Replace module or parts' Documentation

Release 1.0

Vincent van Bergen

May 09, 2014

CHAPTER 1

Class reference

class machine_simulation.simulation.**BreakableComponent**(*env*, *name*, *time_replacement*, *stock*, *mean*, *replace_module*)

A component that breaks down every once in a while

time_to_failure()

Returns the time until next failure if this component, using the mean time to failure of this component. :return: float

class machine_simulation.simulation.**Component**(*env*, *time_replacement*, *stock*)

The specification of a part/module.

replace()

A process which replaces this component by a new one

class machine_simulation.simulation.**ComponentStock**(*env*, *unit_purchase_costs*, *delivery_time*, *unit_holding_costs*, *capacity=inf*)

Implementation of a (S-1,S) inventory management system

get(*amount*)

Parameters **amount** – the amount of stock requested

Returns returns a ContainerGet event

inventory()

A process which keeps track if inventory holding costs

order(*amount*)

A process to order an amount of items :param amount: the number of items to order

class machine_simulation.simulation.**Factory**(*env*, *number_maintenance_men*, *module*, *costs_per_unit_downtime*, *number_of_machines*, *operator_salary*, *maintenance_man_salary*)

Factory consisting of multiple Machines, operators and maintenance men

costs()

Calculates total costs for this factory :return:

track_salary()

A process which keeps track salary costs

class machine_simulation.simulation.**Machine**(*env*, *module*, *costs_per_unit_downtime*, *factory*)

A machine

process_downtime_costs()
Keeps track of downtime costs of machine

repair(*broken_component*)
Will repair machine by either replacing the module or the broken part

run()
Break machine every once in while

class machine_simulation.simulation.**Module**(*env*, *time_replacement*, *stock*, *breakable_components*)
A container for multiple breakable components

break_module(*machine*)
Waits till first part gets broken and tells this to the machine :param machine: the machine which should be broken when this breaks

CHAPTER 2

Tests

2.1 unittests Module

class machine_simulation.tests.unittests.**TestComponentStock**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Tests whether ComponentStock behaves like (S-1,S) inventory system and tests the inventory holding costs

setUp()

test_empty_stock()
Tests whether stock behaves correctly when out-of-stock

test_inventory()
Tests whether inventory holding costs are accounted for

test_multiple_get()
Tests whether stock is refilled after multiple items have been retrieved

test_single_get()
Check whether stock is refilled after an item has been retrieved

class machine_simulation.tests.unittests.**TestMachine**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Tests Machine class

setUp()

test_policy_a()
Tests situation where module is replaced when first part is broken

test_policy_b()
Tests situation where module is replaced when second part is broken

test_policy_c()
Tests situation where module is always replaced

test_policy_o()
Tests situation where part always gets replaced

class machine_simulation.tests.unittests.**TestModule**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Tests Module class

setUp()

test_break_module()
Tests whether the part of the module that is broken first is indeed return by a function inside break_module process

2.2 integrationtests Module

class machine_simulation.tests.integrationtests.**IntegrationTest**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test boundary cases

setUp()
Sets up machine with module consisting of 2 parts with non-zero inventory holding costs and non-zero purchase costs

test_zero_maintenance_men()
Succeeds if a Factory with 0 maintenance men will have only inventory holding costs

class machine_simulation.tests.integrationtests.**TestRandomSeed**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_random_seed()
Tests whether simulation results are the same for 2 simulations if a random seed is set

CHAPTER 3

Indices and tables

- *modindex*

m