European Journal of Operational Research 254 (2016) 517–531

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

Test-driven simulation modelling: A case study using agent-based maritime search-operation simulation

Bhakti Stephan Onggo[a,*], Mumtaz Karatas[b]

[a] Department of Management Science, Lancaster University Management School Lancaster University, Lancaster LA1 4YX, United Kingdom
[b] Department of Industrial Engineering, Turkish Naval Academy, Tuzla, Istanbul 34942, Turkey

ARTICLE INFO

Article history:
Received 17 August 2015
Accepted 31 March 2016
Available online 8 April 2016

Keywords:
Agent-Based Simulation
Model validation
Maritime search operations
Repast

ABSTRACT

Model verification and validation (V&V) is one of the most important activities in simulation modelling. Model validation is especially challenging for Agent-Based Simulation (ABS). Techniques that can help to improve V&V in simulation modelling are needed. This paper proposes a V&V technique called Test-Driven Simulation Modelling (TDSM) which applies techniques from Test-Driven Development in software engineering to simulation modelling. The main principle in TDSM is that a unit test for a simulation model has to be specified before the simulation model is implemented. Hence, TDSM explicitly embeds V&V in simulation modelling. We use a case study in maritime search operations to demonstrate how TDSM can be used in practice. Maritime search operations (and search operations in general) are one of the classic applications of Operational Research (OR). Hence, we can use analytical models from the vast search theory literature for unit tests in TDSM. The results show that TDSM is a useful technique in the verification and validation of simulation models, especially ABS models. This paper also shows that ABS can offer an alternative modelling approach in the analysis of maritime search operations.

1. Introduction

Agent-Based Simulation (ABS) has become one of the commonly used tools to model and understand complex and nonlinear systems (North & Macal, 2007). ABS provides a controlled environment for systematic experimentation using a simulation model that is formed from a set of interacting agents. There is no consensus on the definition of an agent in the ABS literature (Macal & North, 2010). Instead, we have observed a spectrum of complexity in its definition. At one extreme, an ABS model is formed by a set of agents with a set of simple attributes (such as speed and detection range) and simple behaviours (such as move and rescue). At the other extreme, an ABS model can be composed of a set of agents with complex attributes (such as memory and bounded rationality) and complex abilities (such as planning and learning). However, most researchers agree that an agent is an autonomous entity (i.e. it makes independent decisions without any central control), has a set of objectives and interacts with other agents and its environment.

One of the most important activities in ABS is model validation. There are similarities between ABS and Discrete-Event Simulation (DES). Both are able to represent stochastic dynamic systems and can track individuals' states throughout their lifecycles in the model. Hence, a number of validation techniques proposed for DES are also suitable for ABS, such as face validity, operational validity, white-box validation and black-box validation. Balci (1995), Kleijnen (1995) and Sargent (2013) provide a list of validation techniques in the context of stochastic dynamic simulation which are applicable to DES and ABS. All good simulation textbooks have at least one chapter that discusses validation techniques. Law (2014, Chap. 5) and Banks, Carson, Nelson, and Nicol (2010, Chap. 10) discuss various techniques, such as increasing the face validity of a model (e.g. by involving domain experts and model users), checking the validity of model assumptions, validating the components of a model, comparing the behaviour of a model with a real system (or something that can represent a real system if a system does not exist). Pidd (2004, pp. 233–246) divides validation techniques into white-box and black-box validation. White-box validation techniques aim to ensure that the internal working of a simulation model can be justified. Black-box validation techniques compare the output of a simulation model with the output of a benchmark (such as a real-world system similar to the system being modelled or an analytic model).

Despite the similarities, some researchers (e.g. Duong, 2010; Klugl, 2008; Ormerod & Rosewell, 2006; Windrum, Fagiolo, & Moneta, 2007) have noted that model validation in ABS is especially

* Corresponding author. Tel.: +44 1524 594237; fax: +44 1524 592060.
E-mail address: s.onggo@lancaster.ac.uk, stephan.onggo@gmail.com (B.S. Onggo).

http://dx.doi.org/10.1016/j.ejor.2016.03.050
0377-2217/© 2016 Elsevier B.V. All rights reserved.

challenging and identified a number of common challenges. First, we often need to represent the behaviours of agents and the interactions between agents using a set of logical rules. It is challenging to extract this information from social and intelligent agents, such as people and organisations, especially if the agents do not want to be exposed (such as pirates or human traffickers). Furthermore, real-world agents are often heterogeneous. Hence, it is challenging to validate whether the rules used in an ABS model represent the rules used by most real-world agents and whether we have represented the heterogeneity of real-world agents correctly. Second, there is a need to validate ABS models at various levels (agent/micro level, system/macro level and intermediate/meso level). It is challenging to validate behaviour at the system level based solely on knowledge of the behaviours of individual agents. For example, Duong (2010) explains that emergence does not exist before a simulation is run (it might not even exist in the modeller's mind); hence, techniques such as structured walkthrough to analyse emergence from a model without running it would be virtually impossible. Even if we can generate traces during a simulation run, it is still a great challenge to explain how behaviour at a lower level can cause emergence at a higher level. Finally, an ABS model often requires high-fidelity data. Although the collection of high-fidelity data has become very common, qualitative behavioural data from heterogeneous agents in a population are rarely available. Hence, empirical validation may not be possible. The difficulty in validating an ABS model is reflected somewhat in the survey done by Heath, Hill, and Ciarallo (2009). They surveyed 279 research articles and found that only 35 per cent of the models were validated both conceptually (white box) and operationally (black box). Windrum et al. (2007) conduct an interesting discussion about the methodological issues surrounding the empirical validation of ABS. Hence, the challenge is not simply one of data availability, it is also methodological. Given these challenges, an automated tool that can help modellers to validate ABS models is useful.

The objective of this paper is to propose a technique called Test-Driven Simulation Modelling (TDSM) for the verification and validation (V&V) of an ABS model. TDSM's basic principle is that a test case for a simulation model has to be specified before the simulation model is implemented. Hence, the main advantage of TDSM is that the V&V process is explicitly embedded in the simulation-model development process. Modellers are forced to think about how their model is going to be verified and validated, even before they begin to develop it. The second advantage is that TDSM can be implemented using various unit-testing tools, or incorporated into a framework such as the one proposed by Gurcan, Dikenelli, and Bernon (2013), Niazzi, Hussain, and Kolberg (2009) and Thiele, Kurth, and Grimm (2014). To demonstrate how the proposed validation technique can be used in practice, we will show a few examples in which parts of the Agent-Based Simulation software called MASSIM (MAritime Search SIMulation) are verified and validated using TDSM.

The analysis of search operations is a classic part of Operational Research (OR) and has a long history commencing with military operations during the Second World War. Since then, the applications and tools have continued to be mostly nautical, e.g. military assets looking for enemy submarines or maritime pirates, coastguards conducting search-and-rescue (SAR) operations, and patrol boats protecting ports or high-value assets. Since a search operation requires considerable time and effort, it needs to be planned and conducted efficiently. A tool that can help to analyse the expected performance of a search-operation strategy is very useful. Tools that use ABS in the maritime search domain and sea-patrol operations are surprisingly scarce in comparison to other OR techniques, as confirmed by Davidsson, Henesey, Ramstedt, Törnquist, and Wernstedt (2005) and Vaněk, Jakob, Hrstka, and Pěchouček (2013). One of the main reasons is the difficulty in validating the ABS model. This is unfortunate because ABS has unique characteristics (such as the explicit specification of individuals' behaviours and their interactions) which offer an alternative modelling approach. Hence, the findings from this paper can also contribute to the application of ABS in maritime-search operations modelling.

This paper is organised as follows. In Section 2, we review related work in TDSM and the application of ABS in maritime search operations. We explain our generic tool for maritime search operations, called MASSIM, in Section 3. Section 4 discusses how TDSM is used to validate a number of scenarios in MASSIM. Finally, we present our conclusions and recommendations for future work in Section 5.

2. Related work

2.1. Test-Driven Development in Simulation Modelling

The idea behind Test-Driven Simulation Modelling (TDSM) comes from Test-Driven Software Development, known simply as Test-Driven Development (TDD) in software engineering. There are a number of definitions and aliases for TDD (Janzen & Saiedian, 2005). However, the main principle is that a test case for computer code must be created before the computer code is developed (Beck, 2003). The test case is implemented as a unit test. Unit testing is a method in software engineering that is used to test an individual unit of computer code. The individual unit can be a function, a procedure or a class. Because the code does not exist when the unit test is created, the first test will always fail. A programmer will then refine the code until it passes the unit test. Subsequently, a new unit test is created and the process is repeated. At some point, the programmer may need to refactor the code. Refactoring changes the internal structure of the code without changing its observable behaviour (i.e. the code still carries out the same function that it did before). The objective of refactoring is to make the code more readable and easier to maintain. These steps are commonly known as Red-Green-Refactor, which signifies the cycle of creating unit tests that fail, writing code that passes tests and refactoring the code. TDD is a practice that is commonly used with other practices in a software development process.

TDD is suitable for a software development process that is iterative, incremental and evolutionary (Janzen & Saiedian, 2005). A development process is iterative when it involves the repetition of development tasks, usually with an incremental set of requirements. Each increment in the requirements results in a new software release. An evolutionary development process uses feedback from previous iterations to improve the software and guide future iterations. Simulation modelling is also an iterative, incremental and evolutionary process. Simulation modellers often revisit stages in a simulation modelling process (e.g. conceptual modelling, computer implementation) iteratively. In each iteration, specifications may be modified and new specifications may be added based on new information or feedback from relevant stakeholders. Hence, the simulation-modelling process is likely to benefit from TDD.

Collier and Ozik (2013) conducted the first exploratory study to investigate how TDD and unit testing could be used to verify an ABS model written in the Repast simulation library. They argued that by focusing on writing small unit tests, complex simulation code could be decomposed into smaller and more manageable components. Asta, Özcan, and Siebers (2014) followed the same approach and applied it to a Discrete-Event Simulation (DES) model written using AnyLogic. Unlike Repast, AnyLogic is a visual interactive modelling software. Hence, the unit-test code was not written directly as simulation source code. Instead, they wrote each unit test using a user-defined function in AnyLogic to verify a component in the model. Onggo, Indriany, and Gunal (2014) proposed