Scientific Journals
of the Maritime University of Szczecin

Zeszyty Naukowe
Akademii Morskiej w Szczecinie

2015, 42 (114), 101–111
ISSN 1733-8670 (Printed)
ISSN 2392-0378 (Online)

Agent-based approach to illegal maritime behavior modeling

Ondřej Hrstka[1], Ondřej Vaňek[1], Štěpán Kopřiva[1], Jiří Zelinka[2], Jan Faigl[1]
Michal Pěchouček[1]

[1] Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Computer Science Technicka 2, Praha 6, Czech Republic,
e-mails: {hrstka; vanek; kopriva; faigl; pechoucek}@agents.fel.cvut.cz
[2] Baktun s.r.o, Pražská 596, Městec Králové, Czech Republic, e-mail: jiri.zelinka@blindspot-solutions.com

Key words: simulation, multi-agent systems, behavior modeling, drug interdiction, illegal behavior, maritime

Abstract

Maritime shipping is a set of complex activities with a large number of actors involved. We focus on a subset of illegal maritime activities, such as armed robberies, maritime piracy or contraband smuggling. To fight against them and minimize their negative impact naval authorities typically introduce a number of countermeasures, such as deployed patrols or surveillance agents. Due to very high costs of countermeasures it is often beneficial to evaluate their impact using a simulation, allowing what-if analysis and evaluation of a range of scenarios before actually deploying the countermeasures.

We introduce BANDIT, an agent-based computational platform, which is designed to evaluate scenarios with an accent on the modeling of different types of illegal behavior and on the interaction between agents. The platform consists of an agent behavior modeling system and a multi-agent maritime simulator. The platform allows the definition of a number of scenarios through a simple configuration and it offers the means to run these scenarios in a single or a batch mode and evaluate the results as single or aggregate data sets respectively. We demonstrate the usefulness of the platform on the scenarios of the drug smuggling problem in the seas surrounding Central America. Senario outcomes (e.g., heatmaps of activities, set of trajectories etc.) are subsequently used to help with the design of effective countermeasures, i.e., allocating naval patrols and planning their patrol routes.

Introduction

Currently decision makers allocating expensive assets to intercept possible maritime smuggling activities face the problem of evaluating daily strategies for their potential efficiency before actual deployment. To be able to evaluate the efficiency of the allocations and strategies, a computational platform able to assess the quality of the solution is required. The platform is required to model the behavior and decision-making process of drug smugglers as realistically as possible and it should allow the modeling of various scenarios compatible with available INTEL (Intelligence) and METOC (Meteorological and Oceanographic data) provided by the decision makers.

In this paper we present the Behavioral Agents for Drug Interdiction (BANDIT) platform. As the name suggests the platform was designed to help in drug interdiction activities, however the software was designed to be domain independent – it can be used in other areas with minimal effort. BANDIT allows the design of scenarios in the maritime domain with agents representing the individual vessels. The behavior of each agent is described by the Behavior State Machine. The BSM is a framework we have designed and implemented that allows the user to quickly define the complex behavior of the agent (its lifecycle and reactions to external events). This approach allows a scenario to be defined with great complexity, which is achieved by a combination of individual agents' behaviors.

The paper is organized as follows. In the section *Maritime drug smuggling problem – war on drugs* we summarize the background of the war on drugs with a focus on the maritime area around the

Central America region. In the section *Related work* we describe AgentC, which is a predecessor of BANDIT focused on the piracy related simulations around the Horn of Africa. We also describe the work at the Navy Research Lab in Monterey (NRL), since BANDIT was developed with the needs of the NRL in mind. The system architecture of the BANDIT platform is described in the section *System architecture*. The BSM framework is described in the section *Agent behavior models*. Individual agent behavior model examples are introduced in the section *Behavior models of traffickers*. We have created several behavior models of the drug traffickers, in which the behavior is based on documented real-world cases. In the section *Scenarios and evaluation* we describe the scenarios which are evaluated using the BANDIT framework.

Maritime drug smuggling problem – war on drugs

The war on drugs in the United States dates back to the 1970s era of President Nixon. His administrative reacted to the rise of drug use among US citizens by organizing law forces and creating new departments (e.g. the Drug Enforcement Administration – DEA). The escalation of law enforcement resulted in a further increase of illegal drug activities which led to the lasting war on drugs – the never ending cat and mouse game which continues today. *"There were 80% more arrests for drug possession or use in 2010 (1,336,530) than in 1990 (741,600). Between 1990 and its peak in 2006, the arrest rate for drug possession or use increased 75%. The arrest rate declined between 2006 and 2010, ending in 2010 at 46% above its 1990 level and at a level similar to those seen between 1997 and 2002"* (Drugwarfacts, 2015a). In the year 2012, 418 tons of cocaine was seized in South America (Drugwarfacts, 2015b).

During these decades the US authorities, as one of the counter moves against the drug business, tried to deter smuggling operations within US borders, as well as in the Central America region. Movement of drugs can be made by airplane, over land or by ship. A particular drug smuggling operation can be performed by several follow-up methods. For example, the drugs shipment can be sailed from Columbia to the southern part of Mexico, then taken by truck to the Mexico-US border, where it can be smuggled by aerial means, including by drone (BBC, 2015).

In this paper we focus on the maritime domain, which is hard to control because of the vast areas and the lack of the awareness due to limited sensor and equipment capabilities.

Tactics of smugglers

Knowledge about the tactics and reasoning of the smugglers is quite limited (Decker & Chapman, 2008, p. 3). Decker and Chapman try to fill this void, where they report interviews with convicted drug smugglers imprisoned in the US. They provide an insight into different stages of smuggling operations such as drug movement, recruitment of the crew etc.

Methods used by smugglers to avoid the authorities can be categorized into two types: speed or stealth. Using speed a smuggler tries to cross the monitored area as fast as possible in the hope that the patrol will be looking elsewhere. With the stealth approach the smuggler tries to blend into the environment, either by appearing to be a legal vessel (i.e. fishing vessel) or by camouflaging himself, so he cannot be spotted by a patrol. These two approaches are often used together and are dependent on environmental conditions. For example the *go-fast* boat with *tarping* described below.

Go-fast: The go-fast boats are super speed boats used by smugglers. They are the most basic surface vessel used by the narco navy (Elkus, 2012). A typical go-fast boat is 30 to 50 feet long with a narrow beam and powerful engines delivering up to 1000 hp (Tunaley, 2010). They usually travel at night and achieve high speeds to reach long distances.

The original go-fast boats were refurbished boats, for example the Eduardoo fishing vessel. During recent years drug smugglers have started to use a new kind of go-fast boat called the *picuda*. The picuda is a go-fast boat made out of fiberglass. Fiberglass construction results in a very low radar profile. It also makes the ship lighter so it can achieve greater speeds, carry more cargo and consumes less fuel (Fiegel & Picuda, 2014).

There is also another type of smuggler boat known as the *panga*. A panga is a small boat with a narrow beam. In some sources pangas are a slower variant of the go-fast boats, whereas other sources consider panga and go-fast equal in terms of speed.

To avoid being spotted by coastguard patrol aircraft during the day, smugglers use so called tarping tactics. At sunrise the go-fast crew stops the engines and covers the hull with a blue tarp (Alvarez, 2014). Then they drift with engines turned off until sunset, when they remove the tarp, start their engines and continue. Because of their low radar profile and because of the absence of the wake

behind the vessel when drifting, it is nearly impossible to spot the boat during the day.

Environmental factors

We are not sure which environmental factors are significant in the behavior of the drug smuggler because of the lack of information about their reasoning. However, we can safely assume that the weather will play a pivotal role.

The weather acts as a double-edged sword. Clear weather conditions (clear surface, low wind and currents) increases the safety of the smuggler vessel on one hand, but on the other hand this also increases the ability of the authorities' surveillance technology. Hence smugglers appear to prefer those conditions which are still possible to pass through, but reduce the surveillance technology ranges. In some cases the smuggler can overestimate his abilities, which can have disastrous consequences. There have been reported findings of capsized go-fast boats drifting in the sea.

Another environmental factor is the date, with regard to public holidays or sea races. The smugglers use increased traffic to blend in and avoid the patrols.

Data sources

This section summarizes two external data sources used in the BANDIT platform: the OpenStreetMap data source used to obtain shorelines data and the data source for Meteorological and Oceanographic (METOC) data source.

1) *OpenStreetMap data:* The OpenStreetMap (OSM) project is an open source collaborative project to collect and maintain various map data around the world[1]. The project is under the Open Data Commons Open Database License[2]: *"You are free to copy, distribute, transmit and adapt our data, as long as you credit OpenStreetMap and its contributors. If you alter or build upon our data, you may distribute the result only under the same licence..."*

For the BANDIT project we have used the shorelines data from the OSM. The data can be accessed from the website http://openstreetmapdata.com/data/land-polygons. The pre-processing of the data for the simulation is described in the section *System architecture – Environment: Geography*.

2) *Meteorological and Oceanographic data:* The BANDIT platform is designed to work with the Meteorological and Oceanographic (METOC) data from the US Navy Fleet Numerical Meteorology and Oceanography Center. *"Fleet Numerical's primary mission is to provide the highest quality, most relevant and timely worldwide Meteorology and Oceanography support to US and coalition forces from our 24x7 Operations Center in Monterey, California."* The data from this service are not publicly available, however the platform is developed to work with data formats used by the Naval Research Laboratory and tested on dummy data.

The METOC data contains information about three weather elements: currents, wind and wave height. The currents and wind information are in the form of a vector, whereas the wave height is scalar information. The METOC is sampled in time and space. The space is represented by a rectangular grid. One METOC record contains the information about the world weather situation and weather prediction as well. This allows us to work with historic weather data, if we simulate the events in historic date and time and use weather prediction for simulations for the near future.

Related work

Firstly, we summarize here related work in the drug-smuggling and maritime piracy domain. We then follow this with related work focused on methods and techniques, used mainly on finite state machines and agent-based simulation.

AgentC

AgentC is an agent-based large-scale simulation of maritime traffic in the Indian Ocean developed under the supervision of the Principal Investigator at the Czech Technical University in Prague (Jakob et al., 2009; Jakob et al., 2010). The simulation contains several components allowing effective modeling of maritime traffic with adversarial forces present: (1) a set of behavioral models of pirates active in the Indian Ocean, (2) a multi-objective planner of maritime transit routes through piracy infested waters, (3) a set of optimization modules allowing game-theoretic path planning, asset allocation and transit grouping.

Currently, the AgentC framework is focused on the Indian Ocean only and it is not easily transferable to other areas (such as the Gulf of Guinea where piracy is on the rise, or to the Eastern Pacific where drug smuggling activities are of a major concern). Additionally, the framework does not provide sufficient scenario scripting capabilities, such as scenario-based behavior description and environment definition. The environment model does not fully integrate METOC information and does not take into account environmental influence of vessel movement.

[1] http://www.openstreetmap.org/
[2] http://opendatacommons.org/licenses/odbl/

Related work of the Naval Research Laboratory, Monterey

1) *Pirate Attack Risk Surface*: Jim Hansen has worked for a number of years on a predictive model of maritime pirates' activity called Piracy Attack Risk Surface (PARS) (Hansen et al., 2011). The model combines a range of data layers, including wave and ocean current information with historic and recent pirate activity. The resulting index communicates the suitability of pirate activity as a function of location and time and is used operationally by US, EU, and NATO interdiction forces.

Other academics are using PARS to find optimized allocation of assets active in the area (e.g. An et al., 2012).

2) *Counter-smuggling Problem*: Prof. Pattipati et al. (2013), Mishra et al. (2014) and Sidoti et al. (2014) focus on applying optimization techniques for counter-smuggling operations in EASTPAC and CARRIB. The key objective is to find routes for a set of assets, given the start and end locations, such that the total traversal time, dispatch time and the wait time at each intermediate location is minimized. Given a task graph over Time-dependent Multi-objective (TM) risk maps, they formulate and solve a Time-dependent Multi-objective Shortest Path (TMSP) problem to determine asset routes in a multi-task scenario. They employ the method of compromised solution along with mixed integer linear programming to solve this NP-hard problem.

Hansen (2014) works on intelligent assets allocation for counter-smuggling operations for JIATF South. He focuses on modeling the uncertainty of intelligence provided, radar detection range evolving in space and time and how the approach should be used in an operational environment. Hansen also considers difficulties in deploying the system for everyday users and operators.

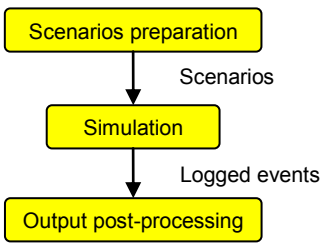

Figure 1. The BANDIT lifecycle visualization. Each part of the simulation is separated from others which allows a modular approach to the platform

Agent-based approach to modeling

The use of agent-based or simulation-based models to support policy design and operational management has a very long-standing tradition in the transportation field. The use of simulation as a validation or solution quality evaluation tool is relatively novel and is not a fully explored research branch. In this section, we first look at agent-based simulations themselves: multi-agent simulation design and internals. Then we provide an overview of the most relevant simulations (focusing on the transportation domain, infrastructure security and maritime piracy).

1) *Multi-Agent Simulation Concept:* Multi-Agent Systems (MAS) provide a descriptive framework that is appropriate for many real-world systems consisting of a set of interacting autonomous actors. Human and animal societies form prominent and intuitive examples of real-world multi-agent systems. Klügl (2009) provides very detailed insights into design and "engineering" of multi-agent simulations. In every MAS there are four aspects that are relevant for capturing the notion of multi-agent systems and agent-based software: the *Agents* forming an *Multi-Agent System* based on their *Interactions* and situated within an *Environment*.

System architecture

In this section we describe high-level architectural elements of the BANDIT platform. This description does not include any implementation details, which are beyond the scope of this paper.

BANDIT platform

The BANDIT platform is intended to be run as a web service on the simulation server. It awaits a request to execute a simulation given by the scenario specification. A typical scenario describes Monte Carlo simulation (Mahadevan, 1997) by specifying a number of iterations and parameters of the simulation where some of these parameters might be specified as random distributions. From this scenario multiple simulation instances are sampled and executed. Results are then statistically processed.

The simulation execution can be separated into three phases (Figure 1):

1) Scenarios sampling;
2) Simulation of the scenarios;
3) Output post-processing.

The scenarios sampling phase is responsible for processing the input data and generating the scenarios to simulate. This is performed by sampling the batch scenario to get specific simulation instances with specific parameter values that are ready to be executed. The batch scenario describes parameters of the Monte Carlo simulation, whereas the simulation instance is the BANDIT data structure representing the simulation that is ready to be executed.

Therefore the simulation execution works with the "scenarios" that have all parameters instantiated. It does not allow an input with random values. The simulation can use random number generators for its execution, however because the random generator initial seed is also a parameter of the simulation input, the simulation itself is repeatable.

There is a singular case if the number of iterations in the input is exactly one. The post-processing of this type of scenario in BANDIT is different from the ordinary post-processing. To differentiate between this singular and regular cases, we will call *single* scenario the singular case and *batch* scenario the regular case (with iterations greater than one).

Although the first phase generates scenarios by sampling the set of probability distributions, it is not the only way to accomplish this task. Another approach would be to receive a set of already sampled single scenarios on the input and just parse them into BANDIT internal data structures. This illustrates the options of the modular approach to the task of processing the input.

The second phase (simulation of the scenarios) executes all prepared simulations. Since the BANDIT platform was designed to support parallel execution of simulations, we execute multiple simulations in parallel. The number of threads can be set as a parameter of the BANDIT platform.

The last phase (output post-processing) processes all measured data from simulations into suitable outputs. This is usually statistical data processing. See section *System architecture – Output* for details.

Platform architecture

The BANDIT platform consists of two main components: the simulator and the agent behavior models. The simulator (section *System architecture – Simulator*) is responsible for simulation of the virtual world, whereas the behavior models (section *Agent behavior models*) control simulated vessels using artificial intelligence – planning the actions, reacting to the inputs etc. These components are strictly separated and all communication is through a specified interface. This approach allows a component to be replaced, should such a need arise, and to test the behavior models independently of the simulation.

System modules

Both major components – the simulator and the behavior model – communicate through an interface called the Controller. The architecture is depicted in Figure 2.

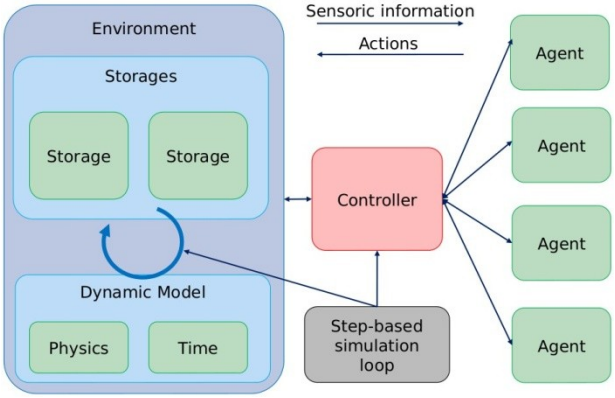

Figure 2. High level architecture of the BANDIT platform. The arrows represent how the data and function calls are propagated through modules of the architecture. Note that the Controller is the interface that connects agents with the environment

The figure depicts the main modules of the BANDIT platform. The environment represents physical items in the simulated world. It consists of two components: the *storages* and the *model*.

a) *Storages:* The storage is an aggregate data structure responsible for keeping the actual state variables of the simulated entities. The state variables describe the physical part of the virtual simulated world, i.e., the environment and the physical embodiment of the modeled agents.

b) *Model:* The model describes the dynamics of the system. At each simulation step, each model is called with the update of the simulation time. The model is then responsible for updating the storage with new state variables. The variables are updated based on the modeled behavior, i.e., on the selected actions of the agent.

c) *Controller:* The controller is an interface between the agent and the environment. It forwards sensory information from the environment to the agent and propagates action calls from the agent to the environment.

Simulator

One of the key issues in the process of designing the simulator is how to represent the flow of time. The two basic approaches for a multi-agent simulation are (1) a discrete event simulation and (2) a time stepped simulation.

The discrete event simulation is based on the fact that important state changes occur only in specific discrete time instants. This change is called the event. Between two consequent events the state variables remain the same. Therefore the simulation clock can jump directly to the time of the next event.

The time step simulation updates the state periodically for a given period Δt (Sislak et al., 2009).

After each update, the agents are given control to update their internal state and execute actions to modify the environment. The value of Δt determines the temporal resolution of the simulation. If Δt is too big, there is a chance that an important event in the simulation will be missed (e.g. two vessels will come into each other's sensor ranges without noticing it). On the other hand, if Δt is too small this results in lots of updates where there is no significant change to the model.

For the BANDIT we have chosen the time stepped approach to be compatible with the NRL current approach. However, because of the decomposition of the simulator, we can switch to the event based approach with only a minimal effort.

Environment

1) *Geography:* The simulation requires real world data about location of coast lines. We use data from OpenStreetMap (described above) for this purpose. The OSM service provides land polygons data that can be directly downloaded from the website. The data however contained too much detail for the BANDIT platform. Since the simulator's purpose is to simulate activities on open seas, we had to simplify the dataset to prevent performance issues.

2) *METOC:* The BANDIT platform supports working with METOC data records. The METOC data records are taken at regular intervals (in this case 6 hours – at 00, 06, 12 and 18 hours). Each record contains the wind, current and wave data. The first two are vectors which represent the direction and magnitude (in knots) while the latter is only a scalar representing the height (in feet).

Each record also contains the weather prediction for the following 72 hours. The prediction is sampled at 6 hour intervals as well. Therefore the data forms a hyper cube where the dimensions are position, timestamp of the snapshot and the time of the prediction.

The movement model

The movement model is responsible for correct movement of vessels in the water. There is one model for each vessel in the simulation. During each step, the model calculates the vessel's next position from the propulsion action and the wind and wave information from METOC data provided. The METOC data are weighted by coefficients `pushed_by_wind` and `pushed_by_waves` to calculate the influence of wind and waves upon the vessel respectively. The model also ensures that the vessel will not sail behind the coastline. If the agent were to try to issue such an action, the vessel would simply stop on the coastline.

When the model indicates that the vessel has reached a desired target it sends the respective event to the controller. It is then up to the controller (or the agent) to decide what the next movement action will be.

Interaction

The interaction model handles all interactions between agents in the simulation. The interaction is typically started with an action from one of the agents. The interaction model receives the action event (output event) from the controller and handles the action according to the simulated environment. Usually this results in one or more reaction events (input events) that are dispatched after duration Δt. Unlike the movement model, there is only one interaction model in the simulation instance shared by all the entities.

We demonstrate here the interaction on the example. Let's have two agents: agent A and agent B. Agent A initiates a transshipment interaction. The interaction model receives the `transshipment_start` event and sends the `transshipment_initiated` event to the agent B. It also schedules sending event `transshipment_finished` to be sent to agents A and B after duration Δt, together

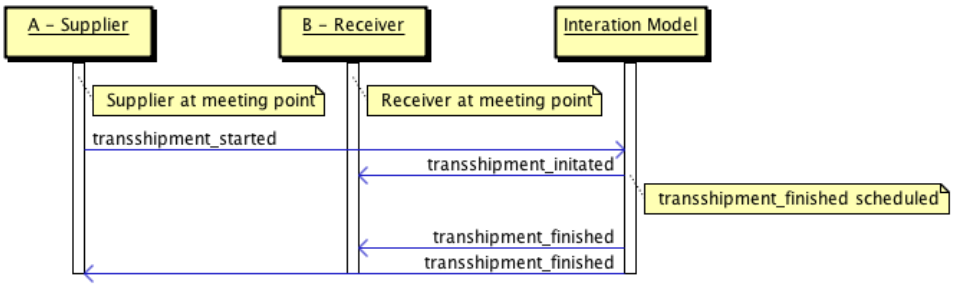

Figure 3. Interaction diagram of the transshipment example. The diagram shows interaction between all three participants of the transshipment. The time flows from top to the bottom, each interaction action is represented by an arrow from sender (initiator) of the action towards the receiver

with changing the environment information about the owner of the transshipped cargo. If there is no interruption of the transshipment operation, the scheduled events will be sent and the transshipment interaction will be finished. This example is depicted in Figure 3.

The interaction model also handles the alarm functionality, which is a special kind of interaction where an agent interacts with itself. Since the agents are strictly reactive, we need to have a mechanism to schedule operations in the future. For this, an agent can schedule to have sent a desired event at a given time (or after a given period) using an alarm. This mechanism is used for example for the implementation of the tarping behavior described above.

Output

The output format of the simulation is according to the requirements specified by the NRL. The individual simulation results are aggregated into a probabilistic representation.

The results form a list of probabilistic grids, where each grid represents data for certain simulation time windows (i.e. 3 hour time windows). The grid values represent the probability of the occurrence of the trafficker given the coordinates. Both temporal (size of the time window) and spatial (size of the grid cell) values are parameters of the post processing algorithm and are now set to values used by the NRL.

The simulation offers the output in the JSON format or in the KML format. The latter is usually used for the visualization (debugging, presentation etc.) rather than for machine processing, since the JSON format is easier to process for this kind of data.

Agent behavior models

The agent behavior model describes how an agent should react on given input from the simulation. The inputs are, for example, notifications that the agent's vessel has reached its destination or that another vessel is within the sensor range. In the BANDIT platform we have developed the Behavior State Machines which are based on experience gained from the AgentC development.

Behavior State Machines

The behavior state machines (BSM) are based on a hierarchical finite state machines approach (hFSM) (Girault, Lee & Lee, 1999). However the states are more complex than the states in hFSM. We explain the difference in the following paragraphs. First we describe a simple BSM without hierarchical property, and then we will generalize to the hierarchical BSM.

From the outside point of view BSM can be perceived as a black-box that receives *input* events and responds by *output* events. Just like FSM, BSM can be pictured as an oriented graph, where nodes represent states and edges represent transitions. The BSM is purely reactive – it can send output events only as a response to an input event. When the BSM receives an input event it will return a list of output events. The list can be of any length, including an empty list. During the processing of the input event, BSM can also switch to another state.

The important part of BSMs are so called *guards*. The guard is a Boolean function (predicate) applied on an event. If the event passes the guard (return value is `true`), it will evoke some kind of reaction. The guard can be an arbitrary complex function ranging from simple matching of the type of event, to complex evaluation of the parameters of the event. For example, the guard can compare only the fact that the event is of type `vessel reached designated waypoint` or it can compare parameters of the `distance event` if another vessel is in range (*less than x*).

The guards can be attached to both transition edges (transition guards) and states (internal response guards). For the transitions the guard can trigger transition between the states. The internal response guards can trigger the function that returns the list of the output events, which represents the response of the state to a given input event.

The input event is processed in a cascade manner. First the event is compared against transition

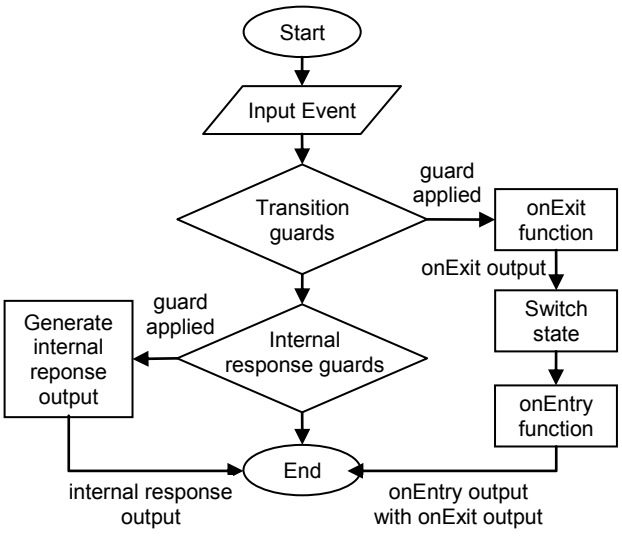

Figure 4. Event processing schema of the BSM. The flow diagram describes the processing algorithm for one event received by the BSM

guards for the given state. If no such guard is satisfied, the event is compared against internal response guards. If the input event did not trigger either any transition guards or any response guard, the event is ignored. The whole process is depicted in Figure 4.

The transitions between states are an ordered set of tuples (`guard`, `target state`). The guards are applied on the event in the order in which they are stored in the set. When the guard that fits the event is reached, the evaluation loop is stopped and the state is switched to the target state from the current tuple (`guard`, `target state`). During the process of state switch the `onExit` and `onEntry` functions are called.

The `onEntry` and `onExit` functions are called when the appropriate state is entered or exited respectively. Both functions take one argument of type of input event and return a list of output events. Therefore, during the state transition the agent may generate output events.

If the event is not "consumed" by any transition guard it is evaluated against response guards in the same manner. However the responses are an ordered set of tuples (`guard`, `response function`), where the response function takes the input event as argument and returns a list of output events (therefore it is of the same type as `onEntry` and `onExit` functions).

Hierarchical behavior state models

The BSM can be hierarchical in the same manner as hFMS. This means that each state can contain another BSM that represents its internal responses. When the input event reaches the second phase of the processing (meaning it did not trigger any transition guard), the event is handed "down" to the BSM inside the state which reacts accordingly.

The hierarchical BSM raises the question of how to handle the state of the internal BSMs. This question is pointless for simple states where the internal variable can be affected by `onEntry` and `onExit` functions. However, these are not always convenient for hierarchical states. For this reason each BSM also includes the `stateSelector` function. This function determines into which state the BSM switches if it is activated by the event. The activation by the event occurs when the BSM switches into hierarchical state.

The `stateSelector` is defined by the user. For example, it can leave the BSM in the state in which it was before. This is useful for situations when the agent needed to respond to an input event and after reaction it wants to continue with the previous activity. Or it can always select one state. This would represent a situation when an agent needs to start over and cannot (or does not want to) continue with previous unfinished activity.

1) *Testing of the BSM:* Because the BSMs (and therefore agents) in the BANDIT platform are strictly separated from the simulation part, BSMs can be tested separately of the simulation. To test the behavior it is only necessary to supply input events and observe if the output events match the prescribed behavior.

Behavior models of traffickers

In this section we present several behavior models used in the BANDIT platform. The behavior models can be either individual states or BSMs that are used as a state. Functionality of these models can be combined together.

Waypoint-based navigation

The waypoint-based navigation behavior model is defined by a list of waypoints specified in GPS coordinates. The agent follows the shortest path connecting the ordered list of waypoints – when the agent reaches the designated waypoint it will follow to the next waypoint in the list.

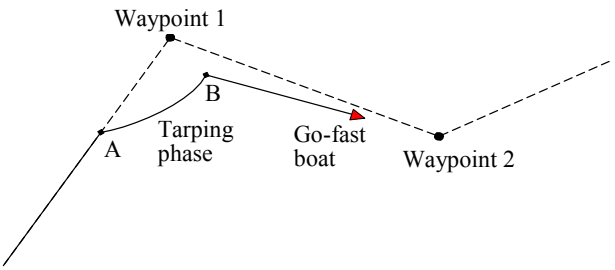

Figure 5. The situation where the agents drifts around the waypoint during the tarping. The tarping happens between point *A* and *B*. After the tarping the agent does not come back to the *Waypoint 1* but continues to the *Waypoint 2* instead

During the movement the agent's vessel is affected by sea currents, winds and waves. The agent compensates the effect of these environmental factors in order to keep the intended course. This results in changes of speed, but the direction remains the same.

Tarping behavior

The tarping behavior models the tarping technique of the smugglers. At sunrise the agent interrupts its current activity and starts tarping. The agent basically leaves the vessel to drift, its movement only being affected by the environment. We

do not simulate the camouflage part of the tarping at this stage (as we do not have information on current sensors). The tarping is finished at sunset, when the agent returns to its previous activity.

During the tarping it may be that the environmental factors would be pushing the vessel in the correct direction and at some point it would drift around its designated waypoint. When the tarping is finished it would not be rational to head back to the waypoint and then turn back to the next waypoint. Therefore, when this situation occurs, the agent is able to detect it and switch to follow on to the next waypoint. The situation of this "over-tarping" of the waypoint is depicted in Figure 5, where the agent starts tarping before reaching the *Waypoint 1* at a location denoted by the letter *A* and stops the tarping at the location denoted by the letter *B*.

Transshipment behavior

The transshipment behavior models the situation when two agents (typically smugglers) need to transship some cargo in the open sea. The type of cargo is a parameter of the behavior. It could be drugs shipments or fuel from the support ship for the go-fast boat.

Another parameter of the transshipment is the location and time of the meeting. Agents try to arrive at the location on time, however they might be late because of environmental factors. In that case the first agent waits for the second one. When both agents are in place, the transshipment begins. The handover of cargo takes a non-zero duration. When the transshipment is finished, both agents switch to their next respective behavior according to the agent model.

Combining behavior models together

By combining the individual behavior models together we can assemble various complex agent models. For example, we combine the waypoint-based navigation behavior with the tarping behavior to create the typical go-fast agent. For the transshipment situation we can combine waypoint-based navigation behavior, tarping behavior and the transshipment behavior for one agent representing a go-fast boat going to refuel whereas the second agent would be composed of the waypoint-based navigation behavior and the transshipment behavior only.

Scenarios and evaluation

In this section we present several scenarios defined in the BANDIT platform. Unfortunately the NRL scenario definitions cannot be publicly released, hence we cannot also present them. Therefore we describe the scenarios that were made to validate and demonstrate the platform capabilities.

Go-fast single scenario

The go-fast single scenario represents drug smuggling by go-fast vessels. Individual smugglers are described in the scenario configuration. The scenario may contain an arbitrary number of smugglers.

Each individual smuggler is specified by the following parameters:

Parameter	description
`pushed-by-ww`	how much the wind and currents affect the vessel
`speed`	travel speed of the vessel
`origin`	the start location of the vessel
`waypoints`	the waypoints of the intended trajectory
`destination`	the destination location

The `pushed-by-ww` is a linear weight parameter in a range [0; 1]. The weight describes how much the vessel is affected by wind and currents. If the value is 0, the vessel is not affected at all and without any other propulsion it will stay at its current location. If the value is 1 the vessel will be affected by the full strength of the combined forces of the wind and the current. If the vessel was without any propulsion, its speed vector would match the sum of the speed vectors of the wind and currents. If the vessel was to try to move, this summed vector would be added to its movement.

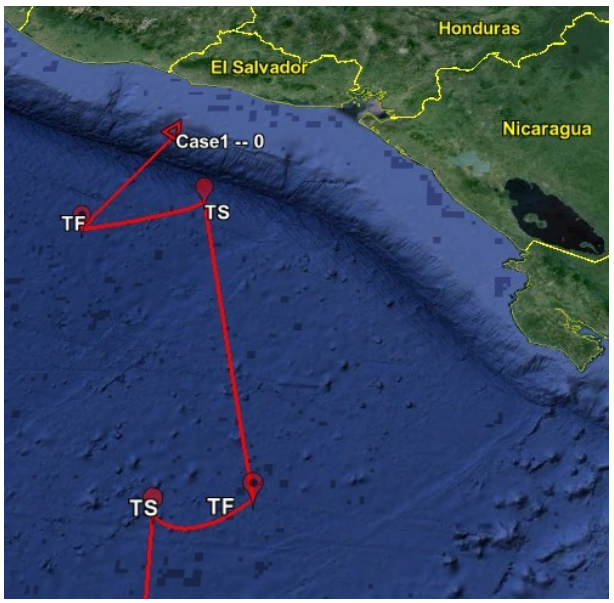

Figure 6. Visualization of the single scenario output. The TS and TF labels stand for Tarping Start and Tarping Finished

The `speed` parameter is the maximal speed of the vessel (in knots). We assume that the vessel travels at a maximal speed for the whole journey. However, the speed is affected by the wind and currents (assuming that the `pushed-by-ww` is non-zero).

The `origin`, `waypoints` and `destination` parameters together specify the route which the smuggler will follow. The `origin` and `destination` parameters are single 2-tuples representing longitude and latitude. The `waypoints` parameter is the list of 2-tuples representing position (same as `origin` and `destination`). The `waypoints` represent intermediate points along the route. This list is allowed to be empty, which would mean that the route is specified only by the `origin` and by the `destination`.

Go-fast batch scenario

The go-fast batch scenario is an extension of the go-fast single scenario. Unlike in the single scenario, the simulation in the batch scenario is executed multiple times, each time with randomly sampled parameters from a given distribution (Monte Carlo simulation).

As specified in the section *System architecture – Output* the output is a grid structure sampled in time. The output JSON file contains a list of 2-tuples where the first element specifies the time window and the second element is a JSON object containing the probabilistic grid.

In the case of the KML output, the grid is serialized into a set of `placemarks`. Each `placemark` is a visualization for one cell of the grid. We utilize the `timespan` parameter of the KML format to define visualization of the time windows. For a single scenario case we utilize the `gx:track` KML entity. More details about the KML format can be found on the website https://developers.google.com/kml/documentation/kmlreference. The example of the output KML is shown in Figure 7, the color represents the probability of the appearance of a vessel for a given time and space (the color scale goes from green to red, with transparent color for zero probability).

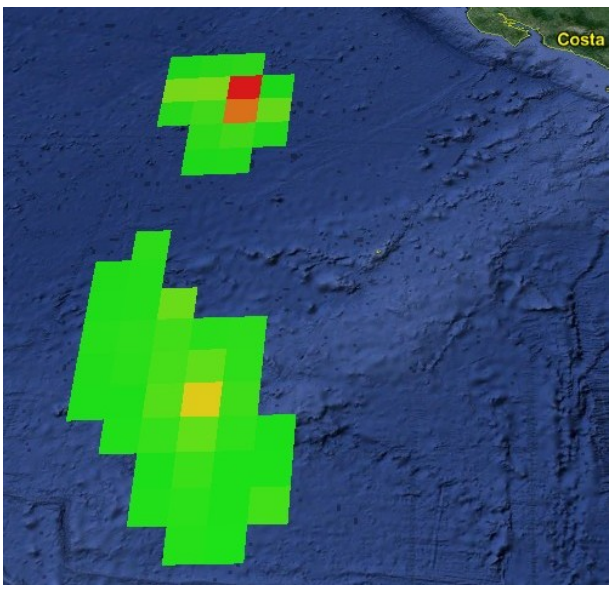

Figure 7. Visualization of the batch scenario output. The color of the grid represents probability of the appearance of a vessel

Transshipment scenario

The transshipment scenario is inspired by documented cases of smuggling drugs from Decker and Chapman (2008, p. 79). According to the authors, the vessel heading for Cuba smuggled three thousand kilos in a false wall of a container with cement. When the vessel was off the coast of Florida, the load was thrown overboard, attached to a rope. It was then picked up by a speedboat and brought into Florida.

We have replicated this scenario in the BANDIT platform. We had to make a few assumptions since the book does not provide all the necessary details. The first assumption is that the cargo vessel sailed from Mexico. This assumption was based on the fact that the route from Mexico to Cuba leads relatively near Florida. Also Mexico is known as an important source of narcotics import into the US.

Another assumption was the origin and destination locations of both vessels. For a cargo ship we chose a random point off of the east coast of Mexico and Havana, Cuba as the target. For the speedboat the origin and destination were the same location – the mouth of the Tampa Bay. All these locations were chosen randomly. The trajectories from the scenario are depicted in Figure 8.

Figure 8. Visualization of the transshipment scenario. The transshipment location is market by the label "*i*"

This scenario demonstrates the technical ability of the BANDIT platform. Some of the states used for the speedboat agent are the same as for the go-fast agents. The act of transshipment demonstrates the ability to model agent-to-agent interaction.

Conclusions

In this paper we have described the BANDIT simulation platform. The platform is designed to be easily adapted for various maritime domains with impact on inter-agent interaction. As part of the BANDIT we have developed the Behavior State Modeling system that allows rapid prototyping of the agent behavior in various domains. One domain upon which the platform was tested is the drug interdiction domain (motivated by the needs of Navy Research Lab in Monterey, USA), where we have implemented several types of scenarios.

The platform itself can be used in a variety of tasks, ranging from getting insight into a problem, through to testing what-if scenarios, to evaluating strategies for the blue forces. In the last case, the Monte Carlo approach can also be used to find configuration of the variables for parametric law enforcement strategy.

Acknowledgments

The research presented in this paper was supported by the Office of Naval Research grant No. N62909-14-1-N231 and by the Czech Science Foundation (GAČR) under research project No. 13-18316P.

References

1. DRUGWARFACTS. (2015a) *Crime, arrests, and us law enforcement – drug war facts*. [Online] Available from: http://www.drugwarfacts.org/cms/Crime. [Accessed: 12th March 2015]
2. DRUGWARFACTS. (2015b) *Interdiction of drugs – drug war facts*. [Online] Available from: http://www.drugwarfacts.org/cms/Drug_Interdiction. [Accessed: 12th March 2015]
3. BBC. (2015) *Drug delivery drone crashes in Mexico*. [Online] January 2015. Available from: http://www.bbc.com/news/technology-30932395. [Accessed: 12th March 2015]
4. DECKER, S.H. & CHAPMAN, M.T. (2008) *Drug smugglers on drug smuggling*. Philadelphie: Temple University Press.
5. ELKUS, A. (2012) *The Rise of the Narco Navy*. *USNI News*. [Online] Junuary 2012. Available from: http://news.usni.org/2012/06/15/rise-narco-navy. [Accessed: 12th March 2015]
6. TUNALEY, J. (2010) *Smuggler and pirate go-fast boats*. [Online] January 2010. Available from: http://london-research-and-development.com/GOFAST.pdf. [Accessed: 12th March 2015]
7. FIEGEL, B. (2014) *The Picuda: A Wave Breaking Go-Fast Wonder That Defies Radar Detection*. [Online] May 2014. Available from: http://fmso.leavenworth.army.mil/OEWatch/201405/ Latin-America 04.html. [Accessed: 12th March 2015]
8. ALVAREZ, L. (2014) *The Picuda: A Wave Breaking Go-Fast Wonder That Defies Radar Detection*. [Online] May 2014. Available from: http://www.nytimes.com/2014/05/30/us/in-puerto-rico-cocaine-gains-access-to-us.html. [Accessed: 12th March 2015]
9. JAKOB, M., VANĚK, O., BENDA, P., URBAN, S. & PĚCHOUČEK, M. (2009) *Adversarial modeling and reasoning in the maritime domain year 1 report*. ATG, CTU, Prague, Tech. Rep..
10. JAKOB, M., VANĚK, O., BOŠANSKÝ, B., HRSTKA, O. & PĚCHOUČEK, M. (2010 *Adversarial modeling and reasoning in the maritime domain year 2 report*. ATG, CTU, Prague, Tech. Rep.
11. HANSEN, J., HSU, L., DYKES, J., DASTUGUE, J., ALLARD, R., BARRON, C., ABRAMSON, M., RUSSELL, S. & MITTU, R. (2011) Information domination: Dynamically coupling METOC and INTEL for improved guidance for piracy interdiction. in *NRL Review*. Naval Research Laboratory. pp. 109–115.
12. AN, W., AYALA, D.F.M., SIDOTI, D., MISHRA, M., HAN, X., PATTIPATI, K.R., REGNIER, E., KLEINMAN, D.L. & HANSEN, J.A. (2012) Dynamic asset allocation approaches for counter-piracy operations. in: *Information Fusion (FUSION). 15th International Conference*. IEEE. pp. 1284–1291.
13. PATTIPATI, P.K.R., KLEINMAN, P.D.L., BALASINGAM, D.B., HAN, X., MISHRA, M., AYALA, D.F.M. & SIDOTI, D. (2013) Optimization-based model- ing framework for dynamic asset allocation algorithms in heterogeneous mission environments. in: *METOC Asset Allocation and Risk Assessment*, 2013.
14. MISHRA, M., HAN, X., AYALA, D.F., SIDOTI, D., PATTIPATI, K., AN, W. & KLIENMAN, D.L. (2014) Multi-objective asset routing problem within a dynamic environment. in: *Advance Computing Conference (IACC)*. IEEE. pp. 79–84.
15. SIDOTI, D., AYALA, D.F., SANKAVARAM, S., HAN, X., MISHRA, M., AN, W., KELLMEYER, D., HANSEN, J. & PATTIPATI, K.R. (2014) Decision support infor- mation integration platform for context-driven interdiction operations in counter-smuggling missions. in: *System Integration (SII). IEEE/SICE International Symposium*. IEEE. pp. 659–664.
16. HANSEN, J. (2014) Us navy metoc-informed decision guidance:products_ and user considerations. in: *DESRAAP Workshop*.
17. KLÜGL, F. (2009) *Agent-based simulation engineering*. Ph.D. dissertation, Habilitation Thesis, University of Würzburg.
18. MAHADEVAN, S. (1997) Monte Carlo Simulation. In: Cruse T.A. (ed.) *Reliability-Based Mechanical Design*. New York: Marcel Dekker. pp. 123–146.
19. SISLAK, D., VOLF, P., JAKOB, M. & PECHOUCEK, M. (2009) Distributed Platform for Large-Scale Agent-Based Simulations. pp. 16–32.
20. GIRAULT, A., LEE, B. & LEE, E.A. (1999) Hierarchical finite state machines with multiple concurrency models. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions*. 18, 6. pp. 742–760.