QUANTICOL

A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours

quantiCOL

http://www.quanticol.eu

TR-QC-02-2015

Modelling disease spread in CARMA

Author(s): Ludovica Luisa Vissat, Christopher D Williams

Publication date: 10th April 2015

Funding Scheme: Small or medium scale focused research project (STREP)
Topic: ICT-2011 9.10: FET-Proactive 'Fundamentals of Collective Adaptive Systems' (FOCAS)
Project number: 600708

Coordinator: Jane Hillston (UEDIN)
e-mail: Jane.Hillston@ed.ac.uk
Fax: +44 131 651 1426

Part. no.	Participant organisation name	Acronym	Country
1 (Coord.)	University of Edinburgh	UEDIN	UK
2	Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo"	CNR	Italy
3	Ludwig-Maximilians-Universität München	LMU	Germany
4	Ecole Polytechnique Fédérale de Lausanne	EPFL	Switzerland
5	IMT Lucca	IMT	Italy
6	University of Southampton	SOTON	UK
7	Institut National de Recherche en Informatique et en Automatique	INRIA	France

COOPERATION

1 Introduction

In this report we briefly present models of a simple scenario of disease spread. The scenario which we consider has three types of agents:

S, **Susceptibles:** individuals who do not currently have the disease but who are susceptible to infection.

I, **Infectives:** individuals who have the disease and may actively spread it by passing it on to susceptible individuals.

R, **Recovereds:** individuals who have had the disease and recovered and who are currently immune from infection.

2 CARMA SIRS example

We present three different ways of representing a simple SIRS model using the language CARMA. The SIRS model is a simple extension of the epidemiological SIR model: it allows agents that are recovered to be free of infection and then become susceptible again. We will represent the same action types and the same mechanism in all the models but we will describe them in different ways.
The possibility of describing the same model in different ways is an interesting feature of CARMA.

2.1 First model

In the first model here each agent has a single attribute, zone, representing its location. Here we present one agent of each type in each location and our agents are located in a grid like the one represented in the following figure. All the agents can either move in their Von Neumann neighbourhood or remain in the same location. The *contact* action happens only between S and I agents that are in the same location while *recovery* and *susceptible* are directly executed by an agent acting alone.

1	2
3	4

The CARMA code describes the SIRS model as follows:

```
//Component declarations
//Susceptibles
(S,{zone := 1..4});
//Infectives
(I,{zone := 1..4});
//Recovereds
(R,{zone := 1..4});

//Process definitions
S = contact*[z == zone](z).I +
[zone == 2 || zone == 3]move*{zone := Uniform(1,zone,4)}.S +
[zone == 4 || zone == 1]move*{zone := Uniform(2,zone,3)}.S;
```

```
I = contact*<zone>.I +
[zone == 2 || zone == 3]move*{zone := Uniform(1,zone,4)}.I +
[zone == 4 || zone == 1]move*{zone := Uniform(2,zone,3)}.I +
recovery*.R;

R = susceptible*.S +
[zone == 2 || zone == 3]move*{zone := Uniform(1,zone,4)}.R +
[zone == 4 || zone == 1]move*{zone := Uniform(2,zone,3)}.R;
```

The *movement* is defined in the Von Neumann neighbourhood and it changes the location but not the state of the agent. We added a guard to the *movement* action to represent the different possible movements depending on the current location of the agent. Therefore, the different movements can happen only if the predicate is satisfied. This action is modelled as a broadcast action: it is a non-blocking action so we do not need any receiver process to execute it.

The action *contact* is modelled as a broadcast input/output where the agent I sends the zone where he is located and the agents S in the same zone (they satisfy the predicate guard) will update their state, becoming infected. For the *recovery* and *susceptible* actions we used the predicate "False" since they do not need to synchronise. These actions are denoted as *spontaneous actions*. For different types of infection, e.g. sexually transmitted diseases, we might choose to represent the contact as a unicast action.

2.2 Second model

The second model presents a variation on the description of the same system. We present just one process with two different attributes, the location and the state. We indicate with 1, 2, and 3 the susceptible, the infective and the recovered states respectively. The agents can still move in their Von Neumann neighbourhood and we insert the guards to allow only some types of agent to make some different types of actions. The contact can influence just susceptible agents and it can be "sent" only from infective ones, while the agent has to be infective to recover and has to be recovered to become susceptible. The ellipsis notation is short hand for multiple declarations of a Component with different valued attributes, the values range over the numbers given. Below the total number of Components will be 12, representing each zone having one of each type of state.
This model is written in CARMA as follows:

```
//Component declaration
//state: 1 = susceptible, 2 = infectives, 3 = recovered
(A, {zone := 1..4, state := 1..3});

//Process definition
A = [zone == 2 || zone == 3]move*{zone := Uniform(1,zone,4)}.A +
[zone == 4 || zone == 1]move*{zone := Uniform(2,zone,3)}.A +
contact*[z == zone && state == 1](z){state := 2}.A +
contact*[state == 2]<zone>.A +
recovery*[state == 2]<>{state := 3}.A +
susceptible*[state == 3]<>{state := 1}.A;
```

This model is more compact than the previous one, since we transformed the state into an attribute. All the actions have the same effects and updates as in the previous model but we describe some of them in a different way. They require some guards, since they are specific actions for specific types of

agent; for example, the contact can happen only between S and I agents. Furthermore, the *recovery* action can be made only by infective agents and the *susceptible* action only by recovered ones.

2.3 Third model

In this third model we will illustrate the idea of having a parallel composition of behaviour affecting state and location, two processes within a component that can evolve independently, representing different aspects of behaviour. We can describe this using CARMA as follows.

```
//Function definition
fun Move(zone){
  if(zone == 2 || zone == 3){
      return Uniform(1,zone,4)
  } else {
      return Uniform(2,zone,3)
  };
};

//Component declarations
(S|A,{zone:=1..4});
(I|A,{zone:=1});
(R|A,{zone:=1});

//Process definitions
S = contact*[z == zone](z).I;
I = contact*<zone>.I + recovery*.R;
R = susceptible*.S;
A = move*{zone := Move(zone)}.A;
```

We describe the different agents using a parallel composition. The first process describes the state and it evolves as usual ($S \to I \to R \to S$). The second process A describes the movement and the update will affect the locations. Function definitions provide the modeller with greater expressive power than update expressions. In this case the function definition provides an if-case, which returns a uniform distribution allowing for randomised Von Neumann movement.

2.4 Environment

Each of these models defines a collective and these collectives can all be considered in the same environment. The environment consists, in general, of a global store and an evolution rule. In this case the global store is empty, and therefore omitted. The evolution rule can have three functions: prob, rate and update. Here the update is not needed because there is no global store. The function "prob" is for defining the probability of a message being received, in our model we only have one input action, and we have expressed the default of 1.0, messages for all actions will always be received. The function "rate" defines the rates for our actions, the boolean predicate provides a look-up in order to have different rates over actions, move* has 3 different possible rates depending on the zone the owning Component can be found in.

```
environment{

  prob{
    default := 1.0;
  }

  rate{
    [this.zone == 1],move* := 0.01;
    [this.zone == 2],move* := 0.01;
    [this.zone == 3],move* := 0.02;
    [this.zone == 4],move* := 0.03;
    [True],contact*<> := 0.015;
    [True],recovery* := 0.2;
    [True],susceptible* := 0.2;
    [True],contact*() := 0.015;
    default := 1.0;
  }
}
```