, but these are one-off investments, after which automation is possible. Although such automation is not a replacement of actual user testing, this would enable approximate but fast feedback e.g. after changes in the design, without the designers needing to setup a costly user testing. PX-MBT opens the scope of PX testing and automation. The software can contribute further scientific discoveries in game-testing across multiple dimensions:

- *Verification of PX-testing requirements.* One potential research avenue we are exploring is the formal verification of PX requirements. This involves creating a domain-specific language to express refined temporal and spatial requirements specified by game designers. This language can be inspired by Linear Temporal Logic or Computational Tree Logic [22], traditionally used for functional verification. The goal is to have an intuitive language that allows designers to define complex PX specifications and verify them using the facilities that PX-MBT provides, such as automated model-based test generation and intelligent emotional agents. Moreover, determining a sensible test suite for PX testing and the likelihood of their occurrence impacts the reliability of the verification. One possible approach is to employ probabilistic model checking to assess the likelihood of certain behaviors within the game, allowing a more refined analysis of certain experiences.
- *Advance modeling of players.* PX-MBT provides a foundation for further research in fine-tuning the computational models of emotions and players' character ('player characterization'), by addressing personalities, moods, expectations, and motivation, each serving as a separate domain to explore.
- *Game personalization.* Understanding players' emotional experiences assists in the personalization of games across various aspects, including adaptive game-play and difficulty adjustment. For instance, in the context of dynamic content generation in video games through procedural content generation, understanding players' emotional experiences and testing PX requirements becomes an important consideration.

#### 5. Conclusions and future work

We present PX-MBT, an open source framework for automated player experience testing, in particular automated verification of emotion requirement, using a computational model of emotions and model-based test generation targeting a subset of human players' behaviors. PX-MBT is still actively in development in several directions, including the introduction of a domain-specific language for

<sup>8</sup> <https://youtu.be/iTkawww9mp0?si=uBu90qz0p66zsjWs>.

<sup>9</sup> <https://github.com/iv4xr-project/MiniDungeon>.

<sup>10</sup> <https://github.com/iv4xr-project/iv4xr-se-plugin>.



specifying and verifying emotion requirements with both temporal and spatial constraints to capture complex aspects of emotional experience within games. We are also considering the use of probabilistic models as an interesting future work. This would allow a more refined analysis of PX behavior, e.g. by exploiting probabilistic model checkers such as PRISM [27]. To enhance the translation of models across various model-based tools, a model transformation language such as Xtend [28] can be considered.

### CRedit authorship contribution statement

**Saba Gholizadeh Ansari:** Formal analysis, Investigation, Methodology, Software, Writing – original draft, Writing – review & editing. **I.S.W.B. Prasetya:** Funding acquisition, Software, Supervision. **Mehdi Dastani:** Supervision. **Gabriele Keller:** Supervision. **Davide Prandi:** Resources, Supervision. **Fitsum Meshesha Kifetew:** Resources, Supervision. **Frank Dignum:** Supervision.

### Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: ISWB Prasetya reports financial support was provided by European Commission. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

### Acknowledgements

This work is funded by EU H2020 iv4XR project, grant nr. 856716.

### References

- [1] X. Fang, S. Chan, J. Brzezinski, C. Nair, Development of an instrument to measure enjoyment of computer game play, *Int. J. Hum.-Comput. Interact.* 26 (9) (2010) 868–886.
- [2] K. Procci, A.R. Singer, K.R. Levy, C. Bowers, Measuring the flow experience of gamers: an evaluation of the dfs-2, *Comput. Hum. Behav.* 28 (6) (2012) 2306–2312.
- [3] C. Jennett, A.L. Cox, P. Cairns, S. Dhoparee, A. Epps, T. Tijs, A. Walton, Measuring and defining the experience of immersion in games, *Int. J. Hum.-Comput. Stud.* 66 (9) (2008) 641–661.
- [4] G.J. Myers, C. Sandler, T. Badgett, *The Art of Software Testing*, John Wiley & Sons, 2011.
- [5] A.P. Vermeeren, E.L.-C. Law, V. Roto, M. Obrist, J. Hoonhout, K. Väänänen-Vainio-Mattila, User experience evaluation methods: current state and development needs, in: *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, 2010.
- [6] R. Alves, P. Valente, N.J. Nunes, The state of user experience evaluation practice, in: *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*, 2014, pp. 93–102.
- [7] C. Guckelsberger, C. Salge, J. Gow, P. Cairns, Predicting player experience without the player. an exploratory study, in: *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, 2017.
- [8] L.E. Nacke, Games user research and physiological game evaluation, in: *Game User Experience Evaluation*, Springer, 2015, pp. 63–86.
- [9] P. Mirza-Babaei, L.E. Nacke, J. Gregory, N. Collins, G. Fitzpatrick, How does it play better? Exploring user testing and biometric storyboards in games user research, in: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 1499–1508.
- [10] A. Ortony, G. Clore, A. Collins, *The Cognitive Structure of Emotions*, Cambridge University Press, Cambridge, England, 1988.
- [11] C.D. Elliott, The affective reasoner: a process model of emotions in a multiagent system, Ph.D. thesis, Northwestern University, 1992.
- [12] C. Bartneck, Integrating the occ Model of Emotions in Embodied Characters, 2002.
- [13] W.S. Reilly, Believable social and emotional agents, Tech. Rep., Carnegie-Mellon Univ Pittsburgh pa Dept of Computer Science, 1996.
- [14] M. Ochs, C. Pelachaud, D. Sadek, An empathic virtual dialog agent to improve human-machine interaction, in: *7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 2008.
- [15] V. Demeure, R. Niewiadomski, C. Pelachaud, How is believability of a virtual agent related to warmth, competence, personification, and embodiment?, *Presence* 20 (5) (2011) 431–448.
- [16] L. Nacke, C.A. Lindley, Flow and immersion in first-person shooters: measuring the player's gameplay experience, in: *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, 2008, pp. 81–88.
- [17] P. Desmet, P. Hekkert, Framework of product experience, *Int. J. Des.* 1 (1) (2007).
- [18] S.G. Ansari, Toward automated assessment of user experience in extended reality, in: *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, IEEE, 2020, pp. 430–432.
- [19] S.G. Ansari, I.S.W.B. Prasetya, M. Dastani, F. Dignum, G. Keller, An appraisal transition system for event-driven emotions in agent-based player experience testing, in: *9th International Workshop on Engineering Multi-Agent Systems (EMAS), Revised Selected Papers*, Springer Nature, 2021.
- [20] R. Ferdous, C.-k. Hung, F. Kifetew, D. Prandi, A. Susi, Evombt: evolutionary model based testing, *Sci. Comput. Program.* 227 (2023) 102942.
- [21] A. Panichella, F.M. Kifetew, P. Tonella, Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets, *IEEE Trans. Softw. Eng.* 44 (2) (2017).
- [22] C. Baier, J.-P. Katoen, *Principles of Model Checking*, MIT Press, 2008.
- [23] J. Callahan, F. Schneider, S. Easterbrook, et al., Automated Software Testing Using Model-Checking, *Proceedings 1996 SPIN Workshop*, vol. 353, Citeseer, 1996.
- [24] S.G. Ansari, I. Prasetya, D. Prandi, F.M. Kifetew, M. Dastani, F. Dignum, G. Keller, Model-based player experience testing with emotion pattern verification, in: *International Conference on Fundamental Approaches to Software Engineering*, Springer Nature, 2023.
- [25] I. Prasetya, M. Dastani, R. Prada, T.E. Vos, F. Dignum, F. Kifetew, Aplib: tactical agents for testing computer games, in: *International Workshop on Engineering Multi-Agent Systems*, Springer, 2020.
- [26] R. Ferdous, F. Kifetew, D. Prandi, A. Susi, Towards agent-based testing of 3D games using reinforcement learning, in: *37th IEEE/ACM International Conference on Automated Software Engineering*, 2022.
- [27] M. Kwiatkowska, G. Norman, D. Parker, Prism: probabilistic symbolic model checker, in: *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, Springer, 2002.
- [28] R. Hebig, C. Seidl, T. Berger, J.K. Pedersen, A. Wasowski, Model transformation languages under a magnifying glass: a controlled experiment with Xtend, ATL, and QVT, in: *26th ESEC/FSE*, 2018.