


RatKit: A Novel Methodology for Verifying, Validating, and Testing Agent-Based Simulations: the Boids Case Study


İbrahim Çakırlar

(SAP Concur, Paris, France)

 <https://orcid.org/0009-0002-6962-9256>, icakirlar@gmail.com


Sevcan Emek

(Manisa Celal Bayar University, Manisa, Türkiye)

 <https://orcid.org/0000-0003-2207-8418>, sevcan.emek@cbu.edu.tr


Şebnem Bora

(Ege University, İzmir, Türkiye)

 <https://orcid.org/0000-0003-0111-4635>, sebnem.bora@ege.edu.tr

Oğuz Dikenelli

(Ege University, İzmir, Türkiye)

 <https://orcid.org/0000-0001-7948-7453>, oguz.dikenelli@ege.edu.tr

Abstract: This study introduces a novel methodology and framework for the verification, validation, and testing of agent-based simulation models: RatKit. Building on repeatable automated testing in ABMS, the present contribution significantly extends the foundation by proposing an integrated metamodel and systematic development methodology that embeds these activities throughout the simulation lifecycle. The RatKit methodology is both general, in that it applies to a wide range of agent-based simulation models using a well-defined metamodel, and comprehensive, in that it addresses the macro-level (societal), the meso-level (interaction) and the micro-level (agent) aspects of simulations. It also provides a generic infrastructure to be able to support various VV&T techniques. RatKit is designed as a general VV&T framework for all ABM frameworks. The methodology comes with a dedicated implemented framework. It is implemented by selecting the Repast ABM development framework. RatKit is demonstrated through a detailed case study of the Boids model, where the dynamics of alignment, cohesion, and separation are examined. Results from the case study show that a test-driven approach can enhance model reliability and ensure that individual agent behaviors coalesce into realistic emergent phenomena. Experiences and feedback obtained during the case studies show that developing ABM with a test-driven method based on VV&T facilitates the creation of desired models.

Keywords: Agent-based simulation, VV&T (Verification, Validation, Testing), RatKit, Metamodel, Boids

Categories: I.6, I.6.4

DOI: 10.3897/jucs.148927

1 Introduction

Agent-based simulation (ABS) highlights new opportunities (much closer to reality concerning other modelling techniques) to model complex systems and complex adaptive systems [Macal, 09]. ABS is increasingly being used in various application domains [Epstein, 07; Gürcan, 14; Grimm, 05; Niazi, 10]. However, despite their increasing popularity, ABS models are often criticized in terms of verifiability and validity of models [Niazi, 11; Naylor 67; Sargent, 91; Law, 03].

Validation and verification (V&V) is an important activity in ABS development as in other simulation techniques. Although almost all simulation researchers agree that it is important and mandatory, there is no agreement on how to conduct the V&V in the development cycle yet. Model verification is the process of determining that the model meets intended modelling objectives and is correctly implemented concerning the conceptual model [Law, 07]. In addition, model validation is the process of determining the degree to which the model is an accurate representation of the real system concerning the intended modelling objectives (theoretical validation), conceptual model (conceptual model validation), simulation model, and simulation results [Sargent, 91; Law, 03, 07; Carley, 96]. Model testing is executing the model, a part of the model or a model component under specified conditions and comparing their results against experimental or real system results to evaluate its functionality. Validation, verification and testing (VV&T) activities are complementary tasks of the development life cycle [Balci, 94, 95]. These complementary and mandatory activities refer to the integrative activity of the ABS development life cycle.

A critical question therefore arises: “Is the simulation model an accurate representation of the real system?” If the model falls short, the insights gained are either misleading or entirely inappropriate [Troitzsch, 04]. To address these challenges, we propose embedding verification, validation, and testing throughout the simulation lifecycle by introducing a systematic, metamodel-driven methodology. In this paper, we detail the new technical contributions of our work: a refined metamodel, an integrated methodology, and a comprehensive framework implemented as RatKit. We further illustrate this approach through the Boids case study, emphasizing the key behavioural attributes—alignment, cohesion, and separation—that underpin flocking behaviour.

VV&T of the simulation model is not a trivial activity. Numerous techniques have been developed for this purpose [Troitzsch, 04; Sargent, 13; Office USGA, 87; Banks, 10; DoDI, 09; Riedmaier, 21; Olsson, 22; Bakar, 18; Sudeikat, 09; Herd, 14]. Some approaches involve collaboration with domain experts or reviewers, auditing, conducting Turing tests, writing comprehensive model documentation, or debugging. Others focus on face validation, model testing, evaluating assumptions and components, and comparing model behavior or outputs with the real system or other models [Hunter, 20; Roungas, 17; Curreli, 21]. These are often referred to as dynamic techniques [Troitzsch, 04], which require executing the agent-based simulation (ABS) model, observing its outputs, and comparing the results against reference data. At this stage, there is a strong need for ABS developers to support the implementation of various methods through a structured VV&T framework.

There are few frameworks that handle conducting validation, verification and testing of ABSs [Niazi, 09; Gürcan, 13; Wright, 12; Khattak, 15; Drchal, 16]. These

frameworks are developed to meet specific needs—particularly model testing—but they fall short when it comes to addressing the broader demands of VV&T automation and integration, including design, execution, and observation. As system complexity increases, there is a growing need to equip ABS developers with effective techniques and automated tools that help produce high-quality (accurate and sufficiently credible) models. Beyond the urgent need for a dedicated framework, another key gap is the absence of a comprehensive methodology. Existing techniques in the literature approach VV&T from differing perspectives, each outlining its own set of application steps. This inconsistency leaves ABS developers uncertain about how to perform these activities and how to apply chosen techniques effectively. A unified methodology— independent of individual techniques—should define the core activities in an ABS study and systematically connect them with suitable methods. Such integration must be guided by a metamodel constructed with reference to existing approaches, ensuring that both technique selection and execution align with developer needs.

In this paper, we introduce the RatKit methodology and demonstrate its application through a supporting framework. We also define a metamodel for VV&T of ABS, based on existing studies and a range of established techniques. Furthermore, we elaborate on the methodology by detailing the fundamental activities derived from the metamodel. These activities represent the core tasks of the quality assurance process and are designed to help ABS developers carry out model validation, verification, and testing more easily and systematically.

RatKit's key advantages center around its modular design and extensibility. Its architecture is built from distinct components—such as the scheduler, logging, observation, and visualization modules—that can be independently updated or replaced to accommodate different agent-based modeling (ABM) platforms. This modularity not only simplifies maintenance but also facilitates integration with a variety of simulation environments. Additionally, RatKit's comprehensive metamodel framework rigorously encapsulates the essential evaluation concepts, enabling a systematic, test-driven development lifecycle. This systematic approach allows developers to embed evaluation tasks throughout the simulation process, thereby promoting early error detection, enhanced model reliability, and iterative refinement of both individual components and emergent system behaviours. Despite these significant strengths, there are several limitations associated with the current implementation of RatKit. First, the framework currently relies on the Repast simulation platform, which may restrict its seamless adoption in environments where other ABM frameworks predominate. Secondly, while RatKit is designed to support extensive evaluation activities, scalability constraints may emerge when handling extremely large-scale models or simulations with thousands of agents, potentially impacting performance. Furthermore, although many evaluation tasks are automated, automating more complex evaluation processes—particularly those requiring real-time adaptation or handling ambiguous behavioural outputs—remains challenging. These limitations point to areas for future improvement, including broadening support for additional simulation platforms, optimizing performance for large-scale implementations, and enhancing the automation of evaluation procedures.

The rest of the paper is organized as follows. The next section reviews existing frameworks in the domain. Section 3 outlines the essential requirements for such frameworks, from both architectural and user perspectives. Section 4 introduces the proposed VV&T metamodel. Section 5 presents the RatKit methodology, highlighting

its guidance for evaluating ABS models. Section 6 describes the RatKit framework in detail. Section 7 demonstrates the methodology's applicability through a well-known case study: Boids. Finally, Section 8 concludes the paper.

2 Related Work

VV&T is often treated as external or final steps in the development of agent-based simulations (ABS). The studies discussed in this section offer limited and specific capabilities, generally treating these quality assurance activities as separate from the core development process. While there is some work on supporting frameworks, it is relatively scarce—and the focus is more on toolkits and infrastructure than on underlying techniques. These efforts can be broadly classified into two categories: monitoring frameworks and model-based testing frameworks.

The testing framework for ABS models is MASTER which is integrated into the MASON framework. The framework handles the testing activities as the last activity of the ABS development. The main focus is testing completed ABS models. The user defines normal situations, facts, constraints and abnormal situations for the overall model. The framework executes the model and evaluates the execution according to the user-defined assertions. It focuses on the completed simulation models and only performs external validation [Wright, 12].

VOMAS is another monitoring-based framework for ABSs. VOMAS proposes using a group of specialized monitoring agents over an overlay network, to perform VV&T tasks. Each agent in the overlay network monitors the ABS model agents execution to detect unusual behaviours and reports these violations (in the same cases, creates logs). The VOMAS approach is based on monitoring and logging [Niazi, 09]. However, VOMAS aims to perform only external validation. Moreover, it's not clear how the constraints are defined for the ABS models, the overlay agents monitor the model execution and perform VV&T tasks.

Monitoring-based approaches carry out quality assurance through observation and logging, detecting suspicious or unexpected behaviours during simulation runs. Although monitoring and logging are essential for ensuring model integrity, the manner in which constraints and assertions are defined plays a critical role. These elements help establish boundaries for model evaluation. Importantly, monitoring should be non-intrusive; frameworks must observe ABS execution as it naturally occurs, without altering agent behaviours or event scheduling. Intervening in this way risks invalidating the model's behaviour and diverging from the intended simulation logic. Therefore, such frameworks should operate as a separate layer above the simulation development environment, keeping the validation and verification processes isolated from model execution.

Model-based testing frameworks are model-driven development environments to ensure model-based testing for simulations [Schieferdecker, 12]. To spread ABS development in the various application domains and to simplify model development, ABS frameworks aim to support model-driven development (MDD) [Ozik, 08]. VeriTAS is the only MDD framework for modelling simulation tests [Djanatliev, 11]. This support not only contains designing simulation tests visually but also contains the execution of simulation tests corresponding to the defined test models. VeriTAS aims to define UML models for both simulation development and test models. Test cases are

generated according to the defined test models. However, it's not clear how the constraints and assertions are defined. And also, it's not clarified how the framework monitors, and tests the execution of the simulation model. On the other hand, VeriTAS claims to apply simulation model testing (it also covers verification and validation) iteratively and systematically during the development cycle. This proposal is similar to the RatKit methodology.

Despite various tools and techniques, there remains a significant gap: the absence of a clear methodology that outlines how to integrate quality assurance activities throughout the ABS development lifecycle [Niazi, 09; Wright, 12; Djanatliev, 11; Gürcan, 13]. A robust framework must be paired with a methodology that not only supports all stages of development—from design to implementation—but also helps developers ensure that the resulting model aligns with their original intent. Most existing solutions focus on specific needs, such as external validation [Carley, 96], and thus fail to meet the broader requirements of integration and automation. What's still missing is a comprehensive, systematic framework that enables structured, consistent, and reliable verification, validation, and testing across all stages of ABS development.

3 Requirements of VV&T Frameworks

The essential requirements of VV&T frameworks are defined in related studies [Gürcan, 13; Dikenelli, 14; Troitzsch, 96]. In this section, architectural and user points of view are presented by evaluating different perspectives.

3.1 Architectural Requirements

These requirements are functional requirements that state essential functionalities of VV&T frameworks.

- **Interoperability & Compatibility:** VV&T tasks are integral and interdependent components of ABS development process. Rather than being standalone tasks, they must be tightly coupled with the simulation environment to ensure correctness and coherence throughout model development. Therefore, a VV&T framework must be seamlessly integrated with the simulation platform, enabling bidirectional information exchange for output evaluation and interpretation based on defined rules.
- **Composability:** Composability refers to assembling components or elements to be combined or connected in various ways to satisfy user requirements [Harkrider, 99]. Modularity is a key principle in model design. Components should be independently developed and assessed, while interacting with the rest of the model solely through well-defined interfaces. VV&T processes must support this modularity by operating at multiple levels of abstraction. The model can be developed, verified, tested and validated systematically. Composability is the ability to combine an ABS model with well-defined and VV&T applied sub models or entities.
- **Autonomy:** Autonomy is one of the mandatory attributes of agents. The autonomy of an agent is operating on its behalf in the environment individually without any interference. The autonomy of agents must be preserved. Agents act independently within the environment, and any framework must avoid interfering

with their execution or scheduling. Interventions can compromise the fidelity of the simulation and reduce the trustworthiness of the results.

- **Monitorability:** The main task is monitoring the simulation model executions: output of agent behaviours, or occurrence of special or unexpected cases, etc. The main functionality of this requirement is gathering and providing useful evidence to the ABS developers about the simulation model execution to evaluate models. The monitoring results here are named as observations in the simulation literature [Balci, 95].
- **Traceability:** Traceability is the ability to keep track of the history of an ABS model execution. The ABS model consists of autonomous agents. The autonomy of an agent makes ABS model results unpredictable. Therefore, tracing model executions in a controlled way make ABS developer's life much easier. Traceability can be achieved by the logging support by the developers. Logging should be optional and should support different log levels to avoid confusion.

3.2 User Requirements

User requirements are shaped by growing adoption and are directly derived from practical needs in ABMS domain. These frameworks should be comprehensive, incorporating a wide range of existing VV&T techniques to support diverse use cases. User requirements define the essential capabilities needed to make quality assurance tasks more accessible and efficient.

- **Usability:** Performing VV&T in the early steps in the ABS development lifecycle makes both the development and design decisions much easier and realistic. It should be identical to the model development to address all users especially, non-computer scientists. Thus, users do not need to any extra effort to perform VV&T.
- **Automation:** Automation is the capability of executing model verifications, validations, and tests together and individually without any manual effort. Therefore, automated VV&T is an essential requirement for monitoring the impact of the new model component or behaviour without any extra effort. As a result of an automated execution of all tasks, the user can trace the effect and easily notice the underlying cause of any change in model behaviours.
- **Presentation of Observations:** Presenting observation is the reflection of the monitoring results. From the user's point of view, the data that is generated by the simulation model during the execution should be easily accessible to evaluate models. Evaluation of the model observations against the real data is subject to verification, validation and testing [Balci, 95]. Model observations relate the model's parameters to observations based on data captured at any time during model execution.
- **Parameter Tuning:** While developing a simulation model, it's important to select appropriate values for model parameters. Model parameters are important to provide the system's behaviour. A dedicated testing framework should the potential to provide parameter tuning find optimum parameter values, show the domino effect between parameters, test the variety of parameter values, draw the boundaries for the parameter value set, test the parameter sensitivity, etc. To find

the appropriate values, the different initial values run multiple times to perform parameter tuning [Dikenelli, 14].

- Visualization: VV&T tasks do not only focus on quantitative techniques [Niazi, 09]. Graphically showing observation history helps developers review simulation execution or the behaviours of the agents. A VV&T framework offering visualization support provides a broader perspective for evaluating ABS models from various aspects. In this way, users can visually monitor the behaviours of agents with different conditions without any extra effort [Dikenelli, 14].

4 VV&T Metamodel for Agent-Based Simulations

VV&T metamodel is an extension of the UML testing profile (UTP) [Baker, 08]. UTP aims to capture all information that would be needed by different test processes. UTP is based on UML 2.0. Thus, it aims to enable test definition and test generation based on UML models. Despite this, UTP is not enough for ABSs. Based on the evaluation of various VV&T techniques, we enrich the UTP according to the requirements of ABS [Troitzsch, 04; Sargent, 13; Office USGA, 87; Banks, 10; DoDI, 09].

The VV&T metamodel aims to capture all information that would be needed by different processes and provides a set of VV&T related concepts in the definition of tasks for ABS models. It is independent of ABS platforms and technologies and can be applied in a variety of domains of ABS. The concepts are gathered to define analysis, design, and implementation in the ABS. VV&T metamodel is shown in Figure 1.

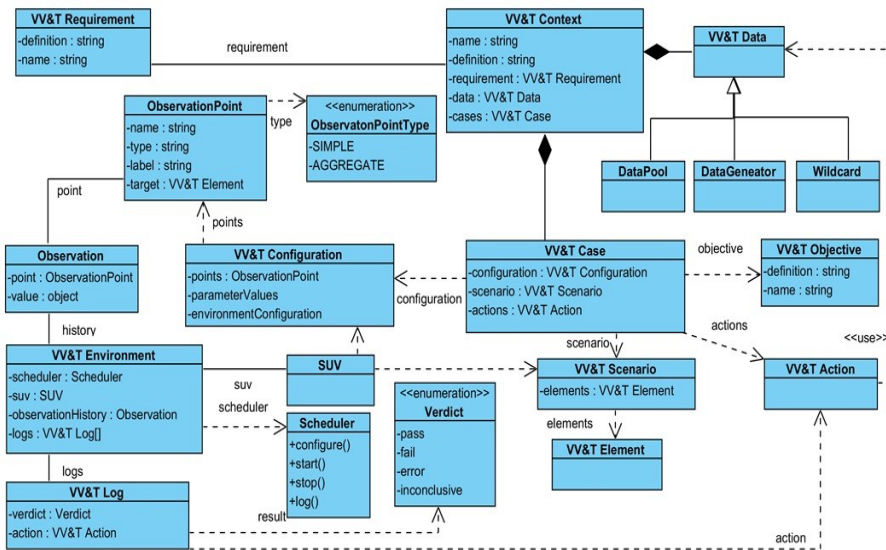


Figure 1: VV&T metamodel for agent-based simulations

VV&T context (VV&T Context) is the outmost concept of the metamodel. This context includes requirements, cases, and data. According to the application domain and requirement complexity, VV&T context consists of a set of cases. For each requirement, a context that contains a set of cases that will be verified, validated or

tested against the provided VV&T data is defined. Most of the development requirements of the ABS model should be converted into a set of requirements.

The VV&T data (VV&T Data) can be organized as a data pool, data generator or wildcard. The data pool expresses the data gathered from the real system under study. Data Pool can be a database or a file according to the data type and application domain. The data generator expresses the data generator part of the data side. In some domains, VV&T data can be generated according to the predefined rules, functions or statistical distributions. Wildcards are used in the loose specification of data values. In some application domains it's helpful not to force users to specify each possible test data value [Bakar, 18]. Wildcards are user-defined data specialized for an application domain.

VV&T case (VV&T Case) which is designed corresponding to a VV&T objective (VV&T Objective), consists of VV&T scenario, VV&T actions and VV&T configurations. VV&T objective expresses the main objective of the actions and forms the content of the scenario. In a VV&T scenario, ABS model elements to be verified validated, or tested (VV&T Element), simulated environments (or a part of the simulated environment) are defined. VV&T Element expresses an ABS model element or a part of ABS model elements according to the context. In some cases, VV&T elements can be defined as fake elements [Niazi, 09]. Fake elements are model elements that behave like real elements in VV&T scenarios for VV&T purposes [Feathers, 04]. Another important part of the VV&T case is the configuration (VV&T Configuration).

A configuration element defines the configuration of a specific VV&T scenario and the environmental resources. In this configuration, model and environment parameters, and environmental resources are defined to configure the scenario. Configuration is used to define the system under verification, validation or test (SUV/SUT). SUV/SUT contains whole system or subsystem elements to create/define a specific scenario for VV&T purposes. Another important part of the configuration is the definition of observation points (ObservationPoint). Observation points are information-gathering definitions that provide required actual data for VV&T actions. VV&T Actions (ValidationAction) express the VV&T tasks, especially assertions and visual results to evaluate behaviours or outputs of the VV&T elements.

VV&T environment (VV&T Environment) expresses the execution environment. The VV&T environment contains SUV/SUT, scheduler, observation history and VV&T logs. SUV/SUT expresses the active VV&T scenario that is created by the scheduler according to the configuration of the VV&T case. The scheduler (Scheduler) is responsible for managing the life cycle of the SUV/SUT in the VV&T environment. Observation history is a collection of data that is gathered during the whole lifecycle of the SUV/SUT from observation points. Observation history is used by the VV&T actions (VV&T Action).

The scheduler schedules the VV&T actions which are responsible for evaluating expected behaviours of SUV/SUT according to the actual observations. As a result of the evaluation, VV&T logs which define the VV&T results correspond to the user-defined assertions and visualizations. VV&T action is used to set/calculate the verdict of the VV&T case. The verdict is an enumeration data type for the possible results of the VV&T actions and suggests how VV&T case execution is carried out. Pass value expresses the validation case was successful and SUV/SUT responded as expected. Fail expressed that the SUV/SUT did not act by the case specification and violated the assertions. Inconclusive is used to de- fine that VV&T execution cannot determine

whether the SUV/SUT is successful or failed. And, the error expresses the VV&T environment itself is has failed, not the SUV/SUT.

5 RatKit Methodology: How to Verify, Validate or Test an ABS Model?

There are various techniques suggested for the verification, validation, and testing of simulation models [Sargent, 91, 13; Office USGA, 87; Banks, 10; DoDI, 09]. However, it's still unclear for simulation developers how to conduct VV&T in their actual studies. In many ABS studies, even when suitable validation and verification techniques are applied, the absence of a clear methodology remains the main barrier to obtaining reliable results. Each technique recommends specific objectives and activities from its own viewpoint. However, factors such as the nature of the simulation domain, its goals, the availability of real data, and knowledge about the actual system influence both ABS development and the quality of validation efforts. Thus, simulation developers need a methodology to conduct these activities systematically. This methodology should be technique-independent and provide a general framework that supports the application of various approaches. The RatKit methodology was developed with this purpose in mind, aiming to make the process more systematic, accessible, and adaptable across different techniques and domains [Dikenelli, 14; Çakırlar, 15]. Figure 2 illustrates the steps of the RatKit methodology.

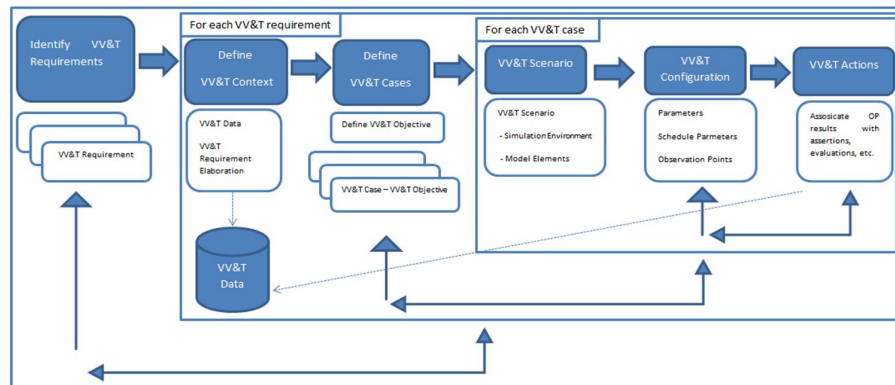


Figure 2: RatKit methodology

The first activity of the methodology is the identification of VV&T requirements based on the objectives of the ABS model and the model requirements. The following activities of the RatKit depend on the definition of the requirements. The requirement derives the rest of the remaining activities like the selection of the VV&T technique and the definition of the VV&T context. However, the direct transformation of a user requirement to the requirement is not suggested. According to the metamodel, each VV&T requirement corresponds to the context. Thus, the requirement should be defined based on data, technique and model objectives.

6 A comprehensive VV&T Framework: RatKit

RatKit (Repeatable Agent Testing Toolkit) is an open-source VV&T framework that provides a feature set to perform verification, validation, and testing of ABS models [36]. RatKit focuses on providing features to the ABS developers in order to perform various verification, validation and testing tasks from a broader perspective. RatKit is licensed under the GNU GPL license. All of the implementations discussed in this article, including the source code, are available directly from the website [Çakırlar, 25].

Figure 3 illustrates the high-level architecture of the RatKit. There are five main components in the RatKit framework. Each component has a dedicated functionality. The framework design is revised and extended according to the metamodel and previous experiences [Dikenelli, 14; Çakırlar, 15]. RatKit v1.0 is integrated into the Repast 2.032 [Repast Symphony, 24]. But, the RatKit is designed as a generic framework for all ABMS platforms. The architecture of the RatKit is suitable for integrating another ABMS framework easily. RatKit uses the Junit testing structure to use evaluation mechanisms for ABS models and also to support unit testing functionality.

The scheduler component is responsible for the schedule management of cases in the VV&T lifecycle. The scheduler is the concrete implementation of the metamodel scheduler element. The scheduler creates SUT/SUV instances and initiates the VV&T lifecycle of the VV&T cases. The logging component is responsible for logging the SUT/SUV execution according to the configuration of the case.

The observation component is responsible for the monitoring of the SUV/SUT in the VV&T environment. The observation point definitions in the VV&T configuration are used by the observation component. The observation component creates observations according to these configurations and these results are presented to the VV&T actions and also to the visualization component. The visualization component is responsible for the visualization of the observations as a result of execution to the user. The configuration has a definition to indicate the visualization request of the user. When the visualization is requested as a visual action, observations are visualized to the user and the user decides whether the current results are valid or not.

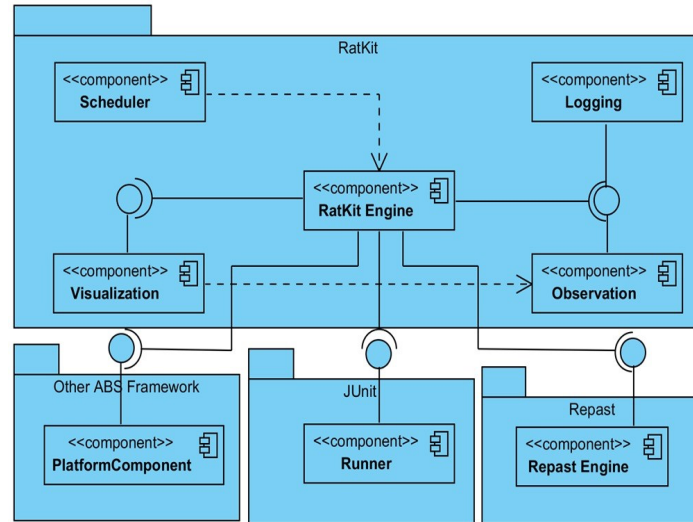


Figure 3: RatKit architecture

The RatKit engine is the core component of the framework. The engine component reflects the innovative and abstract features of the framework. The engine performs its tasks by using other components. The engine component initiates the environment instance, according to the underlying ABMS framework. The observation of the execution is handled by using the observation component and also logging is achieved by using the logging component. The visual VV&T is provided by the visualization component.

The innovative feature of the framework is using a specialized agent (VV&T Agent). When the scheduler initiates the SUT/SUV, according to the VV&T configuration, the VV&T agent is also scheduled to the environment. All actions are executed by the agent in the environment. The results of the actions are generated by the VV&T agent as a VV&T log in the environment. These results are visualized by the engine to provide the user with an execution history. Errors and failures inside the actions are also presented to the user. The next section explains the details of the RatKit in detail based on the case study.

7 Case Study – Boids Model

In this section, we demonstrate the effectiveness of the Ratkit and its applicability in a well-known case study: the Boids model.

This model is an example of a complex system that exhibits emerging behaviour as a result of simple rules. The Boids model aims to model the steering behaviour of flocking birds. Agents of the model are called Boid (Bird + Android = Boid). Each boid follows simple rules and as a result of interactions according to these rules, flocks emerge naturally. There is one assumption defined in the model: “Each boid has the information of its neighbours in a predefined viewpoint.”. The model is based on three simple rules [Reynolds, 87]:

- Separation: Each boid steers itself to avoid crowding in its viewpoint.
- Alignment: Each boid steers itself to align its heading as an average of its neighbours.
- Cohesion: Each boid steers itself to align itself to the average position of its neighbours.

Even, boids model is a simple model that contains three rules and one assumption; it's really hard to implement the model without performing VV&T. The main objective of the model is generating flocking behaviour which is an emerging behaviour. So, the model developers need a VV&T method to develop an ABS model that exhibits the real system steering behaviour. According to the model definition, an ABS developer should perform activities for each separate agent behaviour rule (alignment, cohesion, and separation) and for the whole model (flocking) during the ABS model development. Each requirement defines a separate context with meaningful VV&T data. The VV&T data expresses the real system data. In this case study, we used a data generator that is produced concerning the statistical approach [Bialek, 12]. Each context has its own data set to verify, validate and test the current ABS model properly.

7.1 Creation of Meta data

7.1.1 VV&T Context

Each VV&T context handles VV&T tasks from the perspective of specific objectives. Previously, we defined four different requirements: alignment, cohesion, separation, flocking. In the context, a model developer shall focus on the only alignment behaviour of the boids. For the VV&T data, we only need alignment behaviour-related data, if possible. The data is only useful when a model developer trying to validate boid's alignment. The movement of a boid is defined as "Each boid has a location (X, Y) at a time (t) . At $t + 1$ time the location of the boid is updated".

Corresponding to the design decision, a model developer needs to perform some activities for verification, testing, and validation.

7.1.2 VV&T Objectives

Each case is designed to achieve a specific VV&T objective. However, the selection of verification, validation, and testing depends on the approach of the model developer, existing information about the system, existing data and preferences. For the boids model, we define one example for verification, testing, and validation. To perform verification, it needs to verify the movement behaviour of the boid concerning the modelling requirements. As a result of this, a model developer defines a case. Another example is chosen for model testing: testing of boid alignment. In a VV&T scenario that contains a set of boids, each boid has a random heading value at the time t and the heading values of boids in a given viewpoint based on more neighbourhoods. Lastly, for model validation, the scenario that is constructed with a set of boid agents whose location and heading values are assigned according to the data generator can be used. As a result of this step, the following VV&T objectives are defined;

- Verify the change of boid's location changes between t and $t + 1$ time.
- Test the alignment of a boid in a given range with random values.

- Validate the alignment of boids in a given range with generated values according to the real bird alignments.

7.1.3 VV&T Scenarios

After the definition of objectives, it's easy to design a VV&T scenario. The scenario is a generic definition of a specific ABS model or submodel to perform VV&T as a result of model execution. For each case following scenarios should be defined;

- A boid that has predefined or parametric location information.
- A set of boid (a parameter can express the numbers of the boids in the scenario) with random location and heading information.
- A set of boid with real location and heading information.

7.1.4 VV&T Actions

VV&T actions contain comparison, assertion and evaluation definitions to perform VV&T depending on the case, application domain, etc. The definition of the actions is one of the most important steps of the VV&T method. Each action should be capable of handling the observation results in the correct way. In the previous three cases we defined the following actions:

- Verify the change of boid location between t and $t + 1$.
- Test the change of boids' location properly and test the heading values of boids' are updated as the average heading value between t and $t + 1$.
- Validate change of boids' location properly and validate the heading values of boids' are updated as the generated heading value.

7.1.5 VV&T Configurations

The VV&T configuration consists of model parameters, observation points and model schedule definitions. According to the designed scenario elements and actions, the corresponding VV&T configuration needs change. For our scenarios that are designed in the previous step, the needs of the actions specify the content of VV&T configurations. In this scenario, the value assignment to the location of the boid, as an observation point boid's location attribute should be defined and the schedule of the scenario can be defined as one tick (simulation time). Location and heading values are assigned as random values. The only need is to define simple observation points for the location and the heading values of each boid in the scenario. And for average heading calculation, it's needed to define an aggregate observation point to calculate the average value in each tick.

The sub-steps should also be followed for each defined VV&T context. After the design of cases, actions, and configurations the responsibility of a comprehensive framework starts immediately. The whole ABS model is developed concerning the defined VV&T method iteratively. The implementation details of the case study and more information about the framework are accessible on the RatKit [Çakırlar, 25].

7.2 Verification of a boid movement in the environment

In Section 5, we define a VV&T case in order to verify the movement behaviour of a boid in the environment. Figure 3 illustrates a visual representation of the VV&T case in the life cycle of the RatKit framework. According to the scenario, we only need to create a boid instance (namely boid1) in the Boids environment. The boids environment represents the virtual environment of the model that expresses the simulated environment of the real system. In the scenario definition, we only define an observation point that monitors the location value of boid agents. The RatKit monitors the boid agent instances (boid1) in the environment according to the observation definitions and creates observations in the environment. As shown in the Figure 4, as an example of observation history the location observations are listed. The agent uses these observation results in order to execute the actions defined by the developers.

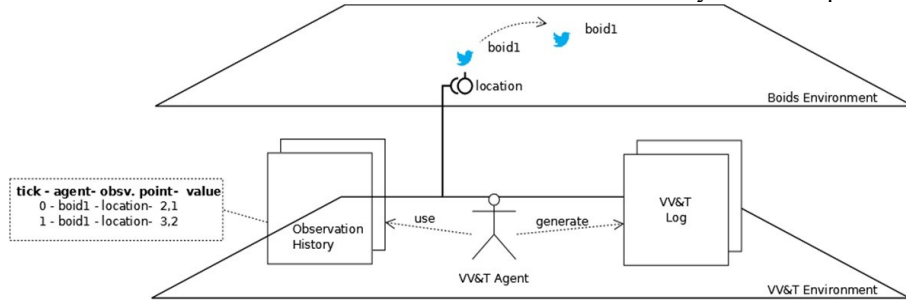


Figure 4: Verification scenario execution for the Boid movement

In the verification case, the location of the boid1 is expected to change in consecutive simulation runs. In this case study, the action focuses on the evaluation of location data in 0th and 1th ticks. According to the observation history that is provided to the agent, the agent creates a new VV&T log for the cases using pass verdict in the environment. Because our assertion is correct according to the given observation values.

7.3 Test of Boid's Alignment Behaviour

The role of the RatKit in execution, evaluation and application is illustrated in Figure 5. In the boids environment (Figure 5), four boids instances (namely boid1, boid2, boid3, and boid4) are created using the scenario. According to the configuration, the RatKit monitors two observations; the location and the heading attributes of the agent instances in each simulation period (tick). As defined in the scenario, the location and

heading values are assigned random values. Thus, the aim is to focus on the testing of the alignment behaviour exhibited.

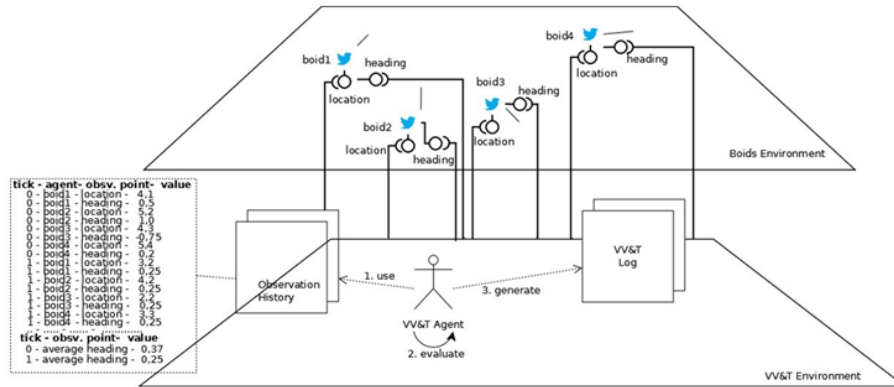


Figure 5: Testing of Boid's alignment behaviour

A sample list of observation history recorded by the VV&T environment is listed on the left side of Figure 5. The table summarizes the role of RatKit in the testing of ABS models. Previously defined VV&T actions are executed by the VV&T agent in the VV&T environment using the observation history. The responsibility of the VV&T agent in the VV&T life cycle is as follows: 1) Reach observation history, (2) Execute actions according to the observation results, (3) Create a new VV&T log as a result of corresponding to the action executions.

According to VV&T design for the testing of boid's alignment and actual observation result the following evaluations can be achieved. In the initial state of the SUT (tick = 0), we have a list of observations for the location, the heading and the average heading value. Also, we have the same values for the first tick (tick = 1). As a result of these values, it's possible to test the alignment behaviour. According to the alignment behaviour definition, we need to test the alignment as an update of the heading to the average heading value inside the viewpoint. In the model implementation, we define the viewpoint as the Moore neighbourhood. So, we can select the boid2 as the main actor of the SUT. We can calculate the average heading value according to the instance heading values. So, in the VV&T action, it's possible to evaluate the average calculation result as changed in the neighbour boid instances.

RatKit also calculates the average heading value. Using another action, we can also test the same value concerning the calculated value. The execution of the actions is handled by the agent and a new VV&T log that refers to the pass is created in the environment. Because the average heading value and the heading values of the selected agent instance's values make our assertion valid.

7.4 Validation of Boids Alignment Behaviour

In the previous section, we define case, scenario, configuration and actions needed to perform validation of boid's alignment behaviour. The role of the RatKit in execution, evaluation and VV&T data usage is illustrated in Figure 6.

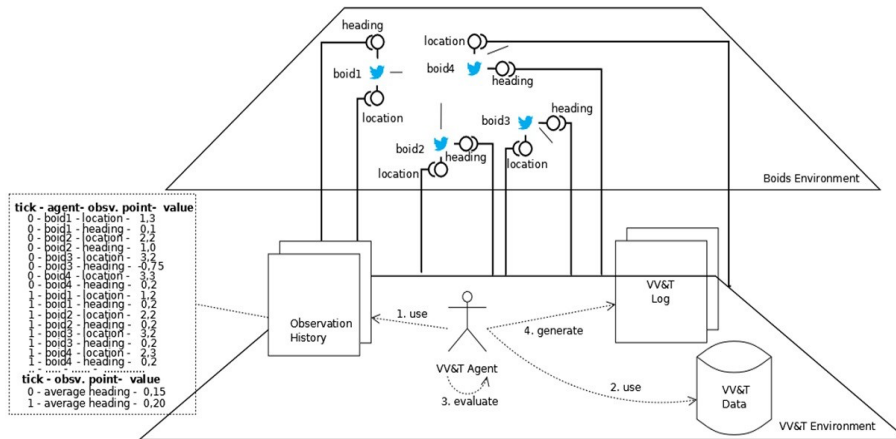


Figure 6: Validation of Boid's alignment

In the boids environment, four boids instances (namely boid1, boid2, boid3, and boid4) are created using the scenario. According to the configuration, the RatKit monitors two observations; the location and the heading attributes of the agent instances in each simulation period (tick). As we defined in the scenario, we decided to assign the location and heading values using a data generator. Thus, to perform validation the initial conditions are initialized based on the experimental data.

As shown in Figure 6, the execution of validation activities is different from other activities. The evaluation of action needs one more extra input; the VV&T data. In this case, the aim is a validation of alignment behavior based on the experimental data. The average heading value which is computed by the environment is evaluated according to the VV&T data. By the way, the alignment behavior output is evaluated according to the real system outputs and the result is generated.

7.5 Completion of the Case Study

In the context of the Boids case study, we present a series of evaluation activities designed to demonstrate the applicability and benefits of RatKit. The predefined evaluation contexts are developed according to a structured schedule, reinforcing that embedding evaluation tasks during ABS model development can yield a more robust and reliable output than traditional development alone. For example, once a developer has completed and validated the alignment behaviour using our methodology, all tasks required to ensure the correct implementation of alignment are finalized. When the developer transitions to the cohesion behaviour context, any necessary changes in the movement implementation are immediately visible through the persistence of

alignment evaluation results. This continued evaluation guarantees that earlier implemented behaviours remain valid and deviations are promptly identified.

Moreover, after all evaluation contexts—encompassing alignment, cohesion, and separation—are implemented, the resulting model outputs become increasingly reliable and valuable for developers, as they closely mirror both the modelling objectives and the real-world system characteristics. Even in a relatively simple agent-based model such as Boids, distinguishing and monitoring emergent behaviours can be extremely challenging. Therefore, the comprehensive evaluation framework provided by RatKit is crucial for achieving accurate and consistent model development.

To further enhance the analysis, developers can incorporate quantitative measures and statistical error metrics that compare the Boids model's behaviour with real system data. Specifically, graphs, tables, and formal tests to provide a deeper analytical perspective on how closely the simulation replicates critical phenomena such as alignment, cohesion, and separation. However, it is important to note that some emergent behaviours are inherently difficult to express in a repeatable, test-driven development framework. For instance, formulating repeatable test data and establishing clear measurement criteria for outputs like cohesion or separation often require explicit assumptions regarding agent influence ranges, interaction thresholds, or inter-agent distances. These assumptions, which are critical for the formulation of repeatable tests, are explicitly detailed as part of our methodology. While many aspects of model behaviour can be rigorously coded and tested, certain emergent phenomena remain challenging to capture fully via automated coding techniques alone.

This enhanced evaluation approach, which combines quantitative metrics with the necessary qualitative assumptions, aligns with earlier work on visual agent-based model development and underscores the practical utility of the RatKit framework [North, 07]. By integrating these strategies, our methodology provides both a rigorous analytical basis and a realistic account of the constraints inherent in automated evaluation of complex behaviours.

8 Conclusions

In this study, we present the evaluation of the RatKit from a testing framework to a VV&T framework. The VV&T metamodel and VV&T method are discussed in detail and described with an example case study. VV&T method is applied on the boids model. It is important for the Boids model to realize the three basic behaviours (alignment, cohesion and separation). Therefore, the focus throughout the study is on the coordination of the three basic behaviours of the model. In terms of the VV&T method, the targets of simulation provide the most important input for each stage of the method. In this study, the determination of targets and situations, verification, testing and validation situations are discussed for the Boids model, respectively.

For Boids models, when a simulation model consisting of thousands of individuals is operated, it is quite difficult to decide whether the alignment, separation or cohesion targets are really realized as defined in the simulation targets by looking only at the visual outputs of the model. When the validation and verification process is assumed to be carried out only according to the outputs of the simulation model, it is important to define the produced output correctly and to operate the parameters and behaviours that affect the behaviour correctly.

As the simulation model behaviours become more complex, it becomes quite difficult to interpret the simulation model outputs (both observed and produced outputs). The boids model is one of the basic models used for modelling complex systems with ABM. However, even when preparing such a simple simulation model, it is quite difficult to decide whether the three different behaviours are actually described as they should be by looking at the visual operation of the model. In models where the scale of the model and the behaviours are difficult to capture, or even dependent on many constraints or rules, it is quite difficult to perform VV&T at the final stage.

References

- [Bakar, 18] Bakar, N. A., Selamat, A.: Agent systems verification: systematic literature review and mapping, *Applied Intelligence*, 2018; 48, 1251–1274.
- [Baker, 08] Baker, P., Dai, Z. R., Grabowski, J., et al.: *Model-Driven Testing Using the UML Testing Profile*, 2008.
- [Balci, 94] Balci, O.: Validation, verification, and testing techniques throughout the life cycle of a simulation study, In: *WSC'94*, 1994, pp. 215–220.
- [Balci, 95] Balci, O.: Principles and techniques of simulation validation, verification, and testing, In: *WSC'95*, 1995, pp. 147–154.
- [Banks, 10] Banks, J., Carson, J. S., Nelson, B. L., et al.: *Discrete-Event System Simulation*, 5th ed. Englewood Cliffs, NJ: Prentice-Hall, 2010.
- [Bialek, 12] Bialek, W., Cavagna, A., Giardina, I., et al.: Statistical mechanics for natural flocks of bird, *Biological Sciences*, 2012; 109: 4786–4791.
- [Carley, 96] Carley, K. M.: *Validating Computational Models*, In: Carnegie Mellon University, 1996. <https://api.semanticscholar.org/CorpusID:5888357>.
- [Curreli, 21] Curreli, C., Pappalardo, F., Russo, G., et al.: Verification of an agent-based disease model of human Mycobacterium tuberculosis infection, *International journal for numerical methods in biomedical engineering*, 2021; 37: 3470.
- [Çakırlar, 15] Çakırlar, I.: *Development of Test Driven Development Methodology For Agent-Based Simulations*, PhD thesis, 2015, Ege University, Izmir, Turkey.
- [Çakırlar, 25] Çakırlar, I.: *RatKit*, <https://github.com/dr-ibrahim-cakirlar/ratkit> [accessed 1 July 2025]
- [Dikenelli, 14] Dikenelli, O., Gürcan, O., Çakırlar, I., et al.: *RatKit: A Repeatable Automated Testing Toolkit for Agent-based Modeling and Simulation*, In: *the 15th International Workshop on MultiAgent Simulation (MABS 2014)*.
- [Djanatliev, 11] Djanatliev, A., Dulz, W., German, R., et al.: *VeriTAS - a versatile modeling environment for test-driven agile simulation*, In: *Proceedings of the 2011 Winter Simulation Conference (WSC '11)*, IEEE Press, 2011, pp. 3657–3666.
- [DoDI, 09] DoDI.: *DoD Modeling and Simulation Verification, Validation, and Accreditation*. U.S. Department of Defense, 2009, DoDI 5000.61.
- [Drchal, 16] Drchal, J., Čertický, M., Jakob, M.: *Data Driven Validation Framework for Multi-agent Activity-Based Models*, In: Gaudou, B., Sichman, J. (eds) *Multi-Agent Based Simulation XVI. MABS 2015, Lecture Notes in Computer Science*, 2016, pp. 55–67.

- [Epstein, 07] Epstein, J. M.: Agent-based computational models and generative social science, *Introductory Chapters in Generative Social Science Studies in Agent-Based Computational Modeling*, Princeton University Press, 2007.
- [Feathers, 04] Feathers, M.: *Working Effectively with Legacy Code*, Prentice Hall, 2004.
- [Gürcan, 13] Gürcan, O., Dikenelli, O., Bernon, C.: A Generic Testing Framework for Agent-Based Simulation Models, *Journal of Simulation*, 2013; 7: 183–201.
- [Gürcan, 14] Gürcan, O., Türker, K. S., Mano, J-P., et al.: Mimicking Human Neuronal Pathways in Silico: an emergent model on the effective connectivity, *Journal of Computational Neuroscience*, 2014; 36: 235–257.
- [Grimm, 05] Grimm, V., Revilla, E., Berger, U., et al.: Pattern-oriented modeling of agent-based complex systems: Lessons from ecology, *Science*, 2005; 310: 987–991.
- [Harkrider, 99] Harkrider, S. M., Lunceford, W. H.: Modeling and simulation composability, In: 1999 interservice/industry training, simulation and education conference, 1999.
- [Herd, 14] Herd, B., Miles, S., McBurney, P., et al.: Verification and validation of agent-based simulations using approximate model checking, In: *Multi-Agent-Based Simulation XIV, Lecture Notes in Computer Science (LNCS)*, 2014, pp. 53–70.
- [Hunter, 20] Hunter, E., Kelleher, J. D.: A Framework for Validating and Testing Agent-based Models: a Case Study from Infectious Diseases Modelling, In: *34th. Annual European Simulation and Modelling Conference- Toulouse*, 2020.
- [Khattak, 15] Khattak, A. S., Khiyal, M. S. H., Rizvi, S. S.: Verification and validation of agent-based model using E- VOMAS approach, *International Journal of Computer Science and Network Security (IJCSNS)*, 2015; 15: 29–35.
- [Law, 03] Law, A. M.: Model Verification and Validation, In: *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sanchez, D. Ferrin, and D. J. Morrice, (eds.), 2003, pp. 66–70.
- [Law, 07] Law, A. M.: *Simulation Modeling and Analysis*, 4th edition. Boston, MA: MacGraw-Hill, 2007.
- [Macal, 09] Macal, C. North, M. J.: Agent-based modeling and simulation, In: *IEEE Proceedings of the 2009 Winter Simulation Conference (WSC)*, 2009, pp. 86–98.
- [Naylor 67] Naylor T. H.: Finger JM, McKenney JL, et al. Verification of Computer Simulation Models, *Management Science*, 1967; 14: 92–106.
- [Niazi, 09] Niazi, M. A., Hussain, A., Kolberg, M.: Verification and Validation of Agent-Based Simulation using the VOMAS approach, *ArXiv*, 2009, abs/1708.02361.
- [Niazi, 10] Niazi, M. A., Hussain, A.: A Novel Agent-Based Simulation Framework for Sensing in Complex Adaptive Environments, *IEEE Sensors Journal*, 2010; 11: 404–412.
- [Niazi, 11] Niazi, M. A., Hussain, A.: Agent-based Computing from Multi-agent Systems to Agent-Based Models: A Visual Survey, *Scientometrics*, 2011; 89: 479–499.
- [North, 07] North, M.J., Tatara, E., Collier, N. T., Ozik J.: Visual Agent-based Model Development with Repast Symphony, *Proceedings of the Agent 2007 Conference on Complex Interaction and Social Emergence*, Argonne National Laboratory, Argonne, IL USA (November 2007)
- [Office USGA, 87] Office USGA.: *DOD Simulations: Improved Assessment Procedures Would Increase the Credibility of Results*, 1987, Report ID: PEMD-88-3. Washington, DC.

- [Olsson, 22] Olsson, D., Rylander, E.: A Framework for Verification and Validation of Simulation Models in Esmini with an Example of Framework Application, PhD thesis, 2022, Chalmers University of Technology and University of Gothenburg.
- [Ozik, 08] Ozik, J. North, M. J.: Agent-based Modeling with a Dynamic Language: Platform Support for Modeling Endogenous Coordination, In: Proceedings of the Second World Congress on Social Simulation, 2008.
- [Repast Symphony, 24]. Repast Symphony.: <https://repast.github.io/> [accessed 12 January 2025]
- [Reynolds, 87] Reynolds, C. W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. *ACM Siggraph Computer Graphics*, 1987; 21: 25–34.
- [Riedmaier, 21] Riedmaier, S., Danquah, B., Schick, B., et al.: Unified framework and survey for model verification, validation and uncertainty quantification, *Archives of Computational Methods in Engineering*, 2021; 28: 2655–2688.
- [Roungas, 17] Roungas, B., Meijer, S., Verbraeck, A.: A Framework for Simulation Validation & Verification Method Selection, In A. Ramezani, E. Williams, & M. Bauer (Eds.), In: Proceedings of the 9th International Conference on Advances in System Simulation, SIMUL, 2017.
- [Sargent, 91] Sargent, R. G.: Simulation model verification and validation, In: 1991 Winter Simulation Conference Proceedings, 1991, pp. 37–47. 0-7803-0181-1.
- [Sargent, 13] Sargent, R. G.: Verification and validation of simulation models. *Journal of Simulation*, 2013; 7: 12–24.
- [Schieferdecker, 12] Schieferdecker, I.: Model-Based Testing, *IEEE Software*, 2012; 29: 14–18.
- [Sudeikat, 09] Sudeikat, J., Renz, W.: A Systemic Approach to the Validation of Self-Organizing Dynamics within MAS, In: Luck, M., Gomez-Sanz, J.J. (eds) *Agent-Oriented Software Engineering IX. AOSE 2008*, In: Lecture Notes in Computer Science, 2009, pp. 31–45.
- [Troitzsch, 96] Troitzsch, K.G., Mueller, U., Gilbert, N., Doran, J.E.: *Social Science Microsimulation, Multilevel simulation*, 1996, Springer.
- [Troitzsch, 04] Troitzsch, K. G.: Validating Simulation Models, Proceedings of 18th European Simulation Multiconference on Networked Simulation and Simulation Networks, In: SCS Publishing House, 2004, pp. 265–270.
- [Wright, 12] Wright, C. J., McMinn, P., Gallardo, J.: Towards the automatic identification of faulty multi-agent based simulation runs using MASTER, In: *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, 2012, pp. 143–156.