## Excessive Infiltration

### Description

def description

return "Excessive infiltration around the building envelope occurs by the unintentional introduction of outside air into a building, typically through cracks in the building envelope and through use of windows and doors. Infiltration is driven by pressure differences between indoors and outdoors of the building caused by wind and by air buoyancy forces known commonly as the stack effect (ASHRAE Handbook Fundamentals, 2005). Excessive infiltration can affect thermal comfort, indoor air quality, heating and cooling demand, and moisture damage of building envelope components (Emmerich et al., 2005). The fault intensity is defined as the percentage of excessive infiltration around the building envelope compared to the non-faulted condition."

end

### Modeler Description

def modeler\_description

return "The user input of the percentage of excessive infiltration is applied to one of either four variables (Design Flow Rate, Flow per Zone Floor Area, Flow per Exterior Surface Area, Air Changes per Hour) in ZoneInfiltration:DesignFlowRate object and one variable (Effective Air Leakage Area) in ZoneInfiltration:EffectiveLeakageArea depending on the user’s choice of infiltration implementation method to impose fault over the original (non-faulted) configuration. The modified value (Infil\_m) is calculated as Infil\_m = Infil\_o \* (1+F/100), where Infil\_o is the original value defined in the infiltration object and F is the percentage of excessive infiltration."

end

### Measure Type

OpenStudio Measure

**Taxonomy**

Envelope.Infiltration

### Arguments

def arguments(model)

args = OpenStudio::Ruleset::OSArgumentVector.new

#make a choice argument for model objects

space\_type\_handles = OpenStudio::StringVector.new

space\_type\_display\_names = OpenStudio::StringVector.new

space\_type\_display\_names << "\*Entire Building\*"

#putting model object and names into hash

space\_type\_args = model.getSpaceTypes

space\_type\_args\_hash = {}

space\_type\_args.each do |space\_type\_arg|

space\_type\_args\_hash[space\_type\_arg.name.to\_s] = space\_type\_arg

end

#looping through sorted hash of model objects

space\_type\_args\_hash.sort.map do |key,value|

#only include if space type is used in the model

if value.spaces.size > 0

space\_type\_handles << value.handle.to\_s

space\_type\_display\_names << key

end

end

#add building to string vector with space type

building = model.getBuilding

space\_type\_handles << building.handle.to\_s

#make a choice argument for space type

space\_type = OpenStudio::Ruleset::OSArgument::makeChoiceArgument("space\_type", space\_type\_handles, space\_type\_display\_names)

space\_type.setDisplayName("Apply the Measure to a Specific Space Type or to the Entire Model.")

space\_type.setDefaultValue("\*Entire Building\*") #if no space type is chosen this will run on the entire building

args << space\_type

#make an argument for excessive infiltration percentage

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

space\_infiltration\_increase\_percent.setDisplayName("Space Infiltration Increase (%).")

space\_infiltration\_increase\_percent.setDefaultValue(20.0)

args << space\_infiltration\_increase\_percent

return args

end

### Initial Condition

#Checking the number of infiltration object defined in the model,

runner.registerInitialCondition("The initial model contained #{space\_infiltration\_objects.size + space\_infiltration\_ela\_objects.size} space infiltration objects.")

#If there is no infiltration object defined in the model,

runner.registerInitialCondition("The initial model did not contain any space infiltration objects.")

### Final Condition

#Report number of infiltration object modified,

runner.registerFinalCondition("#{altered\_instances} space infiltration objects in the model were altered.")

### Not Applicable

#When no infiltration objects were modified,

runner.registerAsNotApplicable("No space infiltration objects were found in the specified space type(s) and no life cycle costs were requested.")

### Warning

#When fault intensity (increase of infiltration) is defined too small,

runner.registerWarning("A space infiltration increase percentage of #{space\_infiltration\_increase\_percent} percent is abnormally low.")

#When fault intensity (increase of infiltration) is defined too high,

runner.registerWarning("A space infiltration increase percentage of #{space\_infiltration\_increase\_percent} percent is abnormally high.")

#When infiltration object’s field is not properly defined,

runner.registerWarning("'#{instance.name}' is used by one or more instances and has no load values.")

### Error

#When space type was not chosen,

runner.registerError("No space type was chosen.")

#When chosen space type is not found in the model,

runner.registerError("The selected space type with handle '#{handle}' was not found in the model. It may have been removed by another measure.")

### Information

* Works with,
  + ZoneInfiltration:DesignFlowRate
  + ZoneInfiltration:EffectiveLeakageArea.
* Future refinement item is,
  + "Space" input option (with drop down menu) instead of "SpaceType" option. LCC cost codes are not used.

### Code Outline

* Define arguments (zone where fault occurs, percentage of increased infiltration).
  + Check whether fault intensity (increase of infiltration) is reasonably defined within 0-100.
  + Modify infiltration objects based on “space types”.
    - Read and replace infiltration method defined in ZoneInfiltration:DesignFlowRate.
      * Replace infiltration values based on user defined fault intensity (F)
        + designFlowRate = designFlowRate \* (1 + F/100)
        + flowperSpaceFloorArea = flowperSpaceFloorArea \* (1 + F/100)
        + flowperExteriorSurfaceArea = flowperExteriorSurfaceArea \* (1 + F/100)
        + flowperExteriorWallArea = flowperExteriorWallArea \* (1 + F/100)
        + airChangesperHour = airChangesperHour \* (1 + F/100)
    - Read and replace infiltration method defined in ZoneInfiltration:EffectiveLeakageArea.
      * Replace infiltration values based on user defined fault intensity
        + effectiveAirLeakageArea = effectiveAirLeakageArea \* (1 + F/100)
  + Modify infiltration objects based on “spaces”.
    - Read and replace infiltration method defined in ZoneInfiltration:DesignFlowRate.
      * Replace infiltration values based on user defined fault intensity (F)
        + designFlowRate = designFlowRate \* (1 + F/100)
        + flowperSpaceFloorArea = flowperSpaceFloorArea \* (1 + F/100)
        + flowperExteriorSurfaceArea = flowperExteriorSurfaceArea \* (1 + F/100)
        + flowperExteriorWallArea = flowperExteriorWallArea \* (1 + F/100)
        + airChangesperHour = airChangesperHour \* (1 + F/100)
    - Read and replace infiltration method defined in ZoneInfiltration:EffectiveLeakageArea.
      * Replace infiltration values based on user defined fault intensity
        + effectiveAirLeakageArea = effectiveAirLeakageArea \* (1 + F/100)

### Tests

* Test invalid user argument values to make sure measure fails gracefully
* Test different infiltration methods.