Discrete Event Simulation of a FRAM model in SimPy

Eric van Kleef, Geulwijk 16, 3831 LM Leusden, The Netherlands
eric@vankleefconsultancy.nl

Introduction

In the latest FRAM workshop in Munich 2013, simulation of a FRAM model was discussed. Special points of interest were the implementation of variation and the possible use of fuzzy numbers.

Simulation is a way to show the consequences of assumptions done about reality. In a way, a simulation is a laboratory, in which experiments can be done that can not be done in reality (Sonnessa, 2004). A simulation does not differ from a mathematical solution of a set of equations; the mathematical solution gives not more information about reality than the underlying equations, but it does show the consequences of the equations more clearly. The set of equations can be seen as the hypothesis about reality and the mathematical solution gives the results that can be compared with measurements to falsify the hypothesis. In the same way our FRAM model is our hypothesis about the real system we are studying. The simulation clarifies the consequences of the relations that are described in the FRAM model, which can then be tested against observations in the real world. If the FRAM model holds, it gives the best available model of the reality and the results of the simulation give a believable description of the behaviour of the system.

Of course, the simulation can never be better than the underlying FRAM model. As the FRAM model incorporates many qualitative relations, the results of the simulation should also be understood in a qualitative way. The simulation helps us to understand the behaviour of the system and gives us a laboratory to experiment with that behaviour. The simulation does not aim at quantification of the model, but merely at identifying the possible scenarios.

A clear distinction has to be made between the dynamics of the system during one mission and the dynamics on a life cycle scale. During the life cycle of the system, many missions are carried out. On the life cycle scale the systems adaptiveness should be studied. Knowledge about the dynamic behaviour of the system during one mission is needed in order to be able to assess the consequences of the long scale adaptations (Van Kleef and Stoop,

2014). It is not possible to simulate both the short term mission dynamics and the long term adaptation dynamics in the same model. The short term dynamics require a short time step and long term dynamics require a long simulation time. The combination of the two in one model gives an inhibiting large number of time steps. On a long term time scale, control and resources fluctuate, e.g. by adaptation and drift into failure phenomena (Dekker, 2011). These phenomena should be simulated in a different model and the results of the short term simulation can be used to assess the results of the adaptation model.

At the same time, the influence of variation requires several instantiations to be simulated, further increasing computational load. A possible solution for this dilemma is the use of fuzzy sets to describe variation. Several attempts have been made to introduce fuzzy sets in simulations (Bonarini and Bontempi, 1994; Cheng and Wu, 2006; Olson and Wu, 2006; Vescovi and Travé-Massuyès, 1992; Zhang et al., 2005). Although it seems possible to simulate with fuzzy durations, the calculations become tedious. It seems that non-linearities are difficult to deal with.

Another way of looking at variation in simulations is the Monte Carlo method. In this method, many runs are made with the simulation with slightly different values to get a probability distribution as a result. In those simulation control and resources can be kept as a constant during the simulated mission and the consequences of variation in control and resources can be studied.

Discrete event simulation

In a FRAM model, changes in the system occurring during execution of a function are not described, but only the result of the function is taken into account as a system state change. The sequence in which the functions are executed is not prescribed. On the contrary, different instantiations of the model can have a different execution sequence of the functions. In some instantiations a function will not execute at all. Another characteristic of FRAM modelling is the variation that occurs between instantiations. The output and duration of a function can variate between one instantiation and another. These variations can be a result from the different system states in which the function is called or can be inherent to the function.

The simulation method chosen should reflect these characteristics of a FRAM model. Instantiations are represented by runs in the simulations in a Monte Carlo like manner, the same simulation run many times with variations in the execution of the functions. Due to the complex nature of a model, this can result in instantiations that are very different. Every run of the simulation represents an instantiation of the FRAM model. If a function may or may not execute, some runs should represent the situation in which the function executes and some the situation in which the function does not execute. The result of the simulations are a set of scenarios, enableing us to generate as many scenarios as necessary. The simulation should also restrict itself as much as possible to the description given in the FRAM model.

Fuzzy events

Output of functions are expressed by fuzzy boolean variables, with values between 0 and 1 (Zadeh, 1965). The value 0 means that the result is non existant or completely worthless, the value 1 that the result is exactly what it should be. Values between 0 and 1 represent situations where a result is existant that deviates from the ideal situation because of incomplete data, unclear messages or similar conditions.

The start condition and preconditions may not be clearly defined. This means that it is not crispy clear whether the function should start or not. This is a sitation that will often be met in practice. As an example: “If it rains, you should take your umbrella with you”. It is clear that if rain is pouring down, one should take an umbrella. It is also clear, that if it is a bright sunny day, the umbrella can be left at home. But what if it is a bit drizzling, or it is dry but the sky is very dark? These are situations where the precondition is somewhere between 0 and 1. It is also possible that the start condition is clearly defined but it is not clear whether the condition is met. Example: “If A gets an order from B, he should start”. Fuzzy situations might be if A gets a message from B, but the message is not clear.

A fuzzy precondition is understood as a probability that the function will start. If the precondition is true, than the function will start, if the precondition is false, the function will not start, and if the precondition somewhere in between, the function may or may not start.

Control variables can also be expressed as fuzzy booleans. The value 0 is here to be understood as the control being completely abesent. As a result the output will be a random value. If the control is 1, the output will also be 1.

If the time variable is 1, the duration of the function will equal the nominal duration. If the time variable is smaller than 1, the duration will follow some distribution around the nominal duration. A useful distribution is the lognormal distribution.

Implementation of FRAM model in SimPy

The simulation presented here is a discrete events simulation of the FRAM model, implemented in SimPy. SimPy is a process oriented discrete events simulation package written in standard Python. SimPy was first designed by Klaus Müller and Tony Vigneau in 2002 and is at present developed by Ontje Lünsdorf and Stefan Scherfke (Vigneau et al., 2012). Python is described in the Python language reference. The Pythonista App on iOS was used here (OmzSoftware, 2013). Both SimPy and Python are open source software, thatcan be operated on many platforms, including Windows, OSX and iOS.

In SimPy several building blocks exist for making discrete event simulations. The most important ones for our purpose are processes, events and resources. Processes simulate an entity which evolves in time, resources simulate something that is needed for the process to execute and that has to be queued for. Events are entities that can happen sometime and that can have a value (Matloff, 2008a, b). Functions are represented by processes, messages between functions as fuzzy events. All events take a fuzzy value between 0 and 1. A value zero represents a non existant event, a value one corresponds to an event that has no variation. If an event happens with variation the value is something between 0 and 1. A fuzzy event is implemented as a SimEvent in SimPy. Input, precondition, control, time and output are all of the type SimEvent. This makes it possible to simply use the output of one function as the input, precondition, control or time for another one. In the current implementation it is not posssible to use an output as a resource for another function. Input, precondition, control and time are all lists of events. The output is a list of two events, one that is fired if the function is completed and one that fires if the test on the preconditions fail.

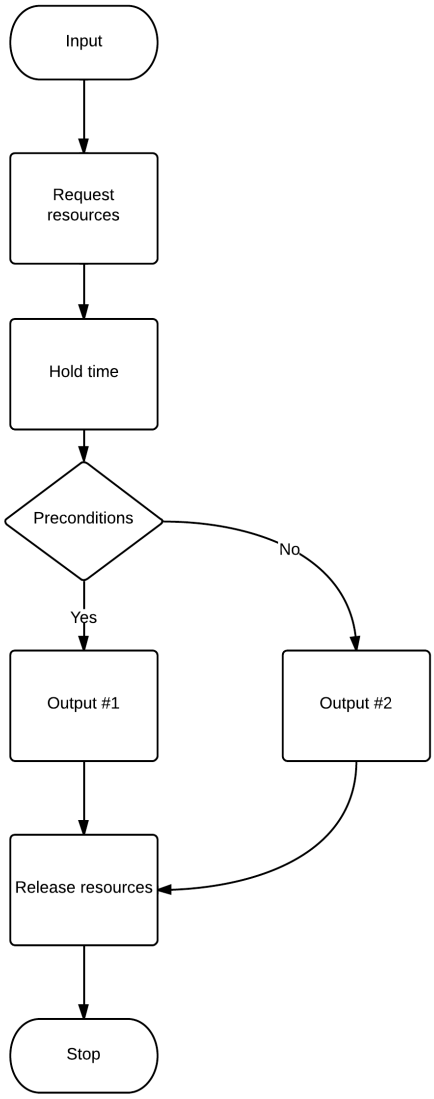

Figure 1 Flow chart of a FRAM function

The function starts as soon as one of the input events is happening (figure 1). The input event is fuzzy. If, for example, someone has to react on a message of somebody else, this message can be ambiguous. In one instantiation the function either starts or doesnot start, so the input has to be defuzzyfied. The defuzzification is implemented as the probability of the function starting being equal to the value of the event triggering the function.

A special point of consideration is what happens if the function is triggered twice. There are several possibilities: the second trigger may be neglected, the function may be executed twice or the second trigger may cancel the first. In the current implementation, the function starts twice.

If the function is triggered, it waits for the resources to become available. A resource can only be used by one function at a time. If all the resources are available, the function continues to the next step. The function checks the preconditions; if they are met, the function continues, if not, the function aborts and fires the second output event. If the preconditions are not crisp, the function may interpret this as a true or a false. The probability of going on is taken as the product of all values of the preconditions. If the function aborts, the resources are released.

What happens in situations like this? A choice has to be made between a few possibilities. The function can start with a delay, the function can start with variation in the output, the function might or might not start. It is the most plausible that the function will either start or not, but that if it starts, the function will not be hampered by the partly lacking precondition. If the condition also influences the output, it should be doubled as an input. If the condition also influences the duration, it should be doubled as a time parameter. If the condition influences the variation, it should be doubled as a control.

In the next step the function fires the output after a certain amount of time. The amount of time is determined by the time list. The first value in the list represents the nominal duration. This is a constant value. The other items in the list are the events that determine the variation in duration. The actual duration for the function is taken as a lognormal distribution with the nominal duration as mean and the variation

The output value is taken as a random draw from $1^{1/c}$, with c the minimum value of the control list. If the control is 1, meaning complete control, the output is 1. If the control = 0, meaning no control at all, the output is completely random, it may be worthless and it may be perfect.

An example: closing a movable water barrier

As an example, the closure procedure of a movable water barrier over a railway is taken. In the Netherlands, several railways cross dykes on places called coupures. These coupures are essentially holes in the dyke. In order to prevent inundation through these coupures, movable water barriers are build that can close the coupure. Closing the water barrier involves a difficult procedure, because the water board is reponsable for closing the water barrier, but the train dispatcher is reponsable for the train safety. Closing the water

barrier involves the cutting of the overhead traction wire. This can only be done when the 25 kV traction power is disconnected. If the overhead wire is still under power, closing the barrier will result in a short circuit and failure of closing the barrier.

If the water level reaches an alarm level, the waterboard orders the preparation of the water barrier. The dispatcher stops the trains and the operator disconnects the power. If the water level reaches the closure level, the waterboard orders the closing of the barrier. The operator will now close the barrier. The corresponding FRAM model is shown in figure 2.

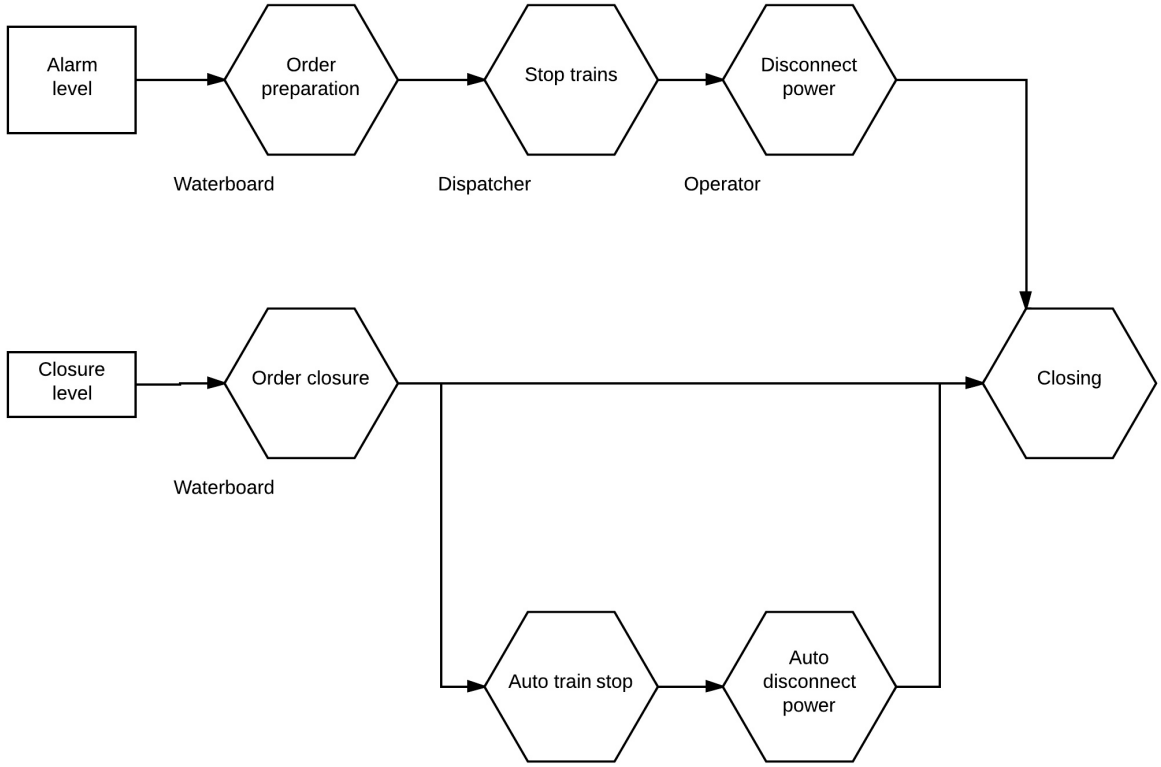

Figure 2 FRAM model of the movable water barrier

The program input and output

The input for the simulation resembles the tabular form of a FRAM model. Alle functions are subsequently written down with their input list, precondition list, resource list, control list, duration list and output list. We can show this by the input list for the above example.

```
Function('waterboard orders barrier preparation')
input         = [alarm_level]
preconditions = []
resources     = [waterboard]
control       = []
duration      = [5, human]
output        = [prepare_for_closure]
endfunction()
```

The label 'duration' is used instead of the label 'time' as used by Hollnagel, because 'time' is a reserved word in SimPy.

The output of the program consists of a list of events happening in an instantiation. This list will be different every time the simulation is run, caused by the variation. One example of the program output are listed below.

```
0.0 start simulation
10.0 alarm level reached
15.0 waterboard orders barrier preparation
25.0 dispatcher stops trains
35.0 operator disconnects traction power
40.0 closure level reached
50.0 waterboard pushes close button
60.0 barrier closed
100.0 end simulation
```

Discussion

It is shown that it is possible to construct a simple program to simulate a FRAM model. The model uses a tabular description of the FRAM model as input and produces instances of the model in descriptive form. The only additional values that have to be given are the nominal durations of the functions.

As the input format of the program is restricted to the standard FRAM descriptions, it should be relatively simple to integrate the program with the FRAM model builder and

visualiser (Hill, 2014). This would make it possible to use the model builder as an input tool. the model builder checks the completeness and integrity of the model, the simulation program will produce the simulation runs resulting from the model enabling a further check on the completeness of the model.

Further research is needed on the output format of the simulation. As for now the program results in a list of instantiations of the model. Possible ways to visualize these results are histograms of the values that several events can have, or the value of the evaluation function dependent on some background variable. The last possibility offers a way to integrate the simulation program with long term system dynamics like models to represent the mechanisms that are responsible for drift of the system parameters. A limitation of the program is that is will not take output of one function as a resource for another one. Further resource is needed to make this possible.

Many scenarios mean a lot of data. Some method is needed to make sense of this data. Some evaluation function is needed for scenarios. This might be the effectiveness of operation or the safety of operation or the duration of a process. The simulation is not dependent of the evaluation function. The simulation is the same whether the resulting data are used for evaluating the safety of the system or for evaluating the work load of some personel.

References

Bonarini, A. and Bontempi, G. (1994) A qualitative simulation approach for fuzzy dynamical models. *ACM Transactions on Modelling and Computer Simulation*, vol. 4, nr. 4, pp. 285-313.

Cheng, T. and Wu, H. (2006) Simulation with fuzzy durations. *International Journal of Applied Science and Engineering*, vol. 4, nr. 2, pp. 189-203.

Dekker, S. (2011) *Drift into failure: From hunting broken components to understanding complex systems*. Ashgate, Farnham, England.

Hill, R. (2014) Instructions for use of the FRAM model visualiser. http://www.functionalresonance.com. Accessed 16 May 2014.

Matloff, N.S. (2008a) Introduction to discrete-event simulation and the SimPy language. University of California, http://heather.cs.ucdavis.edu/~matloff/156/PLN/DESimIntro.pdf. Accessed 17 April 2014.

Matloff, N.S. (2008b) Advanced features of the SimPy language. University of California, http://heather.cs.ucdavis.edu/~matloff/156/PLN/AdvancedSimPy.tex. Accessed 17 April 2014.

Olson, D.L. and Wu, D. (2006) Simulation of fuzzy multiattribute models for grey relationships. *European Journal of Operational Research*, vol. 175, pp. 111-120.

OmzSoftware (2013) Pythonista [Computer program for iOS]. Version 1.3, available at App Store.

Sonnessa, M. (2004) *Modelling and simulation of complex systems*. PhD Torino, Italy.

The Python Language Reference. http://docs.python.org/2/reference/. Accessed 26 February 2014.

Van Kleef, E.A. and Stoop, J.A. (2014) Reliable, resilient: Towards a dialectic synthesis. *46 th ESReDA Conference*, Turino, Italy, 29-30 May 2014.

Vescovi, M.R. and Travé-Massuyès, L. (1992) A constructive approach to qualitative fuzzy simulation. *QR'92, 6th International Workshop on Qualitative Reasoning about Physical Systems*, Edinburgh, United Kingdom, 24-27 July 1992.

Vigneau, T., Müller, K. and Helmbold, B. (2012) SimPy manual. http://simpy.sourceforge.net/old/SimPy_Manual/Manuals/Manual.html. Accessed 29 April 2012.

Zadeh, L.A. (1965) Fuzzy sets. *Information and Control*, vol. 8, nr. 3, pp. 338-353.

Zhang, H., Tam, Z.M. and Li, H. (2005) Modeling uncertain activity duration by fuzzy number and discrete-event simulation. *European Journal of Operational Research*, vol. 164, pp. 715-729.