## Supply Air Duct Leakages

### Description

def description

return "Duct leakage can be caused by torn or missing external duct wrap, poor workmanship around duct takeoffs and fittings, disconnected ducts, improperly installed duct mastic, and temperature and pressure cycling (Roth et al. 2004, 2005). Conditioned air leaking to an unconditioned space in buildings increases the equipment heating or cooling demand and can increase fan power for variable air volume systems. This measure simulates supply air leakage by modifying the ZoneHVAC:AirDistributionUnit object in EnergyPlus. Equation (1) provides an expression for the downstream leakage ratio (r\_(leak,dnst,F)) under faulty conditions in terms of a “normal” leakage ratio (r\_(leak,dnst)) and a fault intensity (F) defined as the ratio of the leakage flow relative to supply flow. r\_(leak,dnst,F) = 1 - ( 1 - r\_(leak,dnst) ) \* ( 1 - F ) ------ (1)"

end

### Modeler Description

def workspaceer\_description

return "Two user inputs are required to simulate the fault. The ZoneHVAC:AirDistributionUnit object has two leakage options (upstream and downstream leakages) available. For supply duct leakage, the leakage ratio (leakage flow relative to supply flow) is applied to the downstream leakage parameter and the upstream leakage parameter is replaced with zero in the object. To use this Measure, choose the AirTerminal object to be faulted and a ratio of leakage flow rate to the airflow directed to the zone upstream to the leak."

end

### Measure Type

EnergyPlus Measure

**Taxonomy**

HVAC.Ventilation

### Arguments

def arguments(workspace)

args = OpenStudio::Ruleset::OSArgumentVector.new

list = OpenStudio::StringVector.new

atsdus = workspace.getObjectsByType("AirTerminal:SingleDuct:Uncontrolled".to\_IddObjectType)

atsdus.each do |atsdu|

list << atsdu.name.to\_s

end

atddcvs = workspace.getObjectsByType("AirTerminal:DualDuct:ConstantVolume".to\_IddObjectType)

atddcvs.each do |atddcv|

list << atddcv.name.to\_s

end

atddvavs = workspace.getObjectsByType("AirTerminal:DualDuct:VAV".to\_IddObjectType)

atddvavs.each do |atddvav|

list << atddvav.name.to\_s

end

atddvavoas = workspace.getObjectsByType("AirTerminal:DualDuct:VAV:OutdoorAir".to\_IddObjectType)

atddvavoas.each do |atddvavoa|

list << atddvavoa.name.to\_s

end

atsdcvrs = workspace.getObjectsByType("AirTerminal:SingleDuct:ConstantVolume:Reheat".to\_IddObjectType)

atsdcvrs.each do |atsdcvr|

list << atsdcvr.name.to\_s

end

atsdvavrs = workspace.getObjectsByType("AirTerminal:SingleDuct:VAV:Reheat".to\_IddObjectType)

atsdvavrs.each do |atsdvavr|

list << atsdvavr.name.to\_s

end

atsdvavnrs = workspace.getObjectsByType("AirTerminal:SingleDuct:VAV:NoReheat".to\_IddObjectType)

atsdvavnrs.each do |atsdvavnr|

list << atsdvavnr.name.to\_s

end

atsdspiurs = workspace.getObjectsByType("AirTerminal:SingleDuct:SeriesPIU:Reheat".to\_IddObjectType)

atsdspiurs.each do |atsdspiur|

list << atsdspiur.name.to\_s

end

atsdppiurs = workspace.getObjectsByType("AirTerminal:SingleDuct:ParallelPIU:Reheat".to\_IddObjectType)

atsdppiurs.each do |atsdppiur|

list << atsdppiur.name.to\_s

end

atsdcvfpis = workspace.getObjectsByType("AirTerminal:SingleDuct:ConstantVolume:FourPipeInduction".to\_IddObjectType)

atsdcvfpis.each do |atsdcvfpi|

list << atsdcvfpi.name.to\_s

end

atsdvavrvsfs = workspace.getObjectsByType("AirTerminal:SingleDuct:VAV:Reheat:VariableSpeedFan".to\_IddObjectType)

atsdvavrvsfs.each do |atsdvavrvsf|

list << atsdvavrvsf.name.to\_s

end

atsdvavhacrs = workspace.getObjectsByType("AirTerminal:SingleDuct:VAV:HeatAndCool:Reheat".to\_IddObjectType)

atsdvavhacrs.each do |atsdvavhacr|

list << atsdvavhacr.name.to\_s

end

atsdvavhacnrs = workspace.getObjectsByType("AirTerminal:SingleDuct:VAV:HeatAndCool:NoReheat".to\_IddObjectType)

atsdvavhacnrs.each do |atsdvavhacnr|

list << atsdvavhacnr.name.to\_s

end

# make choice arguments for fan

airterminal\_choice = OpenStudio::Ruleset::OSArgument::makeChoiceArgument("airterminal\_choice", list, true)

airterminal\_choice.setDisplayName("Select the name of the faulted AirTerminal object")

airterminal\_choice.setDefaultValue(list[0].to\_s)

args << airterminal\_choice

# make a double argument for the leakage ratio

leak\_ratio = OpenStudio::Ruleset::OSArgument::makeDoubleArgument('leak\_ratio', false)

leak\_ratio.setDisplayName('Ratio of leak airflow between 0 and 0.3.')

leak\_ratio.setDefaultValue(0.1) # default leakage level to be 10%

args << leak\_ratio

return args

end

### Initial Condition

#Select air terminal object that is being faulted.

runner.registerInitialCondition("Imposing duct leakages on #{airterminal\_choice}.")

### Final Condition

#Duct leakage applied to the air terminal object.

runner.registerFinalCondition("Imposed performance degradation on #{airterminal\_choice}.")

### Not Applicable

#When selected air terminal is not connected to a return air plenum,

runner.registerAsNotApplicable("#{airterminal\_choice} cannot leak because there are no return plenums for it to leak its airflow. Skipping......")

#If leakage ratio is applied as zero,

runner.registerAsNotApplicable("SupplyAirDuctLeakages is not running for #{airterminal\_choice}. Skipping......")

### Warning

n/a

### Error

#When fault intensity is out of range,

runner.registerError("Fault level #{leak\_ratio} for #{airterminal\_choice} is outside the range from 0 to 0.3. Exiting......")

#When selected air terminal does not have ZoneHVAC:EquipmentList defined,

runner.registerError("Measure AirTerminalSupplyDownstreamLeakToReturn cannot find the ZoneHVAC:EquipmentList that contains #{airterminal\_choice}. Exiting......")

#When selected air terminal does not have AIrLoopHVAC:ZoneSplitter defined,

runner.registerError("Measure AirTerminalSupplyDownstreamLeakToReturn cannot find the AirLoopHVAC:ZoneSplitter that contains #{airterminal\_choice}. Exiting......")

#When additional strings objects to run the model were not inserted properly,

runner.registerError("#{string\_object} inserted unsuccessfully. Exiting......")

### Information

n/a

### Code Outline

* Define arguments (air terminal where fault occurs, fault level in constant value).
* Check constant fault level value (within 0-0.3).
* Replace object to AirTerminal:DingleDuct:ConstantVolume:Reheat if air terminal selected is AirTerminal:SingleDuct:Uncontrolled
  + Read fields from AirTerminal:SingleDuct:Uncontrolled
  + Read fields from ZoneHVAC:AirDistributionUnit if available
  + Change AirLoopHVAC:ZoneSplitter with new node name
  + Create new objects
    - AirTerminal:SingleDuct:ConstantVolume:Reheat
    - Coil:Heating:Electric
    - ZoneHVAC:AirDistributionUnit
* Apply fault to the selected air terminal
  + Find AirLoopHVAC:ReturnPlenum connected to the selected air terminal
  + Check node connections
  + Modify Constant Downstream Leakage Fraction field based on the fault intensity defined by the user
  + Define small number for Nominal Upstream Leakage Fraction

### Tests

* Test model with different air terminal object types
* Test invalid user argument values to make sure measure fails gracefully