## Oversized Equipment at Design

### Description

def description

return "Oversizing of heating and cooling equipment is commonly accepted in real-world applications. In a previous study (Felts and Bailey 2000), more than 40% of the units surveyed were oversized by more than 25%, and 10% were oversized by more than 50%. System oversizing can ensure that the highest heating and cooling demands are met. But excessive oversizing of units can lead to increased equipment cycling with increased energy use due to efficiency losses. The fault intensity (F) for this fault is defined as the ratio of increased sizing compared to the correct sizing."

end

### Modeler Description

def modeler\_description

return "This measure simulates the effect of oversized equipment at design by modifying the Sizing:Parameters object and capacity fields in coil objects in EnergyPlus assigned to the heating and cooling system. One user input is required; percentage of increased sizing. Current measure applicable to following objects; coilcoolingdxsinglespeed, coilcoolingdxtwospeed, coilcoolingdxtwostagewithhumiditycontrolmode, coilcoolingdxvariablerefrigerantflow, coilheatingdxvariablerefrigerantflow, coilheatinggas, coilheatingelectric"

end

### Measure Type

OpenStudio Measure

**Taxonomy**

HVAC.Whole System

### Arguments

def arguments(model)

args = OpenStudio::Ruleset::OSArgumentVector.new

list = OpenStudio::StringVector.new

list << $all\_coil\_selection

#Cooling coils

coilcoolingdxsinglespeeds = model.getCoilCoolingDXSingleSpeeds

coilcoolingdxsinglespeeds.each do |coilcoolingdxsinglespeed|

list << coilcoolingdxsinglespeed.name.to\_s

end

coilcoolingdxtwospeeds = model.getCoilCoolingDXTwoSpeeds

coilcoolingdxtwospeeds.each do |coilcoolingdxtwospeed|

list << coilcoolingdxtwospeed.name.to\_s

end

coilcoolingdxtwostagewithhumiditycontrolmodes = model.getCoilCoolingDXTwoStageWithHumidityControlModes

coilcoolingdxtwostagewithhumiditycontrolmodes.each do |coilcoolingdxtwostagewithhumiditycontrolmode|

list << coilcoolingdxtwostagewithhumiditycontrolmode.name.to\_s

end

coilcoolingdxvariablerefrigerantflows = model.getCoilCoolingDXVariableRefrigerantFlows

coilcoolingdxvariablerefrigerantflows.each do |coilcoolingdxvariablerefrigerantflow|

list << coilcoolingdxvariablerefrigerantflow.name.to\_s

end

#Heating coils

coilheatingdxvariablerefrigerantflows = model.getCoilHeatingDXVariableRefrigerantFlows

coilheatingdxvariablerefrigerantflows.each do |coilheatingdxvariablerefrigerantflow|

list << coilheatingdxvariablerefrigerantflow.name.to\_s

end

coilheatinggass = model.getCoilHeatingGass

coilheatinggass.each do |coilheatinggas|

list << coilheatinggas.name.to\_s

end

coilheatingelectrics = model.getCoilHeatingElectrics

coilheatingelectrics.each do |coilheatingelectric|

list << coilheatingelectric.name.to\_s

end

coil\_choice = OpenStudio::Ruleset::OSArgument.makeChoiceArgument('coil\_choice', list, true)

coil\_choice.setDisplayName("Enter the name of the oversized coil object. If you want to impose the fault on all equipment, select #{$all\_coil\_selection}")

coil\_choice.setDefaultValue("#{$all\_coil\_selection}")

args << coil\_choice

#make an argument for excessive sizing

sizing\_increase\_percent = OpenStudio::Ruleset::OSArgument::makeDoubleArgument("sizing\_increase\_percent",true)

sizing\_increase\_percent.setDisplayName("Sizing Increase (between 0-50%).")

sizing\_increase\_percent.setDefaultValue(10.0)

args << sizing\_increase\_percent

return args

end

### Initial Condition

#When all coils are selected,

runner.registerInitialCondition('Oversized Equipment at Design fault are being applied on all coils......')

#When one coil is selected,

runner.registerInitialCondition("Oversized Equipment at Design fault is being applied to the #{coil\_choice}......")

### Final Condition

#After all coils are imposed with fault,

runner.registerFinalCondition('Oversized Equipment at Design fault applied on all coils......')

#After one coil is imposed with fault,

runner.registerFinalCondition("Oversized Equipment at Design fault applied to the #{coil\_choice}......")

### Not Applicable

#When fault intensity is defined as less than 0.001%,

runner.registerAsNotApplicable("Fault intensity #{sizing\_increase\_percent} is defined too small. Skipping......")

### Warning

n/a

### Error

# When fault intensity is defined outside of range 0-50%,

runner.registerError("Fault intensity #{sizing\_increase\_percent} is defined outside the range from 0 to 50%. Exiting......")

### Information

* Modifies sizing:parameter and capacity field in coil objects based on fault intensity.
* Works with,
  + Coil:Cooling:Dx:SingleSpeed
  + Coil:Cooling:Dx:TwoSpeed
  + Coil:Cooling:Dx:TwoStageWithHumidityControlMode
  + Coil:Cooling:Dx:VariableRefrigerantFlow
  + Coil:Heating:Dx:VariableRefrigerantFlow
  + Coil;Heating:Gas
  + Coil;Heating:Electric

### Code Outline

* Define arguments (coil where fault occurs, percentage of increased sizing parameter).
  + Check whether fault intensity (percentage of increased sizing parameter) is reasonably defined within 0-50%.
  + Modify capacity based on the fault intensity.
    - If all coils are selected,
      * Modify sizing parameter based on fault intensity
        + Heating sizing parameter
        + Cooling sizing parameter
      * Read component names that are defined in the model and modify capacity field in the selected coils that has the same name.
        + Coilcoolingdxsinglespeed
        + Coilcoolingdxvariablerefrigerantflow
        + Coilcoolingdxtwostagewithhumiditycontrolmode
        + Coilcoolingdxtwospeed
        + Coilheatingdxvariablerefrigerantflow
        + Coilheatinggas
        + Coilheatingelectric
    - And if one coil object is selected,
      * Modify capacity field in the selected coil that has the same name.
        + Coilcoolingdxsinglespeed
        + Coilcoolingdxvariablerefrigerantflow
        + Coilcoolingdxtwostagewithhumiditycontrolmode
        + Coilcoolingdxtwospeed
        + Coilheatingdxvariablerefrigerantflow
        + Coilheatinggas
        + Coilheatingelectric

### Tests

* Test invalid user argument values to make sure measure fails gracefully
* Test different coil types.