Master of Science in Informatics at Grenoble
Master Mathématiques Informatique - spécialité Informatique
option Artificial Intelligence and the Web

BDI multi-agent modelling of the population's decision making in a bushfire

Geoffrey Danet

June 22, 2016

Research project performed at Laboratoire d'Informatique de Grenoble (LIG)

Under the supervision of:
Dr Carole Adam, Laboratoire d'Informatique de Grenoble (LIG)

Defended before a jury composed of:
Prof N. Brauner
Prof M. Reza Amani
Prof C. Berrut
Prof J. Euzenat
Prof E. Gaussier
Dr B. Lemaire

June 2016

Abstract

Each summer in Australia, bushfires burn many hectares of forest, causing deaths, injuries, and destruction of property. Unfortunately, the emergency services strategy is made according to their own point of view about the citizens' behaviour, which does not always match reality. In order to raise their awareness about the real population behaviour, we want to provide them with a serious game. The underlying agent-based simulation must be as realistic as possible. The philosophically-grounded BDI architecture provides a high level of abstraction and is therefore the most suitable approach. We proposes a BDI multi-agent simulation of citizen behaviour during a bushfire, based on the witness statements, and validated from statistics.

Acknowledgement

Carole Adam, Julie Dugdale, Benoit Gaudou (GAMA), Patrick Taillandier (GAMA), Philippe Caillou (GAMA), Mathieu Bourgais (GAMA), John Thangarajah (TDF), Julius Millano Bañgate, Antoine Faravelon

Contents

— 1 —
Introduction

On Saturday 7th February 2009, the region of Victoria (Australia) was the victim of violent bushfire causing 173 deaths and many injured. The Victoria area endured the most severe and prolonged heatwaves during the final week of January 2009. The temperature was above 43°C during three consecutive days. The saturday was one of the worse day with in addition to the high temperature (from 40°C the morning to 46.4°C at the hottest period of the day at Melbourne) a strong wind (around 100 km/h). Moreover, the Country Fire Authority (CFA) and the Department of Sustainability and Environment (DSE) warned the citizens that forests and grasslands were the driest they had been since 1983. During this day, the extreme conditions triggered 15 fires. The consequence of the fires was the death of 173 peoples, 414 injured, 450,000 hectares (4,500 km^2) burned. Reports showed that there was a discrepancy between how the emergency services expected the population to behave, and how they really behaved [6]. The problem is that emergency strategies were based on this expected behaviour rather than on the real one.

SWIFT project

The focus of this project is to provide a serious game for the emergency managers in order to test various strategies and finally help them to take the best decision. For that, the model of the population must be as accurate as possible. The SWIFT project is interested in modelling and simulating the population behaviour in the state of Victoria in Australia during a bushfire. The approach is agent-based modelling and simulation, with a BDI (Belief, Desire, Intention) architecture of the agents, to endow them with a realistic behaviour based on psychological theories, as well as on field studies and witness statements made by the Victorian Bushfires Royal Commission (VBRC).

The contributions during this internship were the implementation of a BDI architecture initially based on a finite state machine architecture. We propose as well a methodology in order to convert interview data (textual data) in BDI with a low requirement in computer science.

At first, we will describe the existing studies in simulation for crisis situations. Secondly, we will describe the tools and the methodology used during this project. After what, we will describe the data used and the current implementation. To finish, we show some results and make a conclusion.

— 2 —
State Of the Art

Many crisis situations occur in this world, which can cause both psychological and physical injuries besides considerable damages which have an important cost on the economy. In order to reduce the impact of such situations, many countries are actively searching ways to prevent crisis and decrease the potential number of victims. The observation was the first source of knowledge, but implies victims as well. Quickly, the idea of simulation comes to mind. A full-scale simulation is the closest solution to get realistic results but is very costly, impossible to reproduce exactly, and of course can be very dangerous. On the other hand, we have the computer simulation which provides a cheap, infinitely reusable, and safe solution but with a less realistic result. Moreover, this kind of simulation offers a very high control of the environment and allows to reproduce very precise scenarios easily. Computer simulations such as serious game allow subsequently to increase the understanding of more general causal relationship in emergency management organizations [28].

2.1 Modelling and Simulation

There exist many methods in order to produce simulations. Among them, we can find two main categories of simulation models:

- **Equation-based Model (EBM)**: Model of simulation principally based on differential equation over time. A full model can be composed of a succession of equations. Some existing equation used for equation-based simulation are famous for this, like for instance the Runge-Kutta method defined by C. Runge and M. W. Kutta in 1990, the Euler method based on Leonhard Euler's works (*Institutionum calculi integralis*, 1768) or the Newmark method [20]. This kind of simulation is particulary used in physics for particles simulation [12] or chemistry for molecular dynamics systems[27] for example.

- **Agent-Based Models (ABMs)**: Model of simulation which provide actions and interactions between several autonomous entities called agents. Usually, agents are used in a spatiotemporal environment with an automaton architecture and can have the possibility to interact with their environment.

According to a study made by Parunak et al. [26] the distinction between these two methodologies are essentially in the project requirement. ABM is more appropriate for domains with "*a high degree localization and distribution and dominated by discrete decisions*". EBM is more appropriate for systems where, for exemple, "*the dynamics are dominated by physical laws rather than information processing.*". Moreover, EBM requires a high knowledge of behaviours and is made for one specific purpose, hence, cannot be easily reused. Thus, a complex system such as cognitive entities cannot or is extremely difficult to model using equation in contrary to the ABM modelling systems.

2.1.1 Multi-Agents Systems

An agent is a computer system which can interact with its environment in an autonomous way through actions. The particularity of multi-agent is the possibility to transfer information between agents. Thanks to the data sharing, the agents can, for exemple, collaborate for a task, or ask another agent for information. Thanks to that system, an agent which does not have a specific skill to achieve a task, can ask another agent with the required skill to do it. For exemple, in the context of fire crisis, a citizen can be stuck in a building because of fire (reactive to the environment). In order to stay alive, the agent can need to **inform** firemen about the fire and to **ask** for help. After what, the firemen will extinguish the fire (interaction with the environment). The multi-agent simulation provide to the model an easier interaction between their entities.

The simulation framework of individual cognitive processes attempts to produce emergent social or collaborative behaviour as show the study of Pan et al. [22] through a simulation of building evacuations. Others studies tried to reproduce the interpretation of risk in the model with a descriptive psychology approach during building evacuation.[16].

Unfortunately, these studies are really specific to the building evacuation domain with consequently a portability problem for other researchers who want to use it besides of a higher complexity for the designer. Moreover, the low level of abstraction decrease the behaviour comprehension during the simulation. In addition the behaviour trends are the same for all agents. In contrary to these approaches, the BDI (Belief, Desire, Intention) architecture provides a more flexible solution to make an agent simulation.

2.1.2 Belief Desire Intention (BDI)

For agents using automatons architecture, the behaviour is defined by the states and its transitions. The agent cannot easily change its intention during the simulation if the transitions between these states are not defined. Each transition are determined and defined according to each situation and finally result to a similar agent behaviours. To resume, the flexibility of the decision process is dependent upon the automaton implementation. In contrast with the automatons, the BDI approaches allows agents to have more realistic behaviours through their own decisions. BDI is an agent modelling approaches based on the philosophical theories of Bratman [24]. In this architecture, each agent has **mental states** defined by Belief, Desire and Intention which will affect its behaviour during the simulation.

- **Beliefs** represent the agent's informations about the environment, itself or the other agents. A belief is not always true, it can be **incomplete** or **incorrect**. According to these beliefs, the agent can estimate the feasibility of desires and consequently adapt his intentions.
  e.g. "I believe that fire will come here soon.", "I believe the firemen will deal with it.".

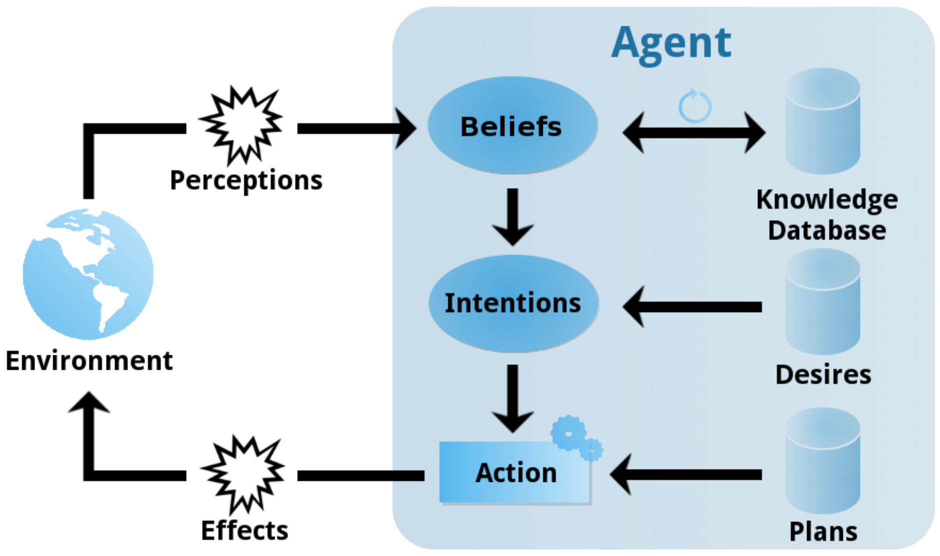


Figure 2.1: Basic representation of the BDI architecture.

- **Desires** represent the agent's motivations to do something. Desires have a direct impact on his intentions. These desires can be **inconsistent** or **unrealistic** and can have differents priorities.
  e.g. "I do not want see my house burning." or "I do not want to be hurt."

- **Intentions** are selected from the desires and beliefs of the agent. An intention can have several associated plans allowing to achieve it.
  e.g. The agent believes that fire will come to its place soon, and has the desire to propect its house and to stay alive. Consequently, the agent according to its priority will choose between protecting its property if its life is not immediately in danger or leaving the house in order to stay alive.

Obviously, this method fit very well for human behaviour simulation and is more intuitive to use than other approaches. Thus, the development of the simulation become easier and more understandable by scientists.

An example from an extract of witness statement about the Black Saturday :

> "I looked out the window and saw some hazy smoke to the north-west. Gary said that he thought it was just dust but we went outside and straight away we noticed that we could smell smoke. It was about 12.45pm when we smelt the smoke and as soon as that happened, Gary agreed to go and get the fire pump."[9]

This sentence can be easily translated using the BDI architecture, Gary **believed** for a while that the hazy smoke was simply dust but after going outside, they smelt smoke (**perception**) and realised that the hazy smoke was indeed coming from a fire (**belief update**). Finally, they decide to defend their property (**intention**), with for first **plan** whose first **action** is to prepare the fire pump.

However, BDI does not suit for building systems which must learn and adapt their behaviour and does not have explicitly multi-agent aspects of behaviour [11].

There already exist some BDI framework [7]:

- Classic framework: Procedural Reasoning System (PRS) which is an architecture for real-time reasoning and system control[15].
- Commercial framework: JACK which inherits many properties from PRS and allows to define Multi-Agent Systems (MAS)
- The most advanced framework : Jadex, an add-on of the JADE framework which has an explicit representation of goals.

2.2 Egress simulation

Egress simulation is a thread which is very studied in crisis simulation and in human behaviors during fire. The most studies in egress simulation are focalized on the study of crowds behaviors, dynamics and more precisely on specifics part such as risk perception [16, 14], communication [23], psychology [23, 17] and sociology [21, 22]. Pan et al. [22] categorize the existing model used for crowd in three main categories:

- Fluid of particle systems: used for evacuation simulation, crowds behaviours are similar to fluids.[18, 13]
- Matrix based systems: The area is represented into cells (grid) where cells represent obstacles, free area, area occuped by a person, ...
- Emergent systems: The interactions of simple parts/elements can create an emergeance phenomena and produce complex system (here crowd dynamics)

Pan et al show that an individual's behaviours follow three basic conventions[21]: People how are following instinct, following experience and people who have bounded rationality. Another studies about crowd simulation, human behaviours and social behaviours shows several aspect of non-adaptive crowd behaviours. Human and social behaviours are categorized in three distinct category: 1) individual (collection of individuals), 2) interactions among individuals and 3) groups. [21] Kuligowski introduce a conceptual model of the behavioural process for building fire composed in four phases: 1) Perception of the cue(s), 2) Interpretation of the cue, situation, and risk, 3) decision-making and 4) Actions. Unfortunately the level of abstraction is too important to aptly represent the real human behaviours. All signals are interpreted in the same way by all agents and does not take in account interpretation such as false warn[17].

— 3 —
Tools and Methodology

In this section we introduce the GAMA framework and the Tactical Development Framework used for the modelling and simulation described in this report.

3.1 Tactics Development Framework

Tactics Development Framework (TDF) is a tactical decision-making modelling tool based on the Prometheus methodology [8], a mature and popular agent-oriented software engineering methodology. Made for modern tactical warfare, TDF provides a new way to support the development life cycle of the simulation. The objective is to simplify the requirement and knowledge elicitation, then to represent this data through diagrammatic specification. Although it was initially made for military tactical purpose, Adam et al. [3] show that TDF suits perfectly for modelling the civilian behaviours.

3.1.1 Diagram description

This section will introduce the different diagrams provided by TDF. Each diagram is illustrated by an example related to this study. The examples and diagrams come from an article which illustrates the methodology used in order to capture civilian's behaviour in the fire with TDF and its mapping with GAML code [4]. The model described in these examples is more elaborate than the model used during this internship.

Analysis overview

The goal of this diagram is to identify the different actors (entities external to the system), the inputs (percepts), and outputs (actions) and to identify senario (similar to use-cases). In our case, the scenarios are exemples of behaviours. A given scenario contains a sequence of steps which include goals (oval shape), actions (right arrow), percepts (bang pictogram) and sub-scenarios. The figure 3.1 shows a part of the analysis overview diagram related to a scenario where a Pedestrian defends its property (steps detailed in the figure 3.2).

Goal overview

The goal overview diagram illustrates how the high-level goals are decomposed into more concrete goals. The figure 5.3 represents the goals that Pedestrian can use. These Pedestrians

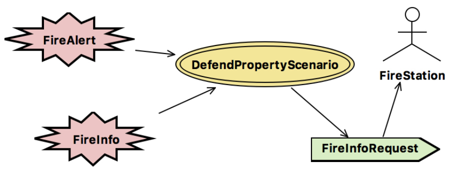
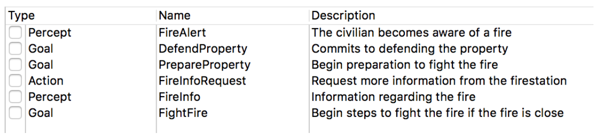


Figure 3.1: Partial Analysis Overview diagram.



Type	Name	Description
Percept	FireAlert	The civilian becomes aware of a fire
Goal	DefendProperty	Commits to defending the property
Goal	PrepareProperty	Begin preparation to fight the fire
Action	FireInfoRequest	Request more information from the firestation
Percept	FireInfo	Information regarding the fire
Goal	FightFire	Begin steps to fight the fire if the fire is close

Figure 3.2: Pedestrian defending property (detailed steps).

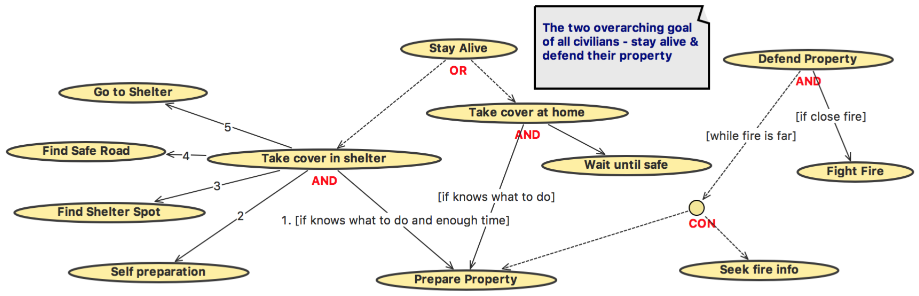


Figure 3.3: Goal overview for Pedestrian (civilians): decomposition of 2 main high-level goals, “Stay Alive”and “Defend Property”.

behaviours are described in the interviews (see Chapter 4) and result from two high-level goals: **Defend property** and **Stay Alive**. The relative priorities of these goals depend on individual differences in various factors: awareness of fire, fire training, motivation to defend property, etc. The sub-goals of a goal can be *OR*, *AND* or *CON* decompositions.

- *OR* which correspond to the disjunction: a Pedestrian can try to stay alive by taking cover at home *or* by going to a shelter (cannot be both).

- *AND* which correspond to the conjunction (can be ordered): to get to a shelter, the Pedestrian agent requires to achieve some tasks before: First prepare their house if they are able and if they have enough time to do it (condition), after what they must prepare themself, find a shelter spot with a safe road. If the Pedestrian clears these tasks, they can finally go to the shelter.

- *CON* which mean that sub-goals can be used in parallel. In our example, to fight a fire during the period when the fire is far, the Pedestrian can simultaneously prepare their property and seek information about fire status.

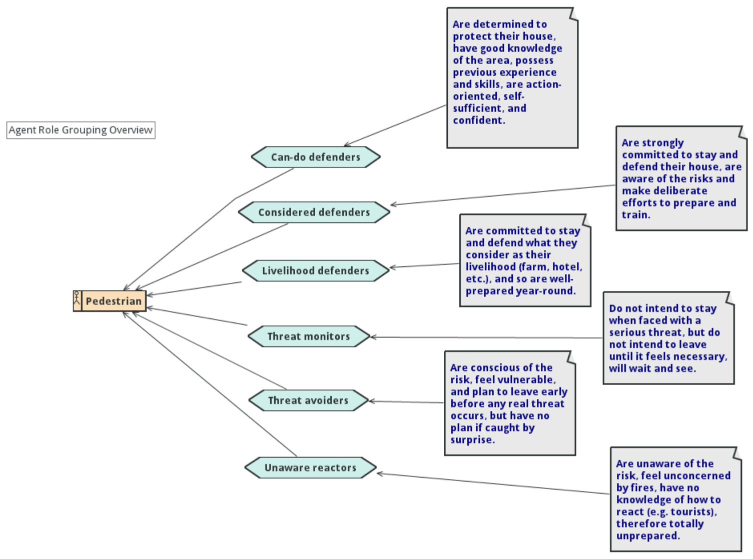

Agent Role Grouping Overview
Are determined to protect their house, have good knowledge of the area, possess previous experience and skills, are action-oriented, self-sufficient, and confident.
Can-do defenders
Are strongly committed to stay and defend their house, are aware of the risks and make deliberate efforts to prepare and train.
Considered defenders
Are committed to stay and defend what they consider as their livelihood (farm, hotel, etc.), and so are well-prepared year-round.
Livelihood defenders
Pedestrian
Do not intend to stay when faced with a serious threat, but do not intend to leave until it feels necessary, will wait and see.
Threat monitors
Are conscious of the risk, feel vulnerable, and plan to leave early before any real threat occurs, but have no plan if caught by surprise.
Threat avoiders
Unaware reactors
Are unaware of the risk, feel unconcerned by fires, have no knowledge of how to react (e.g. tourists), therefore totally unprepared.

Figure 3.4: Agent role overview for Pedestrian.

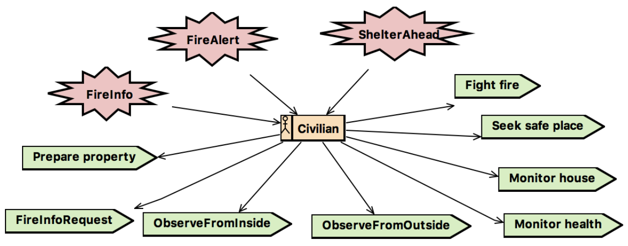


Figure 3.5: System overview.

Role overview

The role overview diagram is used in order to associate the different roles for each agent. The figure 3.4 shows the 6 profiles of behaviours identified by Rhodes[6] from the witness statements. An agent can use only one role at the same time but can change over time.

System overview

The system overview (figure 3.5) allows to represent the relations between each percept (input) and related actions (output) of an agent (can have several agent). This diagram captures the interactions between the agents via protocols. In our case we have only one agent.

Agent overview

This diagram allows to make an overview of an agent with the associated capabilities and plans used to achieve their goals. The percept (input) and plan can be linked to one or several capacities. The agent can use only one of them at the same time. The figure 3.6 shows the overview of the Pedestrian agent with the related percepts and plans relevant to each capability.

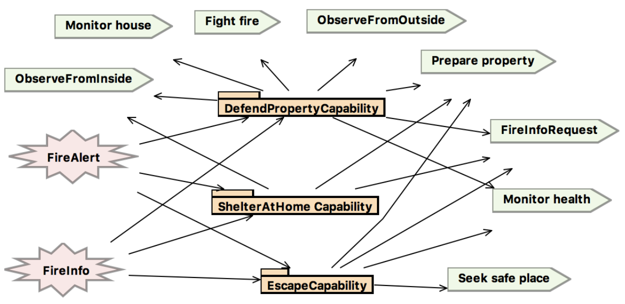


Figure 3.6: Agent Overview, contains the capabilities of a Pedestrian (civilian) agent.

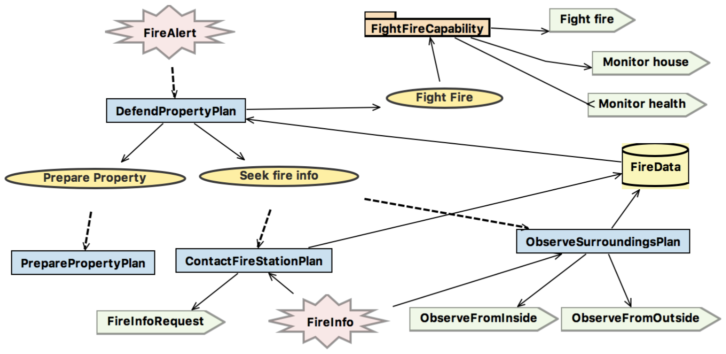


Figure 3.7: Description of the “Defend Property ” capability.

Capability overview

The capability overview represents the details of plans and sub-goals related to a specific capability. The figure 3.7 shows the details of the *Defend property* capability. It shows as well the flow of activities and specifies the required data elements. For example, the “*Fire Data*” is read by the “*Defend Property*” plan and is written by the “*Observe Surroundings*” and “*Contact Fire Station*” plan.

Plan diagrams

The plan diagrams are used in order to describe the plans (kind of UML activity diagram). A plan can be triggered by external percepts or goals, and is composed of several steps which can be sequential or concurrent. A Step can be atomic actions (external), activities (internal processes like “write data”) or sub-goals. The figure 3.8 shows the description of the “*Defend Property*” plan. This plan is triggered by the “*Fire Alert*” percept, and then uses “*Prepare Property*” and “*Seek Information*” goals concurrently. According to the consulted data (*Fire Data*), the agent waits for fire to be close enough in order to use the “*Fight Fire*” goal.

3.2 GIS & Agent-based Modelling Architecture

GIS & Agent-based Modelling Architecture (GAMA) is an open source platform for modelling and simulations. It allows to build multi-agent systems into a spatial environment. Widely used,

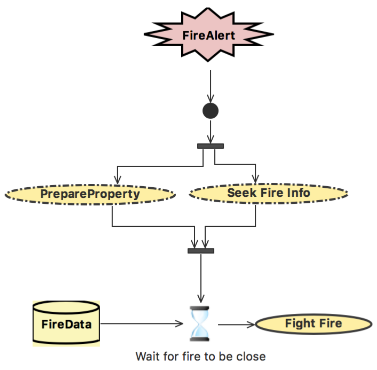


Figure 3.8: Description of the "Defend Property "plan.

his developers community is active and is regularly improving the software. It allows scientists to use GIS data in simulation such as OpenStreetMap (OSM) with provide a fast way to build a precise mapping environment to the simulation. Furthermore, Gama has the advantage to make scalable simulations and to use a very simple and high level programming language called GAML[10, 7, 19].

3.2.1 GAML

GAMA **M**odeling **L**anguage also known as GAML is an agent-oriented language made from the necessity to have an easy language for non-computer scientists. Based on object-oriented languages like JAVA or Smalltalk, GAML is very close to agent-based modelling languages such as for example NetLogo. however, GAML has some additions like the inheritance, type safety or multi-level agency and provides a set of different behavioural architectures for programming agents. Althrough GAML is a programming language, the syntax was made in order to have a short learning curve and allow scientists to use GAMA in a short time.

The first step is to define a *model species* (global model) which will include later the *model agent* (the simulation), *micro-species* and the *experiment plans*.

Basic model structure

The **models** are the specifications of the simulations which can be controlled during an *experiment* called also *experiment plans*.

Model Header

Listing 3.1: Header model syntax.

```
model name_of_the_model

// inclusion of files
import "relative_path_to_a_model_file" [as model_identifier]

global {
    // Definition of [global attributes] (Global Species #declaration)
    //                [actions and behaviors] (Defining Actions And Behaviors)
}
```

A model must begin with the declaration of the name of the model by using the keyword **model** followed by the name. This name is used for building the name of the model species from which the simulations will be instantiated. The user can as well import existing file(s) using **import** and rename it with an alias with **as** (optional). The imported files will be merged in the model as *micro-models*. The global species declaration is made through the **global** block declaration which contains the description of the *model species*. Notice that the import and global declaration is not mandatory.

Species

Listing 3.2: Species declaration.

```
global {

   // definition of global attributes, actions, behaviors. Accessible by all species

   [...]

   species species_name [skills, control, schedules, ...] {
      ...
   }
}
```

Species provide a set of *attributes* (what they know), *actions* (what they can do), *behaviors* (what they actually do) and also specifies properties of their population. The declaration of species is done with the keyword **species**. It is possible as well to define some optional descriptions to the species like **skills** (driving, moving, ...), **controls** (define the agent architecture type) or define some rules for the scheduler with **schedules**. A species can inherit its proprieties from another species called *parent-species* or be included in another one (*micro-species*). Species are used in order to define the agents.

Experiment

An experiment is defined by the keywork **experiment** followed by the name of the experiment and the type (gui or batch). An gui experimentation allows to run one simulation per launch with the possibility to add visual output to interpret the results. In contrary of the gui type, the batch type is used in order to run the simulation severals times usually with graphics or data output.

Listing 3.3: Experiment declaration.

```
experiment experiment_name type: gui (or batch) {

   // Experiments parameters which can be manipulated by the user
   parameter "Parameter's description" var: variable_name [min, max, init, category];
```

```

  output {
    display display_name {
    }
    [monitor statements]
    [file statements]
  }
}
```

BDI

This part will present the BDI implementation syntax through a simple example (see listing 3.10). The aim of this simulation will be to make a dummy BDI agent who will first take a look around its place and try to find fires in a radius area that we will call *perception_area*. After what, the agent will fight the fire if it enters the *defense_radius* area which correponds to the area where the agent thinks necessary to fight the fire. The **fire** will not be described in this part but has for goal to grow and burn all elements around. The first step is to define a new **species** which is called *civilian* in this example. After the declaration of the species, we add the skill *moving* using the keyword **skills** and of course we specify that we want a BDI agent by defining the **control** with *simple_bdi*.

Listing 3.4: BDI agent declaration.

```
species civilian skills:[moving] control: simple_bdi {...}
```

Now, we have a new BDI agent which has the possibility to move. We can now define the required attributes such as the *perception_area* and the *defense_area* which will be used later. The declaration of variables is similar to a standard language such as in JAVA, C++, C ... In our case, we need a **float** or **integer** value in order to represent the radius areas. Moreover, we add an attribute to store known fires represented by a **list** of *fire*. (see listing 3.5)

Listing 3.5: Attributes declaration.

```
float perception_radius <- 10.0;
float defense_radius <- 3.0;
List<fire> known_fires;
```

The next step is to define the **predicates** (see listing 3.6). The civilian agent has two predicates, the first one is the indication that the *fire started*, and the last one will be use in order to define if the agent want to *protect its property*.

Listing 3.6: Predicates declaration.

```
predicate fire <- new_predicate("Fire started"); // the agent believe that the fire started
predicate protect_property <- new_predicate("Protect property"); // the agent desire to fight against the fire
```

We have created a new agent with some attributes and predicates. Now, we will see how to use these parameters in order to get a smarter agent. Indeed, the agent requires to have some instructions in order to know what to do according to the situation. To make this instructions, we will make a basic scenario and add little by little the required instructions. First we need to initialise the agent with an **init** block. In this block, we will define the initial agent's status and add the desire to protect its property. The keyword **do** will give an order to the agent which is to add the desire (**add_desire**) to *protect its property*.

Listing 3.7: Initialisation of the agent.

```
init {
   do add_desire(protect_property);
}
```

Let's suppose that the fire is started, we want the agent to take a look around its current position in order to detect potential fires. To do that, we need to add a **perception** with the **target** set on the *fire*. Moreover, in order to make the simulation more realist, we can define a radius where the agent will not be able to see fires outside it. Thus, we use keyword **in** with the *perception_area* attribute previously declared (listing 3.5). The result of the perception will trigger the belief that the fire is indeed started.

Listing 3.8: Perceptions declaration.

```
perceive target:fire in:perception_radius {
   add self to: myself.known_fires;  // self represent the current target and myself the agent
   ask myself {
      do add_belief(fire);
   }
}
```

Let's assume now that the fire is close to the agent position, it believes that the fire is started and has the desire to defend its property. To satisfy his desire, the agent can use a plan. In order to create a new plan we use the keyword **plan** followed by the name of the plan (here *fight_fire*). After the plan declaration, we need to link it with the associated **intention** *protect_property*. This declaration means that this plan is used in order to satisfy the specified intention which is here *defend its property*. Furthermore, we need to specify that the plan cannot be used without fire at proximity of the agent position. This condition can be expressed with the keyword **when** followed by the condition.
The conditions are the following:

- A known fire must be present in the defense area: To write this condition we need first to get the list of fires in the area using the keyword **at_distance** which will provide a list of fires in the defense radius (right part of the keyword) among the known fires (left part). After getting the list of fires, we check if the list is empty (no known fire at proximity) or not using the negation **not** and the **empty** function.

- The last condition is to check if the agent believes that the fire is started. If the agent does not believe that the fire is currently started, it will think then it is useless to fight the fire without fire at proximity (which makes sense). This verification is made by the function **has_belief** which will be true if the agent at this moment believes the predicate between the parenthesis.

Listing 3.9: Plans declaration.

```
plan fight_fire
   intention: protect_property
   when: (
                not empty(known_fires at_distance defense_radius)
                and has_belief(fire)
            )
   finished_when: empty(known_fires at_distance defense_radius)
{
    ask known_fires at_distance (defense_radius)
    {
        float FightingEffect <- rnd(4.0); // The FightingEffect is a random value between 0 and 4
        intensity <- intensity - fightingEffect;
    }
}
```

The last thing to define is the content of the plan. In order to simulate the fire fighting effect, we will generate a random value corresponding to the effect of the action (here a random value between 0 and 4) on the fire which will be stored in a variable called *fightingEffect*. After what, we just have to reduce the intensity of all known fires in the defense radius (*ask known_fires at_distance* (*defense_radius*)) by the value of the variable previously created. The final result of this agent is represented in the listing 3.10.

Listing 3.10: Exemple of BDI agent.

```
species civilian skills:[moving] control: simple_bdi {
    float perception_radius <- 10.0;
    float defense_radius <- 3.0;
    List<fire> known_fires;

    // Predicates
    predicate fire <- new_predicate("Fire started"); // the agent believe that the fire started
    predicate protect_property <- new_predicate("Protect property"); // the agent desire to fight against the fire

    // Agent initialisation
    init {
        do add_desire(protect_property);
    }

    // Perceptions
    perceive target:fire in:perception_radius {
        add self to: myself.known_fires; // self represent the current target and myself the agent
        ask myself {
            do add_belief(fire);
        }
    }

    // Plans
    plan fight_fire
        intention: protect_property
        when: (
                not empty(known_fires at_distance defense_radius)
                and has_belief(fire)
              )
        finished_when: empty(known_fires at_distance defense_radius)
    {
        ask known_fires at_distance (defense_radius)
        {
            float FightingEffect <- rnd(4.0); // The fightingEffect is a random value between 0 and 4
            intensity <- intensity - fightingEffect;
        }
    }
}
```

Finite State Machine (FSM)

This part describe briefly the FSM architecture of GAML. The following example is based on the ant toy model available on GAMA. In this model, the ants have for objective to find food and bring it back. The declaration of the agent is similar to the BDI declaration but with the keyword **fsm** instead of **simple_bdi**. An ant has for *attribute* a boolean value which determines if the ant currently has food or not.

Listing 3.11: Ant attribute.

```
bool hasFood <- false;
```

To allow ants to have the possibility to pick or drop food, we can for instance use the **action** which is equivalent to a process or function. The pick function will switch the boolean value to *true* in contrary of drop which will switch it back to false.

Listing 3.12: Pick action.

```
//Action to pick the food
action pick {
  hasFood <- true;
}

//Action to drop the food
action drop {
  hasFood <- false;
}
```

The next step is to define the states. The declaration of the state is made by the keyword **state**. For example, we want define the initial step of this simulation, which consist in searching randomly in the area some food or a road made previously by another ant which has discovered food. We create a new state with the name *wandering* and add after the name the keyword **initial** set to *true*. This option defines the state as the initial state. We need now to tell that the ant must search in the area. For that, we use the order keyword **do** following by the order **wander** and define an **amplitude** set to 120 units from the current ant location for instance. Let's now assume that the ant discovers food. We need to change the state of the ant from *wandering* to the next state *carryingFood*. To make the transition, we first use **transition** followed by the name of the new state (here *carryingFood*) and with the condition of the transition (place.food >0) which represents whether the place contains food (0 means that there is no food).

Listing 3.13: Initial state of the FSM.

```
//initial state of the ant that will make it wander until it finds food or a road
state wandering initial: true {
  do wander amplitude:120; // Search randomly in a radius of 120 units around the current position
  transition to: carryingFood when: place.food > 0 {
    do pick;
  }
  transition to: followingRoad when: place.road > 0.05;
}
```

Before bringing back the food, the ant must make a road for the other ants in order to show them the location of the food. In order to do that, we can use the keyword **reflex**. The reflex is a process which will be executed each cycle of the simulation if the condition (optional) is true. In our case, the condition is that the ant must carry food (defined by the boolean *hasFood*).

Listing 3.14: Finite State Machine exemple.

```
//Reflex to allow the diffusion of the road of pheromon by putting pheromon inside a cell
reflex diffuse_road when: hasFood = true {
   ant_grid(location).road <- ant_grid(location).road + 100.0;
}
```

The final step is to create the other states which are: following the road (*followingRoad*) and carrying food (*carryingFood*). Finally, we obtain the following result :

Listing 3.15: Finite State Machine exemple.

```
//Species ant that will move and follow a final state machine
species ant skills: [moving] control: fsm {
   bool hasFood <- false;

   //Reflex to allow the diffusion of the road of pheromon by putting pheromon inside a cell
   reflex diffuse_road when: hasFood = true {
      ant_grid(location).road <- ant_grid(location).road + 100.0;
   }

   //Action to pick the food
   action pick {
      hasFood <- true;
   }

   //Action to drop the food
   action drop {
      hasFood <- false;
   }

   //initial state of the ant that will make it wander until it finds food or a road
   state wandering initial: true {
      do wander amplitude:120; // Search randomly in a radius of 120 units around the current position
      transition to: carryingFood when: place.food > 0 {
         do pick;
      }
      transition to: followingRoad when: place.road > 0.05;
   }

   //State to carry food to the nest once it has been found
   state carryingFood {
      do goto target: center;
      transition to: wandering when: place.isNestLocation {
         do drop;
      }
   }

   //State to follow a pheromon road once it has been found
   state followingRoad {
      location <- (self choose_best_place()) as point;
      transition to: carryingFood when: place.food > 0 {
         do pick;
      }
      transition to: wandering when: (place.road < 0.05);
   }
}
```

3.3 From TDF to GAML

In this part we will explain who TDF diagrams can be used in order to produce GAML code. To illustrate this section, we will use the diagrams previously described in the section 3.1 and make the model of the senario *Defend Property*. We assume that we already created an Pedestrian agent and there exist an fire implementation. We assume as well that each Pedestrian has a house which have to be defended.

To begin, the figure 3.1 and 3.2 provide the global information about the senario we must create. The goal is composed of 2 percepts, 3 goals and 1 action. The figure 3.8 and 5.3 provide more information about the progress of the scenario and about the different sub-goals. We know now that to achive the goal *Defend Property* the agent need to *fight the fire* if its close or to *prepare its property* or to *seek information* about the fire. Moreover, we can deduce from the diagram represented in the figure 5.3 the different predicates.

Predicate

According to the goal overview (figure 5.3) and the plan description (figure 3.8), we can deduce that we need to create 2 predicates. The first one is the predicate which will be used in order to represent the intention to protect the property (called *protect_property*). The second one is the trigger of the scenario *fire_position* which represents the acknowledgement of a fire position (see listing 3.16).

Listing 3.16: Agent predicates.

```
predicate fire_position <- new_predicate("Know the fire position");
predicate protect_property <- new_predicate("Protect property");
```

Percepts

As we have seen previously, the figure 3.7 informs us that the first percept (*FireAlert*) is the trigger of the scenario and the second percept (*FireInfo*) is the result of the information research about a fire. We know as well that its related to the fire position (see figure 3.2). In order to create the perception *FireAlert*, we create at first a "*perceive*" block with for target all *fires* in the *perception area*. The content of the perception is filling the agent's database about fire called *known_fires*, add the belief about the fire position and add the desire to protect the property.

Listing 3.17: Perception FireAlert.

```
perceive target:fire in:perceptionArea {
  add self to: myself.known_fires; // self represent the current target and myself the agent
  ask myself {
    do add_belief(fire_position);
    do add_desire(protect_property);
  }
}
```

Plans

The description of each plan are available on the capability overview (figure 3.7) and the plan diagram (3.8). For example the listing 3.18 correponds to the plan *PrepareProperty*. This goal can be used if the agent are aware of the fire and if the fire is far enough to allows the agent to make some preparation of its property.

Listing 3.18: Plan PrepareProperty.

```
plan PrepareProperty
   intention: protect_property
   when: (
              empty(known_fires at_distance defense_radius)
              and has_belief(fire_position)
            )
{
   do HousePreparation;
}
```

— 4 —
Data description

The goal of our study is to make a BDI multi-agent simulation of citizen behaviour during a bushfire. This study is principally based on related works, on witness statements from the Black Saturday[1] and from the last studies about Australian bushfires[6, 3].

4.1 Witness statements

The witness statements are a good source of information about the peoples behaviour during this day. These witness statements show that there exist several behaviours and profiles. The first main difference is the awareness of the fire risk. The citizens who have made fire training at the CFA know about fire risks and have potentially more chance to know how to fight a fire. According to their priority, if they have a family to protect or if their house is very important (family house, livelihood ...) the person will have for first intention to leave or to protect their property or will simply wait to see if the fire is coming to their direction. Among people who want to stay, we can find two types of person. The first one want to protect their property but does not know how to fight fire and does not have equipped his house for that, these persons have a low danger aversion. The second type want to protect their property and is well prepared. The fact that they know how to fight a fire can produce two types of interpretation during a bushfire. A person who knows how to fire a fire can have a very high confidence in their skill and finally underestimate the intensity of the fire. However, some other people who wanted initially to fight the fire will change their mindset when they realised that the fire is too important in order to be extinguished.

Witness statements were used as well in order to make TDF digrams. An example with the following sentence:

> “While Gary was getting the fire pump ready, I checked the CFA website but I couldn’t find anything that related to our district.”[9]

This sentence show that before the fire was too close from the house, Gary made some preparation (*Prepare Property*) and Sue was searching information with the radio (*seek information*).

4.2 Profiles

A previous study shows there exist six archetypes of population behaviour. These archetypes describe some pattern such as the risk perception, attitudes, intentions, priority and behaviours[6, 3]:

- **Can do defenders**: They are self-sufficient, confident, practical and motivated to deal with the fire in order to protect his property. They have a good knowledge of the area, are experimented with fires and are aware of the risk. They rely on their own judgement and resources. They use official warnings but often rely on local sources and their own observations.

- **Considered defenders**: Although they know the related risk of the fire, they will make a conscious decision to protect their property. They spend time and ressources in order to deal with the fire. They are well connected in their community, are engaged with the fire agencies, participate in programs and seek information regularly. They are familiar with sources of warning but use their own observations to assess the situation.

- **Livelihood defender**: Defend their property which are their livelihood (farmer). Often long-term resident, experimented with the fire and will not give it up easily. They are motivated by the need to protect their livelihood. There are self-reliant, practical with a good local knowledge. They are well connected and are often volunteers in local brigades. They rely on the local knowledge and networks to get information about the threat and make their own judgement of the situation.

- **Threat monitor**: Does not intend to remain but they will not leave until they think it is necessary to leave. They remain in order to prepare their property and will not leave before the completion of their tasks, the proximity of fire, they perceive a danger, or advice from authorities. They are aware about the risk but try to balance the threat against their priorities. They do not have backup plans because they want to leave before it becomes dangerous.

- **Threat avoider**: Are aware of the risk and feel vulnerable. They leave before there is "any source of danger " or as soon as possible. They can have limited capability, responsibility for dependants, undefendable property or the fear of the threat. They support the emergency services but do not seek safety information because they think that leaving will solve the problem. They do various tasks before leaving such as moving important item from the property or make preparation in order to leave. They rely on the official warnings sources and leave if the authorities advise them to do so.

- **Unaware reactor**: Do not think they are leaving in a fire risk area because of the unfamiliarity with the environment and risk and do not feel concerned. Even if they aknowledge about the environment risk which can occur in the area, they will not expect to be affected. They do not seek information about what to do, do not make or prepare plan. They have no idea of what they must do if fires occur and leave if they feel in danger or wait to see they really need to leave. They follow official advice to leave if they receive it but the presence of emergencies services have an influence to lead them to believe the threat is being managed.

Table 4.1: Summary of profiles.

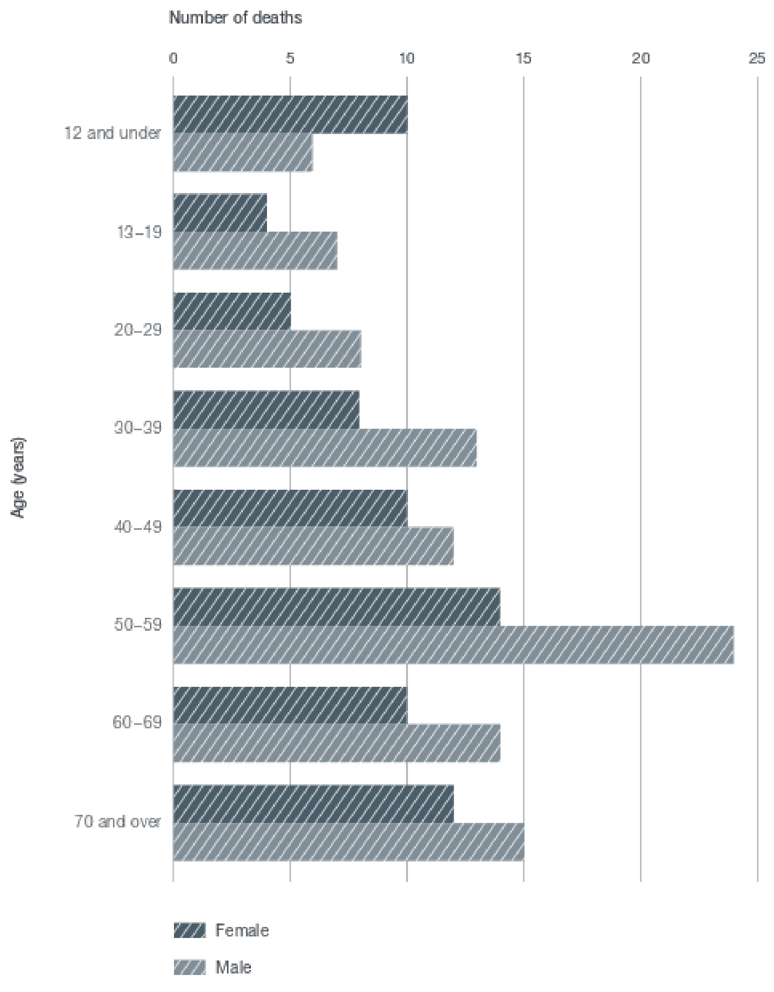
Profile	Experience/capacity	First intention	Determination	Danger aversion
Can do defenders	high	Defend	high	low
Considered defenders	moderate	Defend	moderate	moderate
Livelihood defender	high	Defend	very high	very low
Threat monitor	moderate	Leave if close fire	low	high
Threat avoider	low	Leave	low	very high
Unaware reactor	null	null	null	low

4.3 Statistics

The statistics used in this project have been made by the VBRC, these statistics contain essentially information about the damages caused by the fire during the Black Saturday (number of burnt buildings, number of dead, . . .). The most interesting statistics we can find are data about the causes of the death.

The Professor John Handmer of RMIT University report the following results about the behaviour of the peoples who died during the Black Saturday[2]:

- 58% had made no preparation either for staying and defending or for leaving early (*Threat avoider*). Many of these peoples prepare to leave but was apparently awaiting a warning (*Threat monitors*).
- 20% intended to stay and defend and was well prepared (*Can-do defenders*) and 14% made limited preparation (*Considered defenders*).
- The fire took by surprise 30% of those who died.
- 24% was unaware that they were in a bushfire risk area, 38% did not seem to have a basic knowledge about what they should do in this situation (*Unaware reactors*).
- 14% were fleeing the fire at the time of their death (4% by car and 10% by foot).
- 69% was "passively sheltered" inside a house or other building at time of their death (some of these tried to defend their property).
- 44% was classed as "vulnerable" because of the age (less than 12 or more than 70 years old) or because they were suffering from an acute or chronic illness or disability (more detailsa bout the age and gender repartition on the figure 4.1).
- 32% died of the peoples died on properties whose defendability was questionable.



Source: Exhibit 894 – Review of Fatalities in February 7 2009 Bushfires.[3]

Figure 4.1: Those who died: Age and gender profile. building)[13].

— 5 —
Implementation description

In order to make our simulation, we chose to use the GAMA software which provides a plug-in for BDI implementations with an extension of the GAML language [7]. GAML has the advantage to be simple to use and to understand. The BDI implementation is based on a previous simulation written on GAMA which use an finite-state machine (figure 5.1) for the agents reasoning [3]. The behaviour of agent in the finite state machine implementation is represented by 7 steps. The first step (initial step) is the *Unaware* step. When the agent detect a fire, it switch from *Unaware* to *Aware indecisive*. The decision-making process is done by two random value (*defendMotivation* and *escapeMotivation*). The *flip* function is used in order to generate a number between 0 and the value defined in parameter which corresponds to the probability to get a *true* (boolean value). The higher value between these two flips determine if the agent chooses to defend or leave. This decision making process and the lack of flexibility is the biggest weakness of the finite state machine architecture. Indeed, an agent cannot change its strategy if the FSM does not make the transition explicitly. This is for this reason, the BDI architecture provides a real benefice in term of realism. The BDI implementation use the same environment and fire model.

5.1 Environment description

The figure 5.2 is a graphic representation of the simulation which is the same as the previous version. the map called "world cells", is represented by a green grid where each cell can contain a shelter, a building, a fire, a source of flames or an empty cell. The fire simulation

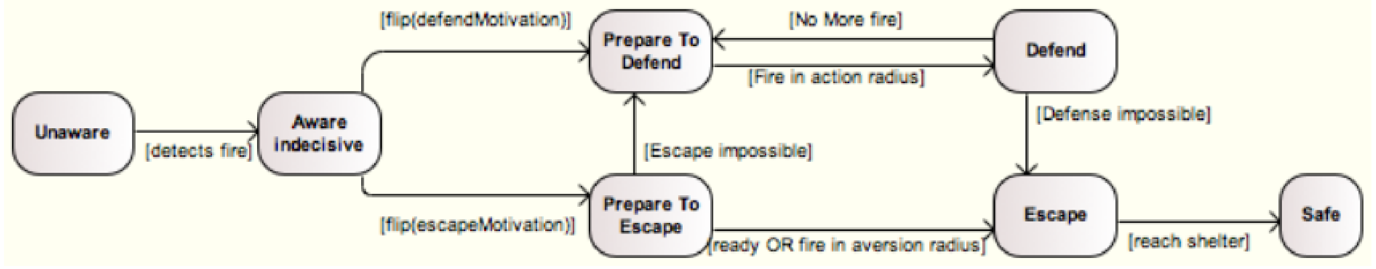


Figure 5.1: Finite-state machine of the agent behaviour[5].

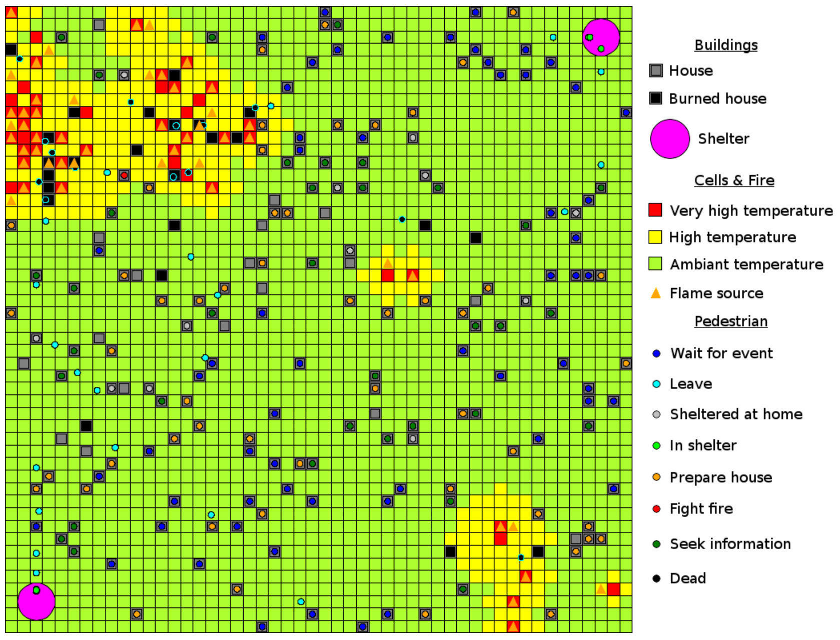


Figure 5.2: Graphical rendering of the BDI simulation.

is quite simple but is good enough to interact with Pedestrian agent (the citizens). During the simulation the fire will grow and propagate. According to the position of the agents and to the centre of the fire represented by an orange triangle, the Pedestrian agent will receive more or less damage. The red cells represent very hot area and the yellow the fire's radiant heat.

The shelter is a circle with a stationary position (on the top right and bottom left corner) defined at the initialisation step. Each agent who enters in a shelter ensures his safety until the end of the fire. Buildings are represented by a square with the same size as a cell. In our simulation, the building represent citizen's house. Each house has a durability attribute which defines the damages caused by the fire. If a building durability decreases to 0, then the building is destroyed and becomes black.

5.2 BDI agent description

The BDI implementation is composed by a main agent called "Pedestrian" which corresponds to the Australian citizens. Each Pedestrian agent has a health value which decreases according to the injuries caused by the fire and has goals to achieve represented by the figure 5.3. At the start, all agent are unaware about the fire and are waiting at home until they perceive a fire. At this moment, the agent (represented with a blue circle) chooses its initial intention which can be simply to stay alive or to defend its property. This initial objective is determined by the attributes of the agent. An agent with a high danger aversion has a bigger chance to leave and take cover. However, an agent who is not afraid to fight fire will probably choose to defend his property.

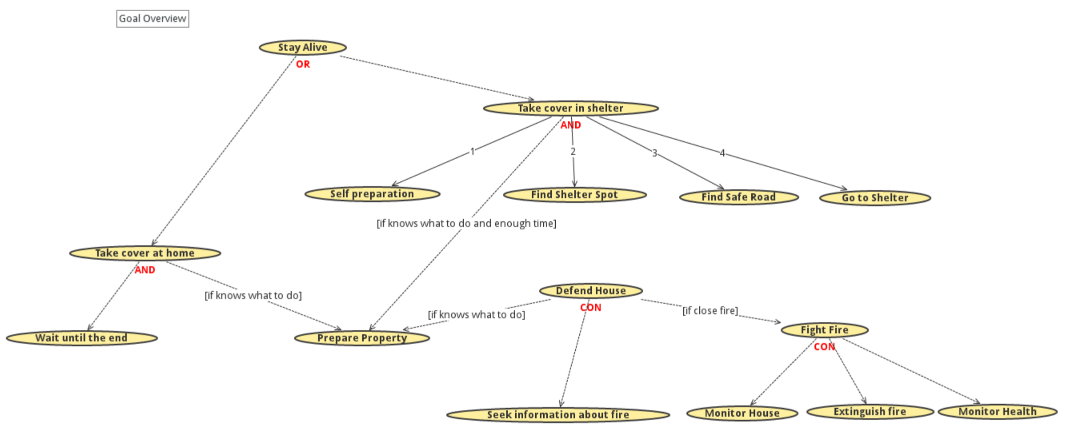


Figure 5.3: Description of the Pedestrian agent goals. Made with the Tactics Development Framework (TDF) plug-in on Eclipse IDE. [25, 8]

5.2.1 Attributes

The Pedestrian agent has two categories of attributes: physical attributes and psychological attributes.
The physical attributes are composed by :

- The health: Corresponds to the health state of the agent. If the health goes down to 0 then the agent dies.
- The injuries: Which contains the total amount of damages the fire caused to the agent.
- The velocity: Corresponds to the agent speed when it is moving.

The psychological attributes :

- The determination: Represents the probability to do the action.
- The persuadabillity: The probability for the agent to be more or less influenced by others agents or the environment.
- The danger aversion: Corresponds to the probability that the agent can accepts the fact that he is in a dangerous situation or efraid of the situation.
- The perception radius: Determines the perception radius where the agent is able to detect fire.
- The danger radius: Correspond to the area where the agent will begin to feel that the situation becomes to be dangerous.
- The defense radius: Determines the area where the agent considers that the fire is close enough and must be fought.

5.2.2 Predicates

In this implementation we made the choice to simplify the model and to reduce the number of predicates in order to keep a simple model. Thus, Pedestrian agents can have only 3 beliefs and 4 desires. The priority of desires can be variable according to the agent attributes. For example, an agent a high danger aversion will have a bigger chance to desire to stay alive.

The agent can believe that :

- it is in a dangerous situation (*belief_danger*),
- it knows a shelter position (*belief_shelter_position*),
- There are fires (*belief_fire_position*).

The desires :

- *waiting_for_event*: The agent can have the desire to wait in order to see how the events will evolve. This desire will change according to the danger aversion. Indeed, a person who is afraid will not stay and see what happens but will have the desire to *stay alive* instead.
- *stay_alive*: The Pedestrian desires to stay alive (variable according to the danger aversion as well).
- *protect_property*: If it is determined enough to deal with the fire, it can have the desire to protect his property.
- *get_information*: A Pedestrian desires to seek information about the current situation.

5.2.3 Plan

Stay Alive

When an agent want to stay alive, he has the possibility to take cover at home or to go in shelters. If the agent knows how to fight a fire, he can choose to prepare his property before leaving or before taking cover in until the end of the fire. Peoples how want to go in a shelters will make preparation for himself such as take water, food or medicines which are represented by the goal “self preparation”. After what, if the agent already knows shelters’ position will choose the closest one and determine the safest path to use. In contrary, if the agent does not know shelters, it can look around and follow another person who seems to know where to go.

Defend House

An agent who chooses to defend its property will have the choice between three plans. The first plan is to look for information using radio, phone or on the web. In this simulation we assume the citizens use only the radio. If the agent learnt how to fight a fire, the agent has the possibility to prepare its house against the fire. Finally if the fire is close to the house, the agent will try to extinguish it. According to its knowledge the fire fighting effect will be more or less efficient. During the fight, the agent regularly checks the condition of the house and its health condition as well. If the agent considers the situation is desperate or if the house burns despite fire fighting, he will change its priority and the desire to stay alive will become more important than the desire to defend its property.

Death

During the simulation an agent may die due to the fire. In this implementation, the colour of the agent will switch to black. The border colour corresponds to the last agent state before its death.

— 6 —
Expectations and Results

The experimentation was made 10 times of 500 cycles (time units) with two different sets of parameters for the fire:

- The first experiment was set in order to have a low propagation rate, a moderate probability for the fire to grow and ungrow (increase or decrease the fire intensity) but with a low initial intensity:
  - Probability of growing: 0.4
  - Probability of ungrowing: 0.1
  - Probability of propagation: 0.1
  - Initial intensity: 1 (max value: 15)
- The second experiment was defined as a violent fire, the propagation rate is 3 times more important than the first experimentation, the fire has a bigger probability to grow (2 times more) with a bigger initial intensity.
  - Probability of growing: 0.8
  - Probability of ungrowing: 0.1
  - Probability of propagation: 0.3
  - Initial intensity: 3 (max value: 15)

Each simulation begins with exactly 10 fire spots (with a maximum of 50 fires), 200 Citizens, 200 building (one per citizen) and 2 shelters. Fires and citizens positions are generated randomly among the remaining free areas, the building positions are exactly the same as the citizen positions. That means that all citizens at the initial state are in their house. Only the shelters have fixed positions on the bottom left and the top right corner of the map.

Gathered data

During each experimentation, we save the last plan used for the agent who died and we categorise each remaining agent which are aware of the fire in 5 distinct categories. We have voluntarily reduced the list of the categories described in the chapter 4 from 6 to 5 and decided to merge the category livelihood defenders with the category Can-do defenders. Indeed, the

difference between these two categories is in the attachment to their home which is not implemented in the currrent simulation.
The results are represented in two differents graphs: The first graph represent the repartition of the last plan used by agents who died during the simulation. The second graph contains the profiles of agent who are aware of the fire according to the following conditions:

- Unaware reactors: Percentage of agents who used the plan *do_nothing*, did not try to get cover at home or in a shelter, did not try to seek information, to protect its property and did not try to fight the fire.
- Threat avoiders: Percentage of agents who go to shelter instead of staying at home. They do not try to seek information or to fight fire. But they can prepare their property before leaving.
- Threat monitors: Percentage of agents who take cover at home or go to a shelter and seek information about the fire. They do not try to fight the fire but can prepare their property if they want to.
- Can-do defenders: Percentage of agents who know what to do during a bushfire (*knows_how_to_fight_fire*) prepare their property and fight the fire. There are no restrictions on the other actions. They can leave, take cover at home or seek information if they wish.
- Considered defenders: Percentage of agents who fight fire without knowing how to do it. They do not prepare their property (see the *prepare_property* plan description) but can seek information, get cover at home or leave as well.

6.1 Expected results

Among the gathered data, we expect to find in the agent behaviour the profiles described previously in chapter 4. Moreover, the behaviour should be different according to the fire intensity. An intense fire must produce a bigger psychological pressure and increase the number of people who leave earlier. Furthermore, the important fire radiance produced by the fire will decrease the number of people who will survive when they choose to take cover at home or when the person tries to fight the fire. To finish, an important fire should increase the number of people who will change their mind when they understand that the fire is too important to be extinguished. If the fire becomes more intense then the level of agent awareness must increase (*Unaware reactors* profil). A low fire intensity should show a bigger number of unaware profiles, less people will leave early or will change their mind if they are fighting the fire. We can expect as well to see more considered defenders during a low intensity fire than an intense one.

6.2 Current results

This section describes the results obtained during the test simulation made. Unfortunatly by lack of time and because of issues with GAMA, we did not have time to gather enough data in order to obtain robust results. Thus, this data may be not representative of the real agent behaviour because of the randomicity of the different parameters. The very low average of agents dead while they fight fire is due to the the randomness of the fire behaviour. The fire

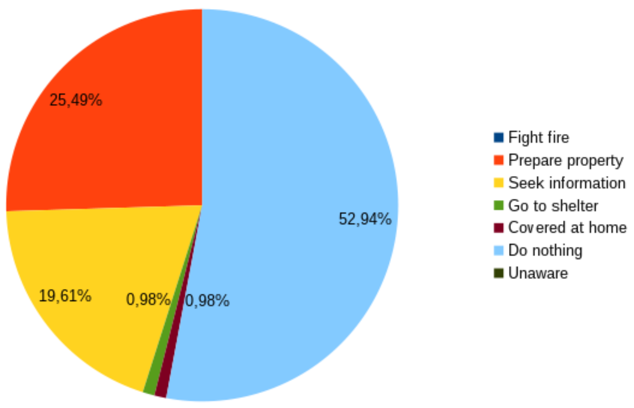


Figure 6.1: Low fire – last plan used before dying.

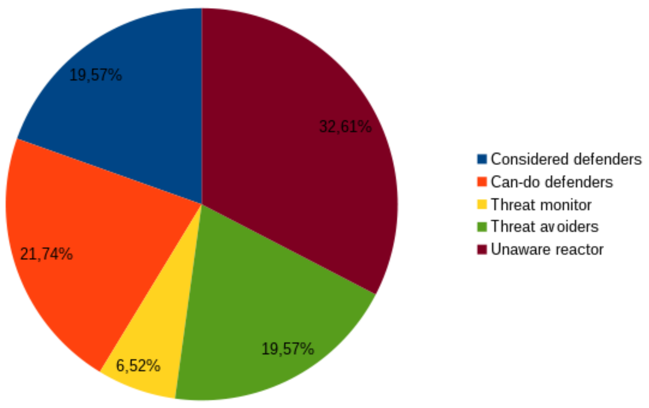


Figure 6.2: Low fire – Agent profile.

can appear near the agent during one cycle and disappear the next cycle. Thus, the agent is fighting fire for only one or two cycles and switches back to another plan after that because of the absence of proximity of fire.

Low fire intensity

The figure 6.1 shows that the biggest majority of the Pedestrian agents are dead because of their inactivity (*Do nothing* 52.94%). 25.49% are dead during the property preparation and 19.61% during the time where they tried to gather information. Very few people are dead during the travel between their house and the shelter and during they were covered in their house (both at 0.98%).
The profile graph (figure 6.2) shows that 32.61% of the agents match with the profile *Unaware reactors*. The *Can-do defenders* are the second more represented profile with 21.74%, followed by the considered defenders and the threat avoiders which are both at 19.57%. Threat monitors are the last profile with 6.52% of the agents.

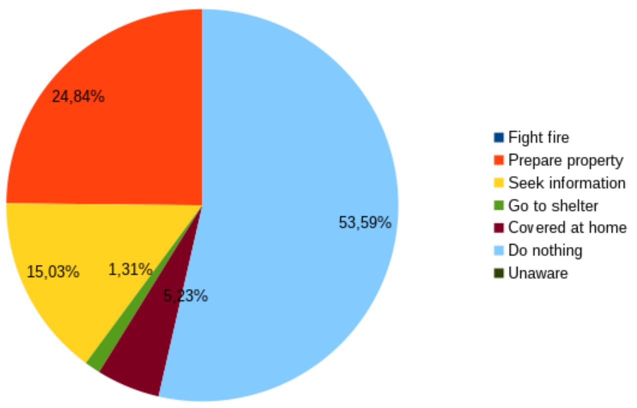


Figure 6.3: High fire – last plan used before dying (percent values).

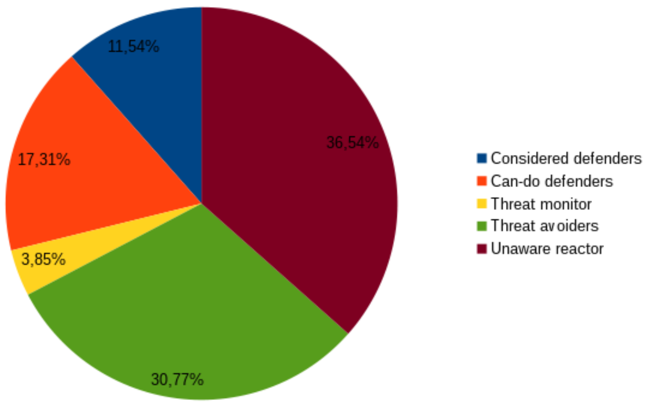


Figure 6.4: High fire – Agent profile (percent values).

High fire intensity

At first glance, the results seem to be quite similar to the small intensity fire. As we can see on the figure 6.3 with an intense fire, more than half of the causes of death are due to an inactivity from the Pedestrian agent (53.59%). The last plan used before dying is the preparation of the property with 24.84%, and during the research of information (15.03%). 5.23% of dead Pedestrian were sheltered in their house against 1.31% who died during the travel from the house to the shelter.
The profiles (figure 6.4) seem to be similar as well. The first more represented profile is the *Unaware reactor* with 36.54%. followed by the *Threat avoiders* with 30.77%. The *Can-do defenders* represent 17.31% against 11.54% for the *Considered defenders*. The *Threat monitors* is again the last profile with 3.85% of the Pedestrian agents.

6.2.1 Comparison

The figure 6.5 show that the fire intensity does not influence the cause of death. The two biggest differences can be noticed on the percentage of Pedestrians who take cover at home, which is more than 5 times more important during an intense fire than during a lower one. This differ-

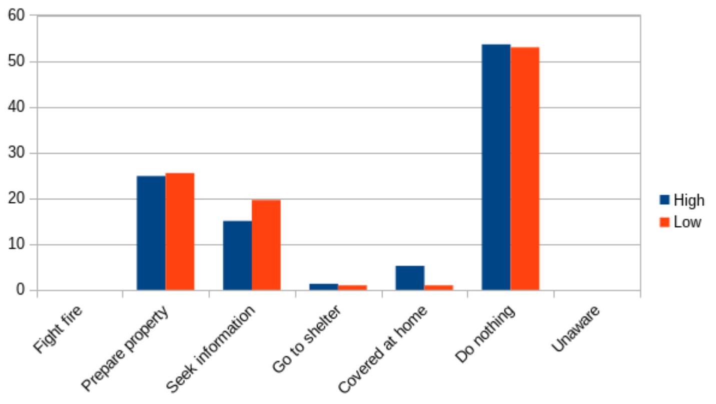


Figure 6.5: Last plan comparison (percent values).

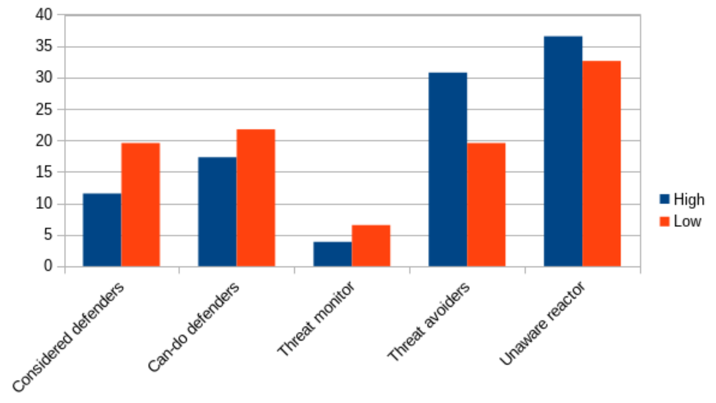


Figure 6.6: Profil comparison (percent values).

ence can be explained by the fact that an agent who is surrounded by fire will prefer to shelter at home instead of crossing the fire in order to go in a shelter. The second notable difference is the percentage of agents who search information during a low fire (+4.58%), which can be explained by the fact that a lower fire includes a lower radiance area than a bigger one. That means an agent in a lower fire will have a bigger probability to stay at the plan *seeks_information* than with an important fire.
The difference between the profiles seems to be more bizarre. Curiously the number of *Unaware reactors* seems to be more important during an intense fire than the low intensity fire. However, the difference for the *Threat avoider* profile is more realistic in the sense where an intense fire increases the chance to scare an agent. But the current implementation does not take in account the fire intensity in the decision making process. This difference is probably due to the low quantity of data gathered with simulations. The last profiles correspond better to the expected result, a lower fire increase the number of agents in the profile *Can-do defenders* and *Considered defenders*.

To conclude, the results shows that the simulation needs to be more detailed on some points and should take more into account the fire parameters in order to improve the decision-making process. The distinction between a low intensity fire and an important one is too weak (or nonexistant). Moreover, if we make a comparison between the statistics from the VBRC and the BDI implementation, we can see that the death causes are not really corresponding. At the exception of the people who were passively sheltered in a building (53% for the simulation against 69% during the Black Saturday) the results seem not match with the reallity. In the futur, we need to define more clarity the related statistics for each profil and death causes.

— 7 —
Conclusion

In this study we showed that the BDI architecture provides a more intuitive simulation implementation than other simulation approaches for human behaviours simulation. However, there are a lack of existing tools for the modelling of BDI agents due to the complexity of the language used. Consequently, scientists who do not have knowledge in computer science cannot use the tools in their full potential. We show an easy way for these scientists to make their own simulation using TDF for the synthesising of textual data such as witness statement and GAMA for the simulation making process. The simulation model is still incomplete and can be improved beside some adjustments for a better matching with reality. Moreover, the simulation's result does not shows enough similarity with the behaviours described in the witness statements despite a too weak distinction between the differents parameters. We tried to have the more simple implementation without lowering descriptiveness in order to find emergent profiles from the model. These profils are indeed emerging but do not seem to be correctly proportioned and require more ajustement and more implementation improvement.

Futur Work

In the future, we would like to provide more possibilities to agents. Some plans are still incomplete such as "seek information" which need an implementation of a radio broadcasting system in order to provide information about fire position to the other agents. Many statements show the importance of relationships between agents, such as family, friends, neighbourhood relationships. We would like as well to make a comparison between the FSM and the BDI implementation. The validation of the simulation will be made by a better comparison with the witness statements and the statistics provided by the CFA.

The last version of the code is available on github : `https://github.com/kust2708/swift.git`.

Bibliography

[1] Witness statement. http://vol4.royalcommission.vic.gov.au/index03a1.html?pid=111.

[2] The lessons learnt. http://www.royalcommission.vic.gov.au/Finaldocuments/volume-1/PF/VBRC_Vol1_Chapter21_PF.pdf, 2009.

[3] Carole Adam, Elise Beck, and Julie Dugdale. *Information Systems for Crisis Response and Management in Mediterranean Countries: Second International Conference, ISCRAM-med 2015, Tunis, Tunisia, October 28-30, 2015, Proceedings*, chapter Modelling the Tactical Behaviour of the Australian Population in a Bushfire, pages 53–64. Springer International Publishing, 2015.

[4] Carole Adam, Geoffrey Danet, John Thangarajah, and Julie Dugdale. Bdi modelling and simulation of human behaviours in bushfires "submitted". 2016.

[5] Carole Adam and Benoit Gaudou. Modélisation de comportements humains en situation de crise à partir d'entretiens : application aux incendies de forêt de melbourne (cnia). July 2016.

[6] Rhodes Alan. Why don't they do what we think they should? In *AFAC*. Emergency Management Victoria, 2014.

[7] Philippe Caillou, Benoit Gaudou, Arnaud Grignard, Chi Quang Truong, and Patrick Taillandier. A Simple-to-use BDI architecture for Agent-based Modeling and Simulation. In *The Eleventh Conference of the European Social Simulation Association (ESSA 2015)*, Groningen, Netherlands, September 2015.

[8] Rick Evertsz, John Thangarajah, Nitin Yadav, and Thanh Ly. A framework for modelling tactical decision-making in autonomous systems. *Journal of Systems and Software*, 110:222–238, 2015.

[9] Sue Exell. Witness statement. http://vol4.royalcommission.vic.gov.au/index03a1.html?pid=111.

[10] GAMA. Gis & agent-based modelling architecture. http://gama-platform.org.

[11] Michael Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Michael Wooldridge. *Intelligent Agents V: Agents Theories, Architectures, and Languages: 5th International Workshop, ATAL'98 Paris, France, July 4–7, 1998 Proceedings*, chapter The Belief-Desire-Intention Model of Agency, pages 1–10. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[12] L Greengard and V Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[13] Dirk Helbing, Illes Farkas, and Tamas Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, September 2000.

[14] Mu H.L., Wang J.H., Mao Z.L., Sun J.H., Lo S.M., and Wang Q.S. Pre-evacuation human reactions in fires: An attribution analysis considering psychological process. *Procedia Engineering*, 52:290–296, 2013. 2012 International Conference on Performance-based Fire and Fire Protection Engineering.

[15] Francois F. Ingrand, Michael P. Georgeff, and Anand S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert: Intelligent Systems and Their Applications*, 7(6):34–44, December 1992.

[16] Max T. Kinateder, Erica D. Kuligowski, Paul A. Reneke, and Richard D. Peacock. A Review of Risk Perception in Building Fire Evacuation. Technical report, National Institute of Standards and Technology, September 2014. NIST Technical Note 1840.

[17] Erica D. Kuligowski. Modeling human behavior during building fires. Technical report, National Institute of Standards and Technology, 2008.

[18] David J. Low. Statistical physics: Following the crowd. *Nature*, 407(6803):465–466, September 2000.

[19] MABS. *A BDI agent architecture for the GAMA modeling and simulation platform*, May 2016.

[20] Nathan M. Newmark. A method of computation for structural dynamics. *Journal of Engineering Mechanics*, 1959.

[21] Xiaoshan Pan, Charles S. Han, Ken Dauber, and Kincho H. Law. Human and social behavior in computational modeling and analysis of egress. *Automation in Construction*, 15(4):448–461, 2006. The first conference on the Future of the {AEC} Industry (BFC05).

[22] Xiaoshan Pan, Charles S. Han, Ken Dauber, and Kincho H. Law. A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations. *AI & SOCIETY*, 22(2):113–132, 2007.

[23] Nuria Pelechano, Kevin O'Brien, Barry Silverman, and Norman Badler. Crowd simulation incorporating agent psychological models, roles and communication. Technical report, Center for Human Modeling and Simulation University of Pennsylvania, 200 S. 33rd St. Philadelphia, PA 19104-6389 USA, 2005.

[24] Anand S. Rao and Michael P. Georgeff. Bdi agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319. IAAA, 1995.

[25] TDF. Tactics development framework. http://agentprojects.com/tdf/.

[26] H. Van Dyke Parunak, Robert Savit, and Rick L. Riolo. *Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide*, pages 10–25. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[27] Wilfred F. van Gunsteren and Herman J. C. Berendsen. Computer simulation of molecular dynamics: Methodology, applications, and perspectives in chemistry. *Angewandte Chemie International Edition in English*, 29(9):992–1023, 1990.

[28] Theo van Ruijven. Serious games as experiments for emergency management research: A review. In *8th International ISCRAM Conference*, Lisbon, Portugal, May 2011. need for realism of underlying model to get valid results, transferable to real life situations.