SALABIM:
OPEN SOURCE DISCRETE EVENT SIMULATION AND ANIMATION IN PYTHON

Ruud van der Ham

salabim.org
Le Maurier - Route du Maurier
04250 Turriers, FRANCE

ABSTRACT

Salabim is a new open source object-oriented package specially developed for discrete event simulation of complex control in logistics and production environments. The choice of Python as the host language means that simulations can be easily combined with powerful packages for statistical processing and presentation, web interfaces, machine learning, databases, etc. The package is built on the solid foundations of the process approach as demonstrated in Simula, Prosim and Tomas. One of the key features of salabim is the integrated animation engine. The package can be used for a wide range of simulation applications, such as (air)ports, hospitals, warehouses, job shop production, distribution systems and communication networks.

1   INTRODUCTION

There are many discrete event simulation packages around, ranging from high end 3D rendering systems to relatively simple language extensions. Commercial packages are often very expensive and interface difficult -if at all- with modern programming languages; and their propriety scripting languages are arguably not powerful enough to cope with the needs of modern simulations. At the other end of the spectrum there are several modules that extend languages like Python, Julia, R, Java and C++ with DES features. These quite often offer just very basic functionality, without addressing the real needs of a modeler. Particularly good queue handling, monitoring and animation are essential. As the project is Open Source Initiative compliant, everything is public and free. There is very active community contributing to the development of the package.

2   APPLICATION

The package can be and is used in a wide rage of simulation applications, such as (air)ports, hospitals, warehouses, job shop production, distribution systems and communication networks. It is also an excellent choice for a simulation curriculum at universities. As salabim is a Python package, the models can use any of the wide range of – nearly all open source – packages that may be relevant for a simulation projects, such as statistical analysis, high end presentations, matrix calculations, GUIs, databases, web interfacing, data science including machine learning, genetic algorithms, (integer) linear programming and convex optimization. The integrated animation engine provides a powerful tool for the model builder as well as a perfect way to show the dynamic behaviour to users and decision makers. The animations can be saved as a video and then be shared via YouTube or similar channels. All salabim models run in the same way under Windows, Linux, OS X and iOS (iPad).

3   SALABIM MODELS

Salabim is built on the solid foundations of the process approach as demonstrated in Simula, Prosim and Tomas. Each salabim model consists of so called components that may be active or just contain data. An active component has one or more process descriptions that can control its own and other components'

flow. Apart from *activate*, *hold*, *passivate* and *cancel*, salabim provides resources and states that can be used to control the flow of the components. Queues play an important role in salabim models and are much more versatile than the standard Python lists and deques, not in the least because of built-in statistics. Sampling from a wide range of distributions and data collection on the model behaviour are standard features. Figure 1 shows a sample model, where a bank office with 3 clerks serves randomly arriving customers who might have to queue. Even this small model can already be used to get statistics in the form of histograms, plots and key figures. In just a couple of lines this model can be animated, showing the customers as they queue or move, the clerks working and the dynamic length of the waiting line versus the time.
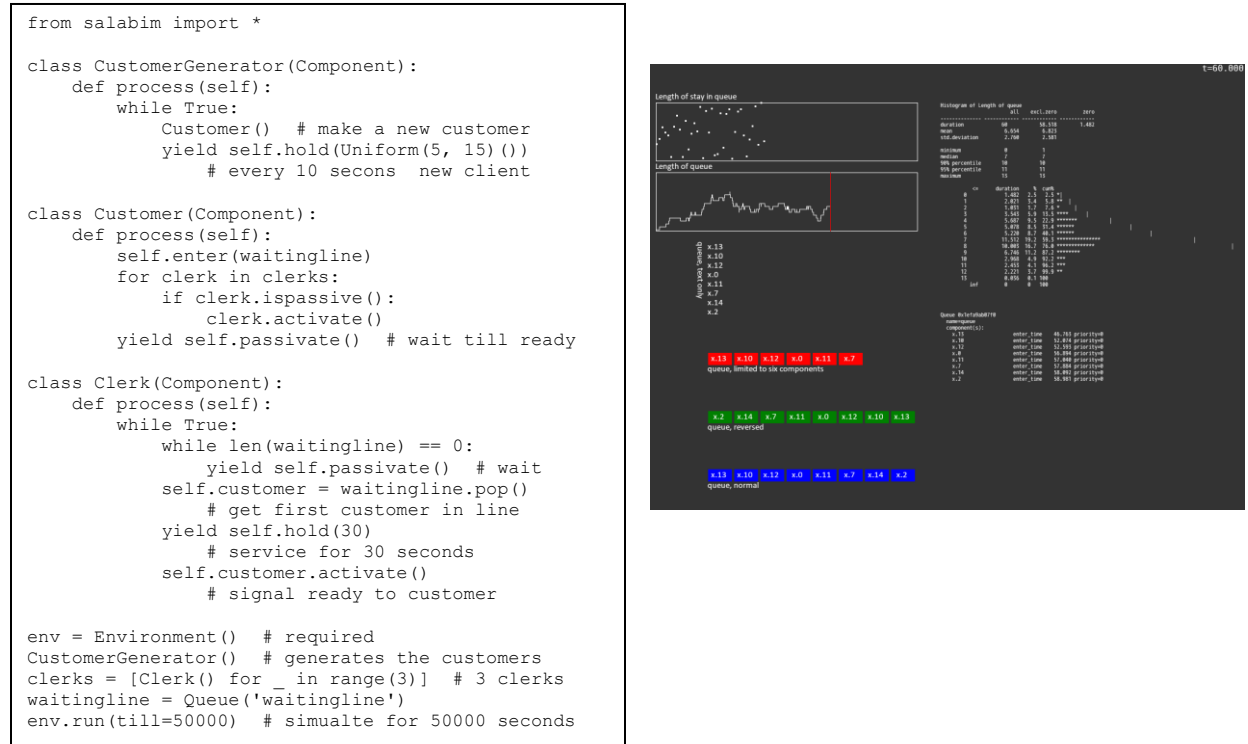
```
from salabim import *

class CustomerGenerator(Component):
    def process(self):
        while True:
            Customer()  # make a new customer
            yield self.hold(Uniform(5, 15)())
                # every 10 secons  new client

class Customer(Component):
    def process(self):
        self.enter(waitingline)
        for clerk in clerks:
            if clerk.ispassive():
                clerk.activate()
        yield self.passivate()  # wait till ready

class Clerk(Component):
    def process(self):
        while True:
            while len(waitingline) == 0:
                yield self.passivate()  # wait
            self.customer = waitingline.pop()
                # get first customer in line
            yield self.hold(30)
                # service for 30 seconds
            self.customer.activate()
                # signal ready to customer


env = Environment()  # required
CustomerGenerator()  # generates the customers
clerks = [Clerk() for _ in range(3)]  # 3 clerks
waitingline = Queue('waitingline')
env.run(till=50000)  # simualte for 50000 seconds
```


Figure 1: A simple example illustrating the use of salabim.

As an example of how optimization could be added to a model, just by inserting

```
import pulp
```

any simulation model can use solvers like CBC, YALMIP, CPLEX (commercial) or XPRESS-MP (commercial).

4 COMPARISON WITH OTHER SIMULATION TOOLS

The poster will contain a clear overview of the features, the fields of application, the structure and a comparison of salabim with various DES packages, including some high end industry standard packages.

REFERENCES

van der Ham, R. 2018. "Salabim: Discrete Event Simulation and Animation in Python". *Journal of Open Source Software* 3(27):767.